為您的負載平衡器選擇黏性策略

AWS 方案指引

AWS 方案指引: 為您的負載平衡器選擇黏性策略

Table of Contents

選擇負載平衡器的黏性策略

Ryan Griffin，Amazon Web Services (AWS)

2024 年 7 月 (文件歷史記錄)

黏性是一個術語，用於描述負載平衡器將流量從用戶端重複路由到單一目的地的功能，而不是平衡多個目的地的流量。例如，來自用戶端 A 的流量可以持續路由至特定伺服器，以便伺服器可以維護工作階段狀態資料。如果來自用戶端 A 的流量路由到兩個不同的伺服器，則每個伺服器可能缺少可供其他伺服器使用的重要資訊。

因此，通常需要透過負載平衡器維持一致的用戶端連線。黏性有兩種：黏性工作階段和目標群組黏性。

- 黏性工作階段 – 在 Amazon Elastic Compute Cloud (Amazon EC2) 執行個體中維護本機工作階段資料，以簡化應用程式架構或改善應用程式效能，因為執行個體可以在本機維護或快取工作階段狀態資訊。 AWS 目前提供兩種黏性工作階段，本指南會詳細討論：應用程式 Cookie 和負載平衡器 Cookie。
- 目標群組黏性 – 在藍/綠部署中，您可能已部署多個版本的應用程式，而且您可能希望用戶端在其工作階段期間繼續使用相同版本的應用程式。在這種情況下，您可以使用目標群組黏性，將所有通訊從用戶端路由到相同的目標群組，而不是相同的 EC2 執行個體。

您可以分別或一起使用這兩種黏性策略。

本指南說明不同類型的負載平衡器黏性與適用的使用案例，以協助您選擇策略。本指南包含說明每個策略的 AWS CloudFormation 範本。

範本程式碼

本指南提供連接的 .zip 檔案，其中包含四個範本，您可以部署這些 AWS CloudFormation 範本來建置基本架構，並嘗試每個黏性策略。我們建議您在實驗室環境中部署這些範本，以測試每個方法。

下載範例程式碼

下載包含這些範本：

- `basic.yml` – 設定 Application Load Balancer 而不黏。
- `targetgroupstickiness.yml` – 根據目標群組示範黏性。

- `stickysessionslb.yml` – 使用負載平衡器產生的 Cookie 示範黏性工作階段。
- `stickysessionsapp.yml` – 使用應用程式型 Cookie 示範黏性工作階段。

