为您的负载均衡器选择粘性策略

AWS 规范性指导

Copyright © 2026 Amazon Web Services, Inc. and/or its affiliates. All rights reserved.

AWS 规范性指导: 为您的负载均衡器选择粘性策略

Copyright © 2026 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

为您的负载均衡器选择粘性策略

Ryan Griffin，Amazon Web Services ()AWS

2024 年 7 月（文件历史记录）

Stickiness 这个术语用于描述负载均衡器的功能，即重复将流量从客户端路由到单个目的地，而不是在多个目的地之间平衡流量。例如，来自客户端 A 的流量可以持续路由到特定的服务器，这样服务器就可以维护会话状态数据。如果将来自客户端 A 的流量路由到两台不同的服务器，则每台服务器都可能缺少可供另一台服务器使用的重要信息。

因此，通常需要通过负载均衡器保持一致的客户端连接。粘性有两种类型：粘性会话和目标群体的粘性。

- 粘性会话 — 在 Amazon Elastic Compute Cloud (Amazon EC2) 实例中维护本地会话数据，以简化应用程序架构或提高应用程序性能，因为该实例可以在本地维护或缓存会话状态信息。 AWS 目前提供两种类型的粘性会话，本指南对此进行了详细讨论：应用程序 Cookie 和负载均衡器 Cookie。
- 目标组粘性-在蓝/绿部署中，您可能部署了多个版本的应用程序，并且您可能希望客户端在会话期间继续使用相同版本的应用程序。在这种情况下，您可以使用目标组粘性将来自客户端的所有通信路由到同一个目标组，而不是同一个实例。 EC2

您可以单独使用这两种粘性策略，也可以一起使用。

本指南描述了不同类型的负载均衡器粘性以及适用的用例，以帮助您选择策略。该指南包括说明每种策略的 AWS CloudFormation 模板。

代码示例

本指南提供了一个附带的.zip 文件，其中包含四个 AWS CloudFormation 模板，您可以部署这些模板来构建基本架构并尝试每种粘性策略。我们建议您在实验室环境中部署这些模板来测试每种方法。

下

载示例代码

下载内容包括以下模板：

- `basic.yml`— 在没有粘性的情况下设置 Application Load Balancer。
- `targetgroupstickiness.yml`— 根据目标群体表现出粘性。

- `stickysessionslb.yml`— 演示使用负载均衡器生成的 Cookie 的粘性会话。
- `stickysessionsapp.yml`— 演示使用基于应用程序的 Cookie 的粘性会话。

