



在上为多租户 SaaS 应用程序实施托管 PostgreSQL AWS

AWS 规范性指导



AWS 规范性指导: 在上为多租户 SaaS 应用程序实施托管 PostgreSQL AWS

Copyright © 2024 Amazon Web Services, Inc. and/or its affiliates. All rights reserved.

Amazon 的商标和商业外观不得用于任何非 Amazon 的商品或服务，也不得以任何可能引起客户混淆、贬低或诋毁 Amazon 的方式使用。所有非 Amazon 拥有的其他商标均为各自所有者的财产，这些所有者可能附属于 Amazon、与 Amazon 有关联或由 Amazon 赞助，也可能不是如此。

Table of Contents

| | |
|---|----|
| 简介 | 1 |
| 目标业务成果 | 1 |
| 为 SaaS 应用程序选择数据库 | 2 |
| 在 Amazon RDS 和 Aurora 之间进行选择 | 3 |
| PostgreSQL 的多租户 SaaS 分区模型 | 5 |
| PostgreSQL 筒仓模型 | 6 |
| PostgreSQL 池模型 | 7 |
| PostgreSQL 桥模型 | 8 |
| 决策矩阵 | 9 |
| 行级别安全建议 | 18 |
| PostgreSQL 可用于池模型 | 20 |
| 最佳实践 | 21 |
| 比较托管 PostgreSQL 的 AWS 选项 | 21 |
| 选择多租户 SaaS 分区模型 | 21 |
| 对池 SaaS 分区模型使用行级安全性 | 21 |
| 常见问题 | 22 |
| 确实 AWS 提供了哪些托管 PostgreSQL 选项？ | 22 |
| 哪种服务最适合 SaaS 应用程序？ | 22 |
| 如果我决定将 PostgreSQL 数据库与多租户 SaaS 应用程序一起使用，我应该考虑哪些独特的要求？ | 22 |
| 我可以哪些模型通过 PostgreSQL 维护租户数据隔离？ | 22 |
| 如何使用由多个租户共享的单个 PostgreSQL 数据库维护租户数据隔离？ | 22 |
| 后续步骤 | 23 |
| 资源 | 24 |
| 参考信息 | 24 |
| 合作伙伴 | 24 |
| 文档历史记录 | 25 |
| 术语表 | 26 |
| # | 26 |
| A | 26 |
| B | 29 |
| C | 30 |
| D | 33 |
| E | 36 |

| | |
|---------|----|
| F | 38 |
| G | 39 |
| H | 39 |
| I | 40 |
| L | 42 |
| M | 43 |
| O | 46 |
| P | 49 |
| Q | 51 |
| R | 51 |
| S | 54 |
| T | 56 |
| U | 58 |
| V | 58 |
| W | 58 |
| Z | 59 |
| | ix |

在上为多租户 SaaS 应用程序实施托管 PostgreSQL AWS

Tabby Ward 和 Thomas Davis , Amazon Web Services (AWS)

2024 年 4 月 ([文档历史记录](#))

在选择用于存储运营数据的数据库时，必须考虑如何构造数据、它将回答哪些查询、提供答案的速度以及数据平台本身的弹性。除了这些一般考虑因素外，还包括软件即服务 (SaaS) 对运营数据的影响，例如性能隔离、租户安全以及多租户 SaaS 应用程序数据的典型特征和设计模式。本指南讨论了这些因素如何适用于使用亚马逊网络服务 AWS 上的 PostgreSQL 数据库作为多租户 SaaS 应用程序的主要操作数据存储。具体而言，该指南重点介绍两个 AWS 托管的 PostgreSQL 选项：兼容亚马逊 Aurora PostgreSQL 的版本和适用于 PostgreSQL 的亚马逊关系数据库服务 (Amazon RDS)。

目标业务成果

本指南详细分析了使用兼容 Aurora PostgreSQL 和亚马逊 RDS for PostgreSQL 的多租户 SaaS 应用程序的最佳实践。我们建议您使用本指南中提供的设计模式和概念，为多租户 SaaS 应用程序提供信息并标准化与 Aurora PostgreSQL 兼容的 Aurora PostgreSQL 或 Amazon RDS for PostgreSQL 的实施。

本规范性指南有助于实现以下业务成果：

- 为您的用例选择最优的 AWS 托管 PostgreSQL 选项 — 本指南将数据库使用的关系和非关系选项与 SaaS 应用程序进行了比较。它还讨论了哪些用例最适合兼容 Aurora PostgreSQL 和亚马逊 RDS for PostgreSQL。这些信息将有助于为您的 SaaS 应用程序选择最佳选项。
- 通过采用 SaaS 分区模型来实施 SaaS 最佳实践 — 本指南讨论并比较了适用于 PostgreSQL 数据库管理系统 (DBMS) 的三种广泛的 SaaS 分区模型：池、桥接和孤岛模型及其变体。这些方法捕获了 SaaS 最佳实践，并在设计 SaaS 应用程序时提供了灵活性。SaaS 分区模型的实施是保留最佳实践的关键部分。
- 在池 SaaS 分区模型中有效使用 RLS — 行级安全 (RLS) 通过根据用户或上下文变量限制可以查看的行，支持在单个 PostgreSQL 表中强制执行租户数据隔离。使用池分区模型时，需要使用 RLS 来防止跨租户访问。

为 SaaS 应用程序选择数据库

对于许多多租户 SaaS 应用程序，选择操作数据库可以提炼为在关系数据库和非关系数据库之间进行选择，或者两者的组合。要做出决定，请考虑以下高级应用程序数据要求和特征：

- 应用程序的数据模型
- 数据的访问模式
- 数据库延迟要求
- 数据完整性和事务完整性要求（原子性、一致性、隔离性和持久性或 ACID）
- 跨区域可用性和恢复要求

