ISVs 运行 Amazon Neptune 数据库的多租户指南

# AWS 规范性指导

# AWS 规范性指导: ISVs 运行 Amazon Neptune 数据库的多租户指南

# Table of Contents

# ISVs 运行 Amazon Neptune 数据库的多租户指南

Amazon Web Services（贡献者）

2024 年 8 月（文档历史记录）

多租户是一种计算机系统架构，其中一个应用程序的单个实例为多个客户提供服务。每个客户都被称为租户。在多租户架构中，应用程序的这些实例在共享环境中运行，在这种环境中，每个租户实际位于同一个基础架构上，但在逻辑上是分开的。

作为独立软件供应商 (ISV)，您可以使用 Amazon Neptune 为需要在高度互联的数据之间导航的应用程序提供支持。您可能正在账户中管理基于云的软件即服务 (SaaS) 应用程序，并为租户提供订阅。然后，租户可以通过互联网或私下访问该服务 AWS PrivateLink。这种模式的经济性对双方都有效，因为租户可以获得比他们购买、构建和维护成本更低的软件。作为 ISV，您可以为订阅收取比创建和维护软件的费用更高的费用。问题在于如何将业务扩展到多个租户。

多租户 ISVs 具有重要的经济和运营优势。多租户架构可为您的组织提供更好的投资回报率 (ROI)。多租户还简化了运营要求，因此您的组织可以更快地采取行动，并降低向租户交付软件的成本。

本文档为如何使用 Amazon Neptune 有效运行多租户 ISV 应用程序提供了指导。本指南基于多年来在支持 ISVs “成功向客户交付 SaaS 解决方案” 方面获得的最佳实践。在组织目标和架构原则的背景下评估本指南将有助于您找到优化解决方案的方法。

> **Note**
>
> 本文档并未提供最佳实践的详尽列表。它为多租户 ISV 工作负载提供了额外的具体AWS 指导，对《为亚马逊海王星应用 Well-Architected 框架》文档进行了补充。我们建议您在设计解决方案时仔细阅读这两个文档中的注意事项。

# SaaS 数据分区模型

SaaS 开发人员面临的挑战之一是设计架构模式，用于在多租户环境中表示和组织数据。这些多租户存储机制和模式通常被称为数据分区。

在多租户 SaaS 环境中，区分数据分区和租户隔离非常重要。这些概念虽然相关，但不是同义词。数据分区是指为每个租户存储数据的方法。但是，仅仅分区并不能保证租户隔离。必须采取其他措施来确保一个租户的数据无法被另一个租户访问。

多租户 SaaS 系统中常用的三种数据分区模型包含三种：孤岛、池和混合。您对任何模型的选择取决于以下因素：

- 合规
- 吵闹的邻居
- 分层策略
- 业务需求
- 租户隔离需求

此外，上可用的每种数据库类型 AWS 通常都提供一组独特的数据分区和租户隔离模型。在查看如何组织租户图以支持解决方案的各种需求时，请考虑 Amazon Neptune 提供的模型。

许多人ISVs从以下断言之一开始在 Neptune 上进行设计：

- 该ISV解决方案要求在不同的集群中对客户进行物理隔离。
- 该ISV解决方案需要诸如命名数据库或传统关系数据库管理系统中的架构之类的结构。

经过考虑，ISVs意识到这些断言是不正确的，因为在几乎所有工作负载下，他们的每个客户的数据库中都有一个断开连接的图表。实施本文档中讨论的数据建模和访问指南可以防止跨越这些数据界限，并维护客户数据隐私。

本指南描述了筒仓模型和池模型，但大多数人ISVs选择池模型是出于成本和运营效率考虑。该指南简要讨论了一种混合模型，该模型结合了筒仓和池模型的各个方面。有些人为其最大的客户ISVs使用混合模型，以满足图表大小的监管或合规性要求。

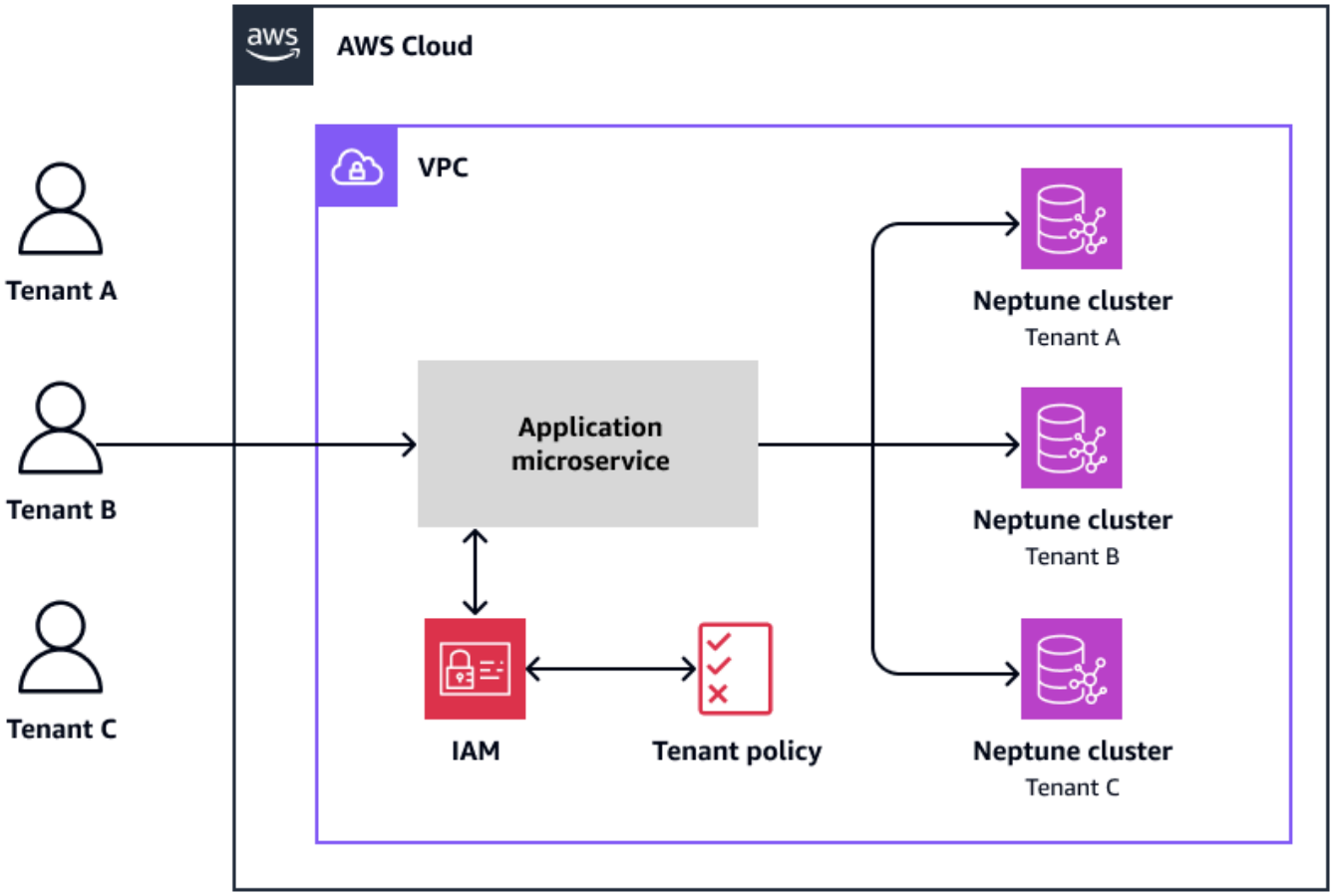
# 孤岛模型多租户

由于合规性和监管要求，某些多租户 SaaS 环境可能需要将租户的数据部署在完全分离的资源上。在某些情况下，大型客户需要专用集群来减少噪音对邻居的影响。在这种情况下，您可以应用筒仓模型。