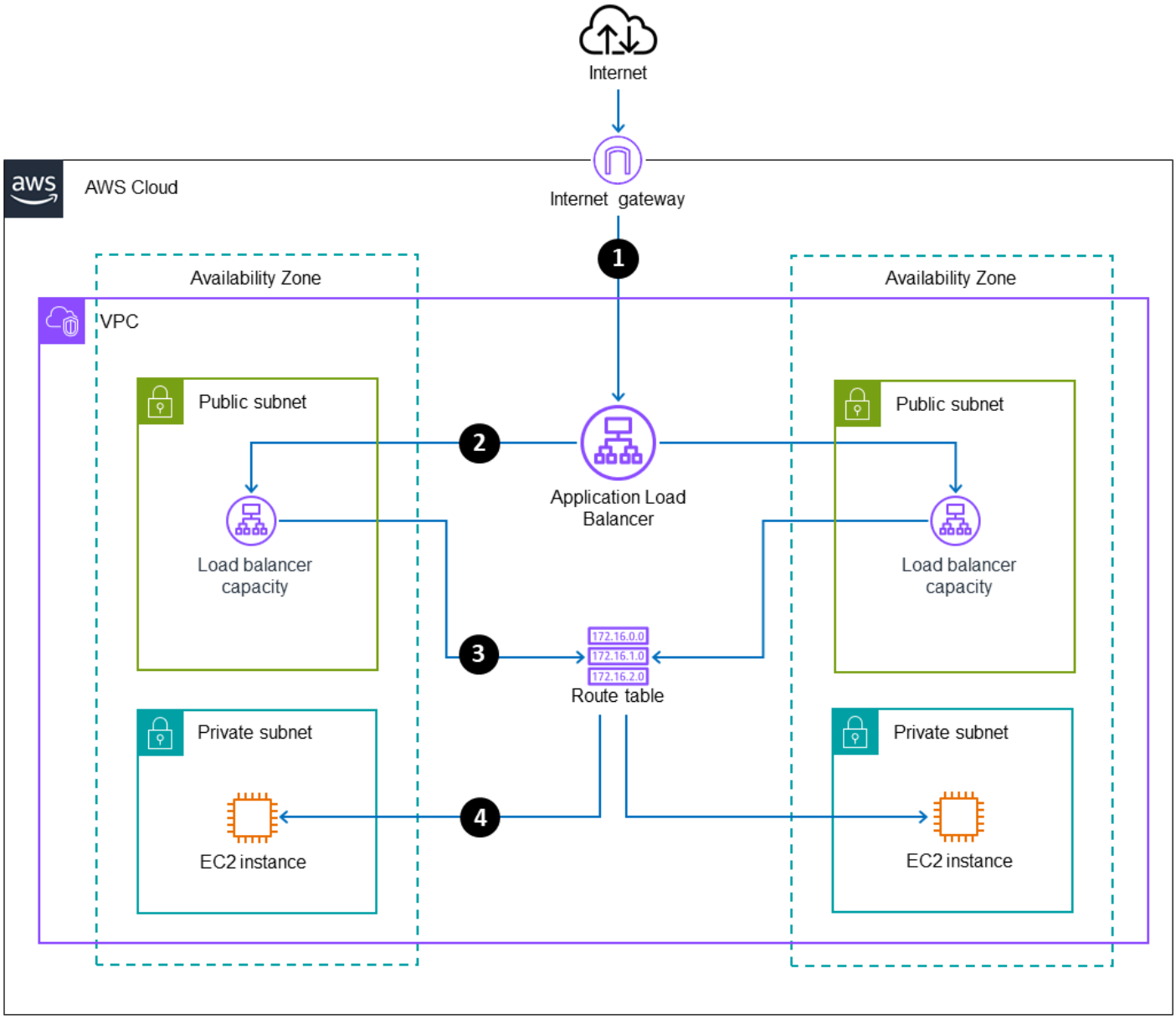
要部署这些模板，您需要一个 active:AWS 帐户和CloudFormation 控制台访问权限。有关部署 CloudFormation 模板的 step-by-steps说明，请参阅 CloudFormation 文档中的创建堆栈。

负载平衡器子网和路由

让我们先举例说明当你配置面向互联网的应用程序负载均衡器时，流量是如何流向私有子网中的亚马逊弹性计算云 (Amazon EC2) 实例的。此架构反映了部署向互联网开放的 Application Load Balancer 时的最佳实践。

