欢迎学习“模块 9：实施弹性、高可用性和监控。”

本模块包含以下章节：

1. 架构需求
2. 扩缩计算资源
3. 扩缩数据库
4. 设计高度可用的环境
5. 监控

本模块还包括：

- 演示如何为 Amazon EC2 Auto Scaling 创建目标跟踪和步进扩缩策略
- 演示如何使用 Application Load Balancer 部署高度可用的 Web 应用程序
- 演示 Amazon Route 53
- 指导实验：创建高度可用的环境
- 挑战实验：为咖啡馆创建可扩展且高度可用的环境

最后，您需要完成一个知识测验，以测试您对本模块中涵盖的关键概念的理解程度。

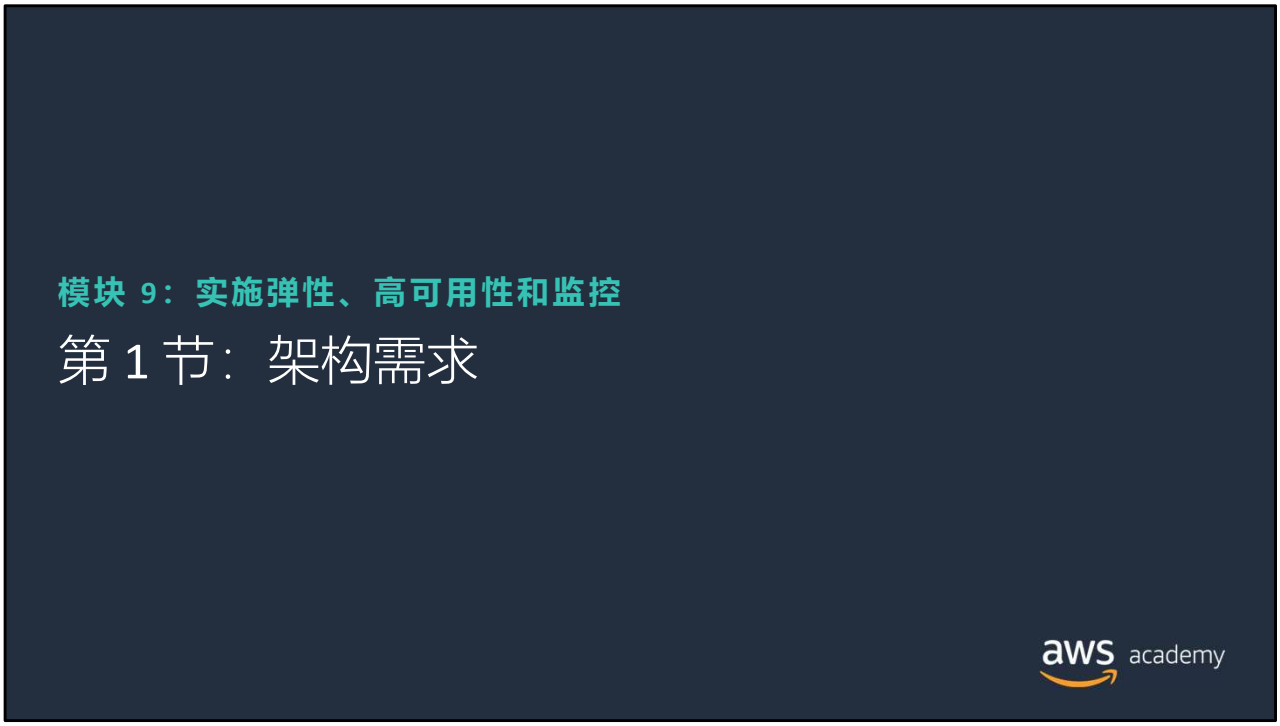
模块目标

学完本模块后，您应该能够：

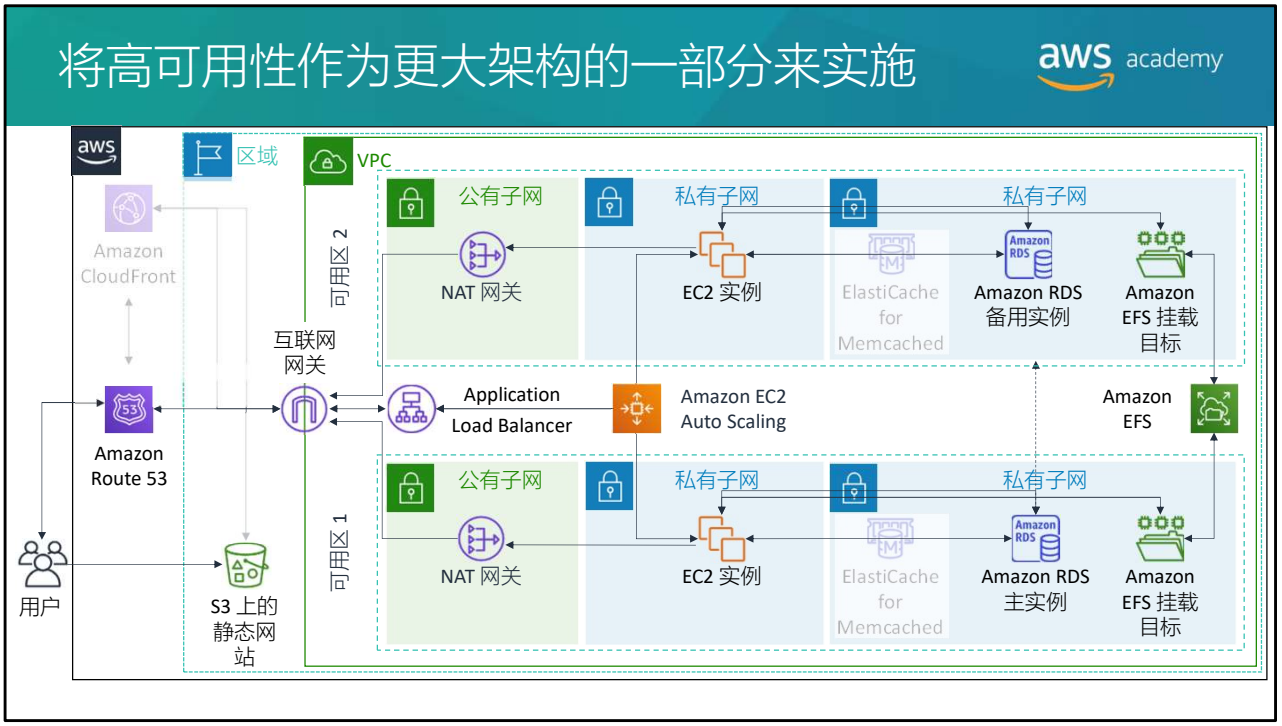
- 在架构中使用 Amazon EC2 Auto Scaling 来提高弹性
- 说明如何扩缩数据库资源
- 部署 Application Load Balancer 以创建高度可用的环境
- 使用 Amazon Route 53 进行域名系统 (DNS) 故障转移
- 创建高度可用的环境
- 设计能够使用 Amazon CloudWatch 监控资源并做出相应反应的架构

学完本模块后，您应该能够：

- 在架构中使用 Amazon EC2 Auto Scaling 来提高弹性
- 说明如何扩缩数据库资源
- 部署 Application Load Balancer 以创建高度可用的环境
- 使用 Amazon Route 53 进行域名系统 (DNS) 故障转移
- 创建高度可用的环境
- 设计能够使用 Amazon CloudWatch 监控资源并做出相应反应的架构



介绍第 1 节：架构需求。



在本模块中，您将学习如何实施富有弹性且响应迅速的反应式架构。讨论使此架构可扩缩且高度可用的组件，例如第二个可用区、Application Load Balancer、Amazon EC2 Auto Scaling 和 Amazon Route 53。

咖啡馆业务要求

咖啡馆将在著名的电视美食节目中亮相。当节目播出时，架构必须处理大幅增加的容量。

咖啡馆很快将在著名的电视美食节目中亮相。当节目播出时，Sofía 和 Nikhil 预计咖啡馆 Web 服务器的用户数量将出现短暂的激增，甚至可能达到数万人。目前，咖啡馆的 Web 服务器部署在一个可用区，他们担心服务器无法应对预期的流量增长。他们希望确保客户在访问网站时拥有出色的体验，不会遇到任何问题，例如下单延迟或延误。

为了确保提供这种体验，网站必须迅速响应，通过扩缩来满足不断变化的客户需求，并做到高度可用。它还必须包含负载均衡。此架构必须跨多个应用程序服务器分发客户订单请求，以便应对需求的增加，而不是让单个服务器过载。

反应式架构

弹性且可扩缩

富有弹性

响应迅速

消息驱动

现代应用程序必须能够处理大量数据，不会停机，响应时间达到亚秒级。为了满足这些需求，您可以实施一个具有弹性、响应迅速、消息驱动的反应式系统。精心设计的反应式架构可以为您节省资金并为用户提供更好的体验。

在本模块中，您将学习如何在 AWS 上构建富有弹性且响应迅速的反应式架构。您将了解消息驱动组件，以便在后面的模块中学习构建解耦架构。

模块 9：实施弹性、高可用性和监控

第 2 节：扩缩计算资源

aws academy

介绍第 2 节：扩缩计算资源。

什么是弹性?

aws academy

弹性基础设施可以随着容量需求的变化而扩展和收缩。

示例:

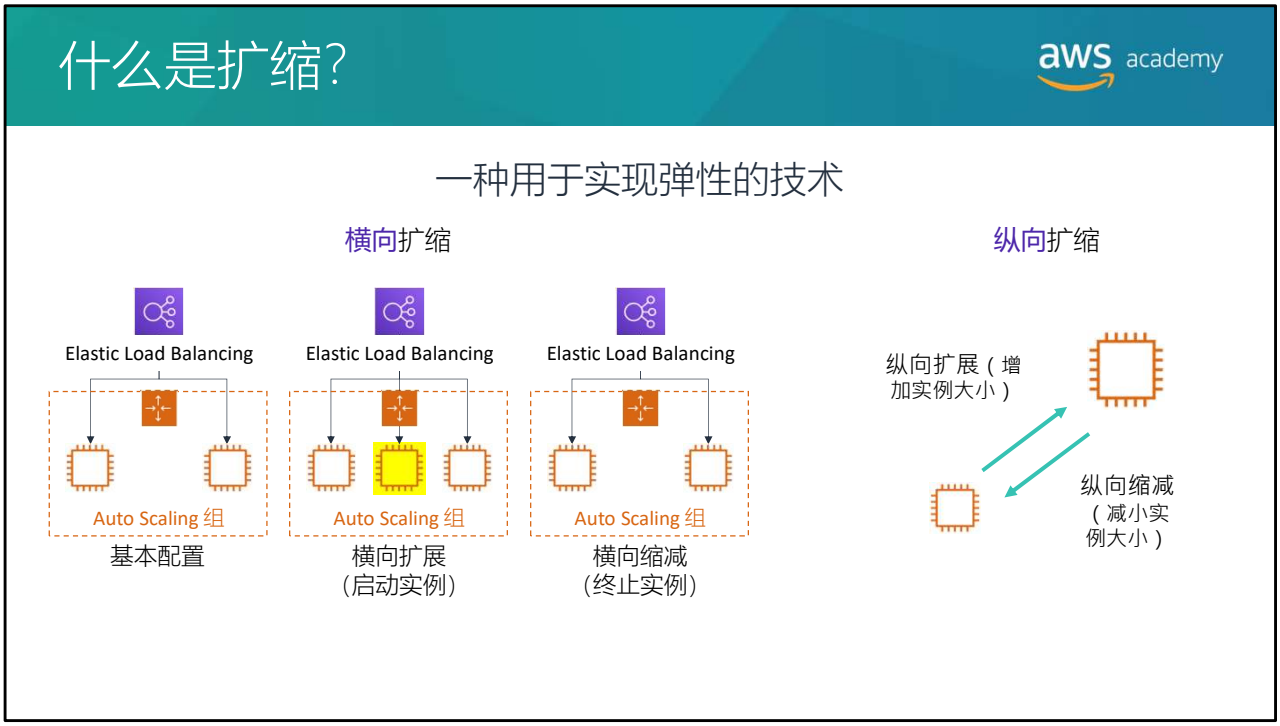
- 在流量激增时增加 Web 服务器的数量
- 在流量下降时降低数据库的写入容量
- 处理整个架构中需求的日常波动

反应式架构的一个特征是弹性。*弹性*意味着，基础设施可随着容量需求的变化而扩展和收缩。您可以在需要时获取资源，在不需要时释放资源。

弹性使您能够:

- 当应用程序流量激增时，增加 Web 服务器的数量
- 在流量下降时降低数据库的写入容量
- 处理整个架构中需求的日常波动

在咖啡馆的例子中，弹性很重要，因为在电视节目播出之后，网站的流量可能会立即增加。流量可能会在一周后下降到正常水平，或者可能在节假日期间再次增加。



*扩缩*是一种用于实现弹性的技术。扩缩是指增加或减少应用程序计算容量的能力。

扩缩有两种类型:

- *横向扩缩*是指添加或删除资源。例如，您可能需要向存储阵列添加更多硬盘驱动器，或添加更多服务器来支持应用程序。添加资源称为*扩展*，而终止资源称为*缩减*。横向扩缩是构建利用云计算弹性的互联网级应用程序的有效方式。

- *纵向扩缩*是指增大或减小单个资源的规格。例如，您可以升级服务器，使其具有更大的硬盘驱动器或更快的 CPU。借助 Amazon Elastic Compute Cloud (Amazon EC2),，您可以停止实例并将其大小调整为具有更多 RAM、CPU、I/O 或联网功能的实例类型。纵向扩缩最终可能达到极限，而且有时不是一种具有成本效益或高度可用的方法。但它很容易实施，对于许多使用案例来说可能已经足够，尤其是在短期内。

Amazon EC2 Auto Scaling

- 根据指定条件 启动或终止实例
- 指定后，使用负载均衡器自动注册新实例
- 可以跨可用区启动

在云中，可以自动处理扩缩。Amazon EC2 Auto Scaling 有助于保持应用程序的可用性，允许您根据您定义的策略、计划和运行状况检查自动添加或删除 EC2 实例。如果您指定了扩缩策略，Amazon EC2 Auto Scaling 可以在应用程序需求增加或降低时启动或终止实例。

Amazon EC2 Auto Scaling 与 Elastic Load Balancing 集成 – 它会自动向负载均衡器注册新实例，以便在实例之间分配传入的流量。

Amazon EC2 Auto Scaling 允许您在一个区域中构建跨多个可用区的高可用性架构。稍后您将在本模块中了解有关高可用性的更多信息。如果一个可用区运行状况不佳或无法使用，Amazon EC2 Auto Scaling 会在未受影响的可用区中启动新实例。当运行状况不佳的可用区恢复到正常运行状态时，Amazon EC2 Auto Scaling 会自动将这些应用程序实例重新平均分配到所有指定的可用区中。

Amazon EC2 Auto Scaling 提供了多种扩缩调整方式来满足您的应用程序需求：

- 计划扩缩 – 使用计划扩缩，扩缩操作将按照日期和时间的函数自动执行。如果您确切地知道应在何时增加或减少组中的实例数量，那么该功能对于可预测的工作负载会非常有用。例如，假设您每周的 Web 应用程序流量在星期三开始增加，星期四仍然保持较高水平，然后在星期五开始减少。您可以根据 Web 应用程序的可预测流量模式来规划扩缩操作。要实施计划扩缩，您可以创建计划操作。

- 动态按需扩缩 – 此方法是更高级的资源扩缩方式。它允许您定义用于控制扩缩过程的参数。例如，您有一个当前在两个 EC2 实例上运行的 Web 应用程序。您希望在应用程序负载变化时将 Auto Scaling 组的 CPU 利用率保持在接近 50%。在根据条件变化进行扩缩，但却不知道条件何时改变时，可以使用这种方法。动态扩缩为您提供了额外的容量来应对流量高峰，而无需维护过多的空闲资源。您可以将 Auto Scaling 组配置为自动扩缩来满足这一需求。

- 预测性扩缩 – 您可以将 Amazon EC2 Auto Scaling 与 AWS Auto Scaling 配合使用来实施预测性扩缩，使用此方式时，容量可根据预测的需求进行扩缩。预测性扩缩使用从您的 Amazon EC2 实际使用情况中收集的数据，而这些数据通过 AWS 的观察得出的数十亿个数据点得到进一步丰富。然后，AWS 使用经过良好训练的机器学习模型来预测您的预期流量（和 Amazon EC2 使用情况），包括每日和每周模式。该模型至少需要 1 天的历史数据才能开始进行预测。每 24 小时重新评估一次，以创建接下来 48 小时的预测。预测过程中会产生可以驱动一组或多组自动扩缩的 EC2 实例的扩缩计划。