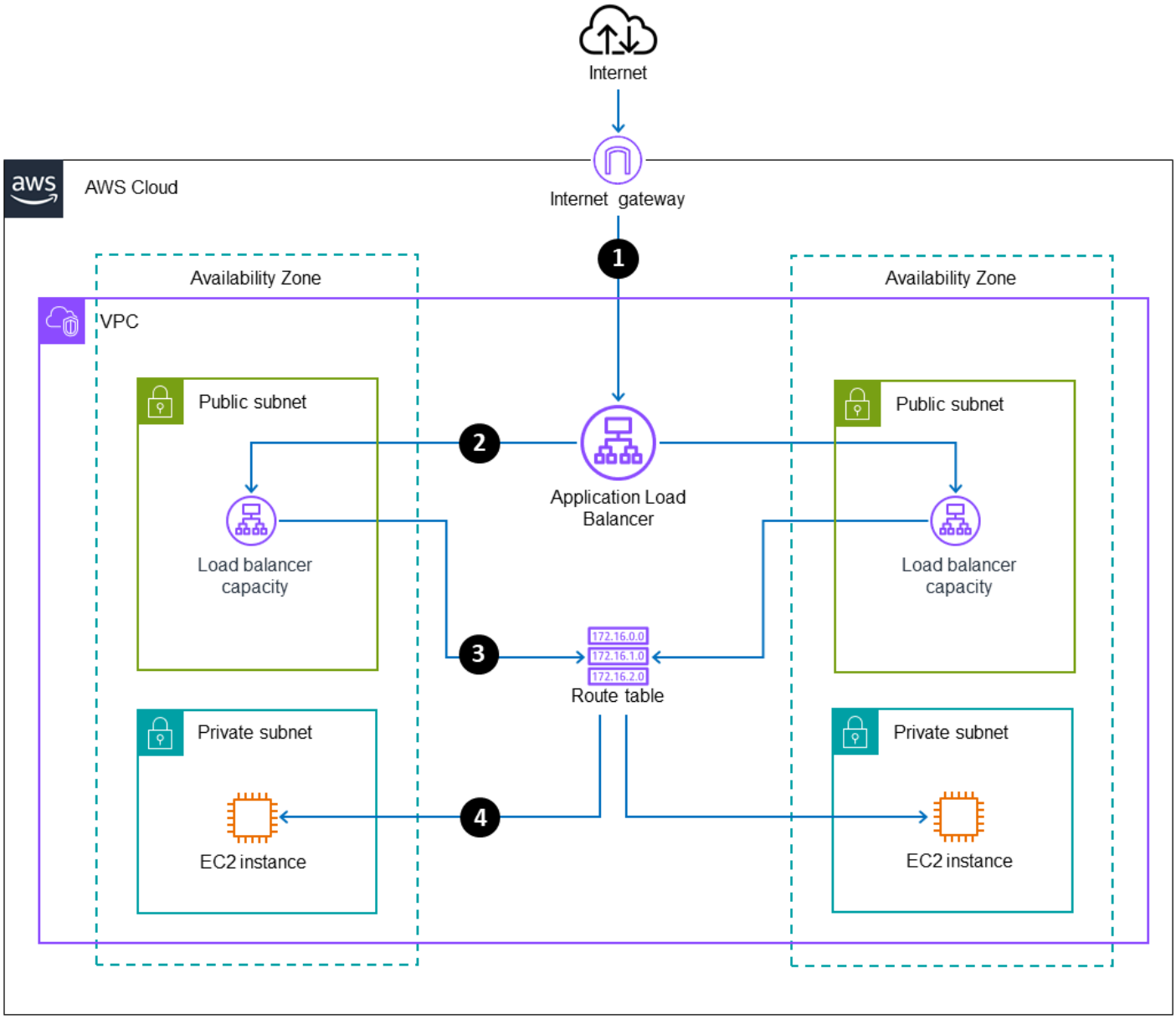
若要部署這些範本，您需要作用中的 ：AWS account 和 CloudFormation 主控台的存取權。如需部署 CloudFormation 範本step-by-steps說明，請參閱 CloudFormation 文件中的建立堆疊。

負載平衡器子網路和路由

讓我們從您設定面向網際網路的 Application Load Balancer 時，流量流向私有子網路中 Amazon Elastic Compute Cloud (Amazon EC2) 執行個體的圖例開始。此架構反映了部署開放給網際網路的 Application Load Balancer 時的最佳實務。