入站流量路径

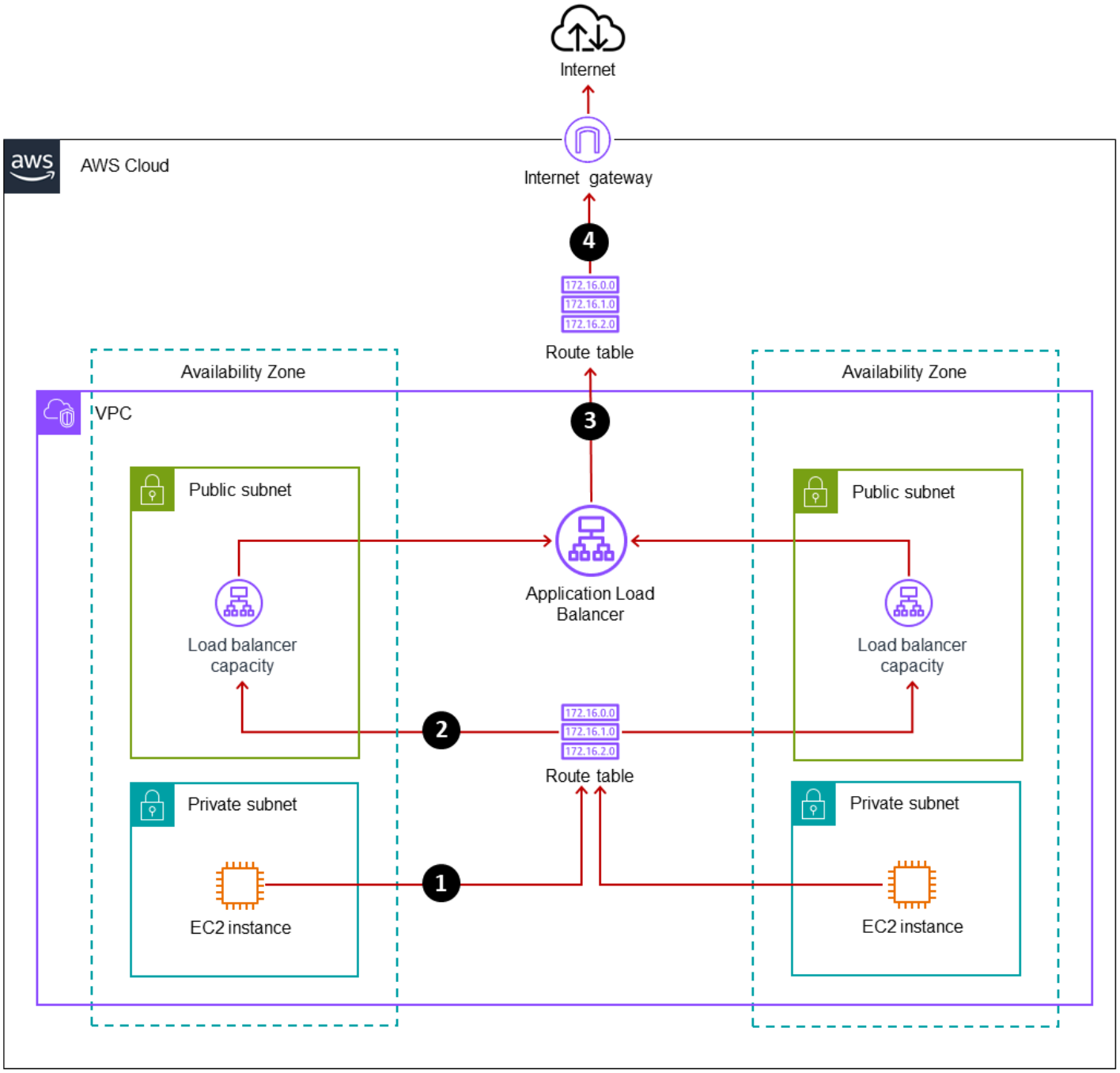
下图说明了与传入流量相关的虚拟私有云 (VPC) 子网和路由，为了清晰起见，从图中删除了返回流量。



1. 来自互联网的流量流入 Application Load Balancer DNS 名称。
2. 如图所示，Application Load Balancer 与两个公有子网关联。Elastic Load Balancing 服务在每个启用的可用区中创建负载均衡器容量，并通过可用区发送流量以确定适当的路由逻辑。Application Load Balancer 使用其内部逻辑来确定将流量路由到哪个目标组和实例。
3. Application Load Balancer 通过与同一可用区中的公有子网关联的节点将请求路由到 EC2 实例。（这些节点由 Elastic Load Balancing 服务进行配置、管理和扩展，对用户不可见。）
4. 路由表将流量路由到 VPC 内本地、公有子网和私有子网之间以及 EC2 实例。