动态扩缩和预测性扩缩可结合使用，以便更快地扩缩基础设施。

最后，您还可以手动添加或删除 EC2 实例。对于手动扩缩，您只指定 Auto Scaling 组的最大容量、最小容量或所需容量的变化。

动态扩展策略类型

- 简单扩缩 – 单次扩缩调整
  - 示例使用案例：新工作负载、突增的工作负载
- 步进扩缩 – 根据警报违规数量调整
  - 示例使用案例：可预测的工作负载
- 目标跟踪扩缩 – 特定指标的目标值
  - 示例使用案例：横向可扩缩应用程序，比如负载均衡应用程序和批数据处理应用程序

动态扩缩意味着调整应用程序的容量以满足不断变化的需求，从而优化可用性、性能和成本。扩缩策略类型决定了如何执行扩缩操作：

- 对于**步进扩缩**和**简单扩缩**策略，您要为触发扩缩过程的 CloudWatch 警报选择扩缩指标和阈值。您还要定义在指定数量的评估期内违反阈值时应如何扩缩 Auto Scaling 组。主要区别在于，步进扩缩允许根据一组扩缩调整值（称为*步进调整*，具体因警报触发情况而异）来增加或减少 Auto Scaling 组的当前容量。

- **目标跟踪扩缩**策略根据特定指标的目标值，增加或减少当前组容量。这种类型的扩缩类似于您家中负责控制温度的恒温器： 您只需选择一个温度，恒温器会替您完成所有其他工作。使用目标跟踪扩缩策略时，您可以选择扩缩指标并设置目标值。Amazon EC2 Auto Scaling 会创建和管理 CloudWatch 警报，这些警报会根据指标和目标值触发扩缩策略并计算扩缩调整值。扩缩策略根据需要增加或减少容量，将指标保持在指定的目标值或接近指定的目标值。除了将指标保持在接近目标值以外，目标跟踪扩缩策略还会针对由于负载模式变化而造成的指标变化进行调整。

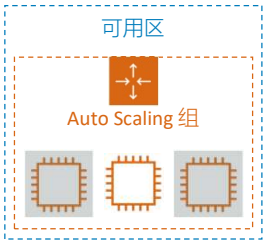
要了解有关目标跟踪扩展扩缩的更多信息，请参阅以下资源：
- Amazon EC2 Auto Scaling 的目标跟踪扩缩策略
- Set it and Forget it Auto Scaling Target Tracking Policies
- AWS re: Invent 2017: Auto Scaling Prime Time: Target Tracking Hits the Bullseye at Netflix

Auto Scaling 组

Auto Scaling 组定义以下内容：

- 最小容量
- 最大容量
- 所需容量*

容量?



*所需容量反映了正在运行并会随对事件的响应而波动的实例数量。

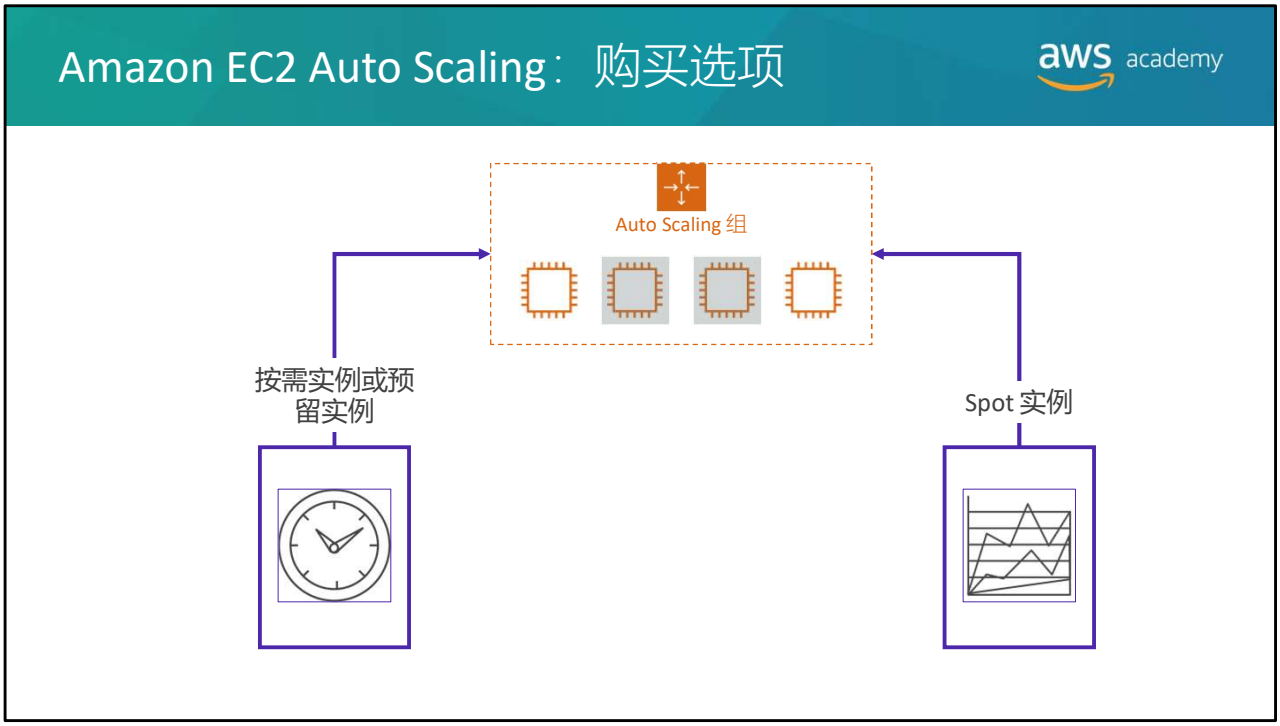
Amazon EC2 Auto Scaling 可帮助您确保拥有适量的 Amazon EC2 实例来处理您的应用程序负载。

您可以创建称为 *Auto Scaling* 的 Amazon EC2 实例集合。您可以指定每个 Auto Scaling 组内的*最小*实例数量，而 Amazon EC2 Auto Scaling 会确保您的组始终不低于此数量。您可以指定每个 Auto Scaling 组内的*最大*实例数量，而 Amazon EC2 Auto Scaling 会确保您的组始终不高于此数量。如果您在创建组时或创建之后指定*所需*容量，则 Amazon EC2 Auto Scaling 会确保您的组内实例始终保持此数量。

请注意，所需容量是基于触发器的设置，可能会随着诸如违反阈值之类的事件而波动。它反映了当时正在运行的实例数量，并且永远不能低于最小值或高于最大值。扩缩策略的作用在于充当您的自动代表，就如何调整所需容量做出决策。然后，Amazon EC2 Auto Scaling 会响应所需容量配置的更改。

首先，设置所需容量以告知 Auto Scaling 组在特定时间运行的实例数量。在 Amazon EC2 Auto Scaling 扩缩之前，当前正在运行的实例数量可能与所需的值不同。

Amazon EC2 Auto Scaling：购买选项



Amazon EC2 Auto Scaling 使您能够根据不断变化的条件横向扩展和缩减基础设施。在配置 Auto Scaling 组时，可以指定其使用的 EC2 实例类型。您还可以指定应由按需实例、预留实例和 Spot 实例满足所需容量的百分比。然后，Amazon EC2 Auto Scaling 将根据这些首选项预置能够达到预期容量的价格最低的实例组合。

您可以仅使用一种实例类型。但最好使用多个实例类型以避免从容量不足的实例池中启动实例。如果 Auto Scaling 组对 Spot 实例的请求无法在一个 Spot 实例池中得到满足，它将继续在其他 Spot 实例池中尝试，而不是启动按需实例。

有关购买选项的更多信息，请参阅具有多个实例类型和购买选项的 Auto Scaling 组。

自动扩缩注意事项

- 多种类型的自动扩展
- 简单、分步或目标跟踪扩展
- 多个指标（不仅仅是 CPU）
- 何时横向扩展和缩减
- 使用生命周期挂钩

在使用 Amazon EC2 Auto Scaling 扩缩架构时，需要考虑以下事项：

- 多种自动扩缩类型 – 您可能需要实施计划扩缩、动态扩缩和预测性扩缩的组合。
- 动态扩缩策略类型 – 简单扩缩策略可根据一个扩缩调整值来增加或减少当前组容量。步进扩缩策略根据一组扩缩调整值（称为*步进调整*，因警报违反情况而异）来增加或减少当前组容量。目标跟踪扩缩策略根据特定指标的目标值，增加或减少当前组容量。
- 多个指标 – 某些架构必须根据两个或更多指标（不仅仅是 CPU）进行扩缩。AWS 建议您使用目标跟踪扩缩策略按某个指标进行扩缩，例如平均 CPU 利用率或 Application Load Balancer 的 RequestCountPerTarget 指标。可通过在容量增加时减少并在容量减少时增加的指标按比例横向扩展或缩减使用目标跟踪的实例数。此类指标有助于确保 Amazon EC2 Auto Scaling 严格遵循应用程序的需求曲线。
- 何时横向扩展和缩减 – 横向扩展要早而快，横向缩减要缓而慢。
- 使用生命周期挂钩 – *生命周期挂钩使您能够在 Auto Scaling 组启动或终止实例时通过*暂停实例来执行自定义操作。当实例暂停时，它将保持等待状态，直到您使用 complete-lifecycle-action 命令或 CompleteLifecycleAction 操作完成生命周期操作，或者超时时段结束（默认为 1 小时）。

演示：为 Amazon EC2 Auto Scaling 创建扩缩策略

现在，讲师可能会选择演示如何为 Amazon EC2 Auto Scaling 创建目标跟踪和步进扩缩策略。

第 2 节要点

aws academy

- 弹性基础设施可以随着容量需求的变化而扩展和收缩
- Amazon EC2 Auto Scaling 会根据您定义的策略、计划和运行状况检查自动添加或删除 EC2 实例
- Amazon EC2 Auto Scaling 提供了多种扩缩选项，以最好地满足您的应用程序需求
- 在配置 Auto Scaling 组时，您可以指定 EC2 实例类型及其使用的定价模型组合

本模块中这节内容的要点包括：

- 弹性基础设施可以随着容量需求的变化而扩展和收缩
- Amazon EC2 Auto Scaling 会根据您定义的策略、计划和运行状况检查自动添加或删除 EC2 实例
- Amazon EC2 Auto Scaling 提供了多种扩缩选项，以最好地满足您的应用程序需求
- 在配置 Auto Scaling 组时，您可以指定 EC2 实例类型及其使用的定价模型组合

模块 9：实施弹性、高可用性和监控

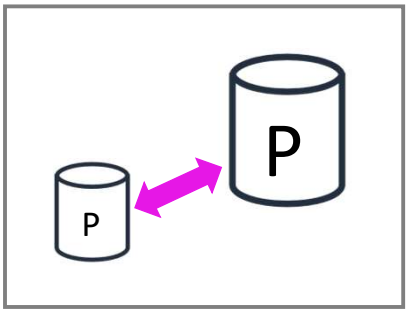
第 3 节：扩缩数据库

介绍第 3 节：扩缩数据库。

在上一节中，您学习了如何扩缩 EC2 实例。除此之外，您还可以扩缩数据库实例。在本节中，您将学习如何扩缩关系数据库和非关系数据库。

使用 Amazon RDS 进行纵向扩缩：按钮式扩缩

- 纵向扩展或缩减数据库实例
- 从 micro 到 24xlarge 以及介于两者之间的所有实例
- 以最短的停机时间纵向扩缩

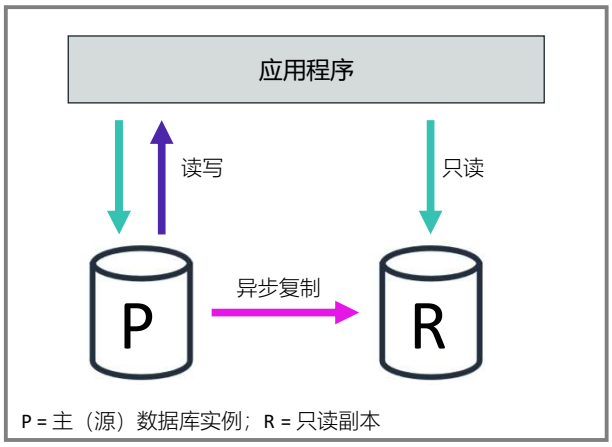


首先，考虑扩缩关系数据库。作为一项托管服务，Amazon Relational Database Service (Amazon RDS) 可以扩缩您的关系数据库，使其能够满足应用程序不断增长的需求。

您可以通过更改 Amazon RDS 实例类来纵向扩缩 Amazon RDS 数据库 (DB) 实例。如果您决定扩展或缩减数据库实例的可用计算资源，注意在修改数据库实例类时，数据库将暂时不可用。这段不可用的时间通常只持续几分钟。除非您指定应立即应用修改，否则它仅在数据库实例维护期间发生。

当您扩展或缩减数据库实例时，存储大小保持不变，且不受更改影响。您可以单独修改数据库实例以增加分配的存储空间或通过更改存储类型来提高性能，例如，从通用固态硬盘 (SSD) 更改为预置的每秒输入/输出操作数 (IOPS) SSD。您还可以使用 Amazon RDS Storage Autoscaling 来自动扩展存储容量，以适应不断增长的数据库工作负载，而不是手动预置存储。

使用 Amazon RDS 进行横向扩缩：只读副本



- 横向扩缩以处理多读少写工作负载
- 最多 5 个只读副本，最多 15 个 Aurora 副本
- 复制是异步的
- 适用于 Amazon RDS for MySQL、MariaDB、 PostgreSQL 和 Oracle

除了纵向扩缩，您还可以横向扩缩数据库。Amazon RDS 使用 MariaDB、MySQL、Oracle 和 PostgreSQL 数据库引擎的内置复制功能，从源数据库实例创建一个特殊类型的数据库实例（称为*只读副本*）。对源数据库实例的更新将异步复制到只读副本。您可以将应用程序发出的读取查询路由到只读副本，以减轻源数据库实例上的负载。

为了提高多读少写数据库工作负载的性能，可以使用只读副本来横向扩缩源数据库实例。您可以为给定源数据库实例创建一个或多个副本（最多 5 个），利用多个数据副本满足大量应用程序读取流量需求，从而增加总读取吞吐量。

如果发生灾难，您可以将只读副本提升为独立数据库实例来提高数据库的可用性。

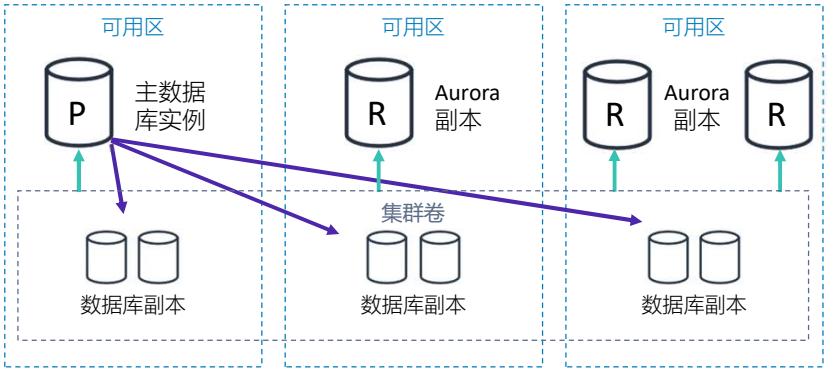
有关只读副本的更多信息，请参阅使用只读副本。

有关如何扩展 RDS 数据库实例的更多信息，请参阅以下资源：

- 修改 Amazon RDS 数据库实例（AWS 文档）
- Scaling Your Amazon RDS Instance Vertically and Horizontally AWS 数据库博客文章

使用 Amazon Aurora 进行扩缩

每个 Aurora 数据库集群最多可包含 15 个 Aurora 副本



Amazon Aurora 通过采用专门为数据库工作负载构建的 SSD 支持的虚拟化存储层，进一步扩展了只读副本的好处。Amazon Aurora 是针对云构建的兼容 MySQL 和 PostgreSQL 的关系数据库引擎。Amazon RDS 可用来管理您的 Amazon Aurora 数据库，处理预置、修补、备份、恢复、故障检测和修复等任务。Amazon Aurora 采用一种具有容错能力并能自我修复的分布式存储系统，这一系统可以将每个数据库实例自动扩展到高达 64TB。 它提供高性能和可用性、时间点恢复、向 Amazon Simple Storage Service (Amazon S3) 的持续备份，以及跨三个可用区 (AZ) 的复制。