傳入流量路徑

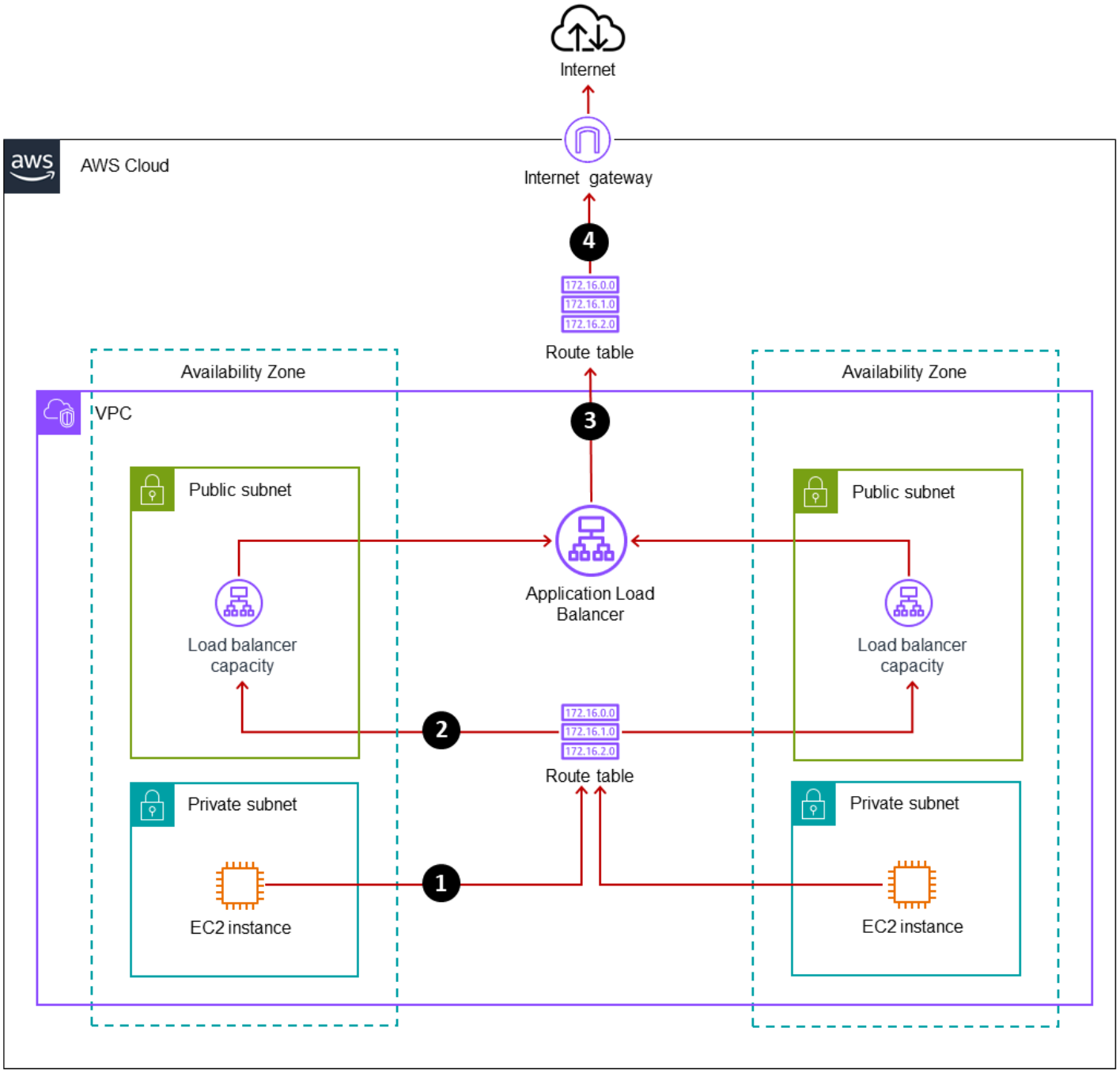
下圖說明與傳入流量流程相關聯的虛擬私有雲端 (VPC) 子網路和路由，為了清楚起見，從圖表中移除傳回流量。



1. 從網際網路流入 Application Load Balancer DNS 名稱的流量。
2. Application Load Balancer 與所說明案例中的兩個公有子網路相關聯。Elastic Load Balancing 服務會在每個啟用的可用區域中建立負載平衡器容量，並透過可用區域傳送流量，以判斷適當的路由邏輯。Application Load Balancer 會使用其內部邏輯來判斷要將流量路由到哪個目標群組和執行個體。
3. Application Load Balancer 會透過與相同可用區域中的公有子網路相關聯的節點，將請求路由至 EC2 執行個體。（這些節點是由 Elastic Load Balancing 服務設定、管理和擴展，使用者看不到。）
4. 路由表會在本機 VPC 內、公有子網路與私有子網路之間，以及 EC2 執行個體之間路由流量。