返回交通路径

下图说明了与返回 Internet 的流量路径相关的 VPC 子网和路由，为清楚起见，传入流量已从图表中删除。

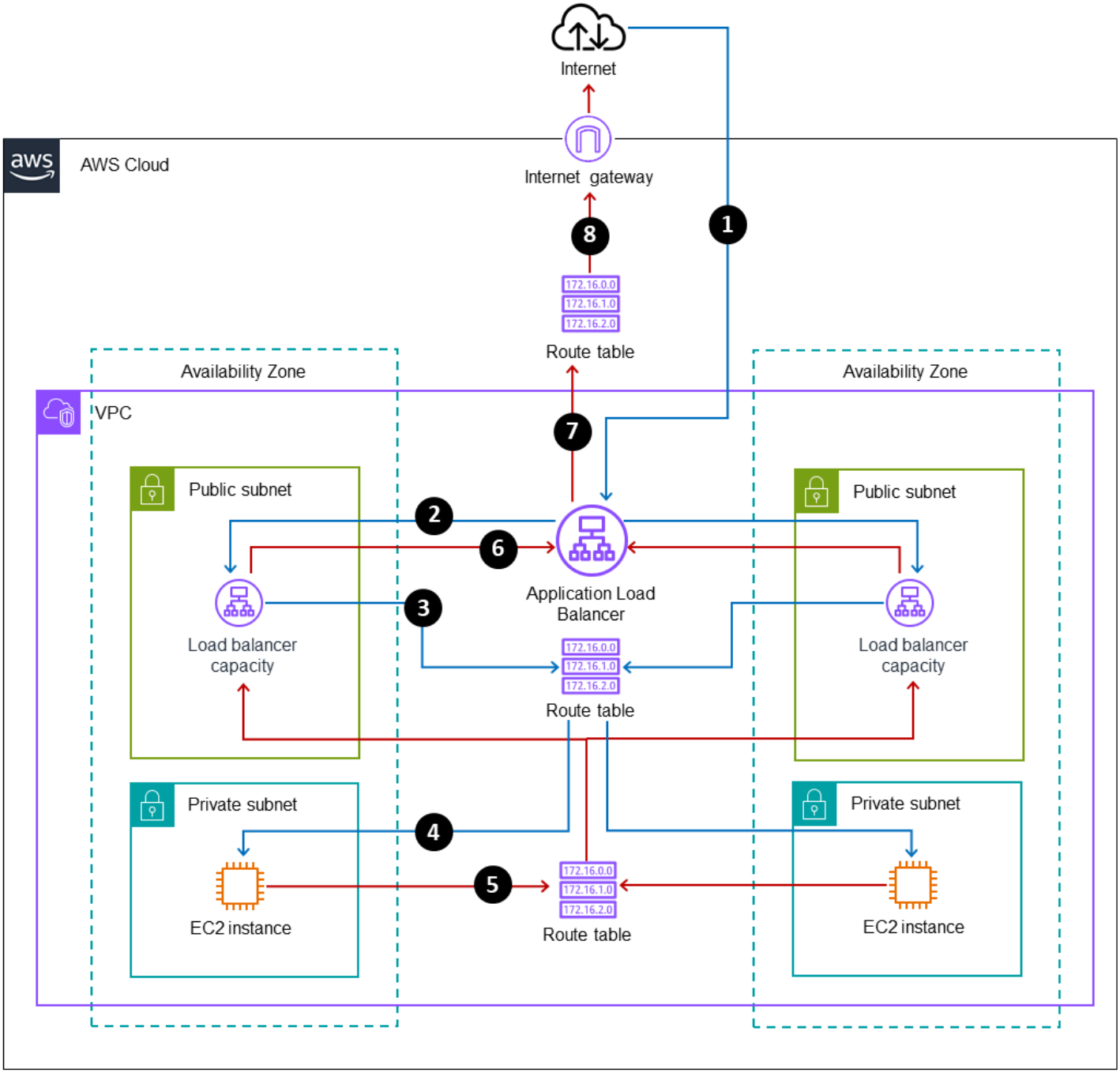


1. 私有子网中的 EC2 实例通过路由表路由出站流量。

2. 路由表中有一条通往公有子网的本地路由。它通过沿着流量进入的路径返回路径来达到相应公有子网中流量输入的 Application Load Balancer 容量。
3. Application Load Balancer 通过其公共接口将流量路由出去。
4. 公有子网的路由表中有一条指向互联网网关的默认路由，该网关将流量路由回互联网。