Amazon Aurora *数据库集群*由一个或多个数据库实例以及一个管理这些数据库实例数据的集群卷组成。

Aurora 数据库集群由两类数据库实例组成：

- **主数据库实例** – 支持读取和写入操作，并执行对集群卷的所有数据修改。每个 Aurora 数据库集群都有一个主数据库实例。
- **Aurora 副本** – 连接到与主数据库实例相同的存储卷，并且仅支持读取操作。除了主数据库实例外，每个 Aurora 数据库集群最多可包含 15 个 Aurora 副本。

您可以选择数据库实例类大小并添加 Aurora 副本以增加读取吞吐量。如果您的工作负载发生变化，您可以修改数据库实例类大小并更改 Aurora 副本的数量。此模型适合可预测的数据库工作负载，因为您可以根据预期工作负载手动调整容量。

Amazon Aurora Serverless

但是，在某些环境中，工作负载可能是间歇性和不可预测的。也许存在可能仅持续几分钟或几小时的大量工作负载的时间段以及少量活动或甚至无活动的长时间段。例如，一些进行间歇性销售活动的零售网站、根据需要生成报告的报告数据库、开发和测试环境，以及具有不确定需求的新应用程序。在这些情况及很多其他情况下，可能很难在正确的时间配置正确的容量。在您为未使用的容量付费时，这还可能会产生更高的成本。

Aurora Serverless 是 Amazon Aurora 的一种按需自动扩缩配置。Aurora Serverless 使您的数据库能够根据应用程序的需求自动启动、关闭以及扩展或缩减容量。它让您可以在云中运行数据库，而无需管理任何数据库实例。Aurora Serverless 可用于不频繁、间歇性或不可预测的工作负载。

您可以创建数据库终端节点，而无需指定数据库实例类大小。您可以指定 Aurora 容量单位 (ACU)。每个 ACU 都是处理和内存容量的组合。数据库存储自动从 10 吉字节 (GiB) 扩展到 64 兆字节 (TiB)，这与标准 Aurora 数据库集群中的存储相同。数据库终端节点连接到代理队列，后者将工作负载路由到资源队列。Aurora Serverless 根据最小和最大容量规格自动扩缩资源。

您需要在数据库处于活动状态期间按照每秒使用的数据库容量付费，并在标准配置和无服务器配置之间迁移。您需要为使用的 ACU 数量付费。

有关 Aurora Serverless 的更多信息，请参阅 Aurora Serverless 的工作原理。

横向扩缩：数据库分片

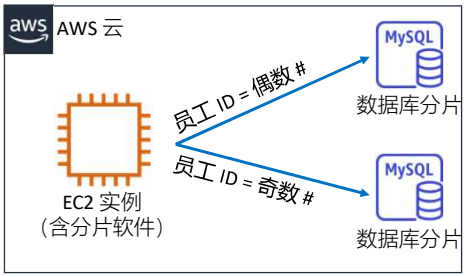
如果没有分片，所有数据都将驻留在一个分区中。

- 示例：一个数据库中的员工 ID

通过分片，数据将被拆分为大块 （分片）。

- 示例：偶数员工 ID 在一个数据库中，奇数员工 ID 在另一数据库中

在许多情况下，分片可提高写入性能。



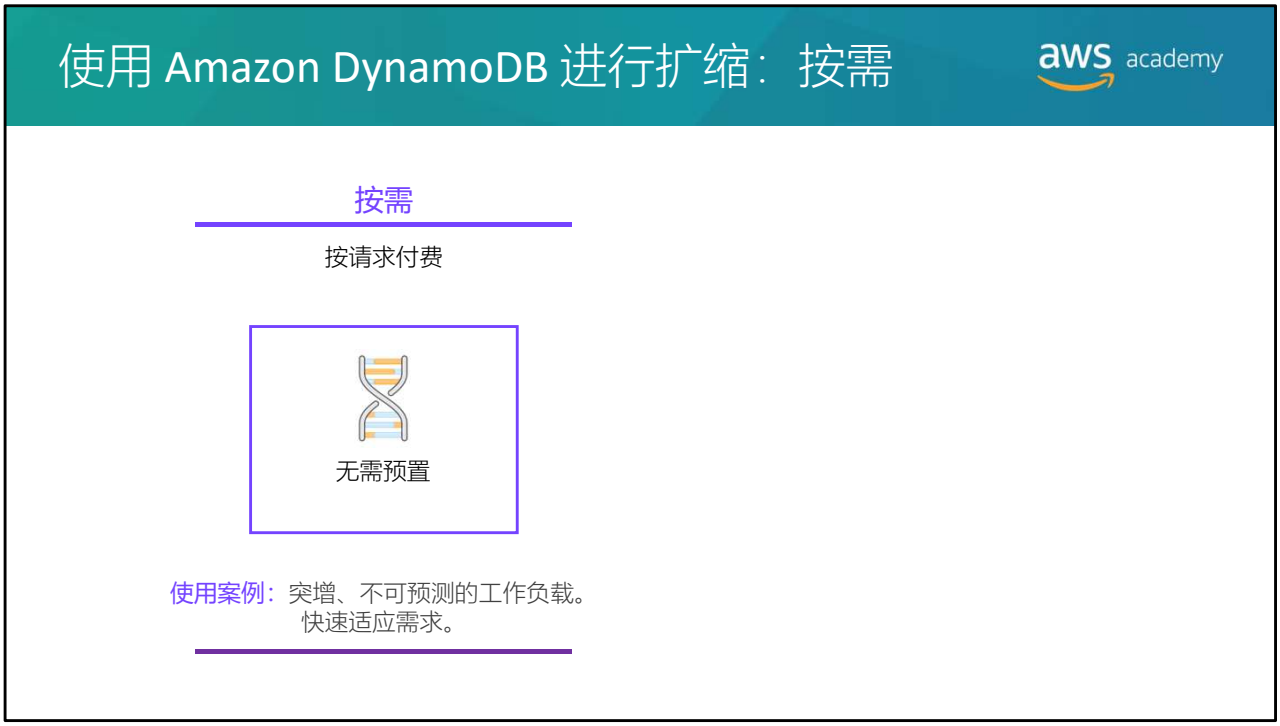
分片，也称为*横向分区*，是关系数据库中常用的扩展方法，可提高写入性能。

分片是一种技术，它将数据拆分成较小的子集，并将它们分布在多个物理上分离的数据库服务器上。每台服务器均称为数据库*分片*。数据库分片通常具有相同类型的硬件、数据库引擎和数据结构，可产生相似的性能水平。但是，它们彼此互不知晓，这是将分片与其他扩展方法（例如数据库集群或复制）区分开来的关键特征。

分片数据库架构提供了可扩缩性和容错能力。通过分片，可以根据需要在多个数据库服务器之间拆分数据。此外，如果一个数据库分片存在硬件问题或经历故障转移，其他分片将不会受到影响，因为单点故障或减速在物理上是隔离的。这种方法的缺点是，由于数据分布在多个分片，必须专门设计数据映射和路由逻辑，以便从多个分片读取或联接数据。与非分片数据库相比，这些类型的查询可能会产生更高的延迟。

有关使用 Amazon RDS 进行分片的更多信息，请阅读这篇 AWS 数据库博客文章。

使用 Amazon DynamoDB 进行扩缩：按需

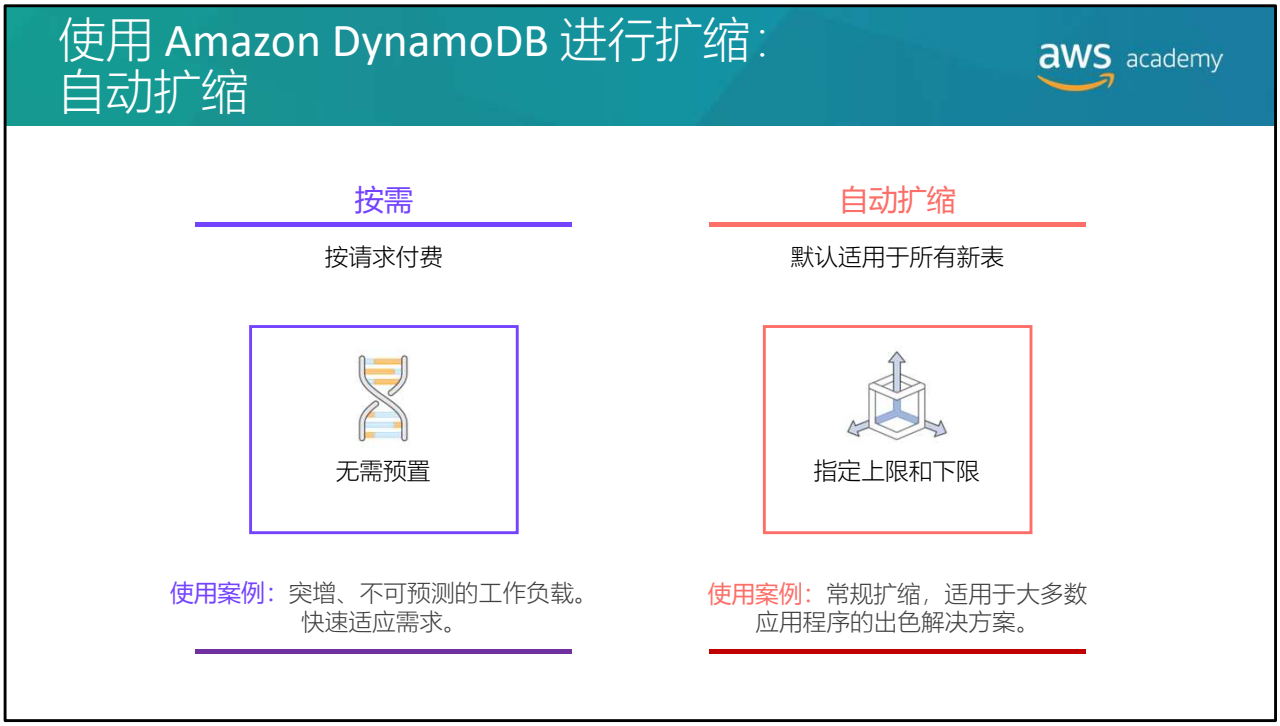


对于需要非关系数据库的不可预测的工作负载，Amazon DynamoDB On-Demand 是 DynamoDB 的灵活计费选项。它可以每秒处理数千个请求，而无需进行容量规划。它采用按请求付费的定价模式，而不是预置的定价模式。

DynamoDB On-Demand 可以观察到流量水平的任何增长或扩展。如果流量水平达到新峰值，DynamoDB 会快速调整以适应工作负载要求。如果您的工作负载难以预测，或在短时间内突增，该功能非常有用。

您可以将表从预置容量更改为按需容量，每天可以更改一次。您可以根据需要随时将按需容量更改为预置容量。

要了解有关 Amazon DynamoDB On-Demand 的更多信息，请阅读这篇 AWS 新闻博客文章。

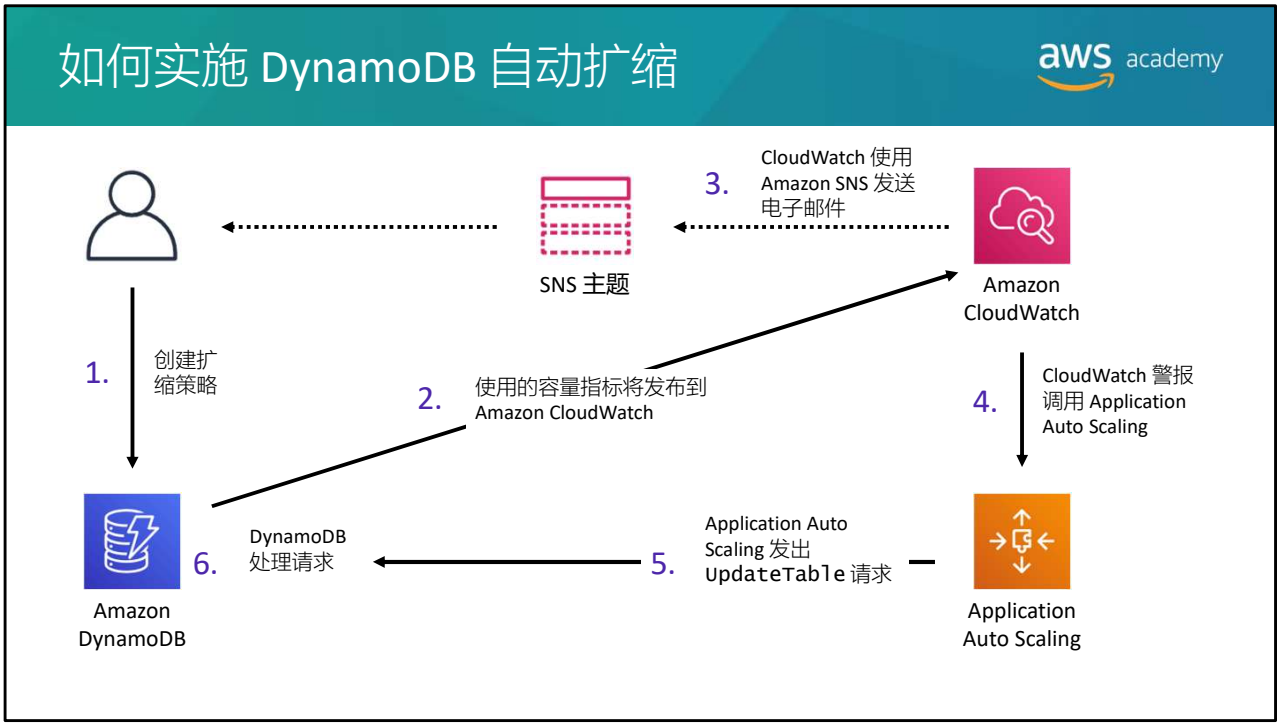


Amazon DynamoDB 还具有一项默认启用的名为 *Auto Scaling* 的功能。Amazon DynamoDB Auto Scaling 可以根据动态变化的请求量自动调整读取和写入吞吐容量，而无需停机。借助 DynamoDB Auto Scaling，您只需设置所需的吞吐量利用率目标以及最小和最大限制 – DynamoDB Auto Scaling 会完成其余工作。

DynamoDB Auto Scaling 与 Amazon CloudWatch 配合使用，可持续监控实际吞吐量消耗。当实际利用率偏离目标时，它会自动扩展或缩减容量。

除了您已为 DynamoDB 和 CloudWatch 警报支付的费用外，使用 DynamoDB Auto Scaling 不会产生任何额外费用。

有关 DynamoDB Auto Scaling 的更多信息，请参阅使用 DynamoDB Auto Scaling 自动管理吞吐容量。



Amazon DynamoDB Auto Scaling 使用 *Application Auto Scaling* 服务代您动态调整预置的吞吐容量，以响应实际的流量模式。此服务使表或全局二级索引 (GSI) 能够增加预置的读取和写入容量，从而不受限制地应对流量的突增。如果工作负载减少，Application Auto Scaling 会降低吞吐量，这样您就无需为未使用的预置容量付费。

要实施 DynamoDB Auto Scaling，请执行以下步骤：

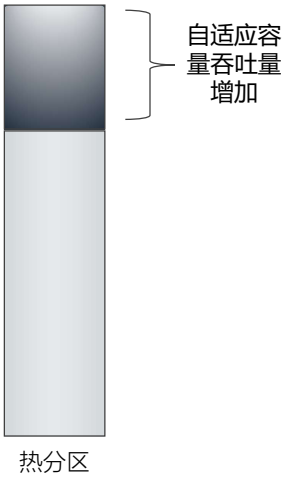
1. 为 DynamoDB 表或 GSI *创建*扩缩策略 。扩缩策略可指定要扩缩读取容量还是写入容量（或二者），并为表或索引指定最小的和最大的预置容量单位设置。
2. DynamoDB 将使用的容量指标发布到 Amazon CloudWatch。
3. 如果表使用的容量在特定时段内超出目标利用率（或低于目标利用率），则 Amazon CloudWatch 会触发警报。您可以使用 Amazon Simple Notification Service (Amazon SNS) 在 Amazon CloudWatch 控制台中查看警报并接收通知。
4. CloudWatch 警报调用 Application Auto Scaling 来评估您的扩缩策略。
5. Application Auto Scaling 向 DynamoDB 发出 *UpdateTable* 请求以调整表的预置吞吐量。
6. DynamoDB 将处理 *UpdateTable* 请求，动态增加（或减少）表的预置吞吐容量，使它接近目标利用率。