傳回流量路徑

下圖說明傳回至網際網路之流量路徑的 VPC 子網路和路由，為了清楚起見，傳入流量已從圖表中移除。

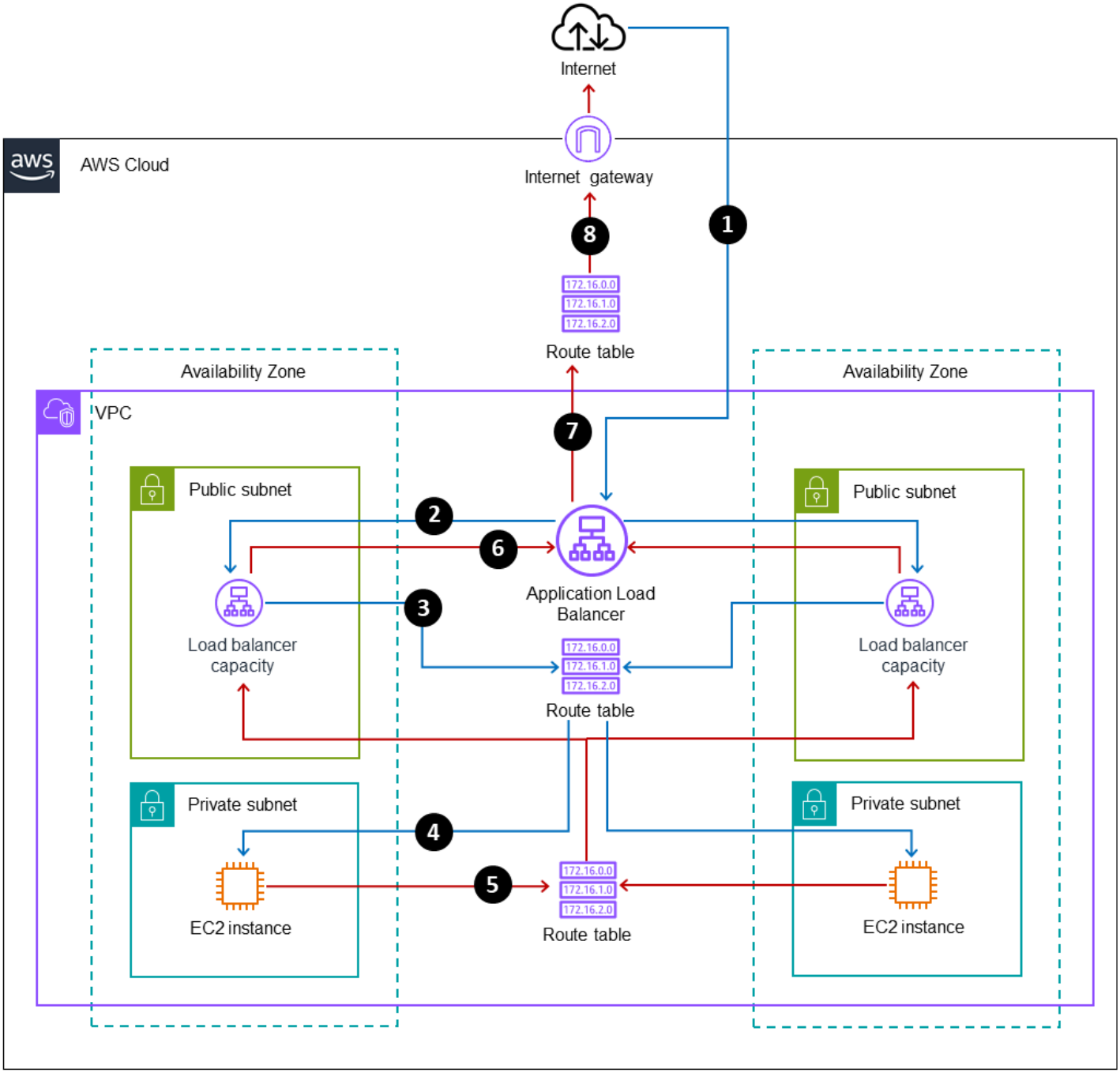


1. 私有子網路中的 EC2 執行個體會透過路由表路由傳出流量。

2. 路由表具有公有子網路的本機路由。它會遵循傳回流量輸入方式的路徑，達到流量在對應公有子網路中進入的 Application Load Balancer 容量。
3. Application Load Balancer 透過其公有界面路由出流量。
4. 公有子網路的路由表具有指向網際網路閘道的預設路由，可將流量路由回網際網路。