下表列出了应用程序数据要求和特征，并在 AWS 数据库产品的背景下对其进行了讨论：兼容 Aurora PostgreSQL 和适用于 PostgreSQL 的 Amazon RDS（关系），以及亚马逊 DynamoDB（非关系）。当你试图在关系型和非关系型操作数据库产品之间做出决定时，你可以参考这个矩阵。

| 数据库 | SaaS 应用程序数据要求和特征 | | | | |
|--|------------------|--------------|--|--------------------|--|
| | 数据模型 | 访问模式 | 延迟要求 | 数据和交易完整性 | 跨区域可用性和恢复 |
| 关系型 (兼容 Aurora PostgreSQL 和适用于 PostgreSQL 的亚马逊 RDS) | 关系型或高度标准化型。 | 不必事先进行彻底的计划。 | 最好是更高的延迟容忍度；默认情况下，使用 Aurora 以及通过实现只读副本、缓存和类似功能可以实现更低的延迟。 | 默认情况下，保持高数据和交易完整性。 | 在 Amazon RDS 中，您可以创建用于跨区域扩展和故障转移的只读副本。 Aurora 主要实现这一过程的自动化 。对于跨多个的主动-主动配置 AWS 区域，您可以将 写入转发与 Aurora |

| | | | | | |
|-----------------------------|--|-----------------------------------|---|------------------------------------|--|
| 非关系型 (亚马逊 DynamoDB) | 通常是非规范化的。这些数据库利用模式对 many-to-many关系 、 大型项目 和 时间序列数据 进行建模。 | 在生成数据模型之前，必须彻底了解数据的所有访问模式 (查询)。 | 借助诸如亚马逊 DynamoDB 加速器 (DAX) 之类的选项，延迟非常低，可以进一步提高性能。 | 以牺牲性能为代价的可选交易完整性。数据完整性问题已转移到应用程序上。 | 全局数据库 结合使用。 使用全局表轻松实现跨区域恢复和主动-主动配置。 (只有在单个 AWS 地区才能实现 ACID 合规性。) |
|-----------------------------|--|-----------------------------------|---|------------------------------------|--|

某些多租户 SaaS 应用程序可能具有独特的数据模型或特殊情况，上表中未包含的数据库可以更好地满足这些需求。例如，时间序列数据集、高度互联的数据集或维护集中式交易账本可能需要使用不同类型的数据库。分析所有可能性超出了本指南的范围。有关 AWS 数据库产品以及它们如何从高层次满足不同用例的完整列表，请参阅《Amazon Web Services 概述》白皮书的“[数据库](#)”部分。

本指南的其余部分重点介绍支持 PostgreSQL 的 AWS 关系数据库服务：兼容 Amazon RDS 和 Aurora PostgreSQL。DynamoDB 需要一种不同的方法来优化 SaaS 应用程序，这超出了本指南的范围。有关 DynamoDB 的更多信息，请参阅博客[文章使用 Amazon DynamoDB AWS 对池化的多租户 SaaS 数据进行分区](#)。

在 Amazon RDS 和 Aurora 之间进行选择

在大多数情况下，我们建议使用兼容 Aurora PostgreSQL 而不是亚马逊 RDS for PostgreSQL。下表显示了在这两个选项之间做出决定时应考虑的因素。

| DBMS 组件 | Amazon RDS for PostgreSQL | 兼容 Aurora PostgreSQL |
|---------|---------------------------------|--|
| 可扩展性 | 复制延迟几分钟，最多 5 个只读副本 | 复制延迟不到一分钟 (对于全局数据库，复制延迟通常小于 1 秒)，最多 15 个只读副本 |
| 崩溃恢复 | 检查点间隔 5 分钟 (默认情况下)，可能会降低数据库性能 | 使用并行线程进行异步恢复，可实现快速恢复 |

| DBMS 组件 | Amazon RDS for PostgreSQL | 兼容 Aurora PostgreSQL |
|-----------|-------------------------------------|---|
| 故障转移 | 除了崩溃恢复时间外，还有 60-120 秒 | 通常大约 30 秒（包括崩溃恢复） |
| 存储 | 最大 IOPS 为 256,000 | IOPS 仅受 Aurora 实例大小和容量的限制 |
| 高可用性和灾难恢复 | 两个可用区，带备用实例，跨区域故障转移到只读副本或复制的备份 | 默认为三个可用区，使用 Aurora 全球数据库进行跨区域故障转移，主动-主动 配置跨 AWS 区域 写入转发 |
| 备份 | 在备份窗口期间，可能会影响性能 | 自动增量备份，不影响性能 |
| 数据库实例类 | 查看 Amazon RDS 实例类列表 | 查看 Aurora 实例类列表 |

在上表中描述的所有类别中，兼容 Aurora PostgreSQL 通常是更好的选择。但是，对于中小型工作负载，Amazon RDS for PostgreSQL 可能仍然有意义，因为它有更多的实例类可供选择，可能会以牺牲 Aurora 更强大的功能集为代价，提供更具成本效益的选择。

PostgreSQL 的多租户 SaaS 分区模型

实现多租户的最佳方法取决于您的 SaaS 应用程序的要求。以下各节演示了在 PostgreSQL 中成功实现多租户的分区模型。

Note

本节中讨论的模型既适用于适用于 PostgreSQL 的 Amazon RDS，也适用于兼容 Aurora PostgreSQL 的。本节中对 PostgreSQL 的引用适用于这两种服务。

您可以在 PostgreSQL 中使用三种高级模型进行 SaaS 分区：孤岛、桥接和池。下图总结了筒仓和池模型之间的权衡。桥梁模型是筒仓和水池模型的混合体。

| 分区模型 | 优点 | 缺点 |
|------|---|--|
| 筒仓 | <ul style="list-style-type: none"> • 合规性 • 没有跨租户影响 • 租户级调整 • 租户级别的可用性 | <ul style="list-style-type: none"> • 敏捷性 • 没有集中式管理 • 部署复杂性 • 费用 |
| 池 | <ul style="list-style-type: none"> • 敏捷性 • 成本优化 • 集中式管理 • 简化部署 | <ul style="list-style-type: none"> • 跨租户影响 • 合规性挑战 • 全有或全无可用性 |
| 桥 | <ul style="list-style-type: none"> • 合规性略有调整 • 敏捷性 • 成本优化 • 集中式管理 | <ul style="list-style-type: none"> • 一些合规性挑战 • 全有或全无可用性（主要是） • 跨租户影响 • 部署复杂性 |

以下各节将更详细地讨论每种模型。

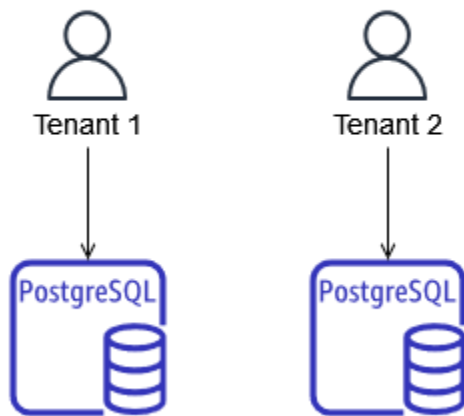
分区模型：

- [PostgreSQL 筒仓模型](#)
- [PostgreSQL 池模型](#)
- [PostgreSQL 桥模型](#)
- [决策矩阵](#)