完整的交通流图

下图结合了入站和返回的流量，提供了负载均衡器路由的完整说明。



1. 来自互联网的流量流入 Application Load Balancer DNS 名称。
2. 如图所示，Application Load Balancer 与两个公有子网关联。Elastic Load Balancing 服务在每个启用的可用区中创建负载均衡器容量，并通过可用区发送流量以确定适当的路由逻辑。Application Load Balancer 使用其内部逻辑来确定将流量路由到哪个目标组和实例。
3. Application Load Balancer 通过与同一可用区中的公有子网关联的节点将请求路由到 EC2 实例。（这些节点由 Elastic Load Balancing 服务进行配置、管理和扩展，对用户不可见。）

4. 路由表将流量路由到 VPC 内本地、公有子网和私有子网之间以及 EC2 实例。
5. 私有子网中的 EC2 实例通过路由表路由出站流量。
6. 路由表中有一条通往公有子网的本地路由。它通过沿着流量进入的路径返回路径来达到相应公有子网中流量输入的 Application Load Balancer 容量。
7. Application Load Balancer 通过其公共接口将流量路由出去。
8. 公有子网的路由表中有一条指向互联网网关的默认路由，该网关将流量路由回互联网。

哪种粘性策略适合你？

回答以下问题将帮助您确定您的负载均衡器粘性策略。

你在用 WebSockets吗？

- 是的：本质上 WebSockets 是粘性的，因此无需使用任何策略。
- 否：继续回答下一个问题。

您是否计划进行蓝/绿部署？

- 是：至少要使用目标群体的粘性，但要继续回答下一个问题。
- 否：继续回答下一个问题。

您是否需要将来自客户端的部分或全部流量重复路由到同一个 EC2 实例？

- 是：继续回答下一个问题。
- 否：您不需要使用粘性会话。

您是否希望您的应用程序代码或客户端使用 Cookie 来控制状态属性和持续时间？

- 是：使用基于应用程序的 Cookie 来启用基于应用程序的粘性会话。
- 否：使用负载均衡器生成的 Cookie 来启用基于持续时间的粘性会话。

没有粘性的 Application Load Balancer

当您使用没有任何粘性的 Application Load Balancer 时，默认情况下，负载均衡器使用轮询方法来确定应将流量路由到哪个 EC2 实例。

模板：使用 CloudFormation 模板`basic.yml`（包含在示例代码.zip 文件中）试用此功能。

> **Note**
>
> 本指南中包含的所有 CloudFormation 模板都部署了自定义 VPC、路由表、路由、Internet 网关、Application Load Balancer、目标组、侦听器和 EC2 实例，以说明特定的负载均衡器粘性策略。

常见使用案例

在以下情况下，使用没有粘性的 Application Load Balancer：

- 您有一份要将流量路由到的目标列表，但这些目标不保持会话状态。
- 您使用的 Web 服务器不维护会话状态。
- 您使用的应用程序服务器不维护会话状态。

步骤

> **备注**
>
> - NAT 网关的费用很小。
> - 与单个 EC2 EC2实例相比，多个实例耗尽免费套餐的小时数更快。

1. 在实验室环境`basic.yml`中部署 CloudFormation 模板。
2. 等待，直到目标组实例的运行状况从初始变为正常。
3. 使用 HTTP (TCP/80) 在网络浏览器中导航到 Application Load Balancer 网址。

   例如：`http://alb-123456789.us-east-1.elb.amazonaws.com/`

该网页显示实例 1- TG1 或实例 2- TG1。

4. 多次刷新页面。

预期结果

加载网页的实例（实例 1 或实例 2）应每次都发生变化，如页面文本所示。负载均衡器逻辑跨多个内部节点管理最后一个目标，这可能会导致同步延迟，因此您可能会被路由到同一个目标。

工作方式

- 在此示例中，将两个 EC2 实例分配给一个目标组。这些 EC2实例安装了 Apache Web 服务器 (`httpd`)，并且每个 EC2 实例上的`index.html`页面文本都经过硬编码以识别该实例。
- Application Load Balancer 运行其内部循环逻辑来确定哪个 EC2 实例应该接收流量。
- 每次重新加载网页时，Application Load Balancer 都会运行其路由逻辑，页面会显示实例 1 TG1 或实例 2- TG1。