在孤岛模型中，租户数据的存储与任何其他租户数据完全隔离。所有用于表示租户数据的结构都被视为该客户端在物理上是唯一的，这意味着每个租户通常具有不同的存储、监控和管理。每个租户还将有一个单独的 AWS Key Management Service (AWS KMS) 密钥用于加密。在 Amazon Neptune 中，孤岛是指每个租户一个集群。

## 每个租户的集群

您可以通过为每个集群设置一个租户来使用 Neptune 实现孤岛模型。下图显示了三个租户在虚拟私有云 (VPC) 中访问应用程序微服务，每个租户都有单独的集群。




每个集群都有其单独的终端节点，以帮助确保不同的接入点，从而实现高效的数据交互和管理。通过将每个租户置于自己的集群中，可以在租户之间创建明确定义的边界，确保客户的数据与其他租户的数据

成功隔离。这种隔离对具有严格监管和安全限制的SaaS解决方案也很有吸引力。此外，当每个租户都有自己的集群时，您不必担心邻居会有噪音，因为一个租户施加的负载可能会对其他租户的体验产生不利影响。

虽然 cluster-per-tenant孤岛模式具有优势，但它也带来了管理和敏捷性方面的挑战。这种模型的分布性质使得汇总和评估所有租户的租户活动和运营状况变得更加困难。部署也变得更具挑战性，因为设置新租户现在需要配置单独的集群。当客户端升级和版本与数据库升级紧密结合时，在具有共享客户端层的环境中，升级变得更具挑战性。

Neptune 支持无服务器集群和预配置集群。评估无服务器实例还是预配置实例可以更好地处理您的应用程序工作负载。通常，如果您的工作负载的需求水平保持不变，则预配置实例将更具成本效益。Serverless 针对要求苛刻、高度可变的工作负载进行了优化，这些工作负载在短时间内大量使用数据库，然后是长时间的轻度活动或没有活动。

在按租户使用 Neptune 预配置的集群时，必须选择与租户需求的最大负载近似的实例大小。这种对服务器的依赖也会对 SaaS 环境的扩展效率和成本产生连带影响。虽然 SaaS 的目标是根据实际租户负载动态调整规模，但 Neptune 配置的集群要求您过度配置，以应对更长的使用时间和负载峰值。过度配置会增加每个租户的成本。此外，由于租户使用量会随着时间的推移而发生变化，因此必须为每个租户分别应用向上或向下扩展集群。

Neptune 团队通常建议不要采用孤岛模式，因为闲置资源会产生更高的成本和额外的运营复杂性。但是，对于需要这种额外隔离的高度监管或敏感工作负载，客户可能愿意支付额外费用。

## 筒仓模型的实施指南

要实现孤 cluster-per-tenant岛隔离模型，请创建 AWS Identity and Access Management (IAM) 数据访问策略。这些策略通过确保租户只能访问包含自己数据的 Neptune 集群来控制对租户的 Neptune 集群的访问。将每个租户的 IAM 策略附加到一个 IAM 角色。然后，应用程序微服务使用 IAM 角色使用 () 的`AssumeRole`方法生成精细的AWS Security Token Service 临时证书。AWS STS这些凭据只能访问该租户的 Neptune 集群，用于连接到租户的 Neptune 集群。

以下代码段显示了基于数据的 IAM 策略示例：

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "neptune-db:ReadDataViaQuery",
```