PostgreSQL 筒仓模型

孤岛模型是通过为应用程序中的每个租户预置一个 PostgreSQL 实例来实现的。该筒仓模型在租户性能和隔离方面表现出色，完全消除了噪音邻居现象。当一个租户对系统的使用影响另一个租户的性能时，就会出现噪音邻居现象。孤岛模型允许您专门为每个租户量身定制性能，并有可能将停机限制在特定租户的孤岛上。但是，通常推动采用孤岛模式的是严格的安全和监管限制。这些限制可能由 SaaS 客户驱动。例如，由于内部限制，SaaS 客户可能会要求隔离其数据，而 SaaS 提供商可能会额外付费提供此类服务。

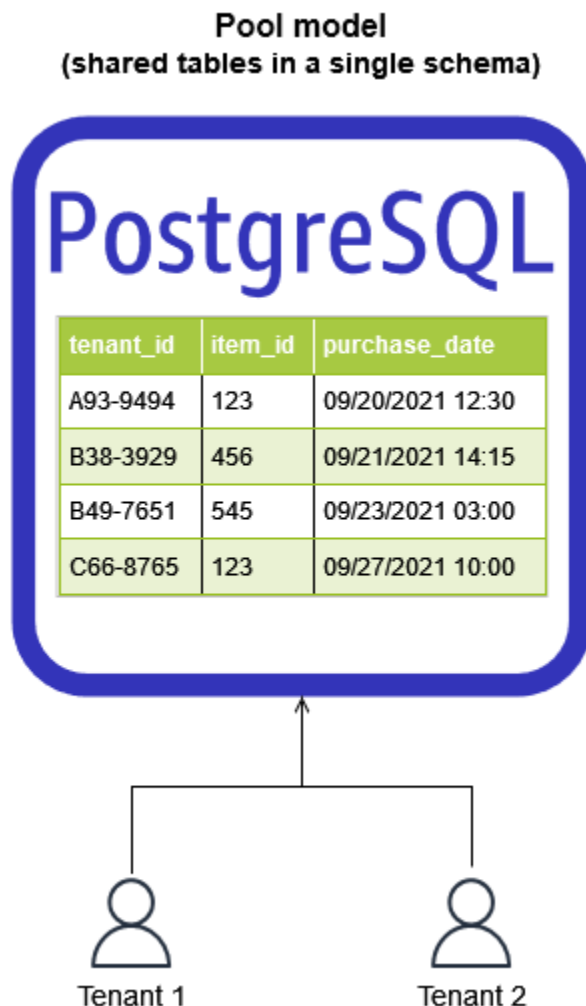
Silo model
(separate PostgreSQL instances or clusters for each tenant)



尽管在某些情况下可能需要筒仓模型，但它有许多缺点。通常很难以经济高效的方式使用孤岛模型，因为管理多个 PostgreSQL 实例的资源消耗可能很复杂。此外，该模型中数据库工作负载的分布式性质使得维护租户活动的集中视图变得更加困难。管理这么多独立运行的工作负载会增加运营和管理开销。孤岛模式还使租户入职变得更加复杂和耗时，因为您必须配置租户特定的资源。此外，整个 SaaS 系统可能更难扩展，因为越来越多的租户专用 PostgreSQL 实例将需要更多的操作时间来管理。最后一个考虑因素是，应用程序或数据访问层必须维护租户与其关联的 PostgreSQL 实例的映射，这增加了实现该模型的复杂性。

PostgreSQL 池模型

池模型是通过配置单个 PostgreSQL 实例 (Amazon RDS 或 Aurora) 并使用[行级安全性 \(RLS\)](#)来维护租户数据隔离来实现的。RLS 策略限制SELECT查询返回表中的哪些行，或者哪些行受到INSERTUPDATE、和DELETE命令的影响。池模型将所有租户数据集中在单个 PostgreSQL 架构中，因此成本效益明显提高，维护所需的运营开销也更少。由于该解决方案的集中化，监视该解决方案也要简单得多。但是，在池模型中监控租户特定的影响通常需要在应用程序中使用一些额外的工具。这是因为默认情况下 PostgreSQL 不知道哪个租户在消耗资源。由于不需要新的基础架构，因此简化了租户入职流程。这种敏捷性可以更轻松地完成快速、自动化的租户入职工作流程。



尽管池模式通常更具成本效益，更易于管理，但它确实有一些缺点。在泳池模型中，噪声邻居现象无法完全消除。但是，可以通过确保 PostgreSQL 实例上有适当的可用资源以及使用策略来减少 PostgreSQL 中的负载（例如将查询转移到只读副本或 Amazon ElastiCache）。有效

的监控在响应租户性能隔离问题方面也起着作用，因为应用程序工具可以记录和监控租户特定的活动。最后，一些 SaaS 客户可能认为 RLS 提供的逻辑分离不够，可能会要求采取额外的隔离措施。

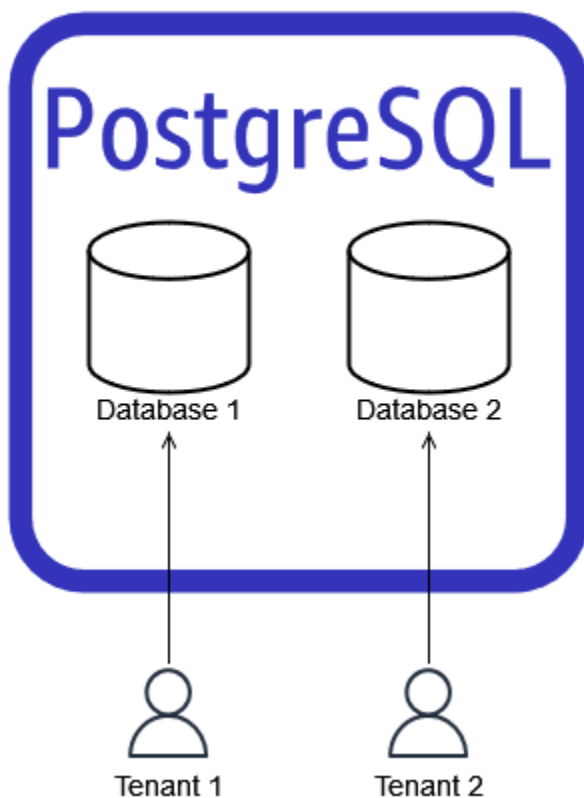
PostgreSQL 桥模型

PostgreSQL 桥接模型是合并方法和孤立方法的组合。与池化模型一样，您可以为每个租户预置一个 PostgreSQL 实例。为了保持租户数据隔离，您可以使用 PostgreSQL 逻辑结构。在下图中，PostgreSQL 数据库用于在逻辑上分隔数据。