目标群体的粘性

当您使用具有目标组粘性的 Application Load Balancer 时：

- Application Load Balancer 使用目标组权重来确定如何在目标组之间平衡传入流量。
- 默认情况下，Application Load Balancer 使用循环方法将请求路由到目标目标组中的 EC2 实例。

模板：使用 CloudFormation 模板`targetgroupstickiness.yml`（包含在示例代码.zip 文件中）来尝试目标群体的粘性。

常见使用案例

在以下场景中使用目标群体的粘性：

- 负载均衡器分配了多个目标组，来自客户端的流量应始终如一地路由到该目标组内的实例。
- 蓝/绿部署。

basic.yml 的代码发生了变化

对监听器进行了单一更改：我们修改了 Application Load Balancer 的默认操作，以指定两个权重相等的目标组（`TG1`和`TG2`），并使用粘性配置。

basic.yml

```
Listener1:
  Type: 'AWS::ElasticLoadBalancingV
2::Listener'
  Properties:
    LoadBalancerArn: !Ref ALB
    Protocol: HTTP
    Port: 80
    DefaultActions:
      - TargetGroupArn: !Ref TG1
        Type: forward
```

targetgroupstickiness.yml

```
Listener1:
  Type: 'AWS::ElasticLoadBalancingV
2::Listener'
  Properties:
    LoadBalancerArn: !Ref ALB
    Protocol: HTTP
    Port: 80
    DefaultActions:
      - ForwardConfig:
          TargetGroups:
            - TargetGroupArn: !Ref TG1
              Weight: 1
```

```
            - TargetGroupArn: !Ref TG2
              Weight: 1
          TargetGroupStickinessConfig
:
              DurationSeconds: 10
              Enabled: true
        Type: forward
```

步骤

> **备注**
>
> - NAT 网关的费用很小。
> - 与单个 EC2 EC2实例相比，多个实例耗尽免费套餐的小时数更快。

1. 在实验室环境`targetgroupstickiness.yml`中部署 CloudFormation 模板。
2. 等待，直到目标组实例的运行状况从初始变为正常。
3. 使用 HTTP (TCP/80) 在网络浏览器中导航到 Application Load Balancer 网址。

   例如：`http://alb-123456789.us-east-1.elb.amazonaws.com/`

   该网页显示以下内容之一：实例 1- TG1、实例 2- TG1、实例 3- TG2 或实例 4- TG2。
4. 多次刷新页面。

预期结果

> **Note**
>
> 本示例中的 CloudFormation 模板将粘性配置为持续 10 秒。

加载网页的实例应在 10 秒的持续时间内停留在目标组（TG1 或 TG2）内，如页面文本所示。

大约 10 秒钟后，粘性就会被释放，目标组实例集可能会发生变化。

工作方式

- 在此示例中，四个 EC2 实例被分成两个目标组，每个目标组有两个实例。这些 EC2 实例安装了 Apache Web 服务器 (`httpd`)，并且每个 EC2实例上的`index.html`页面文本都经过硬编码以区分开来。
- Application Load Balancer 为用户与目标目标组的会话创建绑定，并附有过期时间。
- 当您重新加载页面时，Application Load Balancer 会检查绑定是否存在且尚未过期。
  - 如果绑定已过期或不存在，Application Load Balancer 将运行其路由逻辑并确定目标组。
  - 如果绑定尚未过期，Application Load Balancer 会将流量路由到同一个目标组，但不一定要路由到同一个 EC2 实例。

使用负载均衡器生成的 cookie 的粘性

当你将 Application Load Balancer 与负载均衡器生成的 Cookie 一起使用时：

- Application Load Balancer 使用目标组权重来确定如何在目标组之间平衡传入流量。
- 默认情况下，Application Load Balancer 使用循环方法将请求路由到目标目标组中的 EC2 实例。

  在流量最初路由到实例后，后续流量将在指定的持续时间内持续到该 EC2 实例。

模板：使用 CloudFormation 模板`stickysessionslb.yml`（包含在示例代码.zip 文件中）尝试使用负载均衡器生成的 Cookie 进行粘性会话。