```
        "neptune-db:WriteDataViaQuery"
      ],
      "Resource": "arn:aws:neptune-db:us-east-1:123456789012:tenant-1-cluster/*",
      "Condition": {
        "ArnEquals": {
          "aws:PrincipalArn": "arn:aws:iam::123456789012:role/tenant-role-1"
        }
      }
    }
  ]
}
```

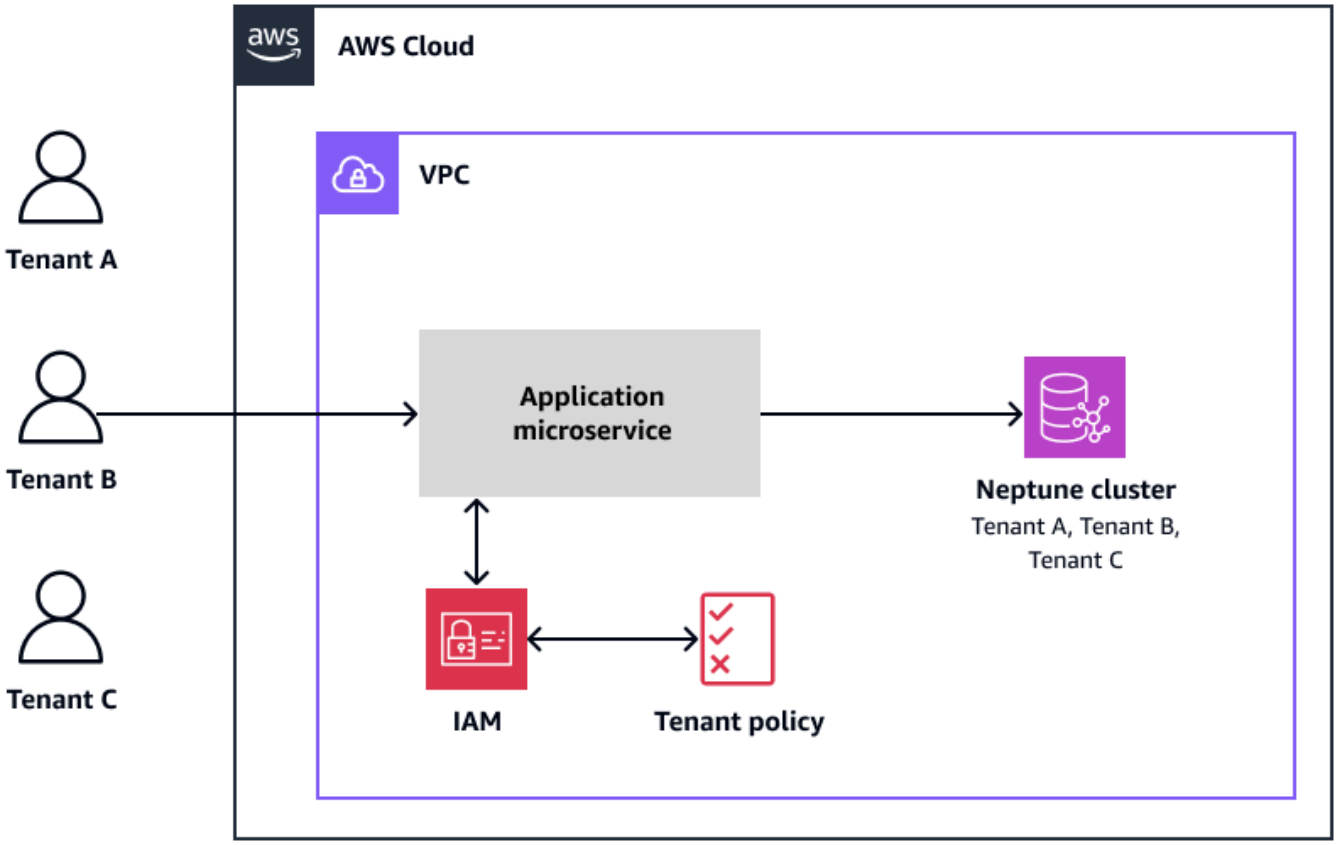
该代码为示例租户提供了对各自的 Neptune 集群的读写查询权限。tenant-1该Condition元素确保只有担任 tentant-1 IAM 角色 () 的调用实体（委托人tenant-role-1）才能访问的 Nep tenant-1 tune 集群。

# 池模型多租户

有时，由于成本或运营开销，实施孤岛模型既没有必要也不可行：

- 您可能没有足够的资源来维护每个租户的单个集群。
- 可能没有必要对每个租户的数据进行物理分离，逻辑分离足以满足他们的需求和合规性要求。

下图显示了池模型，租户数据放在单个 Amazon Neptune 集群中，所有租户共享一个公共数据库。




这种池隔离模型减少了管理开销，并且由于需要管理的集群较少，因此可以提高运营效率。此外，计算资源可以在多个客户之间共享，而不是在客户不活动期间保持闲置状态。

使用池模型时，有两种方法可以对数据进行建模。您的方法取决于您是在构建带标签的属性图 (LPG) 还是使用资源描述框架 (RDF) 创建图表。

## 标注属性图的池模型

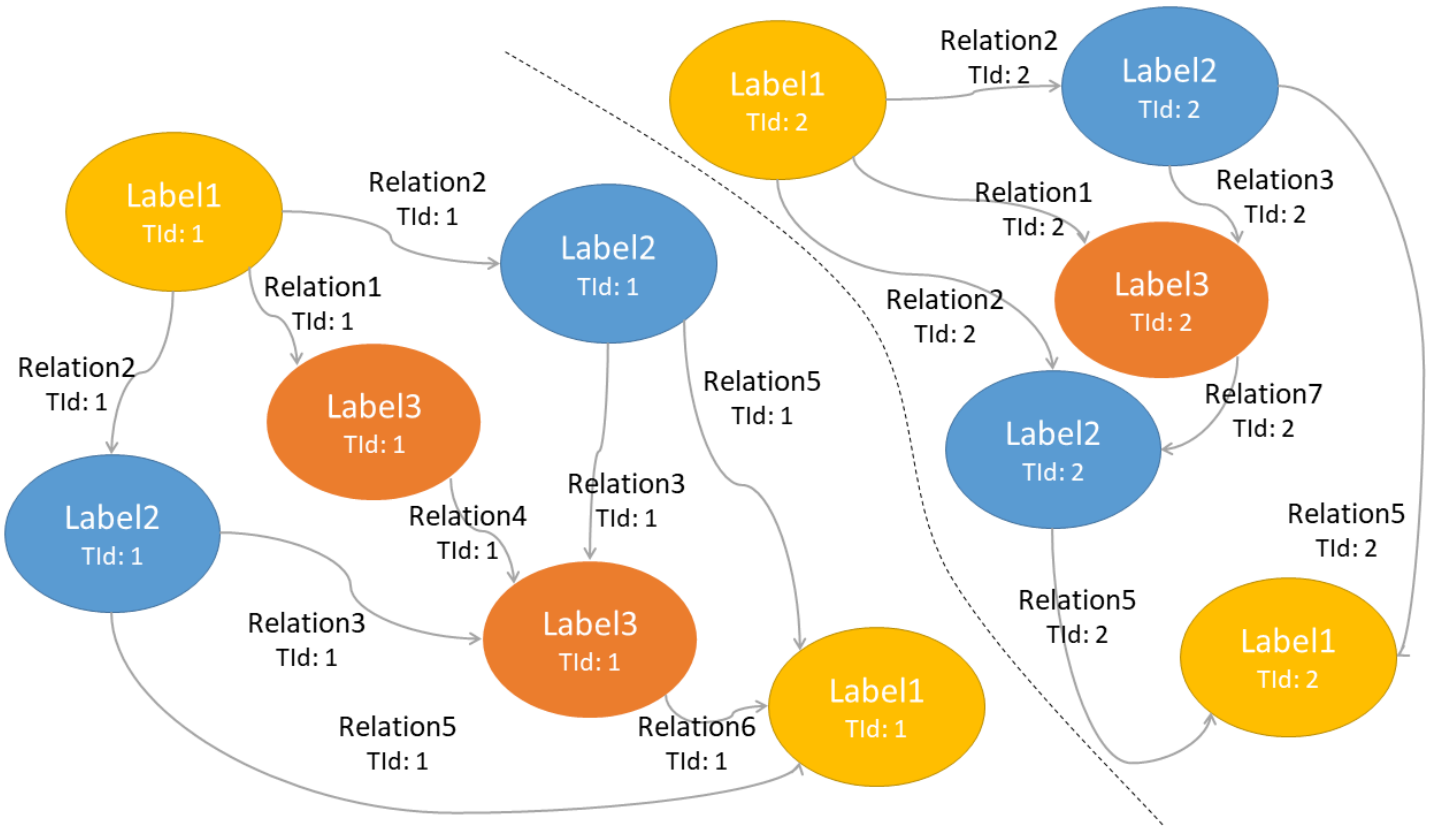
在 Amazon Neptune LPGs 上使用池模型有三种不同的方法：

- 属性策略 – 当您需要优先使用已建立的库结构（例如 Apache TinkerPop Gremlin 语言的）而不是性能时，请选择属性策略。PartitionStrategy
- Prefix-Label 策略 – 我们根据性能和限制噪音邻域效应推荐大多数场景使用前缀标签策略。
- 多标签策略 – 多标签策略的性能与前缀标签策略相比有所提高。它还支持运行跨集群上所有租户的查询（例如，用于报告或监控所有租户的 ISV 查询）。

## 房地产策略

使用 LPGs，用户可以向节点、顶点和边添加键值对属性。为了实现逻辑分离，大多数客户直观地将其建模为每个节点和边缘的唯一属性，并使用公共租户属性密钥。租户属性密钥表示拥有该节点的所有租户。租户标识符是用于标识单个租户的唯一值。

下图显示了此模型。两个断开连接的子图具有不同的标记节点和边，租户属性键用TId表示。一个子图中的每个节点和边的TId1值为。在另一个子图中，每个节点和边的TId2值都为。




在带标签的属性图中，有两种方法可以对此进行管理。Gremlin 查询语言提供了PartitionStrategy遍历库来帮助管理数据的数据分区。以下示例中的代码期望每个节点和边都有一个名为的属性TId：

```
strategy1 = new PartitionStrategy(partitionKey: "TId", writePartition: "1",
 readPartitions: ["1"])
strategy2 = new PartitionStrategy(partitionKey: "TId", writePartition: "2",
 readPartitions: ["2"])
```

写入新节点或边时，根据是否选择了"1"或"2"，将该属性"TId"添加strategy2为strategy1或的值。对于带有 of "TId" 的客户"1"，您可以使用strategy1。以下示例显示了为该客户写入数据：

```
g.withStrategies(strategy1).addV("Label1").property("Value", "123456").property(id,
 "Item_1")
```

对于读取查询，分别使用"TId == '1'"strategy1或将过滤"TId == '2'"器添加到每个节点或边缘遍历strategy2中。这些分区策略可以简化您的代码，但它们不是必需的。使用该策略的好处是，它可以在授权级别注入，然后传递给构成查询的较低级别的代码。这将确定客户标识符 (TId) 的代码与查询逻辑分开。

以下示例代码显示了用于读取数据的 Gremlin 查询：

```
g.withStrategies(strategy1).V().hasLabel("Label1")
```

前面的代码等同于以下示例：

```
g.V().hasLabel("Label1").has("TId", "1")
```

同样，在使用 Gremlin 写入数据时，可以使用以下查询：

```
g.withStrategies(strategy1).addV("Label1").property("Value").property(id, "Item_1")
```

前面的代码等同于以下示例，后者不使用分区策略，因此需要显式写入"TId"属性：

```
g.addV("Label1").property("TId", "1").property("Value").property(id, "Item_1")
```

在 OpenCypher 中，这些库并不存在。您负责编写和修改查询，以将租户标识符作为属性添加到节点和边缘。例如：

```
CREATE (n:Item {`~id`: 'Item_1', Value: '123456', TId: '1'})
CREATE (n:Item {`~id`: 'Item_2', Value: '123456', TId: '2'})
```

请注意没有分区策略的 Gremlin 代码之间的相似之处。然后，您可以使用以下代码读取从第一条CREATE语句中写入的节点：