Note

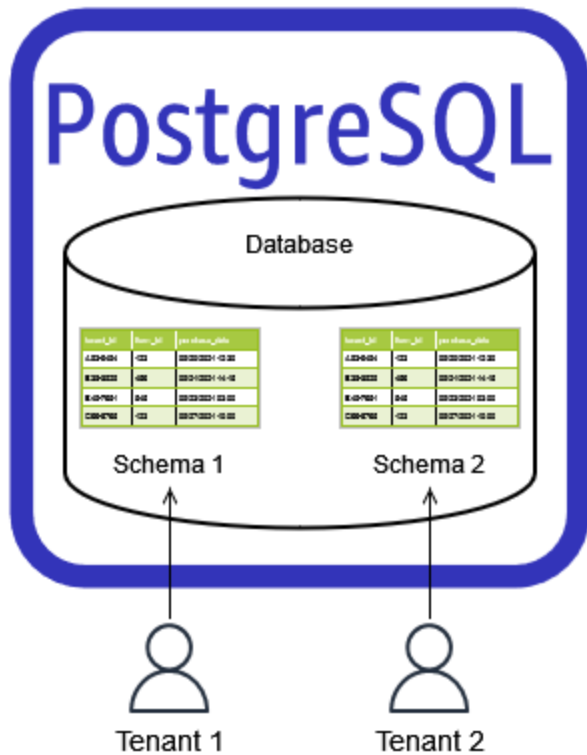
PostgreSQL 数据库不引用单独的 PoAmazon RDS for PostgreSQL 或兼容 Aurora PostgreSQL 的数据库实例。相反，它指的是 PostgreSQL 数据库管理系统的逻辑结构，用于分隔数据。

Bridge model with separate databases (separate databases in a single instance)



您也可以使用单个 PostgreSQL 数据库来实现桥接模型，每个数据库中都有租户特定的架构，如下图所示。

Bridge model with separate schemas (separate schemas in a single database)



桥梁模型与池模型一样存在噪声邻居和租户性能隔离问题。它还要求按租户配置单独的数据库或架构，从而产生一些额外的运营和配置开销。它需要有效的监控才能快速响应租户的性能问题。它还需要应用程序工具来监控租户特定的使用情况。总体而言，桥接模型可以看作是RLS的替代方案，RLS需要新的PostgreSQL数据库或架构，从而稍微增加了租户的入门工作量。与孤岛模型一样，应用程序或数据访问层必须维护租户与其关联的 PostgreSQL 数据库或架构的映射。

决策矩阵

要决定应在 PostgreSQL 中使用哪种多租户 SaaS 分区模型，请查阅以下决策矩阵。该矩阵分析了这四个分区选项：

- Silo-每个租户具有单独的 PostgreSQL 实例或集群。
- 使用不同的数据库进行桥接-单个 PostgreSQL 实例或集群中的每个租户都有一个单独的数据库。

- 使用不同的架构进行桥接 — 单个 PostgreSQL 数据库、单个 PostgreSQL 实例或集群中每个租户的单独架构。
- 池 — 在单个实例和架构中为租户共享表。

| | 筒仓 | 使用单独的数据库进行桥接 | 使用单独的架构进行桥接 | 池 |
|----------------|--|--|---|---|
| 使用案例 | 隔离数据并完全控制资源使用是一项关键要求，否则您的租户非常庞大且对性能非常敏感。 | 数据隔离是一项关键要求，需要对租户的数据进行有限的交叉引用，或者不需要交叉引用。 | 租户数量适中，数据量适中。如果您必须交叉引用租户的数据，这是首选模型。 | 大量租户，每个租户的数据较少。 |
| 新租户入职灵活性 | 非常低。（每个租户都需要一个新的实例或集群。） | 中度地减少固定。（需要为每个租户创建一个新数据库来存储架构对象。） | 中度地减少固定。（需要为每个租户创建一个新架构来存储对象。） | 最快的选项。（需要最低限度的设置。） |
| 数据库连接池配置工作量和效率 | 需要付出大量努力。（每个租户一个连接池。） 效率低。（租户之间不共享数据库连接。） | 需要付出大量努力。（除非您使用 Amazon RDS 代理 ，否则每个租户只能配置一个连接池。） 效率低。（租户和连接总数之间不共享数据库连接。所有租户的使用量取决于数据库实例类。） | 所需的工作量更少。（适用于所有租户的一个连接池配置。） 效率适中。（仅在会话池模式下通过 SET ROLE 或 SET SCHEMA 命令重用连接。SET 命令在使用 Amazon RDS Proxy 时也会导致会话固定，但可以取消客户端连接池，并且可以为每个 | 只需最少的精力。 最高限度地减少固定（一个连接池供所有租户使用，并在所有租户之间高效地重复使用连接。数据库连接限制取决于数据库实例类。） |