有关如何实施 DynamoDB Auto Scaling 的更多信息，请参阅使用 DynamoDB Auto Scaling 自动管理吞吐容量。

扩缩吞吐容量：DynamoDB 自适应容量

- 无限制启用对热分区的读写操作
- 自动增加分区的吞吐量以接收更多流量*
- 针对每个 DynamoDB 表自动启用

*流量不能超过表的总预置容量或分区的最大容量。



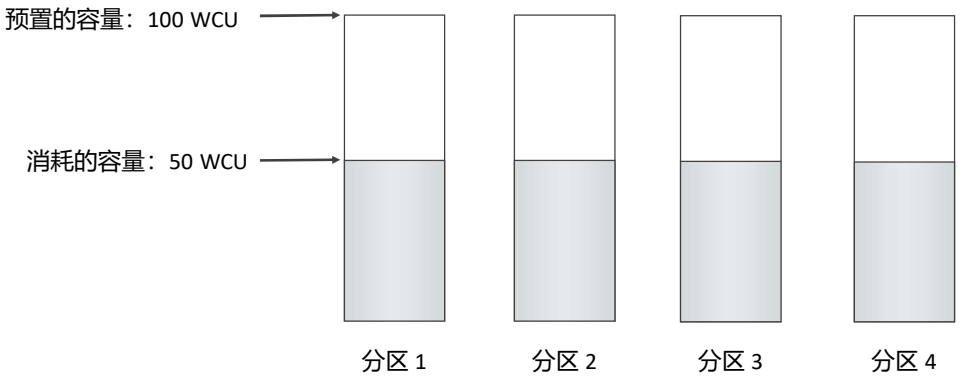
读写活动有时无法在各个分区之间平均分配。当数据访问不平衡时，某个分区可以接收高于其他分区的读取和写入流量，这称为*热分区*。在极端情况下，如果单个分区接收到超过 3000 个读取容量单位 (RCU) 或 1000 个写入容量单位 (WCU)，则会发生限制。

为了更好地适应不均匀的访问模式，DynamoDB 自适应容量让您的应用程序可以不断对热分区进行读写而不受限制。但是，流量不能超过表的总预置容量或分区的最大容量。自适应容量的工作原理是，自动增加分区的吞吐容量来接收更多流量。

系统会自动为每个 DynamoDB 表启用自适应容量，因此您无需显式启用或禁用它。

自适应容量示例 (1/3)

具有自适应容量的示例表
预置的总容量 = 400 WCU
消耗的总容量 = 200 WCU

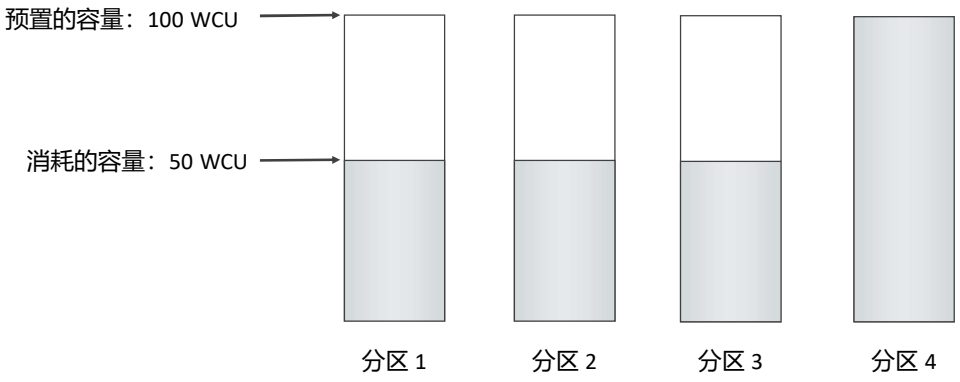


上图展示了自适应容量的工作原理。

示例表预置了 400 个 WCU，这些容量单位均匀分布在 4 个分区中，每个分区最多支持每秒 100 个 WCU。

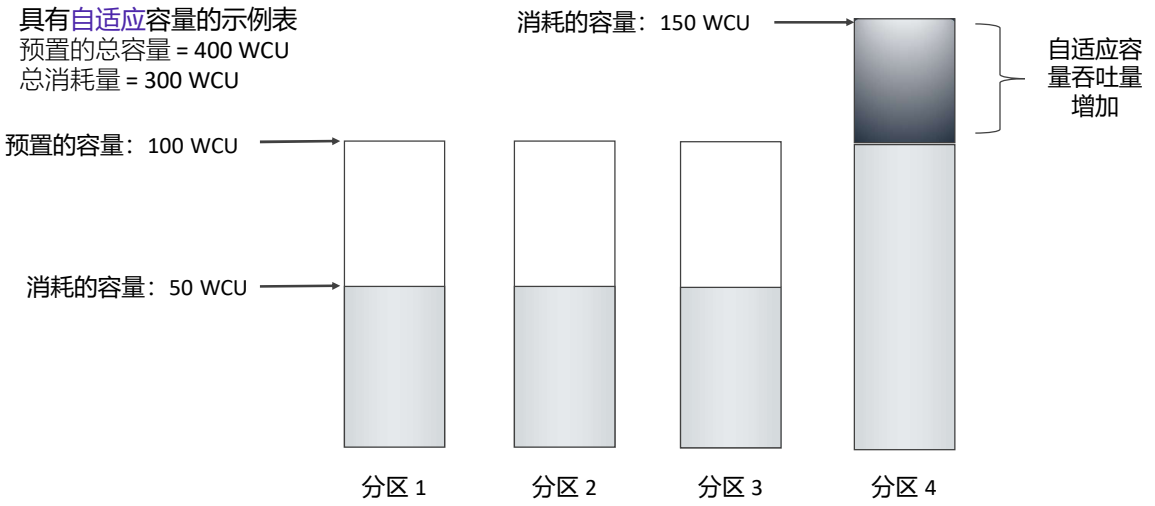
自适应容量示例 (2/3)

具有自适应容量的示例表
预置的总容量 = 400 WCU
消耗的总容量 = 250 WCU



分区 1、2 和 3 每秒接收的写入流量为 50 个 WCU。分区 4 每秒接收 150 个 WCU。此热分区可以在接收写入流量的同时仍具有未利用的突增容量；但是，它最终会限制每秒超过 100 个 WCU 的流量。

自适应容量示例 (3/3)



DynamoDB 自适应容量通过增加分区 4 的容量来做出响应，因此分区 4 可以承受每秒 150 个 WCU 的更高工作负载，而不会受到限制。

有关 DynamoDB 自适应容量的更多信息，请参阅了解 DynamoDB 自适应容量。

自适应容量不能修复热键和热分区

分区键值	统一性
用户 ID，应用程序有许多用户	好
状态代码，只有几个可用的状态代码	差
项目创建日期，四舍五入到最近的时间段（例如，日、小时或分钟）	差
设备 ID，每个设备以相对相似的间隔访问数据设备	好
设备 ID，即使跟踪许多设备，其中一台也比所有其他设备更常用	差

表的主键的分区键部分决定了存储表数据的逻辑分区。这反过来会影响底层物理分区。表的预置 I/O 容量在这些物理分区之间平均分配。因此，如果分区键设计无法均匀分发 I/O 请求，则可能会创建热分区，从而导致预置的 I/O 容量使用受到限制且效率低下。

表的预置吞吐量的最佳使用情况不仅取决于各个项目的工作负载模式，还取决于分区键设计。这既不意味着您必须访问所有分区键值才能达到高效的吞吐量级别，也不意味着访问的分区键值的百分比必须很高。而是*意味着工作负载访问的分区键值越不同，这些请求在分区空间中的分布就越广*。一般来说，随着访问的分区键值占分区键值总数的比率增大，您使用预置吞吐量的效率就越高。

该表显示了一些常见分区键 schema 的预置吞吐效率对比。

有关设计分区键的更多信息，请参阅设计分区键以均匀分发工作负载。

第 3 节要点

- 您可以使用按钮式扩缩来纵向扩缩 RDS 数据库实例的计算容量
- 您可以使用只读副本或分区来横向扩缩 RDS 数据库实例
- 通过 Amazon Aurora，您可以选择数据库实例类的大小和 Aurora 副本的数量（最多 15 个）
- Aurora Serverless 根据最小和最大容量规格自动扩缩资源
- Amazon DynamoDB On-Demand 提供按请求付费的定价模式
- DynamoDB Auto Scaling 使用 Amazon Application Auto Scaling 动态调整预置的吞吐容量
- DynamoDB 自适应容量的工作方式是自动增加分区的吞吐容量，从而接收更多的流量

本模块中这节内容的要点包括：

- 您可以使用按钮式扩缩来纵向扩缩 RDS 数据库实例的计算容量
- 您可以使用只读副本或分区来横向扩缩 RDS 数据库实例
- 通过 Amazon Aurora，您可以选择数据库实例类的大小和 Aurora 副本的数量（最多 15 个）
- Aurora Serverless 根据最小和最大容量规格自动扩缩资源
- Amazon DynamoDB On-Demand 提供按请求付费的定价模式
- DynamoDB Auto Scaling 使用 Amazon Application Auto Scaling 动态调整预置的吞吐容量
- DynamoDB 自适应容量的工作方式是自动增加分区的吞吐容量，从而接收更多的流量

模块 9：实施弹性、高可用性和监控

第 4 节：设计高度可用的环境

介绍第 4 节：设计高度可用的环境。

高度可用的系统

- 可以承受某种程度的降级，同时保持可用
- 最大限度地减少停机时间
- 需要极少的人为干预
- 在可接受的性能下降时间内从故障中恢复或转移到辅助源

正常运行时间百分比	每年最长停机时间	每天等效停机时间
90%	36.5 天	2.4 小时
99%	3.65 天	14 分钟
99.9%	8.76 小时	86 秒
99.99%	52.6 分钟	8.6 秒
99.999%	5.25 分钟	0.86 秒

最佳实践是在设计和构建解决方案时，避免单点故障。要遵循此最佳实践，您希望将架构设计为高度可用。

高度可用的*系统可以承受某种程度的降级，同时仍然保持可用。*在高度可用的系统中，停机时间尽可能减至最少，只需极少的人为干预。

高度可用的系统可在反应式架构中实现弹性。当受到负载（更多的服务请求）、攻击或组件故障压力时，弹性工作负载可以恢复。弹性工作负载可以在可接受的性能下降时间内从故障中恢复或转移到辅助源。

Elastic Load Balancing

一种托管的负载均衡服务，可在多个 EC2 实例、容器、IP 地址和 Lambda 函数之间分配传入的应用程序流量。

- 可以面向外部或面向内部
- 每个负载均衡器都会收到一个 DNS 名称
- 发现并响应运行状况不佳的实例

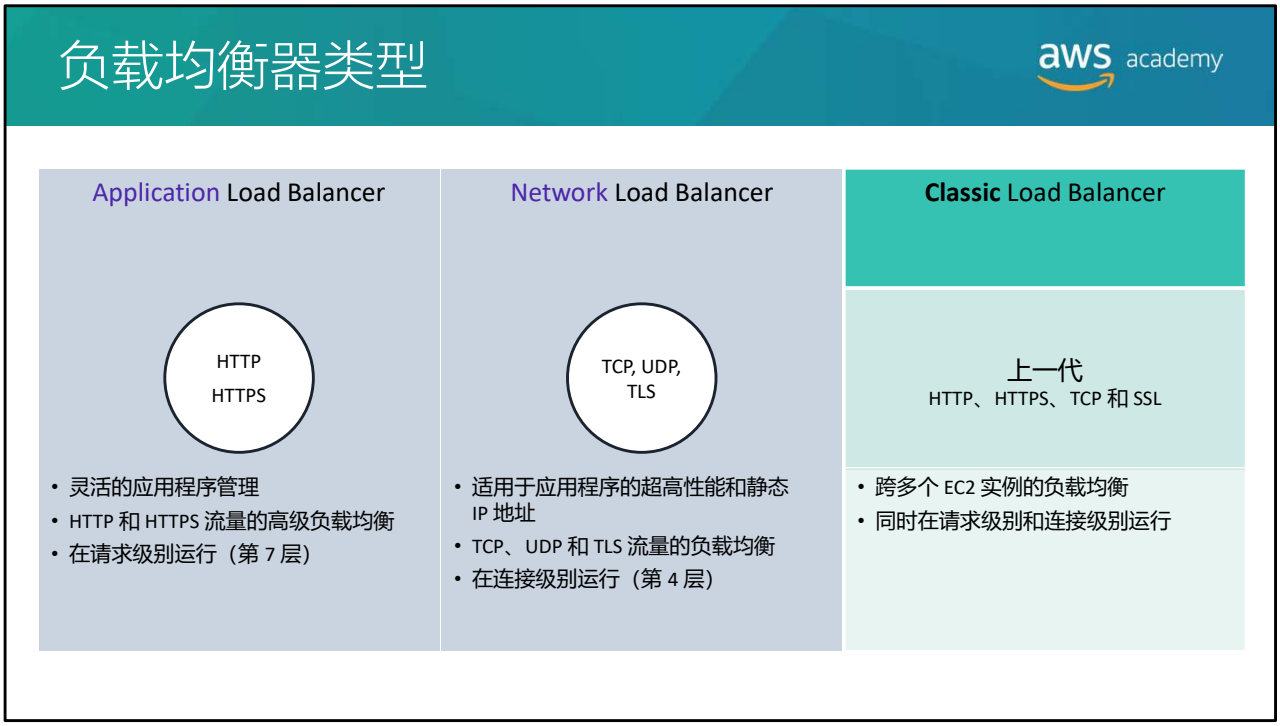
Elastic Load Balancing 是创建高度可用的架构的关键组件。

ELB 可在多个目标（如 EC2 实例、容器、IP 地址和 Lambda 函数）之间自动分发传入的应用程序流量。它可以在单个可用区中或跨多个可用区处理不同的应用程序流量负载。

ELB 提供三种负载均衡器。负载均衡器可以面向外部并分发入站公共流量。它们还可以面向内部并分发私有流量。

每种负载均衡器都会收到一个默认的域名系统 (DNS) 名称。

最后，ELB 自动将传入流量分发到多个可用区中的多个目标，并且仅将流量发送到正常运行的目标。为了查明 EC2 实例的可用性，负载均衡器会定期发送 ping、尝试连接或发送请求来测试 EC2 实例。这些测试称为*运行状况检查*。每个注册的 EC2 实例必须使用 HTTP 状态代码 200 来响应运行状况检查的目标，这样才能被负载均衡器视为运行状况良好。



ELB 提供三种负载均衡器，它们都具有高可用性、自动扩缩功能和强大的安全性，可让您的应用程序实现容错。

- Application Load Balancer *在应用程序级别（开放系统互连 (OSI) 模型第 7 层）运行。*它根据请求的内容将流量路由到目标 – EC2 实例、容器、IP 地址和 Lambda 函数。它非常适合 HTTP 和安全 HTTP (HTTPS) 流量的高级负载均衡。Application Load Balancer 提供面向现代化应用程序架构交付的高级请求路由，包括微服务和基于容器的应用程序。Application Load Balancer 通过确保始终使用最新的安全套接字层/传输层安全性 (SSL/TLS) 密码和协议，简化并提高应用程序的安全性。

- Network Load Balancer *在网络传输级别（OSI 模型第 4 层）运行，将连接路由到 EC2 实例、容器和 IP 地址等目标。*它是传输控制协议 (TCP) 和用户数据报协议 (UDP) 流量实现负载均衡的理想之选。Network Load Balancer 能够在保持超低延迟的同时，每秒处理数百万个请求。Network Load Balancer 针对处理突发和不稳定的网络流量模式进行了优化。

- Classic Load Balancer *可以在应用程序级别和网络传输级别运行，在多个 EC2 实例之间提供基本的负载均衡。*Classic Load Balancer 支持对使用 HTTP、HTTPS、TCP 和 SSL 的应用程序实施负载均衡。Classic Load Balancer 是一种较旧的实施方案。如果可能，AWS 建议您使用专用的 Application Load Balancer 或 Network Load Balancer。

使用 VPC 对等连接，您可以从另一 VPC 访问内部负载均衡器（包括 Classic Load Balancer、

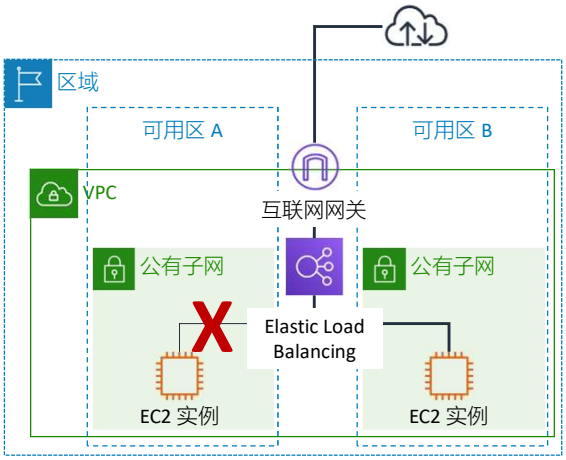
Application Load Balancer 和 Network Load Balancer）。VPC 对等连接可用于本地或跨账户 VPC 的区域内和区域间连接。