```
MATCH (n:Item {TId: '1'})
RETURN n
--or
MATCH (n:Item)
WHERE n.TId == '1'
RETURN n
```

当你想使用原生 TinkerPop Gremlin 构造时，你可以选择属性策略，例如。 PartitionStrategy但是，与前缀标签策略相比，该模型在Amazon Neptune上存在性能缺陷。有关这些性能缺点的讨论，请参阅液化石油气模型的性能影响部分。

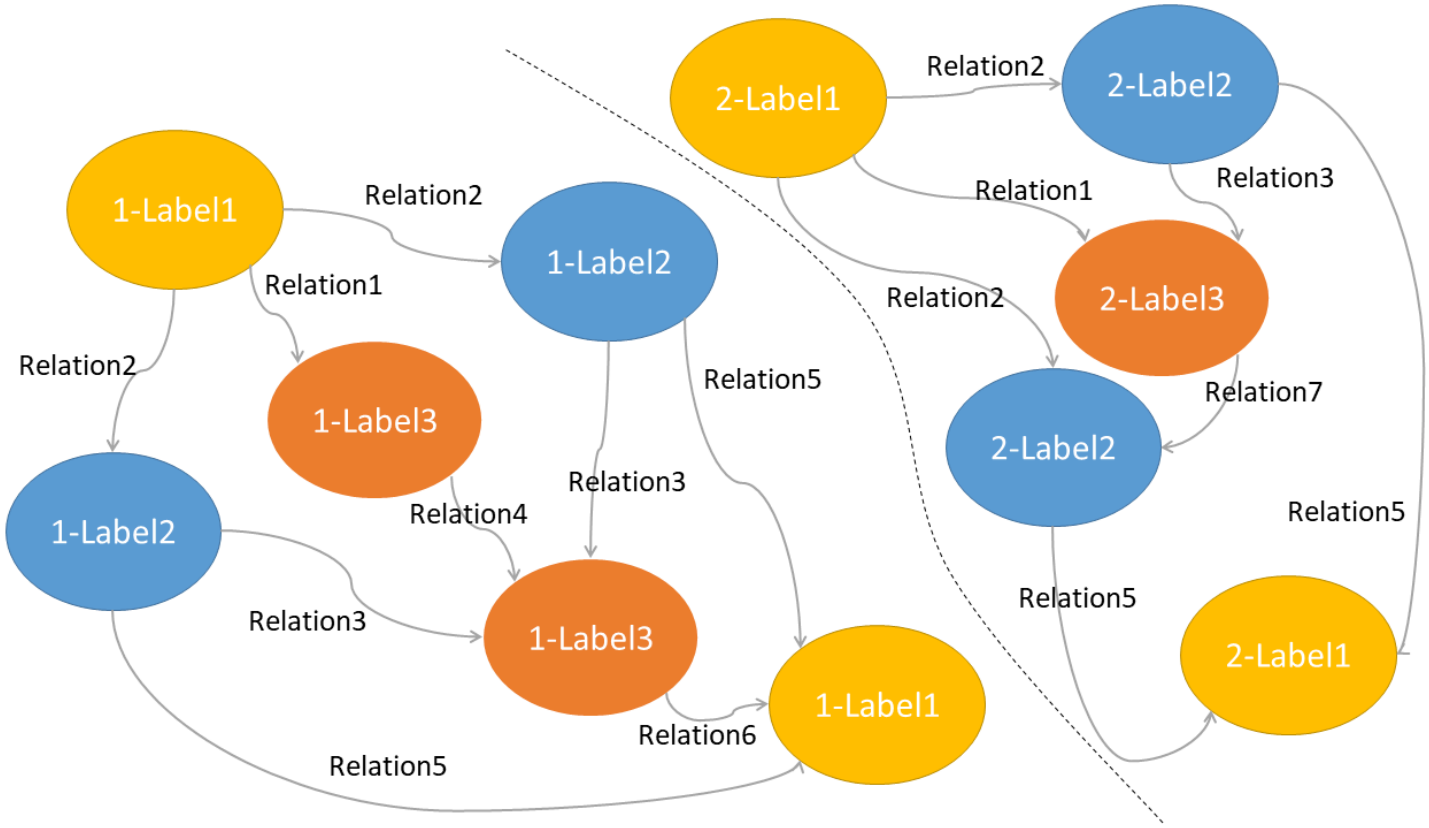
如果满足以下条件，请考虑仅在节点上对属性策略进行建模，而不在边上建模：

- 您的图表的边缘比标签多得多。
- 每个租户都是一个断开连接的图表。
- 您只能通过使用节点作为起点而不是标签来访问图表。

## 前缀标签策略

如果绩效是头等大事，我们强烈建议考虑使用前缀标签策略，而不是房地产策略。

在前缀标签策略中，您可以使用租户标识符和节点标签的组合来标记每个节点。例如，如果租户的标识符为，"1"而节点标签为"Label1"，则将节点标签指定为"1-Label1"。下图显示了使用此模型的两个断开连接的子图。




在 Gremlin 中写入数据时，可以在任何节点的标签上添加标识号：

```
g.addV("1-Label1")
g.addV("2-Label6")
```

查询此图表时，您可以检查节点上是否存在此前缀：

```
g.V().hasLabel("1-Label1")
```

在 OpenCypher 中，你可以使用以下语句写入数据：CREATE

```
CREATE (n:`1-Label1` {`~id`: 'Item_1', Value: 'XYZ123456'})
```

要查询你在 OpenCypher 中写入的数据，请使用以下代码：

```
MATCH n= (:`1-Label1`)
RETURN n
```

前缀标签策略假设所有节点都分配给一个或多个租户，并且不在边缘范围内分配权限。避免在边缘标签上使用此策略，因为这会导致大量谓词，并会对 Neptune 性能产生负面影响。

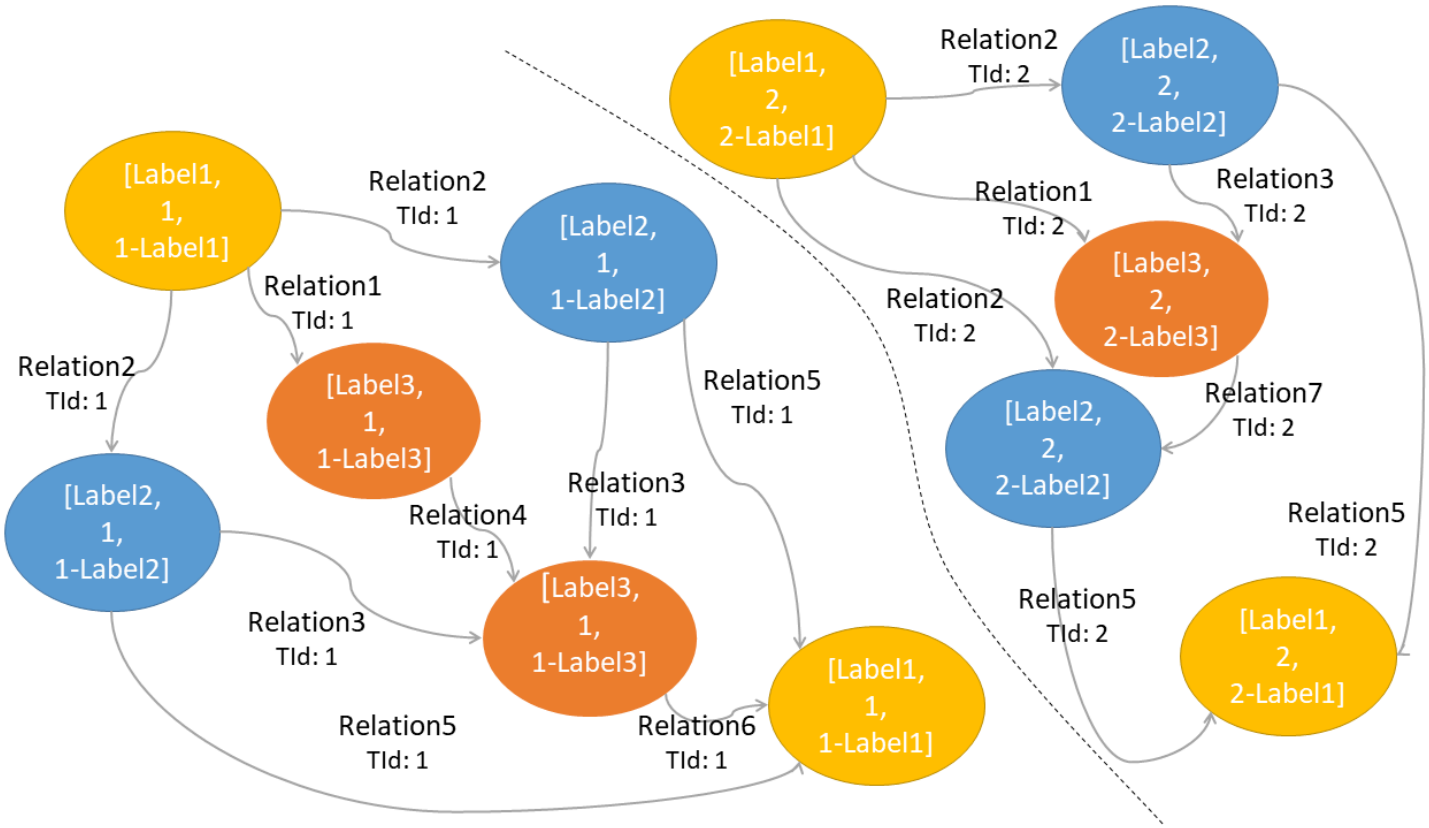
前缀标签方法有两个主要缺点。首先，很难运行跨租户的任何查询。例如，对给定标签的所有节点进行计数以进行报告或监控的查询。如果这是您的用例，请考虑将此策略与多标签策略相结合。有关组合策略的更多信息，请参阅混合模型部分。

其次，前缀标签策略要求控制措施强制在每个查询中正确应用适当的前缀，以防止数据泄露。但是，对于需要低延迟查询的工作负载，这种策略是最有效的选择，我们强烈推荐使用这种策略。液化石油气模型的性能影响部分提供了一些示例，说明为什么这是最有效的策略。

## 多标签策略

第三种选择是使用多标签策略。对于这种方法，您可以为图表上的每个节点添加额外的标签。例如，如果您需要筛选给定租户的所有数据，请添加租户 ID 标签。 如果您需要筛选给定标签的所有数据，而不考虑租户，请添加该标签。下图显示了通过为每个节点使用三个标签所应用的多标签策略。

现在，您可以使用三种不同的模式来访问图表：




- 筛选Label1以返回所有租户中的Label1所有节点。
- 筛选1以返回租户 1 的所有节点。
- 筛选后仅1-Label1返回带有标签的租户 1 的所有节点Label1。

因为 LPGs，有两种方法可以实现这一点。

在 Gremlin 中，您可以使用名为的遍历策略SubgraphStrategy将所有查询的范围限制为仅具有特定标签的顶点，例如："Label1"