| | 筒仓 | 使用单独的数据库进行桥接 | 使用单独的架构进行桥接 | 池 |
|--------------------------------------|----------------------------------|--|---|--|
| | | | 请求建立直接连接，以提高效率。) | |
| 数据库维护 (真空管理) 和资源使用 | 更限度地减少管理。 | 中等复杂性。(可能会导致大量资源消耗，因为之后 <code>vacuum_naptime</code> 必须为每个数据库启动真空工作器，这会导致自动真空启动器的 CPU 使用率很高。清理每个数据库的 PostgreSQL 系统目录表也可能带来额外的开销。) | 大型的 PostgreSQL 系统目录表。(总 <code>pg_catalog</code> 规模以数十 GB 为单位，具体取决于租户数量和亲属关系。可能需要修改与真空相关的参数才能控制工作台膨胀。) | 表可能很大，具体取决于租户数量和每个租户的数据。(可能需要修改与真空相关的参数才能控制工作台膨胀。) |
| 扩展管理工作 | 付出了巨大的努力(针对不同实例中的每个数据库)。 | 大量工作(在每个数据库级别)。 | 工作量最小(在通用数据库中一次)。 | 工作量最小(在通用数据库中一次)。 |
| 更改部署工作 | 重大限度地减少固定(Connect 每个单独的实例并发布更改。) | 重大限度地减少固定(Connect 每个数据库和架构，然后发布更改。) | 适度努力。(Connect 通用数据库并发布每个架构的更改。) | 最少的努力。(Connect 通用数据库并发布更改。) |
| 变更部署 — 影响范围 | 最低。(单个租户受到影响。) | 最低。(单个租户受到影响。) | 最低。(单个租户受到影响。) | 非常大。(所有租户都受到影响。) |

| | 筒仓 | 使用单独的数据 库进行桥接 | 使用单独的架构 进行桥接 | 池 |
|--|-------------------|--|--|---|
| 查询绩效管理 和工作 | 可管理的查询性能。 | 可管理的查询性能。 | 可管理的查询性能。 | 要保持查询性能，可能需要付出大量努力。（随着时间的推移，由于表的大小增加，查询的运行速度可能会变慢。您可以使用表分区和数据库分片来保持性能。） |
| 跨租户资源影响 | 没有影响。（租户之间不共享资源。） | 中度地减少影响。（租户共享常用资源，例如实例 CPU 和内存。） | 中度地减少影响。（租户共享常用资源，例如实例 CPU 和内存。） | 重度地减少固定。（租户在资源、锁定冲突等方面相互影响。） |
| 租户级别的调整 （例如，为每个租户创建其他索引或为特定租户调整数据库参数） | 可能的。 | 可能的原因。（可以对每个租户进行架构级别的更改，但数据库参数在所有租户中都是全局性的。） | 可能的原因。（可以对每个租户进行架构级别的更改，但数据库参数在所有租户中都是全局性的。） | 不可能的。（桌子由所有租户共享。） |

| | 筒仓 | 使用单独的数据库进行桥接 | 使用单独的架构进行桥接 | 池 |
|---------------------------|----------------------------------|--|--|--|
| 为对绩效敏感的租户进行再平衡工作 | 最低。(无需再平衡。扩展服务器和 I/O 资源以处理这种情况。) | 中。(使用逻辑复制或导出 pg_dump 出数据库,但停机时间可能会很长,具体取决于数据大小。您可以使用 Amazon RDS for PostgreSQL 中的可传输数据库功能在实例之间更快地复制数据库。) | 中等,但可能涉及长时间的停机时间。(使用逻辑复制或导出 pg_dump 出架构,但停机时间可能会很长,具体取决于数据大小。) | 意义重大,因为所有租户共享相同的桌子。(对数据库进行分片需要将所有内容复制到另一个实例,另外还需要一个步骤来清理租户数据。) 很可能需要更改应用程序逻辑。 |
| 主要版本升级导致数据库停机 | 标准停机时间。(取决于 PostgreSQL 系统目录的大小。) | 停机时间可能会更长。(根据系统目录的大小,时间会有所不同。PostgreSQL 系统目录表也在数据库之间复制) | 停机时间可能会更长。(根据 PostgreSQL 系统目录的大小,时间会有所不同。) | 标准停机时间。(取决于 PostgreSQL 系统目录的大小。) |
| 管理开销(例如,用于数据库日志分析或备份作业监控) | 重大努力 | 最少的努力。 | 最少的努力。 | 最少的努力。 |
| 租户级别的可用性 | 最低。(每个租户都失败并独立恢复。) | 影响范围更大。(如果出现硬件或资源问题,所有租户都会出现故障并一起恢复。) | 影响范围更大。(如果出现硬件或资源问题,所有租户都会出现故障并一起恢复。) | 影响范围更大。(如果出现硬件或资源问题,所有租户都会出现故障并一起恢复。) |

| | 筒仓 | 使用单独的数据库进行桥接 | 使用单独的架构进行桥接 | 池 |
|--------------------------------|--|--|--|-------------------------|
| 租户级别的备份和恢复工作 | 最限度地减少固定（每个租户可以单独备份和还原。） | 适度努力。（对每个租户使用逻辑导出和导入。需要进行一些编码和自动化。） | 适度努力。（对每个租户使用逻辑导出和导入。需要进行一些编码和自动化。） | 重大限度地减少固定（所有租户共享相同的桌子。） |
| 租户级别的 point-in-time 恢复工作 | 最少的努力。（使用快照使用点对点时间恢复，或者在 Amazon Aurora 中使用回溯功能。） | 适度努力。（使用快照恢复，然后使用导出/导入。但是，这将是一个缓慢的操作。） | 适度努力。（使用快照恢复，然后使用导出/导入。但是，这将是一个缓慢的操作。） | 工作量大，复杂程度高。 |
| 统一架构名称 | 每个租户具有相同的架构名称。 | 每个租户具有相同的架构名称。 | 每个租户具有不同的架构。 | 通用架构。 |
| 按租户自定义（例如，特定租户的附加表列） | 可能的。 | 可能的。 | 可能的。 | 复杂（因为所有租户共享相同的表）。 |
| 对象关系映射 (ORM) 层（例如 Ruby）的目录管理效率 | 高效（因为客户端连接是特定于租户的）。 | 高效（因为客户端连接特定于数据库）。 | 效率适中。（根据所使用的 ORM、用户/角色安全模型和 search_path 配置，客户端有时会缓存所有租户的元数据，从而导致数据库连接内存使用率过高。） | 高效（因为所有租户共享相同的桌子）。 |