完成流量流程圖

下圖結合傳入和傳回流量流程，以提供負載平衡器路由的完整說明。



1. 從網際網路流入 Application Load Balancer DNS 名稱的流量。
2. Application Load Balancer 與所說明案例中的兩個公有子網路相關聯。Elastic Load Balancing 服務會在每個啟用的可用區域中建立負載平衡器容量，並透過可用區域傳送流量，以判斷適當的路由邏輯。Application Load Balancer 會使用其內部邏輯來判斷要將流量路由到哪個目標群組和執行個體。

3. Application Load Balancer 會透過與相同可用區域中的公有子網路相關聯的節點，將請求路由至 EC2 執行個體。（這些節點是由 Elastic Load Balancing 服務設定、管理和擴展，使用者看不到。）
4. 路由表會在本機 VPC 內、公有子網路與私有子網路之間，以及 EC2 執行個體之間路由流量。
5. 私有子網路中的 EC2 執行個體會透過路由表路由傳出流量。
6. 路由表具有公有子網路的本機路由。它會遵循傳回流量輸入方式的路徑，達到流量在對應公有子網路中進入的 Application Load Balancer 容量。
7. Application Load Balancer 透過其公有界面路由出流量。
8. 公有子網路的路由表具有指向網際網路閘道的預設路由，可將流量路由回網際網路。

哪種黏性策略適合您？

回答下列問題可協助您判斷負載平衡器黏性策略。

您是否使用 WebSockets？

- 是：WebSockets本身就黏性，因此不需要使用任何策略。
- 否：繼續下一個問題。

您是否正在規劃藍/綠部署？

- 是：至少使用目標群組黏性，但繼續下一個問題。
- 否：繼續下一個問題。

您需要用戶端的部分或全部流量重複路由到相同的 EC2 執行個體嗎？

- 是：繼續下一個問題。
- 否：您不需要使用黏性工作階段。

您是否希望應用程式程式碼或用戶端使用 Cookie 控制狀態屬性和持續時間？

- 是：使用應用程式型 Cookie 來啟用應用程式型黏性工作階段。
- 否：使用負載平衡器產生的 Cookie 來啟用以持續時間為基礎的黏性工作階段。

沒有黏性的 Application Load Balancer

當您在沒有任何形式的黏性的情況下使用 Application Load Balancer 時，根據預設，負載平衡器會使用循環配置方法來判斷其應路由流量的 EC2 執行個體。

範本：使用 CloudFormation 範本 `basic.yml`（包含在範例程式碼 .zip 檔案中) 來嘗試此功能。

> **Note**
>
> 本指南包含的所有 CloudFormation 範本都會部署自訂 VPC、路由表、路由、網際網路閘道、Application Load Balancer、目標群組、接聽程式和 EC2 執行個體，以說明特定的負載平衡器黏性策略。

常用案例

在這些情況下，使用 Application Load Balancer 而不黏附：

- 您有目標清單可路由流量，但目標不會維持工作階段狀態。
- 您使用的 Web 伺服器未維持工作階段狀態。
- 您使用的應用程式伺服器未維持工作階段狀態。

步驟

> **備註**
>
> - NAT 閘道會產生很小的成本。
> - 多個 EC2 執行個體比單一 EC2 執行個體更快地使用您的免費方案時數。

1. 在`basic.yml`實驗室環境中部署 CloudFormation 範本。
2. 等到目標群組執行個體的運作狀態從初始變更為正常運作。
3. 使用 HTTP (TCP/80) 導覽至 Web 瀏覽器中的 Application Load Balancer URL。