```
g.withStrategies(
    new SubgraphStrategy(
        vertices=hasLabel("Label1")
    )
)
```

不同的是 PartitionStrategy，只 SubgraphStrategy 影响读取数据，不影响写入数据。要写入数据，请在每个查询中手动分配标签：

```
g.addV("Label1").property("Value","XYZ123456")
.addV("Label2").property("Value","XYZ123456")
```

读取数据时，您可以使用以下 SubgraphStrategy 命令查询所有节点"Label1"：

```
g.withStrategies(
new SubgraphStrategy(vertices=.hasLabel("Label1"))
).
V().has("Value","XYZ123456")
```

Neptune 仅返回第一条记录"Label1"，其值为。"XYZ123456"它等同于以下查询，但该查询不使用 SubgraphStrategy：

```
g.V().hasLabel("Label1").hasValue("XYZ123456")
```

在这个基本查询中，使用 SubgraphStrategy 起来似乎更复杂。请记住，您的库可以提供已定义策略g的实例。开发人员不必确保应用了正确的过滤器：

```
def getGraphTraversal():
return g.withStrategies(new SubgraphStrategy(vertices=.hasLabel("Label1"))

getGraphTraversal().has("Value","XYZ123456")
```

OpenCypher 库没有这些结构，因此您必须为每个节点创建多个标签：

```
CREATE (n:`1`:`Label1`:`1-Label1` {`~id`: 'Item_1', Value: '12345'})
```

当您使用这些标签筛选子图时，可以返回具有您正在寻找的客户标签的节点，或者返回与具有该标签的另一个节点共享关系的节点：

```
MATCH n=(:Label1:`1`)
// or
MATCH n=(:`1-Label1`)
```

多标签策略为您提供了最大的灵活性，可以按类型 (Label1) 或租户 (1) 查询节点，或者在性能最重要时使用更有效的前缀标签策略 ()。1-Label1

这种策略的主要缺点是，每个标签都是存储在图表中的额外对象。对象是节点或边上的节点、边或属性 LPGs。摄取速度由每秒对象数来衡量和约束，存储成本取决于消耗的千兆字节数。这意味着额外的物体可能会在大规模上产生可衡量的影响。

## 对液化石油气模型的性能影响

AWS 技能生成器课程 Amazon Neptune 的数据建模深入描述了 Neptune 数据模型的内部结构和建模含义，但我们将在这里总结这些设计的重要注意事项。考虑在单个 Neptune 集群上有三个租户（T1、T2、T3）。这些租户具有以下属性：

- 租户 1 (T1) 总共有 1 亿个节点，其中 1000 万个节点属于 Item 类型。
- 租户 2 (T2) 总共有 1000 万个节点，其中 100 万个节点属于 Item 类型。
- 租户 3 (T3) 总共有 1 亿个节点，其中 100 万个节点属于 Item 类型。

运行查询，使用属性策略为租户 3 检索物品。Neptune 会检查两个索引调用的统计信息：

- 哪里`tenant property key=T3`有 1 亿个结果
- 哪里`label = Item`有 1200 万个结果（来自 T1 的 1000 万个 + 来自 T2 的 100 万个 + 来自 T3 的 100 万个）

Neptune 查询优化器确定最好先应用后一个查询（1200 万个结果），然后检查每个项目。`tenant property key=T3`您检索了 1200 万个项目以查找 100 万个结果。

请注意此查询对邻居的噪音影响。如果每个租户有 1 亿个 Item 节点，则第一个查询将有 3 亿个结果，而不是 1200 万个结果（出于说明目的，这过于简化。 Neptune 优化器可能应用了不同的操作顺序）。

接下来，考虑前缀标签策略。调用单个索引 wherelabel=T3-Item，这将返回 100 万个结果。这实现了与财产策略相同的结果，但它检索的记录减少了1100万条。此外，由于索引中的标签不重叠，因此您不再有噪音邻居顾虑。

与属性策略相比，多标签策略并不能直接提高查询性能。当搜索空间也具有可比性时，按属性值筛选与按标签值筛选相当。相反，多标签策略支持更大的灵活性。 多标签策略提供的性能等同于或标签的前缀标签策略。label=T3 T3-Item多标签策略提供的性能与属性策略相同。label=Item好处是支持各种访问模式。

# RDF 的池模型

资源描述框架 (RDF) 具有命名图的概念，它提供了一种分隔数据的合乎逻辑的方法。在 Amazon Neptune 中，您有一个默认的命名图表和用户定义的命名图表。您可以根据需要创建任意数量的命名图表。它们统称为 RDF 数据集。所有命名的图形，无论是默认的还是用户定义的，都由 RDF 数据集中的国际化资源标识符 (IRI) 定义。在 Neptune 中，除非用户在写入数据时声明了命名图，否则所有三元组都被视为默认命名图的一部分。

命名图有多种用例：

- 数据分区和数据隔离
- 数据来源
- 版本控制
- 推理

本指南重点介绍数据分区用例。我们建议为每个租户创建一个用户定义的命名图。

## 使用 Graph Store HTTP 协议的 SPARQL 查询选项

以下示例查询使用 SPARQL 协议和 RDF 查询语言 (SPARQL) 以及 Graph Store HTTP 协议为租户查询或创建命名图。

- HTTP GET-要检索租户的特定图表，请执行以下操作：

```
curl --request GET 'https://your-neptune-endpoint:port/sparql/gsp/?graph=http%3A//
www.example.com/named/tenant1'
```

- HTTP PUTα 要使用请求中指定的有效负载创建或替换特定的命名图表，请执行以下操作：