| | 筒仓 | 使用单独的数据 库进行桥接 | 使用单独的架构 进行桥接 | 池 |
|----------------------------------|---|---|---|--|
| 合并租户报告工 作 | 重大限度地减少 固定 (您必须使 用外部数据包装 器 [FDW] 来整合 所有租户中的数 据, 或者提取、 转换 [ETL] 并将 其加载到另一个 报告数据库。) | 重大限度地减少 固定 (您必须 使用 FDW 将所 有租户或 ETL 中的数据合并到 另一个报告数据 库。) | 适度努力。 (您 可以使用联合来 聚合所有架构中 的数据。) | 最少的努力。 (所有租户数 据都在同一个表 中, 因此报告很 简单。) |
| 租户特定的只 读报告实例 (例 如, 基于订阅) | 最大限度减少固 定 (创建只读副 本。) | 适度努力。 (您 可以使用逻辑 复制或DatAWS abase Migration Service [AWS DMS] 进行配 置。) | 适度努力。 (您 可以使用逻辑复 制或AWS DMS 进行配置。) | 复杂 (因为所有 租户共享相同的 表)。 |
| 数据隔离 | 最低。 | 较好。 (您可以 使用 PostgreSQ L 角色管理数据 库级权限。) | 较好。 (您可以 使用 PostgreSQ L 角色管理架构 级权限。) | 更糟糕了。 (由 于所有租户共享 相同的表, 因此 您必须实现诸如 行级安全 [RLS] 之类的功能来隔 离租户。) |
| 租户专用的存储 加密密钥 | 可能的。 (每 个 PostgreSQ L 集群可以有自 己的AWS Key Management Service [AWS KMS] 密钥用于 存储加密。) | 不可能的。 (所 有租户共享相同 的 KMS 密钥用 于存储加密。) | 不可能的。 (所 有租户共享相同 的 KMS 密钥用 于存储加密。) | 不可能的。 (所 有租户共享相同 的 KMS 密钥用 于存储加密。) |

| | 筒仓 | 使用单独的数据库进行桥接 | 使用单独的架构进行桥接 | 池 |
|---|--|--|---|--|
| 使用AWS Identity and Access Management (IAM) 对每个租户进行数据库身份验证 | 可能的。 | 可能的。 | 可能 (通过为每个架构设置单独的 PostgreSQL 用户)。 | 不可能 (因为表由所有租户共享)。 |
| 基础设施成本 | 最高 (因为没有共享任何东西)。 | 中。 | 中。 | 最低。 |
| 数据复制和存储使用情况 | 所有租户中总数最高。(PostgreSQL 系统目录表以及应用程序的静态和常用数据在所有租户之间复制。) | 所有租户中总数最高。(PostgreSQL 系统目录表以及应用程序的静态和常用数据在所有租户之间复制。) | 中。(应用程序的静态和通用数据可以采用通用架构, 由其他租户访问。) | 最低。(不重复数据。应用程序的静态数据和通用数据可以位于相同的架构中。) |
| 以租户为中心的监控 (快速找出是哪个租户造成了问题) | 最限度地减少固定 (由于每个租户都受到单独监控, 因此可以轻松检查特定租户的活动。) | 适度努力。(由于所有租户共享相同的物理资源, 因此您必须应用额外的筛选来检查特定租户的活动。) | 适度努力。(由于所有租户共享相同的物理资源, 因此您必须应用额外的筛选来检查特定租户的活动。) | 重大限度地减少固定 (由于所有租户共享包括表在内的所有资源, 因此您必须使用绑定变量捕获来检查特定 SQL 查询属于哪个租户。) |
| 集中管理和运行状况/活动监控 | 做出重大努力 (建立中央监测和中央指挥中心)。 | 工作量适中 (因为所有租户共享同一个实例)。 | 工作量适中 (因为所有租户共享同一个实例)。 | 工作量最小 (因为所有租户共享相同的资源, 包括架构)。 |

| | 筒仓 | 使用单独的数据 库进行桥接 | 使用单独的架构 进行桥接 | 池 |
|---|-----|---|--|---|
| 对象标识符 (OID) 和交易 ID (XID) 交叉的可能 性 | 最低。 | 高。(因为 OID, XID 是单 个 PostgreSQL 集群范围的计数 器, 有效清理物 理数据库时可能 会出现问题)。 | 中。(因为 OID, XID 是单 个 PostgreSQL 集群范围的计数 器)。 | 高。(例如, 单 个表可以达到 40 亿的 TOAST OID 限制, 具体取决 于 out-of-line 列 数。) |

行级别安全建议

需要行级安全 (RLS) 才能在使用 PostgreSQL 的池化模型中维护租户数据隔离。RLS 在数据库级别集中执行隔离策略，并消除了软件开发人员维护这种隔离的负担。实现 RLS 的最常用方法是在 PostgreSQL 数据库管理系统中启用此功能。RLS 涉及根据指定列中的值筛选对数据行的访问权限。您可以使用两种方法来筛选对数据的访问权限：

- 将表中的指定数据列与当前 PostgreSQL 用户的值进行比较。该用户可以访问该列中等于已登录的 PostgreSQL 用户的值。
- 将表中的指定数据列与应用程序设置的运行时变量的值进行比较。在该会话期间，可以访问该列中等于运行时变量的值。

第二个选项是首选，因为第一个选项需要为每个租户创建一个新的 PostgreSQL 用户。相反，使用 PostgreSQL 的 SaaS 应用程序应负责在运行时在查询 PostgreSQL 时设置租户特定的上下文。这将产生强制执行 RLS 的效果。您也可以 table-by-table 逐步启用 RLS。作为最佳实践，您应该在所有包含租户数据的表上启用。

以下示例将创建两个表并启用。此示例将一系列数据与运行时变量的值进行比较 `app.current_tenant`。

```
-- Create a table for our tenants with indexes on the primary key and the tenant's name
CREATE TABLE tenant (
    tenant_id UUID DEFAULT uuid_generate_v4() PRIMARY KEY,
    name VARCHAR(255) UNIQUE,
    status VARCHAR(64) CHECK (status IN ('active', 'suspended', 'disabled')),
    tier VARCHAR(64) CHECK (tier IN ('gold', 'silver', 'bronze'))
);

-- Create a table for users of a tenant
CREATE TABLE tenant_user (
    user_id UUID DEFAULT uuid_generate_v4() PRIMARY KEY,
    tenant_id UUID NOT NULL REFERENCES tenant (tenant_id) ON DELETE RESTRICT,
    email VARCHAR(255) NOT NULL UNIQUE,
    given_name VARCHAR(255) NOT NULL CHECK (given_name <> ''),
    family_name VARCHAR(255) NOT NULL CHECK (family_name <> '')
);

-- Turn on RLS
ALTER TABLE tenant ENABLE ROW LEVEL SECURITY;
```

```
-- Restrict read and write actions so tenants can only see their rows
-- Cast the UUID value in tenant_id to match the type current_setting
-- This policy implies a WITH CHECK that matches the USING clause
CREATE POLICY tenant_isolation_policy ON tenant
USING (tenant_id = current_setting('app.current_tenant')::UUID);

-- And do the same for the tenant users
ALTER TABLE tenant_user ENABLE ROW LEVEL SECURITY;

CREATE POLICY tenant_user_isolation_policy ON tenant_user
USING (tenant_id = current_setting('app.current_tenant')::UUID);
```