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

網頁會顯示執行個體 1 - TG1 或執行個體 2 - TG1。

4. 多次重新整理頁面。

預期結果

載入網頁的執行個體 (執行個體 1 或執行個體 2) 應每次變更，如頁面文字中所反映。負載平衡器邏輯會跨多個內部節點管理最後一個目標，這可能會導致同步延遲，因此您可能會路由到相同的目標。

運作方式

- 在此範例中，會將兩個 EC2 執行個體指派給單一目標群組。EC2 執行個體已安裝 Apache Web 伺服器 (`httpd`)，且每個 EC2 執行個體上的`index.html`頁面文字都經過硬式編碼以識別該執行個體。
- Application Load Balancer 會執行其內部循環配置邏輯，以判斷應接收流量的 EC2 執行個體。
- 每次重新載入網頁時，Application Load Balancer 都會執行其路由邏輯，而頁面會顯示執行個體 1 - TG1 或執行個體 2 - TG1。

目標群組黏性

當您使用具有目標群組黏性的 Application Load Balancer 時：

- Application Load Balancer 使用目標群組權重來判斷如何在目標群組之間平衡傳入流量。
- 根據預設，Application Load Balancer 會使用循環配置方法來將請求路由至目的地目標群組中的 EC2 執行個體。

範本：使用 CloudFormation 範本 `targetgroupstickiness.yml`（包含在範例程式碼 .zip 檔案中）來嘗試目標群組黏性。

常用案例

在這些案例中使用目標群組黏性：

- 有多個目標群組指派給負載平衡器，且來自用戶端的流量應一致路由至該目標群組內的執行個體。
- 藍/綠部署。

來自 basic.yml 的程式碼變更

已對接聽程式進行單一變更：我們修改了 Application Load Balancer 預設動作，以指定兩個同等權重的目標群組 (`TG1` 和 `TG2`)，並具有黏性組態。

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
```

步驟

> **備註**
>
> - NAT 閘道會產生很小的成本。
> - 多個 EC2 執行個體比單一 EC2 執行個體更快地使用您的免費方案時數。

1. 在`targetgroupstickiness.yml`實驗室環境中部署 CloudFormation 範本。
2. 等到目標群組執行個體的運作狀態從初始變更為正常運作。
3. 使用 HTTP (TCP/80) 導覽至 Web 瀏覽器中的 Application Load Balancer URL。

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

   網頁會顯示下列其中一項：執行個體 1 - TG1、執行個體 2 - TG1、執行個體 3 - TG2 或執行個體 4 - TG2。
4. 多次重新整理頁面。

預期結果

> **Note**
>
> 此範例中的 CloudFormation 範本會將黏性設定為持續 10 秒。

載入網頁的執行個體應該在 10 秒內保留在目標群組 (TG1 或 TG2) 內，如頁面文字中所反映。

大約 10 秒後，黏性就會釋放，目標群組執行個體集可能會變更。

運作方式

- 在此範例中，四個 EC2 執行個體會分割為兩個目標群組，每個目標群組有兩個執行個體。EC2 執行個體已安裝 Apache Web 伺服器 (`httpd`)，且每個 EC2 執行個體上的`index.html`頁面文字都經過硬式編碼以區分。
- Application Load Balancer 會建立使用者工作階段對目的地目標群組的繫結，並具有過期時間。
- 當您重新載入頁面時，Application Load Balancer 會檢查繫結是否存在且尚未過期。
  - 如果繫結已過期或不存在，Application Load Balancer 會執行其路由邏輯並決定目的地目標群組。
  - 如果繫結尚未過期，Application Load Balancer 會將流量路由至相同的目標群組，但不一定會路由至相同的 EC2 執行個體。

具有負載平衡器產生的 Cookie 的黏性工作階段

當您使用 Application Load Balancer 搭配負載平衡器產生的 Cookie 時：

- Application Load Balancer 使用目標群組權重來判斷如何在目標群組之間平衡傳入流量。
- 根據預設，Application Load Balancer 會使用循環配置方法來將請求路由至目的地目標群組中的 EC2 執行個體。

  在流量最初路由到執行個體之後，後續流量會在指定的持續時間內黏附至該 EC2 執行個體。

範本：使用 CloudFormation 範本 `stickysessionslb.yml`（包含在範例程式碼 .zip 檔案中），嘗試使用負載平衡器產生的 Cookie 進行黏性工作階段。

常用案例

在這些案例中，使用黏性工作階段搭配負載平衡器產生的 Cookie：

- PHP Web 伺服器
- 維護臨時工作階段資料的伺服器，例如日誌、購物車或聊天對話

來自 basic.yml 的程式碼變更

相關程式碼變更位於目標群組組態中，以將黏性類型設定為 `lb_cookie`，持續時間設定為 10 秒。

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
```