```
curl --request PUT -H "Content-Type: text/turtle" \  --data-raw "@prefix ex: http://
example.com/ . ex:subject ex:predicate ex:object ." \
'https://your-neptune-endpoint:port/sparql/gsp/?graph=http%3A//www.example.com/named/
tenant1'
```

在 RDF 中，物体是三元组。

- HTTP POSTα 要创建一个新的命名图表（如果不存在），或者要与现有图表合并，请执行以下操作：

```
curl --request POST -H "Content-Type: text/turtle" \
--data-raw "@prefix ex: http://example.com/ . ex:subject ex:predicate ex:object ." \

'https://your-neptune-endpoint:port/sparql/gsp/?graph=http%3A//www.example.com/named/
tenant1'
```

## RDF 的租户隔离

要在应用程序层设置必要的防护栏的情况下对数据进行逻辑隔离，请在租户和用户定义的命名图形之间创建映射。在为 RDF 数据集设计多租户时，请注意以下方面 RDF 和 SPARQL：

- 在 Neptune 中，当你在不指定命名图的情况下进行查询时，它会检索数据库中所有命名图形中与模式匹配的所有三元组。
- 在 RDF 中，不同命名图的节点之间的连接没有限制。例如，在上图中，中的节点:G1可以通过边缘连接到:G2中的节点。

例如，如果特定租户的最终用户向 API 提交查询，则 API 应在将查询提交到 Neptune 数据库之前验证以下要求：

- 任何以单个租户为范围的查询都必须指定命名图。否则，您就有可能在租户之间泄露数据。
- 更新或删除查询应始终指定命名图表。
- 边或关系两边的节点都应始终属于正确的命名图。

有关最佳实践的更多信息，请参阅 Neptune 文档。

# 为成长做好准备

成功使用池模型后，最终会超出单个 Neptune 集群的大小。租户增长，或者租户数量增加，所有客户所需的数据摄取率超过了集群的能力。发生这种情况时，您需要将客户分成多个集群。事先针对此配置进行设计，而不必稍后再尝试进行改造。即使您的初始规模是仅使用单个集群，也要模拟将来达到该规模时跨多个集群路由租户所需的组件。

如果您的解决方案需要更多资源，具体取决于您的租户规模，请同时为他们的增长做好准备。如果单个集群上的多个客户显著增长，则该集群可能不再支持您的需求。使用 Amazon Neptune 数据库克隆功能，设计一种策略，将租户移动到另一个集群，或者将现有集群一分为二。

熟悉 Neptune Copy-on-Write 协议，它可以在你实施数据库克隆时为你节省资金。，如果你因为摄取瓶颈而拆分集群，那么只要你的策略允许，不从集群中删除数据可能会更有效率。如果数据页面未更改，则两个集群将共享该页面，但如果数据页面已修改（因为数据页上的某些数据已被删除），则不会共享该页面。

> **Note**
>
> 本指南适用于撰写本文时最新的 Neptune 版本，即 Neptune 版本 1.3.1。随着 Neptune 存储层的发展，该指南可能会在 future 版本中发生变化。

## 多租户场景的限制

请注意，某些 Neptune 功能不是为多租户场景构建的。不应允许租户直接访问池模型中的 Neptune 端点，因为这些多租户策略不是在数据库级别强制执行的。始终在您的客户和 Neptune 端点之间保留某种代理，以强制执行本文档中描述的设计。此类代理的示例包括以下内容：

- 在客户端图层中追加标签过滤器
- 有一个 API 可以将身份验证令牌映射到租户 ID 并将此过滤器注入查询

本指南还适用于让客户直接访问海王星图形笔记本、Neptune 图形浏览器或 Neptun e Streams 等功能。

# 混合模式多租户

SaaS 解决方案通常混合使用孤岛模式和池模式。各种因素会影响在同一环境中何时以及如何同时使用筒仓和池模型的决定。

其中一个因素是分层，即SaaS解决方案为每层租户提供独特的体验。例如，如果您的套餐是免费套餐、标准套餐和高级套餐，则可以使用池模型将免费套餐租户数据存储在共享的 Neptune 集群中。对于标准和高级级别的租户，您可以使用 cluster-per-tenant 孤岛模式。

此外，一些 SaaS 提供商能够在共享的 Amazon Neptune 集群上构建其池解决方案作为基础。随后，他们可以为需要孤立存储的租户创建单独的 Neptune 集群，这通常是出于合规和监管要求。

尽管这可能会增加您的数据访问层和管理配置文件的复杂性，但它也可以为您的企业提供一种对产品进行分层以满足客户要求的方法。

# 的操作最佳实践 ISVs

本节中的许多指南都是适用于所有客户的最佳实践，但它们对它们具有更重要的意义 ISVs。

## 使用最新版本更新你的 Neptune 集群

在 Amazon Neptune 发行说明中，您可以看到每个版本都带来了许多错误修复、性能改进和新功能。尽量让你的 Neptune 集群保持最新版本。

如果您在工作负载中发现了以前未被发现的错误，并且您的集群使用的是最新版本，Neptune 工程师可以为您的集群创建私有补丁（如果需要并且您想要的话）。该补丁可以桥接到下一个版本，届时该修复程序将正式推出。要帮助将集群更新到最新版本，请使用 Nep tune Blu e/Green 解决方案。

## 使用增量而不是删除和替换进行数据摄取

您可以使用多种技术将数据摄取或写入到 Neptune。许多客户试图通过每次收到数据源更改时删除并重新插入图表来简化数据摄取。他们可能会为每个节点添加一个`last-modified`属性，并定期扫描自某个指定日期以来未修改过的节点，然后将其删除。虽然这些技术简化了数据摄取过程，但它们会对您的 Neptune 集群产生长期的运行状况和可扩展性影响。

首先，Neptune 使用字符串的字典编码。除非您明确指定节点和边 IDs 的值，否则 Neptune 会生成一个以 ID 字符串表示的 GUID，并将该字符串存储在字典中。如果您经常删除和添加对象，则自动生成的对象 IDs 会导致字典膨胀。

其次，Neptune 向上扩展，最大每秒可摄取大约 120 K 个物体。如果您不断删除和添加对象，则会将大量带宽消耗在基本未更改的对象上。这限制了你可以在集群上托管的租户数量，需要在集群中使用更大的写入器实例，并且需要更多的 I/O 操作。所有这些因素都会增加您的成本。

我们强烈建议您开发一种方法来计算已更改内容的真实增量，而不是使用删除和添加方法。但是，有些数据源不利于此（例如，返回当前状态的 API 调用，或者无法准确跟踪更改内容的事件）。如果您的原始数据源不利于识别更改，请使用提取、转换和加载 (ETL) 过程来计算增量。例如，您可以以 Parquet 格式维护以前每次数据捕获的快照，用于 AWS Glue 计算这些快照之间的差异，然后仅将差异推送到 Neptune。

# 建模 Neptune 的成本将如何随着租户的变化而变化

无论您使用孤岛、池还是混合模式，您的云成本都将随着租户的规模而扩展。与并发连接较少的租户相比，需要更多并发连接的租户需要更大的实例或更多的只读副本。这同样适用于需要更快地摄取数据的租户。

Neptune 集群成本的三个组成部分是实例大小（和数量）、数据大小（GB-月）和 I/O 操作（每百万个）。虽然这些成本通常因工作负载而异，但它们会随着大小和数据量而扩展，但可以通过使用 AWS 工具来衡量。根据租户规模的关键指标，包括租户规模随时间的变化情况，跟踪和了解规模经济。如果 I/O 费用的不可预测性影响了利润，请考虑选择 Neptune I/O 优化存储，以获得更可预测的成本。

# 根据客户需求扩展您的集群

对于正确调整您的 Neptune 实例大小，没有久经考验或真实的公式。Nep tune 文档提供了指导，但变量太多，无法推荐直接映射。这些变量包括但不限于以下内容：

- 数据模型
- 数据形状
- 查询并发性
- 查询的复杂性

计划测试以确定您的工作负载和租户配置文件的最佳规模。一般而言，我们建议使用预配置实例，以提高成本效益和可预测性。如果您的客户体验目标优先考虑最佳扩展而不是成本，请考虑使用 Neptune Serverless 实例来确保无论工作负载如何波动都能获得更一致的体验。

如果您的租户读取工作负载的峰值和低谷变化很大，请将 Neptune Serverless 实例与 Neptune 自动缩放结合使用。 新的只读副本在初始化后通常需要 10-15 分钟才能上线。这意味着仅靠自动缩放可以应对长时间的流量变化，但这还不足以应对快速变化的活动峰值。通过将 Neptune Serverless 和 Neptune 自动缩放相结合，您可以向上或向下扩展实例，也可以缩小只读副本的数量。

如果您的租户的工作负载配置文件或服务级别协议（SLAs）有很大不同，请考虑使用自定义终端节点和专用只读副本将流量引导到针对该流量进行了优化的实例。优化可以包括不同的实例大小、特定的查询模式或预热缓冲区缓存。

# 后续步骤

如果您刚刚开始为多租户ISV应用程序实施 Amazon Neptune，请多考虑所需的模型。在以后的旅程中，更改模型的成本会更高。

如果您处于旅程的初期，请确认您使用的是最适合您需求的模型，并且您是否遵循了该模型的指导方针。

事先计划。当你处于旅程的早期阶段时，人们很容易推迟跨集群对客户进行分片或优化ETL流程以提供变化的增量，而不是删除和重新添加顶点和边缘的工作。随着规模的扩展，这些决策可能会对性能和成本产生负面影响。

最后，如果您已经踏上了旅程，那么本指南可能会向您保证您的架构是最佳的，或者它可能会提供一些更改以改进您的架构。

如果您对本指南有任何疑问或需要进一步的帮助，请联系您的 AWS 账户 团队并要求与 Neptune 专家进行会谈。

# 资源

- 亚马逊 Neptune 文档
- 亚马逊 Neptune 的数据建模（课程）
- 为亚马逊 Neptune 应用 AWS Well-Architected 框架
- Well-Architected Framework
- 多租户架构指南 AWS
- SaaS 租户隔离策略：在多租户环境中隔离资源
- Apache 文档 TinkerPop
- SPARQL

# 贡献者

本指南的贡献者包括：

- Brian O'Keefe，WW Neptune 校长，SSA AWS
- Veeresham Gande，高级技术客户经理， AWS
- 达娜·欧文斯，创业解决方案架构师， AWS
- Nima Seifi，创业解决方案架构师， AWS

# 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 |
| --- | --- | --- |
| 初次发布 | — | 2024 年 9 月 3 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：将本地 Oracle 数据库迁移到 AWS 云中的 Amazon Relational Database Service（Amazon RDS）for Oracle。
- 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。
- 重新托管（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：将本地 Oracle 数据库迁移到 AWS 云中 EC2 实例上的 Oracle。
- 重新放置（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。您将服务器从本地平台迁移到同一平台的云服务中。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

请参阅托管服务。

ACID

请参阅原子性、一致性、隔离性、持久性。

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。它比主动-被动迁移更灵活，但工作量更大。

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

一种 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

请参阅人工智能。

AIOps

请参阅人工智能运营。

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？

人工智能操作 (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 AIOps AWS 迁移策略中使用的更多信息，请参阅操作集成指南。

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。

权威数据来源

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来源复制到其他位置，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