要详细了解三种负载均衡器之间的差异，请参阅 Elastic Load Balancing 功能页面上的*产品比较*。

实现高可用性

从每个 AWS 区域两个可用区开始。

如果一个可用区中的资源无法访问，应用程序不会出现故障。



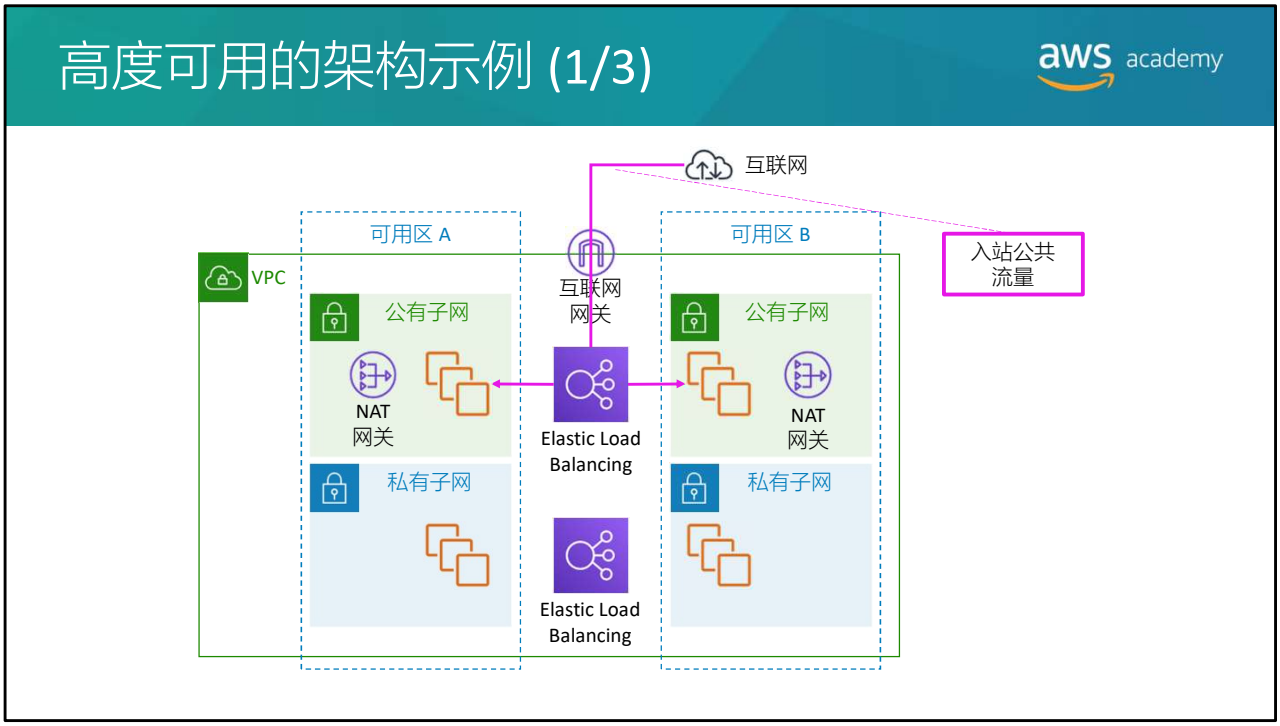
要构建高度可用的应用程序，最佳实践是在多个可用区中启动资源，然后使用负载均衡器在这些资源之间分配流量。在多个可用区中运行您的应用程序可以在数据中心发生故障时实现更高的可用性。

在此基本模式中，两台 Web 服务器在不同可用区中的 EC2 实例上运行。这些实例被置于 Elastic Load Balancing 负载均衡器之后，由该负载均衡器在实例之间分配流量。如果一台服务器不可用，负载均衡器将配置为停止将流量分配到运行状况不佳的实例，并开始将流量路由到运行状况良好的实例。这样，如果其中一个可用区中的数据中心出现故障，则应用程序仍然可用。

大多数应用程序可以设计为每个 AWS 区域支持两个可用区。但如果您使用的是仅支持主或辅助故障转移的数据源，那么使用多个可用区对应用程序而言可能并无益处。由于可用区在物理空间上是分散的，因此在一个 AWS 区域中的三个或更多可用区内复制资源并不会带来太多好处。

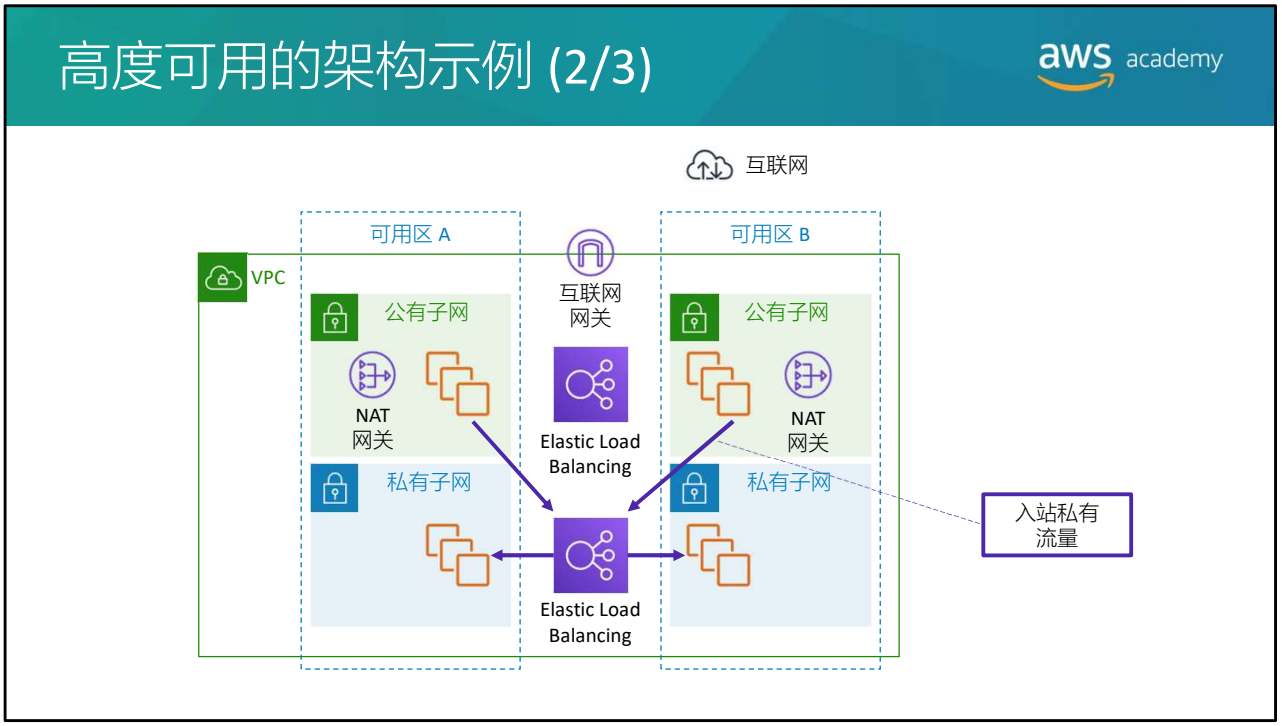
如果您使用大量 Amazon EC2 Spot 实例或使用超出主动或被动范围的数据源（例如 Amazon DynamoDB），那么使用两个以上的可用区可能会颇有助益。

高度可用的架构示例 (1/3)



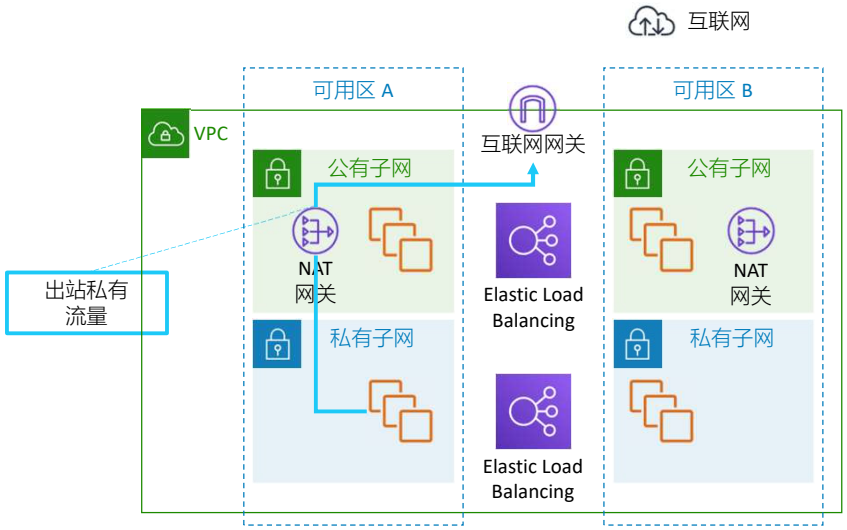
与您刚才考虑的架构一样，EC2 实例被置于负载均衡器之后，由该负载均衡器在实例之间分配入站公共流量。如果其中一台服务器不可用，负载均衡器将配置为停止向运行状况不佳的实例分配流量，然后开始将流量路由到运行状况良好的实例。

高度可用的架构示例 (2/3)



您可以在架构中加入第二个负载均衡器，将入站流量从公有子网中的实例路由到私有子网中的实例。

高度可用的架构示例 (3/3)

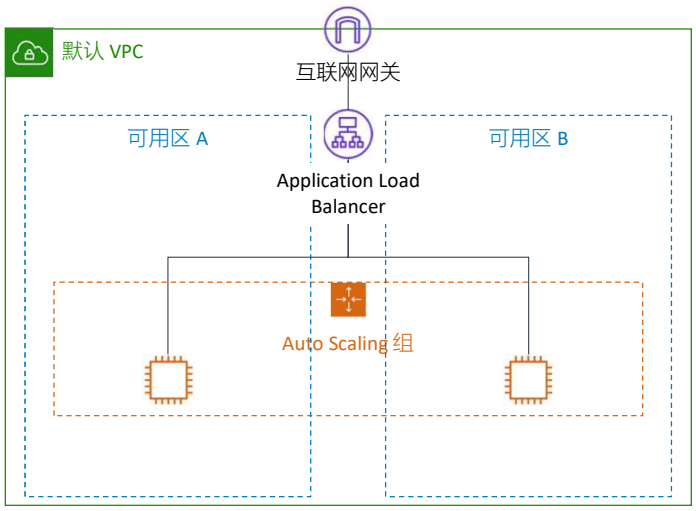


如果您在多个可用区中拥有资源并且它们共享一个 NAT 网关，则当该 NAT 网关的可用区不可用时，其他可用区中的资源将无法访问互联网。最佳实践是在两个可用区中使用 NAT 网关以确保高可用性。

演示：
创建高度可用的 Web 应用程序

现在，讲师可能会选择演示如何通过跨多个可用区在 Application Load Balancer 之后部署 Web 服务器来创建高度可用的 Web 应用程序。

演示架构



本演示展示了如何：

1. 创建并启动 EC2 Web 服务器
2. 通过 EC2 Web 服务器创建 Amazon 系统映像 (AMI)
3. 创建和配置 Application Load Balancer
4. 创建和配置 Auto Scaling 组并将其部署到两个可用区
5. 测试 Application Load Balancer

Amazon Route 53

Amazon Route 53 是一种高度可用且可扩展的云 DNS 服务。

- 将域名转换为 IP 地址
- 将用户请求连接到在 AWS 内部和外部运行的基础设施
- 可以配置为将流量路由到运行状况良好的终端节点，或监控应用程序及其终端节点的运行状况
- 提供域名注册
- 有多个路由选项

您还可以使用 Amazon Route 53 在网络架构中实施多区域高可用性和容错能力。

Amazon Route 53 是一种高度可用且可扩展的云 DNS 服务。该服务旨在提供一种可靠且经济高效的方式，以将用户路由到互联网应用程序。它将名称（如 *example.com*）转换为计算机相互连接所用的数字 IP 地址（如 *192.0.2.1*）。

Route 53 能有效地将用户请求连接到在 AWS 中运行的基础设施上，例如 EC2 实例、ELB 负载均衡器或 S3 存储桶。您也可以使用 Route 53 将用户路由到 AWS 之外的基础设施上。

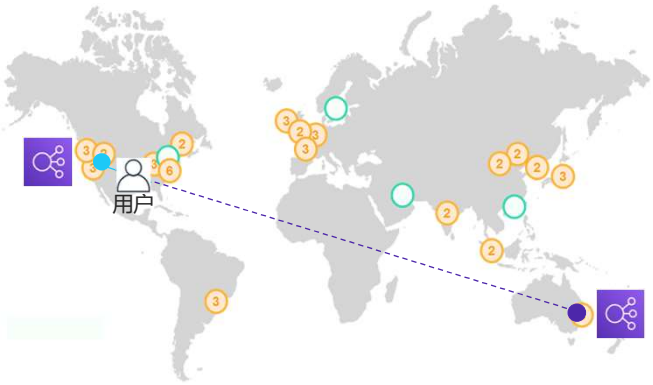
您可以使用 Route 53 配置 DNS 运行状况检查以将流量路由到运行状况良好的终端节点，或者独立监控应用程序及其终端节点的运行状况。

Route 53 还提供域名注册功能。您可以购买和管理域名（如 *example.com*），而 Amazon Route 53 将为您的域自动配置 DNS 设置。

Amazon Route 53 提供了各种路由选项，这些选项可与 DNS 故障转移相结合，以实现低延迟容错架构。有关 Amazon Route 53 路由选项的详细信息，请参阅选择路由策略。

Amazon Route 53 支持的路由

- 简单路由
- 加权轮询路由
- 基于延迟的路由
- 地理位置路由
- 地理位置临近度路由
- 故障转移路由
- 多值应答路由



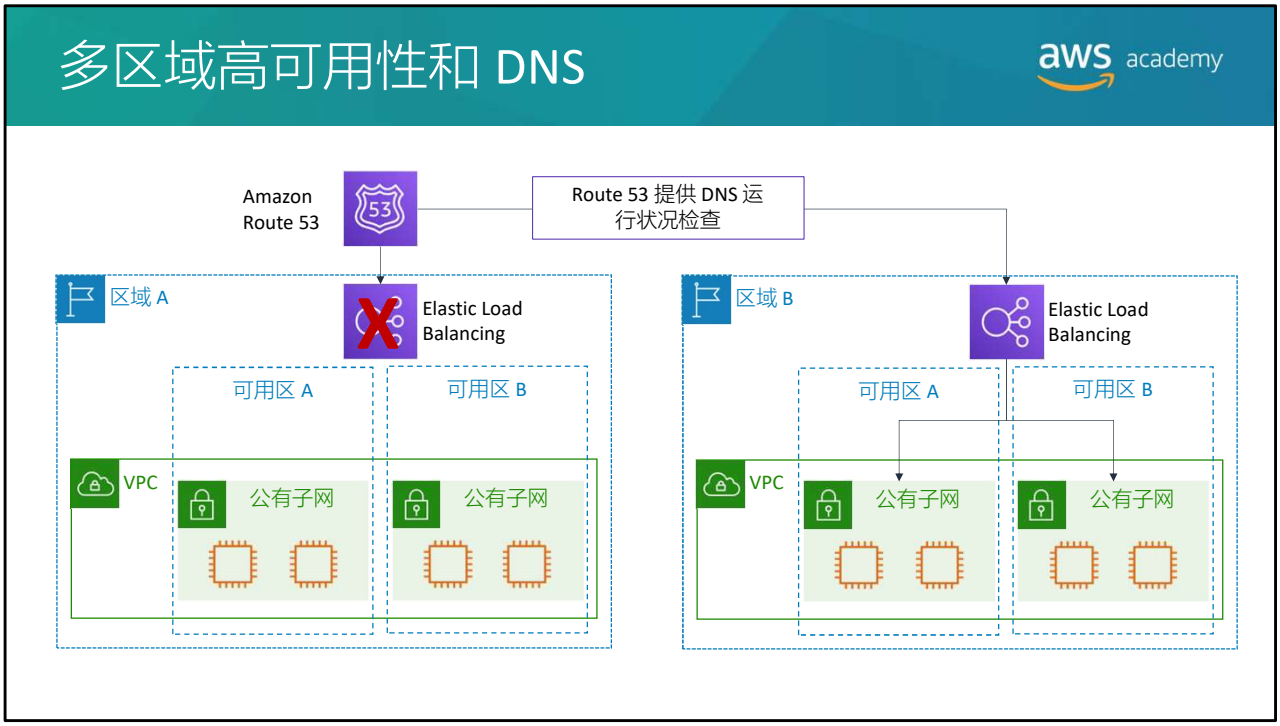
Amazon Route 53 支持多种类型的路由策略，这些策略可确定 Amazon Route 53 如何响应查询：