步驟

> **備註**
>
> - NAT 閘道會產生很小的成本。
> - 多個 EC2 執行個體使用免費方案時數的速度會比單一 EC2 執行個體快。

1. 在`stickysessionslb.yml`實驗室環境中部署 CloudFormation 範本。
2. 等待目標群組執行個體的運作狀態從初始變更為正常運作。
3. 使用 HTTP (TCP/80) 導覽至 Web 瀏覽器中的 Application Load Balancer URL。

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

   網頁會顯示下列其中一項：執行個體 1 - TG1、執行個體 2 - TG1、執行個體 3 - TG2 或執行個體 4 - TG1。
4. 多次重新整理頁面。

預期結果

> **Note**
>
> 此範例中的 CloudFormation 範本會將黏性設定為持續 10 秒。

載入網頁的執行個體應在 10 秒內保持不變，如頁面文字中所反映。大約 10 秒後，粘性就會釋放，而目的地執行個體可能會變更。

運作方式

- 在此範例中，一個目標群組中存在兩個 EC2 執行個體。EC2 執行個體已安裝 Apache Web 伺服器 (`httpd`)，且每個 EC2 執行個體上的`index.html`頁面文字都經過硬式編碼，使其不同。
- Application Load Balancer 會建立使用者工作階段的繫結，其會繫結至目的地，並具有過期時間。
- 當您重新載入頁面時，Application Load Balancer 會檢查繫結是否存在且尚未過期。
  - 如果繫結已過期或不存在，Application Load Balancer 會執行其路由邏輯並決定目的地執行個體。
  - 如果繫結尚未過期，Application Load Balancer 會將流量路由到相同的目的地執行個體。

使用應用程式型 Cookie 的黏性工作階段

當您搭配應用程式型 Cookie 使用 Application Load Balancer 時：

- Application Load Balancer 使用目標群組權重來判斷如何在目標群組之間平衡傳入流量。
- 根據預設，Application Load Balancer 會使用循環配置方法來將請求路由至目的地目標群組中的 EC2 執行個體。
- 流量最初路由至 EC2 執行個體後，EC2 執行個體應用程式回應應包含自訂應用程式 Cookie，該 Cookie 會與自動化 Application Load Balancer Cookie 一起傳回用戶端。
- 如果用戶端傳送傳回應用程式 Cookie 和 Application Load Balancer Cookie，後續流量將黏附至 EC2 執行個體。
- 以應用程式為基礎的 Cookie 在設定的持續時間不使用後過期。

範本：使用 CloudFormation 範本 `stickysessionsapp.yml` (包含在範例程式碼 .zip 檔案中) 來嘗試使用應用程式型 Cookie 的黏性工作階段。

常用案例

當您想要在這些案例中進行其他控制時，請使用黏性工作階段搭配應用程式產生的 Cookie：

- PHP Web 伺服器
- 維護臨時工作階段資料的伺服器，例如日誌、購物車或聊天對話

來自 basic.yml 的程式碼變更

唯一的程式碼變更是在目標群組組態中。我們已將黏性組態新增至 Application Load Balancer 和目標群組屬性。應用程式 Cookie 持續時間已指定，目標群組已啟用應用程式 Cookie 黏性。

basic.yml	**stickysessionsapp.yml**

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

```
      Name: TG1
      Protocol: HTTP
      Port: 80
      TargetType: instance
      Targets:
        - Id: !Ref Instance1
        - Id: !Ref Instance2
VpcId: !Ref CustomVPC
```

```
    Name: TG1
    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
```