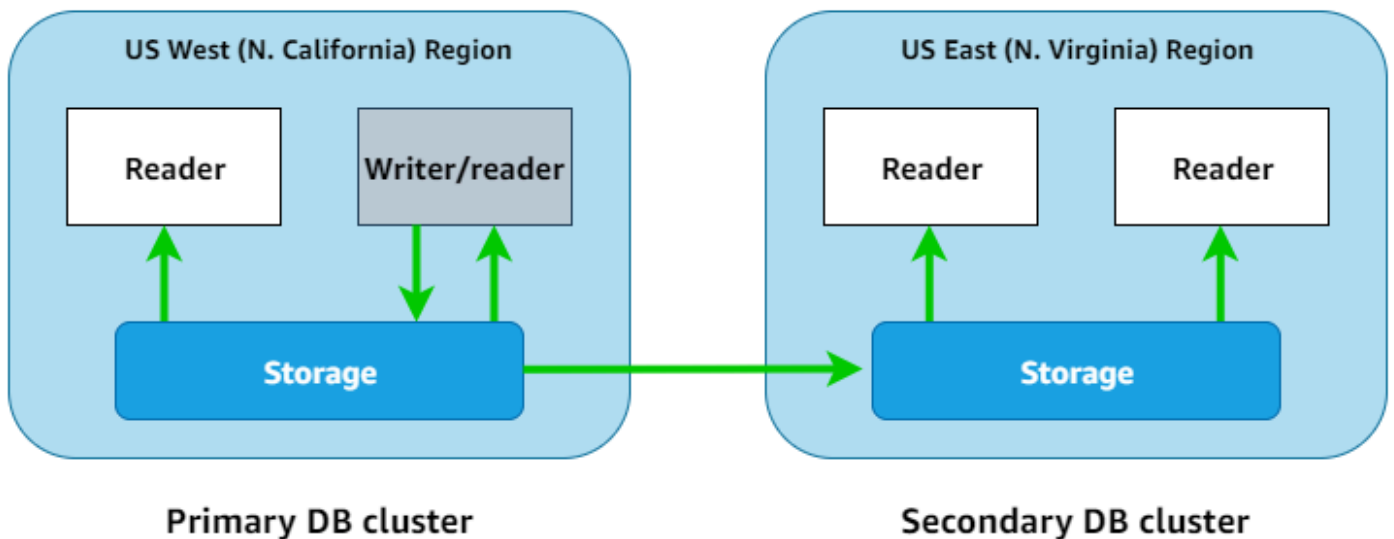
有关更多信息，请参阅博客文章：[使用 PostgreSQL 行级别安全性隔离](#)。SAWS aaS Factory 团队还有一些[示例 GitHub](#)可以帮助实施 RLS。

PostgreSQL 可用于池模型

池模型本质上只有一个 PostgreSQL 实例。因此，设计应用程序以实现高可用性至关重要。池化数据库的故障或中断会导致您的应用程序降级或所有租户都无法访问。

通过启用高可用性功能，可以使适用于 PostgreSQL 的 Amazon RDS 数据库实例在两个可用区之间实现冗余。有关更多信息，请参阅 Amazon RDS 文档中的 [Amazon RDS 高可用性 \(多可用区\)](#)。对于跨区域故障转移，您可以在其他 AWS 区域创建只读副本。（此只读副本必须作为故障转移过程的一部分进行升级。）此外，您可以复制跨 AWS 区域复制的备份以进行恢复。有关更多信息，请参阅 Amazon RDS 文档中的 [将自动备份复制到其他 AWS 区域](#)。

兼容 Aurora PostgreSQL 的 Aurora 会自动备份数据，其方式可以承受多个可用区域的故障。（请参阅 Aurora 文档中的 [亚马逊 Aurora 的高可用性](#)。）为了提高 Aurora 的弹性并更快地恢复，您可以在其他可用区创建 Aurora 只读副本。您可以使用 Aurora 全球数据库将数据复制到另外五个 AWS 区域，以实现跨区域恢复和自动故障转移。（请参阅 [Aurora 文档中的使用亚马逊 Aurora 全球数据库](#)。）此外，您可以使用 Aurora 全局数据库启用 [写入转发](#)，以实现跨多个数据库的高可用性 AWS 区域。



无论您使用的是适用于 PostgreSQL 的 Amazon RDS 还是与 Aurora PostgreSQL 兼容，我们都建议您实施高可用性功能，以减轻任何中断对使用池模型的所有多租户 SaaS 应用程序的影响。

最佳实践

本节列出了本指南中的一些高级要点。有关每个要点的详细讨论，请点击相应章节的链接。

比较托管 PostgreSQL 的AWS选项

AWS提供了在托管环境中运行 PostgreSQL 的两种主要方法。（在这种情况下，托管意味着AWS服务部分或完全支持 PostgreSQL 基础设施和 DBMS。）开启的托管 PostgreSQL 选项AWS具有自动备份、故障转移、优化和对 PostgreSQL 进行部分管理的好处。作为托管选项，AWS提供适用于 PostgreSQL 的 Amazon Aurora Service (Amazon RDS) 和 Amazon Aurora (Amazon RD 您可以通过分析 PostgreSQL 用例从这两个模型中选择最佳选择。有关更多信息，请参阅本指南中有关在 [Amazon RDS 和 Aurora 之间进行选择](#) 部分。

选择多租户 SaaS 分区模型

您可以从三种适用于 PostgreSQL 的 SaaS 分区模型中进行选择：孤岛、桥接和池。每种模型都有优点和缺点，您应该根据自己的用例选择最优的模型。适用于 PostgreSQL 和兼容 Aurora PostgreSQL 的亚马逊 RDS 支持所有三种型号 选择模型对于维护 SaaS 应用程序中的租户数据隔离至关重要。有关这些模型的详细讨论，请参阅本指南中 [PostgreSQL 的多租户 SaaS 分区模型](#) 部分。