常见使用案例

在以下情况下，使用带有负载均衡器生成的 Cookie 的粘性会话：

- PHP 网络服务器
- 维护日志、购物车或聊天对话等临时会话数据的服务器

basic.yml 的代码发生了变化

相关的代码更改位于目标组配置中，将粘性类型设置为，持续时间设置为 10 秒。`lb_cookie`

basic.yml

```
TG1:
   Type: 'AWS::ElasticLoadBalancingV
2::TargetGroup'
   Properties:
     Name: TG1
     Protocol: HTTP
     Port: 80
     TargetType: instance
     Targets:
       - Id: !Ref Instance1
       - Id: !Ref Instance2
```

stickysessionslb.yml

```
TG1:
   Type: 'AWS::ElasticLoadBalancingV
2::TargetGroup'
   Properties:
     Name: TG1
     Protocol: HTTP
     Port: 80
     TargetType: instance
     Targets:
       - Id: !Ref Instance1
       - Id: !Ref Instance2
     VpcId: !Ref CustomVPC
```

```
VpcId: !Ref CustomVPC
```

```
      TargetGroupAttributes:
        - Key: stickiness.enabled
          Value: true
        - Key: stickiness.type
          Value: lb_cookie
        - Key: stickiness.lb_cook
ie.duration_seconds
Value: 10
```

步骤

> **备注**
>
> - NAT 网关的费用很小。
> - 多个 EC2 实例将比单个 EC2实例更快地耗尽您的免费套餐时间。

1. 在实验室环境`stickysessionslb.yml`中部署 CloudFormation 模板。
2. 等待，直到目标组实例的运行状况从初始变为正常。
3. 使用 HTTP (TCP/80) 在网络浏览器中导航到 Application Load Balancer 网址。

   例如：`http://alb-123456789.us-east-1.elb.amazonaws.com/`

   该网页显示以下内容之一：实例 1- TG1、实例 2- TG1、实例 3- TG2 或实例 4- TG1。
4. 多次刷新页面。

预期结果

> **Note**
>
> 本示例中的 CloudFormation 模板将粘性配置为持续 10 秒。

加载网页的实例应在 10 秒的持续时间内保持不变，如页面文本所示。大约 10 秒钟后，粘性就会被释放，目标实例可能会发生变化。

工作方式

- 在此示例中，一个目标组中存在两个 EC2 实例。这些 EC2 实例安装了 Apache Web 服务器 (`httpd`)，并且每个 EC2 实例上的`index.html`页面文本都经过硬编码以区分开来。
- Application Load Balancer 为用户的会话创建绑定，该绑定绑定到目标，并带有过期时间。
- 当您重新加载页面时，Application Load Balancer 会检查绑定是否存在且尚未过期。
  - 如果绑定已过期或不存在，Application Load Balancer 将运行其路由逻辑并确定目标实例。
  - 如果绑定尚未过期，Application Load Balancer 会将流量路由到同一个目标实例。

使用基于应用程序的 Cookie 进行粘性会话

当您使用带有基于应用程序的 Cookie 的 Application Load Balancer 时：

- Application Load Balancer 使用目标组权重来确定如何在目标组之间平衡传入流量。
- 默认情况下，Application Load Balancer 使用循环方法将请求路由到目标目标组中的 EC2 实例。
- 在流量最初路由到实例后， EC2 EC2 实例应用程序响应应包含自定义应用程序 Cookie，该应用程序将与自动的 Application Load Balancer cookie 一起发送回客户端。
- 如果客户端发回应用程序 Cookie 和 Application Load Balancer cookie，则后续流量将持续到该 EC2 实例。
- 基于应用程序的 Cookie 在配置的持续时间内不使用后过期。

模板：使用 CloudFormation 模板`stickysessionsapp.yml`（包含在示例代码.zip 文件中）尝试使用基于应用程序的 Cookie 进行粘性会话。