- *简单路由（轮询）* – 可在所有参与服务器之间尽可能均匀地分配请求数。
- *加权轮询路由* – 允许您为资源记录集分配权重，以便指定提供不同响应的频率。您可以使用此功能进行 A/B 测试，将一小部分流量发送到您已更改软件的服务器。例如，假设您向一个 DNS 名称关联了两个记录集：一个权重为 3，另一个权重为 1。在本例中，75% 的时间内 Amazon Route 53 将返回权重为 3 的记录集，25% 的时间内 Amazon Route 53 将返回权重为 1 的记录集。权重可以是 0 到 255 之间的任意数字。
- *延迟路由 (LBR)* – 如果您的资源位于多个 AWS 区域，并且您想要将流量路由到提供最佳延迟的区域，则可以使用该策略。延迟路由的工作原理是将您的客户路由到 AWS 终端节点（如 EC2 实例、弹性 IP 地址 或负载均衡器），以便根据运行应用程序的不同 AWS 区域的实际性能测量结果提供最快的体验。
- *地理位置路由* – 允许您根据用户的地理位置（DNS 查询的来源）选择提供流量的资源。使用地理位置路由时，您可以对您的内容进行本地化，并使用用户的语言显示您的部分或全部网站。您也可以使用地理位置路由将内容分配限制为仅分配至具有分配权限的位置。另一种可能的用途是以可预测、易于管理的方式在终端节点间均衡负载，以便每个用户位置一致地路由到同一终端节点。

- *地理位置临近度路由* – 如果您使用的是 Route 53 流量，则允许您根据用户与资源之间的物理距离来路由流量。您还可以通过指定正偏差或负偏差来为每个资源路由更多或更少的流量。 创建流量策略时，您可以为每个终端节点指定 AWS 区域（如果您正在使用 AWS 资源）或纬度和经度。

- *故障转移路由（DNS 故障转移）* – 如果您想配置主动-被动故障转移，则可以使用该策略。Route 53 可以帮助检测网站是否中断，并将用户重定向到应用程序正常运行的备用位置。启用此功能后，Route 53 运行状况检查代理将监控应用程序的每个站点或终端节点，以确定其可用性。您可以利用此功能来提高面向客户的应用程序的可用性。

- *多值应答路由* – 如果您想将流量大致随机路由到多个资源（如 Web 服务器），则使用此功能。您可以为每个资源创建一条多值应答记录。您还可以选择将 Route 53 运行状况检查与每条记录相关联。例如，假设您管理着一项使用 12 台 Web 服务器的 HTTP Web 服务，且每台 Web 服务器都有自己的 IP 地址。没有一台 Web 服务器可以处理所有流量。但如果您创建了十几条多值应答记录，则 Route 53 可响应最多具有 8 条运行状况良好记录的 DNS 查询，从而能够响应每个 DNS 查询。Route 53 可为不同的 DNS 解析程序提供不同的应答。如果解析程序缓存响应后 Web 服务器变得不可用，客户端软件可以尝试响应中的其他 IP 地址。



通过 *DNS 故障转移路由*， Route 53 可以帮助检测网站是否中断，并将用户重定向到应用程序正常运行的备用位置。启用此功能后，Route 53 运行状况检查代理将监控应用程序的每个站点或终端节点，以确定其可用性。您可以利用此功能来提高面向客户的应用程序的可用性。

演示：
Amazon Route 53

现在，讲师可能会选择播放视频，演示如何使用 Route 53 进行简单路由、故障转移路由和地理位置路由。

第 4 节要点

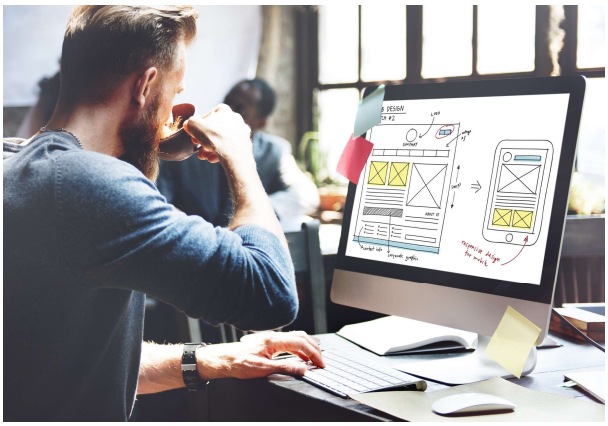
- 您可以将网络架构设计为高度可用，避免单点故障
- Route 53 提供了各种路由选项，可与 DNS 故障转移结合使用，以实现低延迟容错架构

本模块中这节内容的要点包括：

- 您可以将网络架构设计为高度可用，避免单点故障
- Route 53 提供了多种路由选项，这些选项可与 DNS 故障转移相结合，以实现多种低延迟容错架构

模块 9 – 指导实验：创建高度可用的环境

aws academy



现在，您要完成“模块 9 – 指导实验：创建高度可用的环境”。

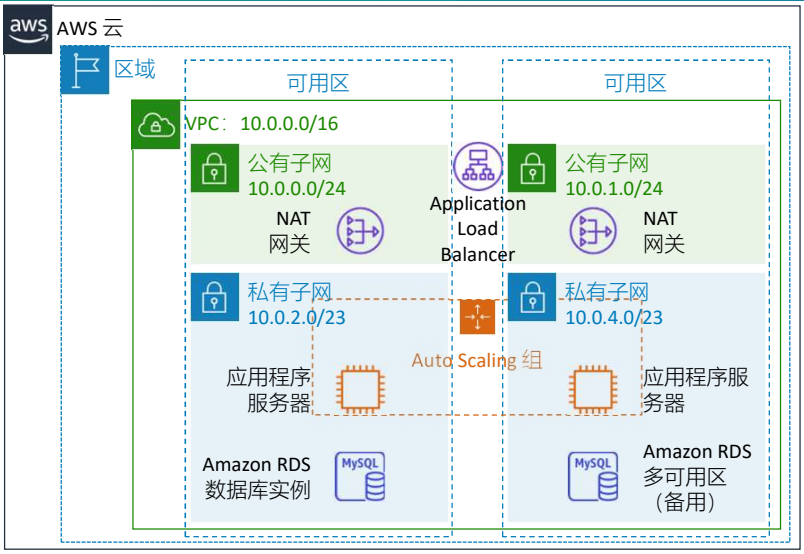
指导实验：任务

1. 检查提供的 VPC
2. 创建 Application Load Balancer
3. 创建 Auto Scaling 组
4. 测试应用程序是否实现了高可用性

在本指导实验中，您将完成以下任务：

1. 检查提供的 VPC
2. 创建 Application Load Balancer
3. 创建 Auto Scaling 组
4. 测试应用程序是否实现了高可用性

指导实验：最终产品

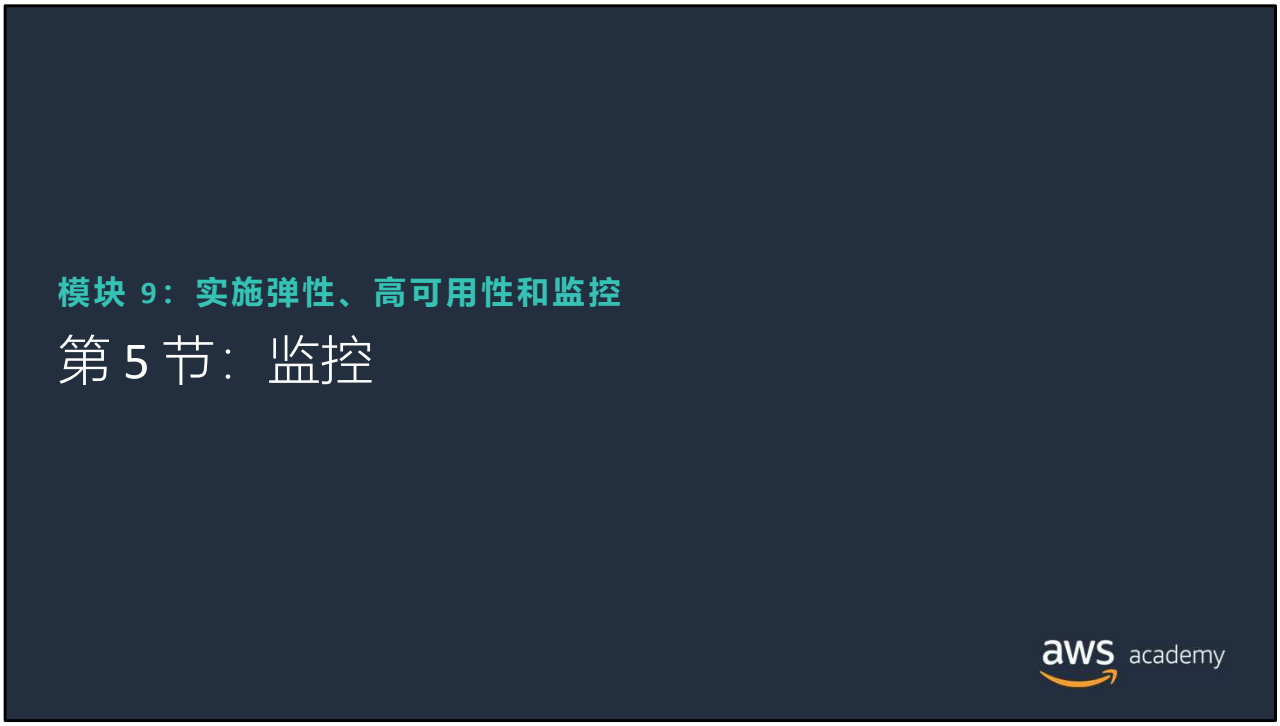


该图总结了您要在实验中构建的内容。

现在可以开始指导实验了。

指导实验总结：要点

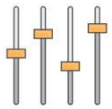
完成这个指导实验之后，讲师可能会带您讨论此指导实验的要点。



介绍第 5 节：监控。

监控使用情况、操作和性能

运行状况



资源利用率

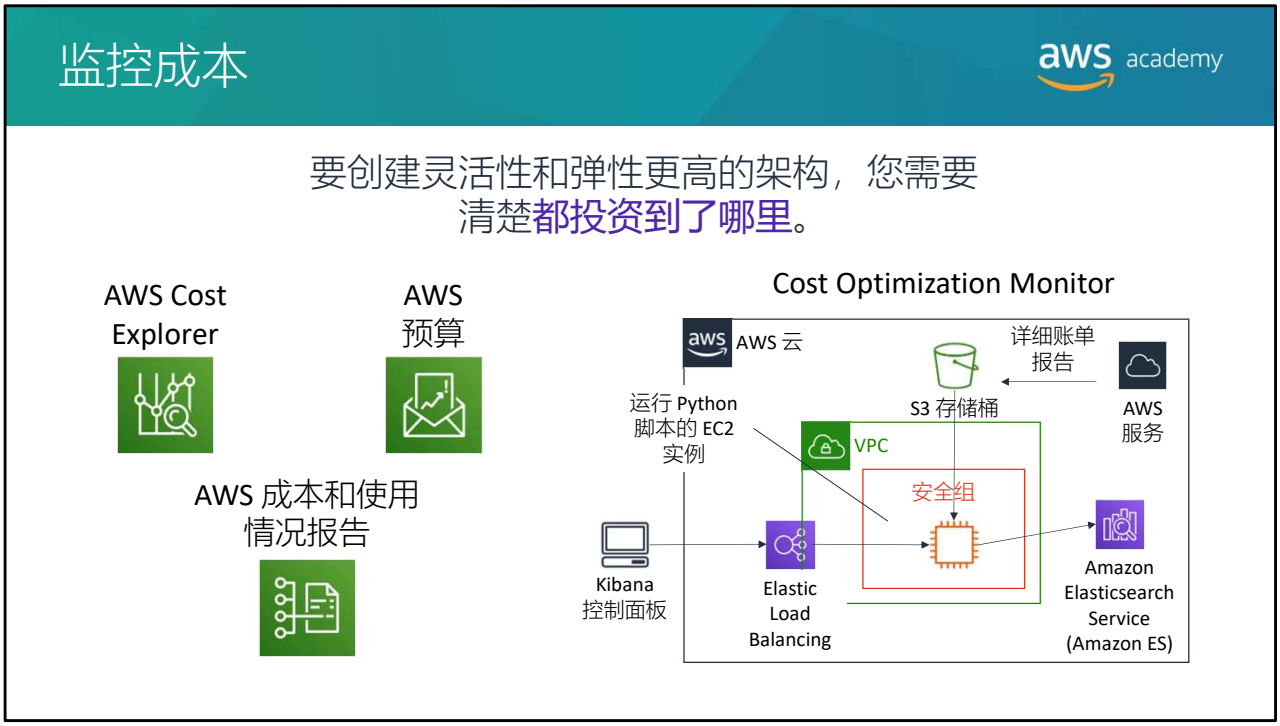
应用程序性能

安全审计

监控是反应式架构的重要组成部分。

监控可以帮助您：

- 跟踪资源的运行和执行情况
- 跟踪资源利用率和应用程序性能，以确保您的基础设施能够满足需求
- 确定为您的 AWS 资源设置哪些权限以实现所需的安全目标

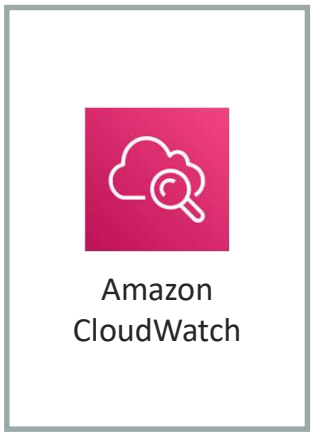


监控还有助于您了解和管理 AWS 基础设施的成本。AWS 提供多种监控和报告工具，其中包括：

- AWS Cost Explorer 可帮助您按天或按月直观地查看、理解和管理 AWS 成本和使用情况。您可以使用它查看过去长达 13 个月的数据，从而了解您在某段时间内使用 AWS 资源的模式。
- AWS 预算使您能够设置自定义预算，以便在成本或用量超过（或预计会超过）您的预算数量时向您发出警报。
- AWS 成本和使用情况报告包含最全面的一组 AWS 成本和使用情况数据，其中包括有关 AWS 服务、定价和预留的元数据。
- Cost Optimization Monitor 是一种解决方案架构，可自动处理详细的账单报告，以提供可在自定义控制面板中进行搜索、分析和实现可视化的精细指标。该解决方案可以让您深入了解服务使用情况和成本，您可以按周期、账户、资源或标签进行细分。

要了解有关如何监控 AWS 基础设施成本的更多信息，请参阅 AWS 成本管理服务。

Amazon CloudWatch



- 收集并跟踪资源和应用程序的指标
- 帮助您关联、可视化和分析指标和日志
- 支持您创建警报并检测异常行为
- 可以发送通知或更改您正在监控的资源

Amazon CloudWatch 是一种面向开发运维工程师、开发人员、站点可靠性工程师和 IT 经理的监控和可观测性服务。CloudWatch 为您提供相关数据和可行见解，以监控应用程序、响应系统范围的性能变化、优化资源利用率，并在统一视图中查看运营状况。

您可以使用 CloudWatch 收集和跟踪指标，这些指标是您用于衡量资源和应用程序的变量。您可以创建用于监控指标并发送通知的 CloudWatch 警报。此外，当超过阈值时，CloudWatch 可以自动更改您正在监控的资源。

例如，您可以监控 CPU 使用情况以及 EC2 实例的磁盘读取和写入情况。然后，您可以使用这些数据来确定是否应启动其他实例来处理增加的负载。您还可以使用这些数据停止未完全利用的实例以节省资金。

除了监控 AWS 随附的内置指标外，您还可以监控自己的自定义指标。您可以使用 CloudWatch 全面了解资源使用率、应用程序性能和运行状况。

有关 CloudWatch 的更多信息，请参阅什么是 Amazon CloudWatch?