对池 SaaS 分区模型使用行级安全性

在使用 PostgreSQL 的池模型中，需要行级安全 (RLS) 来维护租户数据的隔离。这是因为在池模型中，基础架构、PostgreSQL 数据库或架构之间没有基于每个租户的逻辑分离。RLS 在数据库级别集中执行隔离策略，并消除了软件开发人员维护这种隔离的负担。您可以使用 RLS 将数据库操作限制为特定租户。有关更多信息和示例，请参阅本指南中的 [行级安全建议](#) 部分。

常见问题

本节提供了有关在多租户 SaaS 应用程序中实现托管 PostgreSQL 的常见问题的答案。

确实AWS提供了哪些托管 PostgreSQL 选项？

AWS提供[兼容 Amazon Aurora PostgreSQL 和适用于 PostgreSQL 的 Amazon Relational Database Service \(亚马逊 RDS\)](#)。AWS还提供[广泛的托管数据库产品目录](#)。

哪种服务最适合 SaaS 应用程序？

您可以同时使用兼容 Aurora PostgreSQL 和适用 Amazon RDS for PostgreSQL S 用于 SaaS 应用程序以及本指南中讨论的所有 SaaS 分区模型。这两种服务在可扩展性、崩溃恢复、故障转移、存储选项、高可用性、灾难恢复、备份以及每个选项可用的实例类方面存在差异。最佳选择将取决于您的特定的使用案例。使用本指南中的[决策矩阵](#)为您的用例选择最佳选项。

如果我决定将 PostgreSQL 数据库与多租户 SaaS 应用程序一起使用，我应该考虑哪些独特的要求？

与任何用于 SaaS 应用程序的数据存储一样，最重要的考虑因素是维护租户数据隔离的方法。如本指南所述，您可以通过多种方法通过AWS托管 PostgreSQL 产品实现租户数据隔离。此外，对于任何 PostgreSQL 实现，您都应考虑按租户进行性能隔离。

我可以哪些模型通过 PostgreSQL 维护租户数据隔离？

您可以使用孤岛、桥接和池模型作为 SaaS 分区策略，以保持租户数据隔离。有关这些模型以及如何将其应用于 PostgreSQL 的讨论，请参阅本指南中 [PostgreSQL 的多租户 SaaS 分区模型](#) 部分。

如何使用由多个租户共享的单个 PostgreSQL 数据库维护租户数据隔离？

PostgreSQL 支持行级安全 (RLS) 功能，您可以使用该功能在单个 PostgreSQL 数据库或实例中强制执行租户数据隔离。此外，您可以在单个实例中为每个租户预置单独的 PostgreSQL 数据库，或者按租户创建架构来实现此目标。有关这些方法的优缺点，请参阅本指南中的[行级安全建议](#)部分。

后续步骤

AWS 为操作托管 PostgreSQL 提供了两种选项：兼容 Aurora PostgreSQL 和 Amazon RDS for PostgreSQL RDS。我们建议您评估这两种服务，然后选择最能支持您的多租户 SaaS 应用程序特定用例的选项。符合 SaaS 分区模型可以确保使用 PostgreSQL 的 SaaS 应用程序严格遵守最佳实践来维护租赁。SaaS 孤岛、桥接和池分区模型支持许多 SaaS 用例。这些模型在性能隔离、运营开销和租户安全等因素之间具有不同的优势。

后续步骤：

- [评估兼容 Aurora PostgreSQL 和适用 Amazon RDS for PostgreSQL](#)，然后为你的 SaaS 应用程序选择最佳选项。
- [选择符合应用程序要求的 SaaS 分区模型](#)：孤岛、桥接或池。
- 根据你选择的 SaaS 分区模型实现 PostgreSQL。

资源

参考信息

- [SaaS 存储策略：在AWS \(AWS白皮书 \) 上构建多租户存储模型](#)
- [使用适用于亚马逊 Aurora PostgreSQL 的亚马逊 Aurora 全球数据库进行跨区域灾难恢复 \(博客文章 AWS \)](#)
- 通过 [PostgreSQL 行级安全实现多租户数据隔离 \(博客文章 \)](#) AWS
- [使用 Amazon Aurora PostgreSQL \(Aurora 文档 \)](#)
- 亚马逊 RDS [上的 PostgreSQL \(亚马逊 RDS 文档 \)](#)

合作伙伴

- [适用于 PostgreSQL 合作伙伴的 Amazon Aurora](#)
- [适用于 PostgreSQL 的亚马逊 RDS 合作伙伴](#)

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 | 更新以反映 Aurora 中写入转发的可用性。 | 2024 年 4 月 29 日 |
| 更新 | 更新了 Amazon RDS 和 Aurora 比较表 。 | 2022 年 10 月 21 日 |
| 二 | 初次发布 | 2021 年 9 月 30 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：将本地 Oracle 数据库迁移到 Amazon Aurora PostgreSQL 兼容版。
- **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：将您的本地 Oracle 数据库迁移到 AWS 云端适用于 Oracle 的亚马逊关系数据库服务 (Amazon RDS)。
- **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：将您的本地 Oracle 数据库迁移到 AWS 云中 EC2 实例上的 Oracle。
- **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。此迁移场景特定于 VMware Cloud on AWS，它支持虚拟机 (VM) 兼容性和本地环境之间的工作负载可移植性。AWS 在将基础设施迁移到 VMware Cloud on AWS 时，您可以在本地数据中心使用 VMware Cloud Foundation 技术。示例：将托管 Oracle 数据库的虚拟机管理程序重新部署到 VMware Cloud 上。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 (IAM) 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。当您确信时，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 云企业战略博客上的 [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 AWS 云端时通常要经历的四个阶段：

- 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“云企业战略”博客文章 [《云优先之旅和采用阶段》](#) 中定义了 AWS 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-Architected 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。此帐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 Framework 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。

|

laC

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。

IloT

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 (IloT)

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 \(IloT \) 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 AWS 云端的业务案例。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 AWS 云端的方法。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。有关详细信息，请参阅在[AWS 云中评估应用程序的现代化准备情况](#)。

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 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)

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。这些控件会在资源配置之前对其进行扫描。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 [Secrets Manager 文档中的密钥](#)。

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

由接收数据的人在目的地对数据 AWS 服务 进行加密。

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 ynthetic](#) 来创建这些测试。

T

tags

键值对，用作组织资源的元数据。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。