恶意机器人

一种旨在扰乱或伤害个人或组织的机器人。

BCP

请参阅业务连续性计划。

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。另请参阅字节顺序。

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应用程序版本（蓝色），在另一个环境中运行新应用程序版本（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交互的软件应用程序。有些机器人是有用或有益的，例如在互联网上索引信息的 Web 爬网程序。还有一些被称为恶意机器人的机器人，其目的是扰乱或伤害个人或组织。

僵尸网络

被恶意软件感染并受单方（称为僵尸网络控制者或僵尸网络操作者）控制的僵尸网络。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）。

紧急（break-glass）访问

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 AWS Well-Architected Guidance 中的 Implement break-glass procedures 指示器。

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

请参阅 AWS 云采用框架。

金丝雀部署

缓慢而渐进地向最终用户发布版本。当您确信无误后，即可部署新版本，并完全替换当前版本。

CCoE

请参阅云卓越中心。

CDC

请参阅更改数据捕获。

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韧性。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

请参阅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。云计算通常连接到边缘计算技术。

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 云中时通常会经历四个阶段：

- 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，创建着陆区、定义 CCo E、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 云 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

请参阅配置管理数据库。

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 Bitbucket Cloud。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 管线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）

一种 AI 领域，它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，Amazon SageMaker AI 为 CV 提供了图像处理算法。

配置偏移

对于工作负载而言，一种偏离预期状态的配置更改。这可能会导致工作负载变得不合规，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，以自定义您的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、提高生产力、提高代码质量和更快地交付。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

请参阅计算机视觉。

# 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 云 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

请参阅数据库定义语言。

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。

委派管理员

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

请参阅环境。

侦测性控制

一种安全控制，在事件发生后进行检测、记录日志和发出提醒。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

星型架构中的一种较小的表，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）

您用来最大程度地减少由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

请参阅数据库操作语言。

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

请参阅灾难恢复。

偏差检测

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

请参阅开发价值流映射。

# E

EDA

请参阅探索性数据分析。

EDI

请参阅电子数据交换。

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。与云计算比较时，边缘计算可以减少通信延迟并缩短响应时间。

电子数据交换（EDI）

组织之间业务文件的自动交换。有关更多信息，请参阅什么是电子数据交换。

加密

一种将人类可读的纯文本数据转换为加密文字的计算流程。

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

请参阅服务端点。

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。

环境

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

请参阅企业资源规划。

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

星型架构中的中心表。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。

快速失效机制

一种使用频繁且增量式的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 云，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

请参阅分支。

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，请参阅使用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。

少样本提示

在要求 LLM 执行类似任务之前，先向其提供少量示例，以演示任务和预期输出。此技术是上下文内学习的一种应用，其中模型可以从提示中嵌入的示例（样本）中学习。对于需要特定格式、推理或领域知识的任务，少样本提示可能非常有效。另请参阅零样本提示。

FGAC

请参阅精细访问控制。

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，通过更改数据捕获使用连续数据复制，在极短的时间内迁移数据，而非使用分阶段方法。目标是将停机时间降至最低。

FM

请参阅基础模型。

基础模型（FM）

一个大型深度学习神经网络，一直在广义和未标记数据的大量数据集上进行训练。 FMs 能够执行各种各样的一般任务，例如理解语言、生成文本和图像以及用自然语言进行对话。有关更多信息，请参阅什么是基础模型。

# G

生成式人工智能

AI 模型的一个子集，这些模型已经过大量数据训练，可以使用简单的文本提示来创建新的内容和构件，例如图像、视频、文本和音频。有关更多信息，请参阅什么是生成式人工智能。

地理阻止

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工作流程，而基于中继的工作流程则是现代的、首选的方法。

黄金映像

系统或软件的快照，用作部署该系统或软件的新实例的模板。例如，在制造业中，黄金映像可用于在多个设备上预调配软件，并有助于提高设备制造操作的速度、可扩展性和生产效率。

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

帮助管理各组织单位的资源、策略和合规性的高级规则 (OUs)。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。侦测性护栏会检测策略违规和合规性问题，并生成提醒以进行修复。它们通过使用 AWS Config、、Amazon、 AWS Security Hub CSPM GuardDuty AWS Trusted Advisor、Amazon Inspector 和自定义 AWS Lambda 支票来实现。

# H

HA

请参阅高可用性。

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。

保留数据

从用于训练机器学习模型的数据集中保留的一部分标注的历史数据。通过将模型预测与保留数据进行比较，您可以使用保留数据来评估模型性能。

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。

# 我

IaC

请参阅基础设施即代码。

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 云 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

请参阅工业物联网。

不可变基础设施

一种模型，可为生产工作负载部署新的基础设施，而不是更新、修补或修改现有基础设施。不可变基础设施本质上比可变基础设施更一致、更可靠、更可预测。有关更多信息，请参阅 AWS Well-Architected Framework 中的使用不可变基础设施进行部署最佳实践。

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 安全参考架构建议设置您的网络帐户，包括入站、出站和检查， VPCs 以保护您的应用程序与更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 Klaus Schwab 在 2016 年提出，指的是通过连接、实时数据、自动化、分析和 AI/ML 的进步来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调配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) 数字化转型战略。

检查 VPC

在 AWS 多账户架构中，一种集中式 VPC，用于管理对 VPCs（相同或不同 AWS 区域）、互联网和本地网络之间的网络流量的检查。AWS 安全参考架构建议设置您的网络帐户，包括入站、出站和检查， VPCs 以保护您的应用程序与更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，请参阅使用机器学习模型的可解释性 AWS。

物联网

请参阅物联网。

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。

大语言模型（LLM）

一种基于大量数据进行预训练的深度学习 AI 模型。LLM 可以执行多项任务，例如回答问题、总结文档、将文本翻译成其他语言以及完成句子。有关更多信息，请参阅什么是 LLMs。

大规模迁移

迁移 300 台或更多服务器。

LBAC

请参阅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

请参阅 7 R。

小端序系统

一个先存储最低有效字节的系统。另请参阅字节顺序。

LLM

请参阅大型语言模型。

下层环境

请参阅环境。

# 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

请参阅分支。

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权限。恶意软件的示例包括病毒、蠕虫、勒索软件、木马、间谍软件和键盘记录器。

托管式服务

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。Amazon Simple Storage Service（Amazon S3）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

请参阅迁移加速计划。

机制