步驟

> **備註**
>
> - NAT 閘道會產生很小的成本。
> - 多個 EC2 執行個體使用免費方案時數的速度會比單一 EC2 執行個體快。

1. 在`stickysessionslb.yml`實驗室環境中部署 CloudFormation 範本。
2. 等到目標群組執行個體的運作狀態從初始變更為正常運作。
3. 使用 HTTP (TCP/80) 導覽至 Web 瀏覽器中的 Application Load Balancer URL。

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

   網頁會顯示下列其中一項：執行個體 1 - TG1、執行個體 2 - TG1、執行個體 3 - TG2 或執行個體 4 - TG2。
4. 多次重新整理頁面。

預期結果

> **Note**
>
> 此範例中的 CloudFormation 範本已將黏性設定為持續 10 秒。有效的黏性持續時間組態介於 1 秒到 1 週之間。

載入網頁的執行個體應保持不變，如頁面文字所示。

黏性持續時間不會重新整理，但是根據 EC2 執行個體所產生之應用程式 Cookie 在 Application Load Balancer 中設定的過期時間。

範例 1：等待 5 秒以重新整理頁面。將載入相同的執行個體，並再重新整理黏性 10 秒。

範例 2：等待超過 10 秒以重新載入頁面。應用程式 Cookie 會過期，而您會被路由到不同的 EC2 執行個體。這個新執行個體會產生另一個應用程式 Cookie，持續時間為 10 秒。

運作方式

- 在此範例中，EC2 執行個體已安裝 Apache Web 伺服器 (httpd)。`httpd.conf` 檔案設定為將靜態`Set-Cookie`值傳回用戶端 （ 您的 Web 瀏覽器 ）。`Set-Cookie` 值硬式編碼為 `TESTCOOKIE=<somevalue>`。
- 開啟瀏覽器的檢查元素選項，選擇網路索引標籤，然後選擇 Get 方法，以載入頁面。您將看到 Cookie 索引標籤。
- 瀏覽器是一種用戶端應用程式，會自動設定 ，以使用伺服器`Set-Cookie`回應中收到的 Cookie 傳回任何後續更新至伺服器。
- 當您重新載入頁面時，初始頁面載入中收到的 Cookie 會自動傳回 Application Load Balancer。
  - 如果 Cookie 已過期 （ 亦即，自上次呼叫後已經過 10 秒 ），Application Load Balancer 會使用新的邏輯來判斷要將流量路由到哪個 EC2 執行個體。
  - 如果 Cookie 尚未過期，Application Load Balancer 會將流量路由到相同的 EC2 執行個體。

Resources

- Application Load Balancer 的黏性工作階段 (Elastic Load Balancing 文件)

文件歷史紀錄

下表描述了本指南的重大變更。如果您想收到有關未來更新的通知，可以訂閱 RSS 摘要。

變更	描述	日期
更新章節	以新的圖表和說明更新負載平衡器子網路和路由區段。	2024 年 7 月 5 日
更新與修正內容	更新程式碼、圖表和範例。	2023 年 5 月 31 日
更新章節	更新了其在目標群組黏性與具有負載平衡器產生 Cookie 的黏性工作階段中的運作方式章節，以提供更準確且詳細資訊。 https://docs.aws.amazon.com/prescriptive-guidance/latest/load-balancer-stickiness/target-group-stickiness.html https://docs.aws.amazon.com/prescriptive-guidance/latest/load-balancer-stickiness/alb-cookies-stickiness.html	2022 年 7 月 18 日
—	初次出版	2021 年 8 月 5 日

本文為英文版的機器翻譯版本，如內容有任何歧義或不一致之處，概以英文版為準。