CloudWatch 如何响应


指标


日志

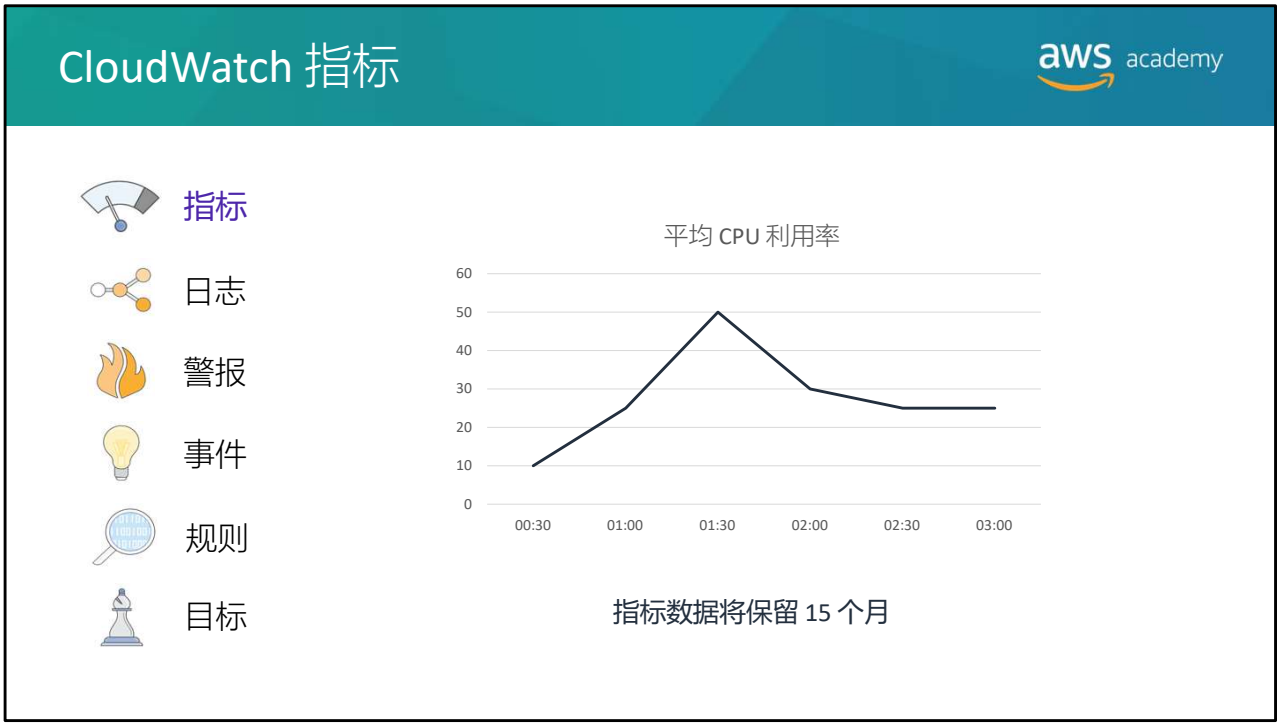

警报


事件


规则


目标

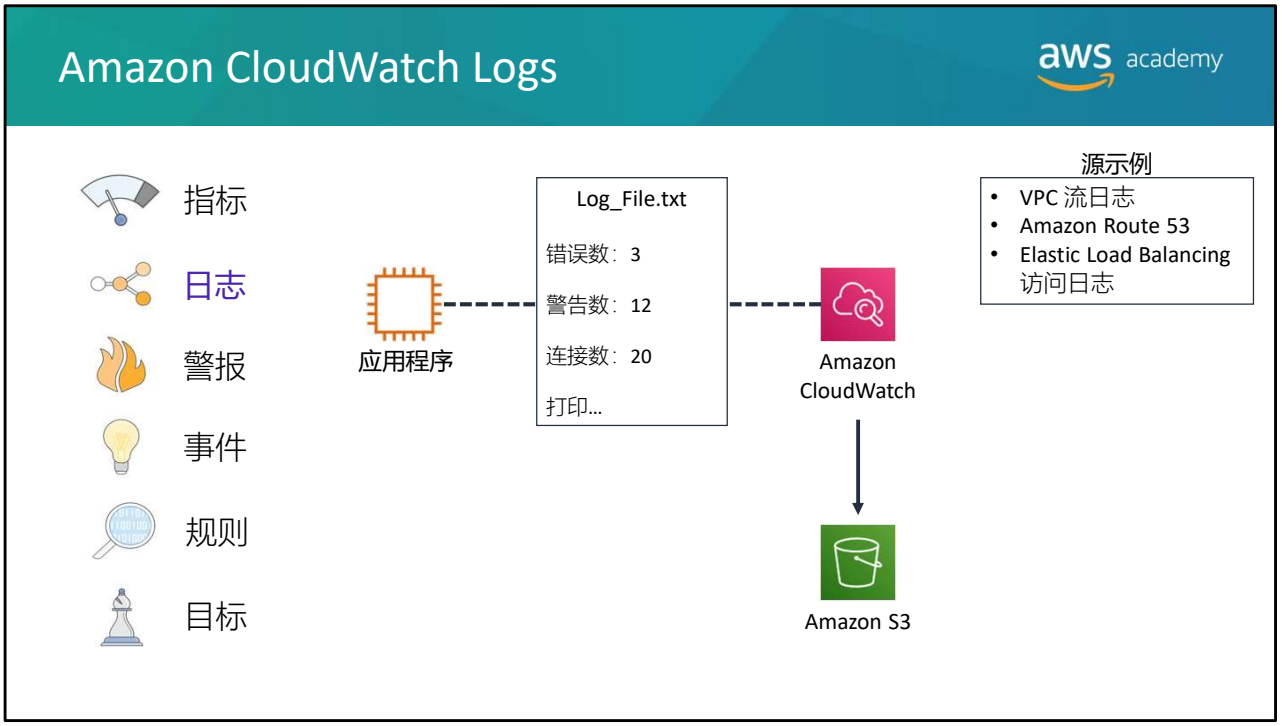
您可以使用多个 CloudWatch 组件来监控资源和应用程序，并响应事件。



*指标*是与系统性能有关的数据。默认情况下，许多 AWS 服务可提供资源指标，例如 EC2 实例、Amazon Elastic Block Store (Amazon EBS) 卷和 Amazon RDS 数据库实例。此外，您还可以为某些资源启用详细监控（例如 EC2 实例），或发布您自己的应用程序指标。CloudWatch 可以加载您账户中的所有指标（包括 AWS 资源指标和您提供的应用程序指标），用于搜索、绘图和发送警报。

指标数据的保留期限为 15 个月，这使您能够查看最新数据和历史数据。

有关指标的更多信息，请参阅使用 Amazon CloudWatch 指标。



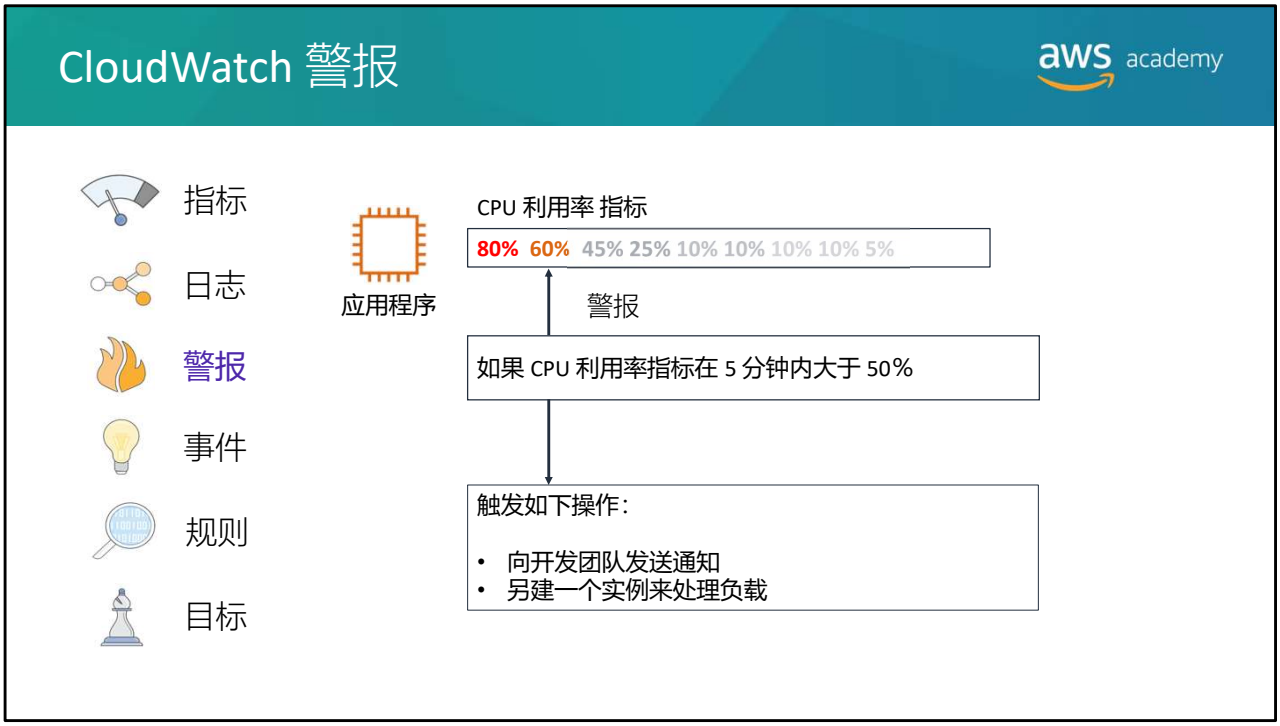
您可以使用 Amazon CloudWatch Logs 监控、存储和访问来自 EC2 实例、AWS CloudTrail、Route 53 和其他 AWS 服务等来源的日志文件。

例如，您可以使用 CloudWatch Logs 通过日志数据监控应用程序和系统。CloudWatch Logs 可以跟踪应用程序日志中发生的错误数。当错误率超过您指定的阈值时，它将发送通知。

此外，您可以使用 CloudWatch Logs Insights 在几秒钟内分析日志。它为您提供快速的交互式查询和可视化效果。您可以使用折线图或堆叠面积图直观呈现查询结果，并将这些查询添加到 CloudWatch 控制面板。

有关 CloudWatch Logs 的更多信息，请参阅以下资源：

- Amazon CloudWatch Logs
- Amazon CloudWatch Logs Insights



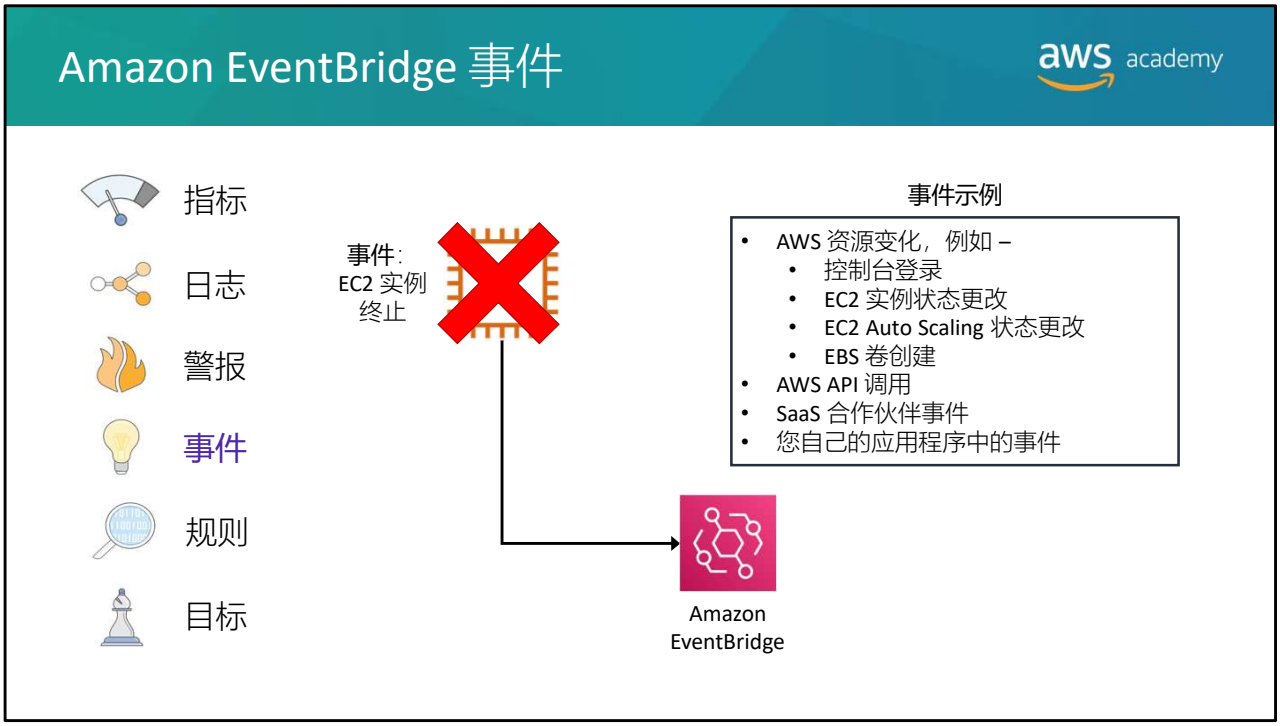
您可以使用警报代您自动发起操作。警报*会在指定时间段内监控单个指标*。它根据相对于某个阈值的指标值在一段时间执行一项或多项指定操作。操作指的是发送至 Amazon SNS 主题或 Auto Scaling 策略的通知。您还可以将警报添加到控制面板。

警报仅在出现持续的状态更改时才会调用操作。CloudWatch 警报不会仅因操作处于某个状态而调用它们。状态必须已改变并已维持了指定的若干个时间段。

在幻灯片所示的示例中，当 CPUUtilization 指标（也即当前正用于实例的已分配 EC2 计算单元的百分比）大于 50% 并持续 5 分钟时将触发警报。警报会触发操作，例如运行 Auto Scaling 策略或向开发团队发送通知。

未触发警报时也可以执行操作。

有关 CloudWatch 警报的更多信息，请参阅使用 Amazon CloudWatch 警报。

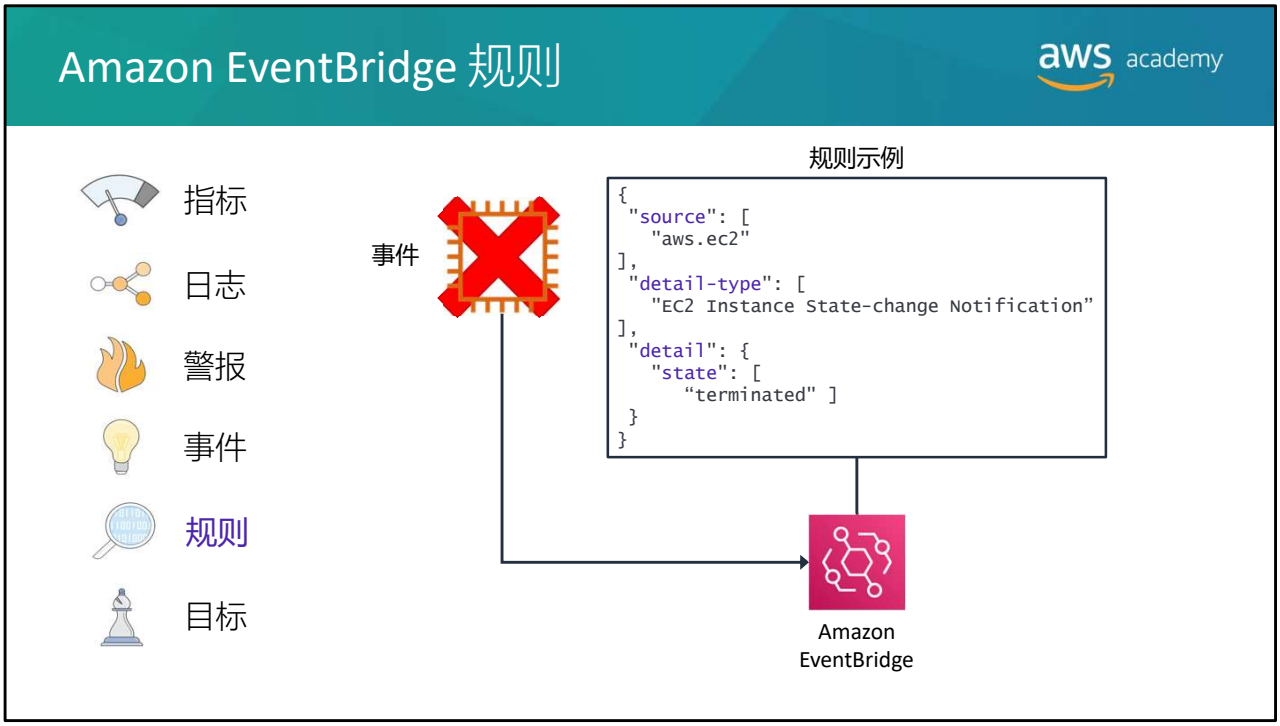


Amazon EventBridge（前身为 Amazon CloudWatch Events）可从您自己的应用程序、软件即服务 (SaaS) 应用程序和 AWS 服务中提取实时数据流。然后，它将数据路由到目标（如 AWS Lambda）。

事件*表示环境发生的变化。*这可以是 AWS 环境、SaaS 合作伙伴服务或应用程序，或您自己的自定义应用程序或服务之一。例如，Amazon EC2 会在 EC2 实例的状态从*等待中*更改为*运行中*时生成事件，Amazon EC2 Auto Scaling 会在启动或终止实例时生成事件。 AWS CloudTrail 在您进行 API 调用时发布事件。您还可以设置定期生成的计划事件。