一个完整的过程，您可以在其中创建工具，推动工具的采用，然后检查结果以进行调整。机制是一种在运作过程中自我强化和改善的循环。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

请参阅制造执行系统。

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

一种小型的独立服务，通过明确的定义进行通信 APIs ，通常由小型的独立团队拥有。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通过定义明确的接口进行通信。 APIs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、开发人员和冲刺 DevOps 领域的专业人员。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，提供了用于验证迁移到 AWS 云的业务案例的信息。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

将工作负载迁移到 AWS 云的方法。有关更多信息，请参见术语表中的 7 R 词条，以及动员您的组织以加快大规模迁移。

ML

请参阅机器学习。

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，请参阅在 AWS 云中实现应用程序现代化的策略。

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

请参阅迁移组合评测。

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？”

可变基础设施

一种用于更新和修改生产工作负载的现有基础设施的模型。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

请参阅来源访问控制。

OAI

请参阅来源访问身份。

OCM

请参阅组织变革管理。

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

请参阅运营集成。

OLA

请参阅运营级别协议。

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

请参阅开放流程通信 – 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 提供了一个包含数据加密、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）

一份问题核对清单和关联的最佳实践，可帮助您了解、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 AWS Well-Architected Framework 中的运营准备情况审查（ORR）。

运营技术（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 转型的关键重点。

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。

ORR

请参阅运营准备情况审查。

OT

请参阅运营技术。

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 安全参考架构建议设置您的网络帐户，包括入站、出站和检查， VPCs 以保护您的应用程序与更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

请参阅个人身份信息。

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

请参阅可编程逻辑控制器。

PLM

请参阅产品生命周期管理。

policy

一个对象，可以定义权限（请参阅基于身份的策略）、指定访问条件（请参阅基于资源的策略）或定义 AWS Organizations 的组织中所有账户的最大权限（请参阅服务控制策略）。

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。

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 `false` 的查询条件，通常位于 `WHERE` 子句中。

谓词下推

一种数据库查询优化技术，可在传输之前筛选查询中的数据。这将减少从关系数据库检索和处理的数据量，并提高查询性能。

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

一种在整个开发过程中都考虑隐私的系统工程方法。

私有托管区

一个容器，其中包含有关您希望 Amazon Route 53 如何响应针对一个或多个 VPCs域名及其子域名的 DNS 查询的信息。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，旨在防止部署不合规资源。这些控制会在资源预置之前对其进行扫描。如果资源与控制不兼容，则不会预置它。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）

对产品在其整个生命周期内的数据和流程的管理，从设计、开发和发布，到增长和成熟，再到衰退和淘汰。

生产环境

请参阅环境。

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。

提示串接

使用一个 LLM 提示的输出作为下一个提示的输入，以生成更好的响应。该技术用于将复杂的任务分解为子任务，或者迭代地完善或扩展初步响应。它有助于提高模型响应的准确性和相关性，并允许获得更精细的个性化结果。

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。

publish/subscribe (pub/sub)

一种支持微服务间异步通信的模式，可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

一系列用于访问 SQL 关系数据库系统中的数据的步骤，类似于指令。

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

请参阅责任、问责、咨询和知情（RACI）。

RAG

请参阅检索增强生成。

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

请参阅责任、问责、咨询和知情（RACI）。

RCAC

请参阅行列访问控制。

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

请参阅 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

请参阅 7 R。

Region

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

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

请参阅 7 R。

版本

在部署过程中，推动生产环境变更的行为。

重新放置

请参阅 7 R。

更换平台

请参阅 7 R。

重新购买

请参阅 7 R。

韧性

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

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

请参阅 7 R。

停用

请参阅 7 R。

检索增强生成（RAG）

一种生成式人工智能技术，其中 LLM 在生成响应之前引用其训练数据来源之外的权威数据来源。例如，RAG 模型可以对组织的知识库或自定义数据执行语义搜索。有关更多信息，请参阅什么是 RAG。

轮换

定期更新密钥以使攻击者更难访问凭证的过程。

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

请参阅恢复点目标。

RTO

请参阅恢复时间目标。

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 管理控制台 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

请参阅监督控制和数据采集。

SCP

请参阅服务控制策略。

机密密钥

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 文档中的什么是 Amazon Secrets Manager 密钥？。

安全设计

一种在整个开发过程中都考虑安全的系统工程方法。

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。安全控制有以下四种类型：预防性、检测性、响应性和主动性。

安全固化

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

一种预定义的程序化操作，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 实例或轮换凭证。

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，用于集中控制组织中所有账户的权限 AWS Organizations。 SCPs 定义防护措施或限制管理员可以委托给用户或角色的操作。您可以使用 SCPs 允许列表或拒绝列表来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。

服务水平指示器（SLI）

对服务性能方面的衡量，例如错误率、可用性或吞吐量。

服务水平目标（SLO）

代表服务运行状况的目标指标，由服务水平指示器衡量。

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。

SIEM

请参阅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

请参阅服务水平协议。

SLI

请参阅服务水平指示器。

SLO

请参阅服务水平目标。

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，请参阅在 AWS 云中实现应用程序现代化的分阶段方法。

SPOF

请参阅单点故障。

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事务数据或测量数据，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。

监督控制和数据采集（SCADA）

在制造业中，一种使用硬件和软件来监控实物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。

系统提示

一种为 LLM 提供上下文、说明或准则以指导其行为的技术。系统提示有助于设置上下文并制定与用户交互的规则。

# T

标签

键值对，用作组织资源的元数据。 AWS 标签有助于您管理、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

请参阅环境。

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

一个网络传输中心，可用于将您的网络 VPCs 和本地网络互连。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

请参阅环境。

# 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

两者之间的连接 VPCs ，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

一种对与当前记录有某种关联的一组行执行计算的 SQL 函数。窗口函数对于处理任务很有用，例如计算移动平均值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。

WORM

请参阅一次写入多次读取。

WQF

请参阅 AWS 工作负载资格鉴定框架。

一次写入多次读取（WORM）

一种存储模型，可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，但无法对其进行更改。此数据存储基础设施被认为不可变。

# Z

零日漏洞利用

一种利用零日漏洞的攻击，通常为恶意软件。

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。

零样本提示

为 LLM 提供执行任务的说明，但没有可以帮助指导的示例（样本）。LLM 必须使用预先训练的知识来处理任务。零样本提示的有效性取决于任务的复杂性和提示的质量。另请参阅少样本提示。

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。