常见使用案例

如果您想在以下情况下进行更多控制，请使用带有应用程序生成的 Cookie 的粘性会话：

- PHP 网络服务器
- 维护日志、购物车或聊天对话等临时会话数据的服务器

basic.yml 的代码发生了变化

唯一的代码更改是在目标组配置中。我们在 Application Load Balancer 和目标组属性中添加了粘性配置。应用程序 Cookie 持续时间已指定，目标群组已启用应用程序 Cookie 粘性。

basic.yml	**stickysessionsapp.yml**

`TG1:`
`  Type: 'AWS::ElasticLoadBalancingV2::TargetGroup'`
`  Properties:`
`  Name: TG1`
	`TG1:`
	`  Type: 'AWS::ElasticLoadBalancingV2::TargetGroup'`
	`  Properties:`
	`  Name: TG1`

```
      Protocol: HTTP
      Port: 80
      TargetType: instance
      Targets:
        - Id: !Ref Instance1
        - Id: !Ref Instance2
VpcId: !Ref CustomVPC
```

```
    Protocol: HTTP
    Port: 80
    TargetType: instance
    Targets:
      - Id: !Ref Instance1
      - Id: !Ref Instance2
    VpcId: !Ref CustomVPC
    TargetGroupAttributes:
      - Key: stickiness.enabled
        Value: true
      - Key: stickiness.type
        Value: app_cookie
      - Key: stickiness.app_coo
kie.duration_seconds
        Value: 10
      - Key: stickiness.app_coo
kie.cookie_name
        Value: TESTCOOKIE
```

步骤

> **备注**
>
> - NAT 网关的费用很小。
> - 多个 EC2 实例将比单个 EC2实例更快地耗尽您的免费套餐时间。

1. 在实验室环境`stickysessionslb.yml`中部署 CloudFormation 模板。
2. 等待，直到目标组实例的运行状况从初始变为正常。
3. 使用 HTTP (TCP/80) 在网络浏览器中导航到 Application Load Balancer 网址。

   例如：`http://alb-123456789.us-east-1.elb.amazonaws.com/`

   该网页显示以下内容之一：实例 1- TG1、实例 2- TG1、实例 3- TG2 或实例 4- TG2。
4. 多次刷新页面。

预期结果

> **Note**
>
> 本示例中的 CloudFormation 模板已将粘性配置为持续 10 秒。有效的粘性持续时间配置介于 1 秒到 1 周之间。

加载网页的实例应保持不变，如页面文本所示。

粘性持续时间不会刷新，而是基于在 Application Load Balancer 中为实例生成的应用程序 Cookie 配置的过期时间。 EC2

示例 1：等待 5 秒钟刷新页面。同一个实例将加载，粘性将再刷新 10 秒钟。

示例 2：等待时间超过 10 秒以重新加载页面。应用程序 Cookie 过期，您将被路由到另一个实 EC2 例。这个新实例会生成另一个持续时间为 10 秒的应用程序 Cookie。

工作方式

- 在此示例中，这些 EC2 实例安装了 Apache Web 服务器 (httpd)。该`httpd.conf`文件配置为向客户端（您的 Web 浏览器）返回静态`Set-Cookie`值。该`Set-Cookie`值被硬编码为。`TESTCOOKIE=<somevalue>`
- 打开浏览器的 Inspect Element 选项，选择网络选项卡，然后选择加载页面的 Ge t 方法。您将看到一个 Cookie 选项卡。
- 浏览器是一个客户端应用程序，它会自动配置为使用服务器`Set-Cookie`响应中收到的 Cookie 向服务器返回任何后续更新。
- 当您重新加载页面时，在初始页面加载时收到的 Cookie 会自动发送回 Application Load Balancer。
  - 如果 Cookie 已过期（即自您上次调用以来已过 10 秒），Application Load Balancer 将使用新的逻辑来确定将流量路由到哪个 EC2 实例。
  - 如果 cookie 尚未过期，Application Load Balancer 会将流量路由到同一个 EC2实例。

资源

- 应用程序负载均衡器的粘性会话（Elastic Load Balancing 文档）

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
更新部分	使用新的图表和说明更新了负载平衡器子网和路由部分。	2024 年 7 月 5 日
更新和更正	更新了代码、图表和示例。	2023 年 5 月 31 日
更新的章节	更新了使用负载均衡器生成的 Cookie 的 “目标群组粘性” 和 “粘性会话” 中的 “工作原理” 部分，以提供更准确、更详细的信息。	2022 年 7 月 18 日
一	初次发布	2021 年 8 月 5 日

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。