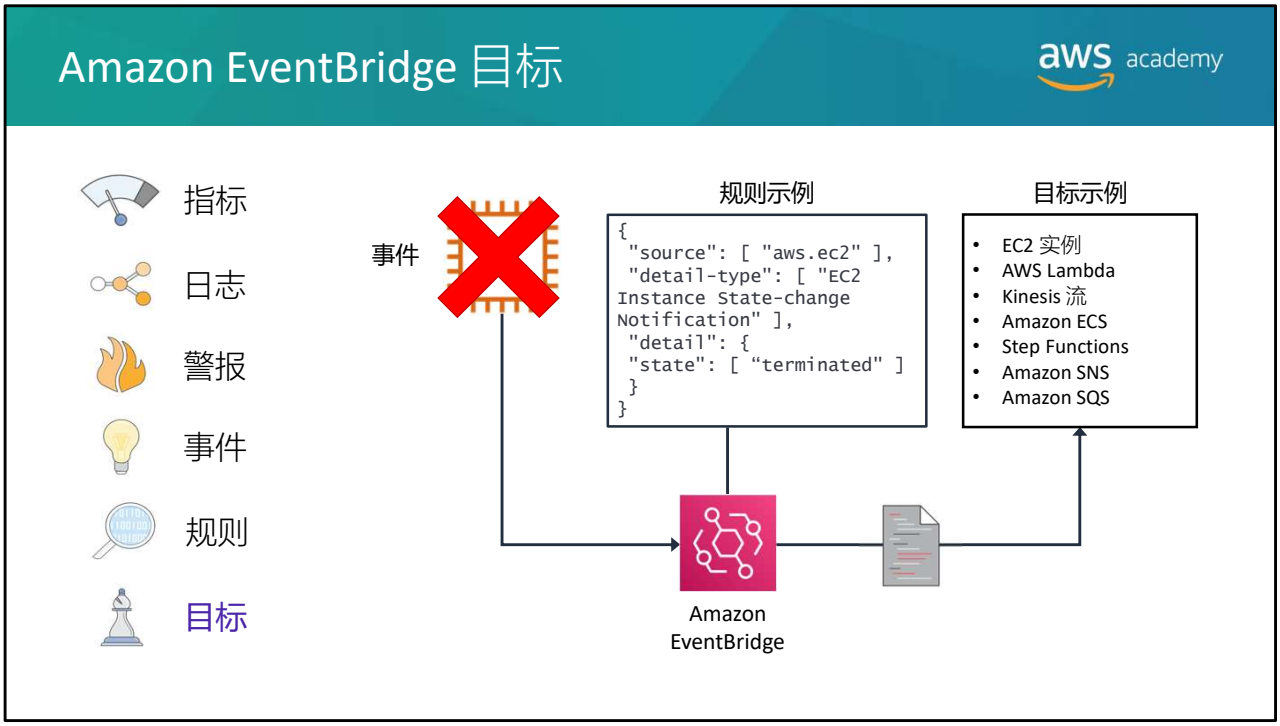
现有的 CloudWatch Events 用户可以在新的 EventBridge 控制台和 CloudWatch Events 控制台中访问其现有默认总线、规则和事件。EventBridge 使用相同的 CloudWatch Events API，因此现有 CloudWatch Events API 的所有使用方式均保持不变。

有关 EventBridge 的更多信息，请参阅什么是 Amazon EventBridge?



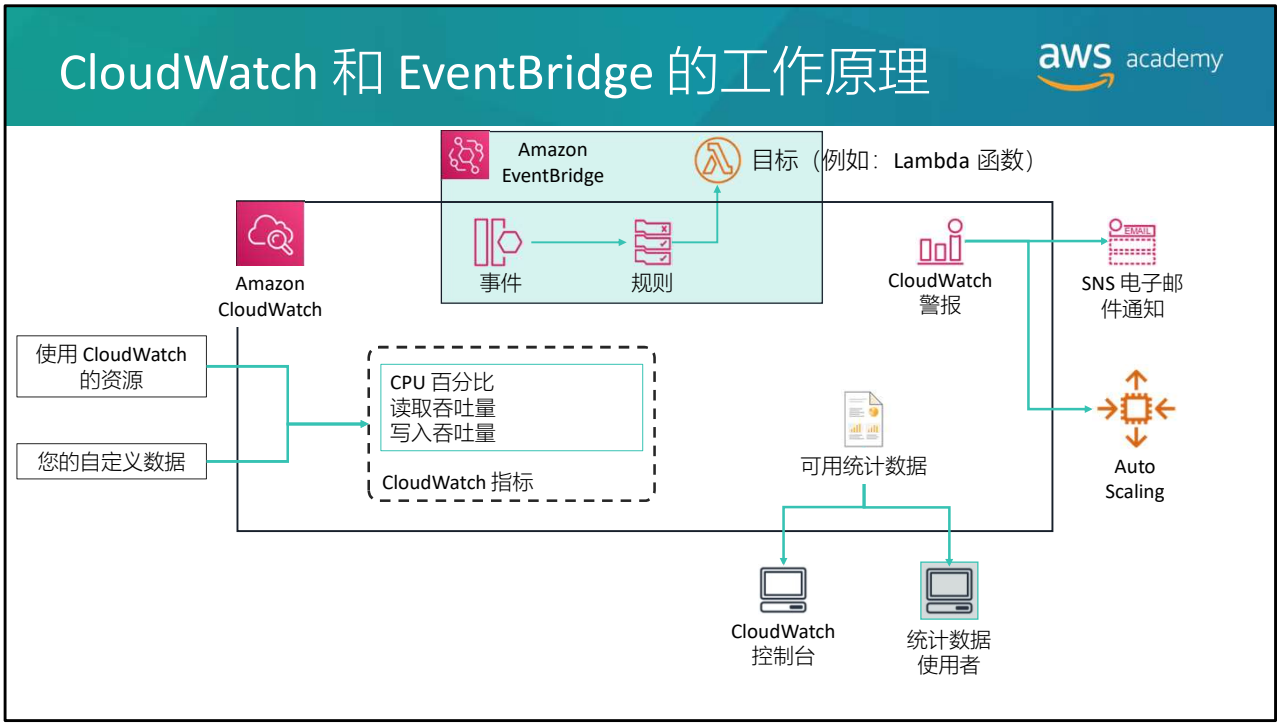
您可以设置路由规则来确定数据发送目的地，以构建能够实时响应所有数据源的应用程序架构。

规则*会匹配传入事件并将它们路由到目标以进行处理。*单个规则可以路由到多个目标，所有目标都是并行处理的。规则并不是按特定顺序进行处理的。因此，组织的不同部门能够查找和处理他们感兴趣的事件。规则可以自定义发送到目标的 JavaScript 对象表示法 (JSON) 消息，方法是仅传递某些部分或使用常量进行覆盖。



目标*负责处理事件*。目标包括 EC2 实例、Lambda 函数、Amazon Kinesis Streams、Amazon Elastic Container Service (Amazon ECS) 任务、AWS Step Functions 状态机、SNS 主题、Amazon Simple Queue Service (Amazon SQS) 队列和内置目标。目标接收 JSON 格式的事件。

创建规则时，将其与特定*事件总线*关联，并且该规则仅与该事件总线接收的事件匹配。



此架构在宏观层面展示了 CloudWatch 和 EventBridge 的工作方式。

CloudWatch 充当指标存储库。AWS 服务（如 Amazon EC2）会将指标放在存储库中，您可以根据这些指标检索统计信息。如果您将自己的自定义指标放在存储库中，则还可以检索有关这些指标的统计数据。

您可以在 CloudWatch 控制台中使用指标计算统计数据，然后以图形化的方式显示数据。

您可以在 EventBridge 中创建匹配传入事件并将其路由到目标进行处理的规则。

在满足特定条件时，您可以配置警报操作以停止、启动或终止 EC2 实例。此外，您还可以创建警报代您启动 Amazon EC2 Auto Scaling 和 Amazon SNS 操作。

第 5 节要点

- AWS Cost Explorer、AWS 预算、AWS 成本和使用情况报告和 Cost Optimization Monitor 可以帮助您了解和管理 AWS 基础设施的成本。
- CloudWatch 以日志、指标和事件的形式收集监控和运营数据。它使用自动化控制面板对数据进行可视化，以便您统一查看在 AWS 和本地运行的 AWS 资源、应用程序和服务。
- EventBridge 是一种无服务器事件总线服务，通过它可以将应用程序与来自各种来源的数据连接起来。EventBridge 可传输来自您自己的应用程序、SaaS 应用程序和 AWS 服务的实时数据流，然后将这些数据路由到目标。

本模块中这节内容的要点包括：

- AWS Cost Explorer、AWS 预算、AWS 成本和使用情况报告和 Cost Optimization Monitor 可以帮助您了解和管理 AWS 基础设施的成本。
- CloudWatch 以日志、指标和事件的形式收集监控和运营数据。它使用自动化控制面板对数据进行可视化，以便您统一查看在 AWS 和本地运行的 AWS 资源、应用程序和服务。
- EventBridge 是一项无服务器事件总线服务，通过它可以轻松将应用程序与来自各种来源的数据相连。EventBridge 从您自己的应用程序、SaaS 应用程序和 AWS 服务中提取实时数据流。然后将这些数据路由到目标。

模块 9 – 挑战实验：为咖啡馆创建可扩展且高度可用的环境

您现在要完成“模块 9 – 挑战实验：为咖啡馆创建可扩展且高度可用的环境”。

业务需求：可扩缩且高度可用的环境

- 咖啡馆很快将在著名的电视美食节目中亮相。
- Sonía 和 Nikhil 想要确保咖啡馆的网站能够应对预期的流量增长。

咖啡馆很快将在著名的电视美食节目中亮相。当节目播出时，Sofía 和 Nikhil 预计咖啡馆 Web 服务器的用户数量将出现短暂的激增，甚至可能达到数万人。目前，咖啡馆的 Web 服务器部署在一个可用区，他们担心服务器无法应对预期的流量增长。他们希望确保客户在访问网站时拥有出色的体验，不会遇到任何问题，例如下单延迟或延误。

为了确保提供这种体验，网站必须迅速响应，通过扩缩来满足不断变化的客户需求，并做到高度可用。它还必须包含负载均衡。此架构必须跨多个应用程序服务器分发客户订单请求，以便应对需求的增加，而不是让单个服务器过载。

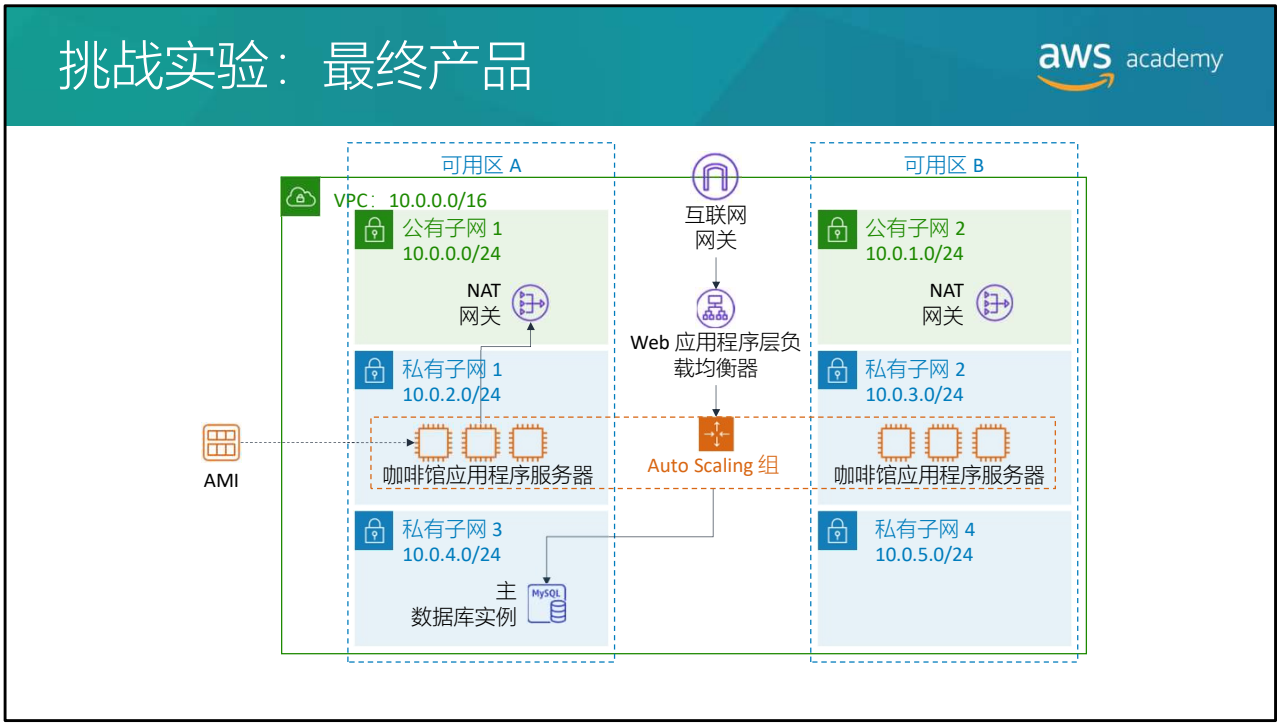
挑战实验：任务

1. 为第二个可用区创建 NAT 网关
2. 在公有子网中创建堡垒主机实例
3. 创建启动模板
4. 创建 Auto Scaling 组
5. 创建负载均衡器
6. 测试 Web 应用程序
7. 测试负载下的自动扩缩

在本挑战实验中，您将完成以下任务：

1. 为第二个可用区创建 NAT 网关
2. 在公有子网中创建堡垒主机实例
3. 创建启动模板
4. 创建 Auto Scaling 组
5. 创建负载均衡器
6. 测试 Web 应用程序
7. 测试负载下的自动扩缩

挑战实验：最终产品



该图总结了您完成实验后将会构建的内容。

现在可以开始挑战实验了。

挑战实验总结：要点

完成这个挑战实验之后，您的讲师可能会带您讨论此挑战实验的要点。

模块 9：实施弹性、高可用性和监控

模块总结

现在来回顾下本模块，并对知识测验和对实践认证考试问题的讨论进行总结。

模块总结

aws academy

总体来说，您在本模块中学习了如何：

- 在架构中使用 Amazon EC2 Auto Scaling 来提高弹性
- 说明如何扩缩数据库资源
- 部署 Application Load Balancer 以创建高度可用的环境
- 使用 Amazon Route 53 进行 DNS 故障转移
- 创建高度可用的环境
- 设计能够使用 Amazon CloudWatch 监控资源并做出相应反应的架构

总体来说，您在本模块中学习了如何：

- 在架构中使用 Amazon EC2 Auto Scaling 来提高弹性
- 说明如何扩缩数据库资源
- 部署 Application Load Balancer 以创建高度可用的环境
- 使用 Amazon Route 53 进行 DNS 故障转移
- 创建高度可用的环境
- 设计能够使用 Amazon CloudWatch 监控资源并做出相应反应的架构

完成知识测验

现在可以完成本模块的知识测验。

样题

aws academy

Web 应用程序使客户能够将订单上传到 S3 存储桶。生成的 Amazon S3 事件会触发一个 Lambda 函数，该函数会将消息插入 SQS 队列。单个 EC2 实例会从队列中读取消息，然后处理消息并将其存储在按唯一订单 ID 分区的 DynamoDB 表中。预计下个月的流量将增加 10 倍，解决方案架构师正在审查架构是否存在可能的扩缩问题。

哪个组件最有可能需要重新架构才能扩展以容纳新的流量？

A. Lambda 函数
B. SQS 队列
C. EC2 实例
D. DynamoDB 表

请查看答案选项，并根据之前突出显示的关键字排除错误选项。

***正确答案是 C**：EC2 实例。*单个 EC2 实例无法扩缩，是架构中的单点故障。更好的解决方案是将 EC2 实例放在两个可用区的 Auto Scaling 组中，使其从队列中读取消息。其他答案都是可配置为扩缩或自动扩缩的托管服务。

其他资源

- Set it and Forget it: Auto Scaling Target Tracking Policies
- Amazon Elastic Load Balancer 简介 – 应用
- 使用 ELB Elastic Load Balancer 配置 Auto Scaling 组
- 什么是 Application Load Balancer?

如果您想了解有关本模块所涵盖主题的更多信息，下面这些其他资源可能会有所帮助：

- Set it and Forget it: Auto Scaling Target Tracking Policies
- Amazon Elastic Load Balancer 简介 – 应用
- 使用 ELB Elastic Load Balancer 配置 Auto Scaling 组
- 什么是 Application Load Balancer?

感谢您完成本模块的学习。