Elegir una estrategia de adherencia para tu balanceador de carga

AWS Orientación prescriptiva

AWS Orientación prescriptiva: Elegir una estrategia de adherencia para tu balanceador de carga

Table of Contents

Elegir una estrategia de adherencia para tu balanceador de carga

Ryan Griffin, Amazon Web Services ()AWS

Julio de 2024 (historial del documento)

La rigidez es un término que se utiliza para describir la funcionalidad de un balanceador de cargas para enrutar repetidamente el tráfico desde un cliente a un único destino, en lugar de equilibrar el tráfico entre varios destinos. Por ejemplo, el tráfico del cliente A se puede enrutar continuamente a un servidor específico, de modo que el servidor pueda mantener los datos del estado de la sesión. Si el tráfico del cliente A se enruta a dos servidores distintos, es posible que a cada servidor le falte información importante que está disponible para el otro servidor.

Por lo tanto, a menudo es necesario mantener una conexión de cliente coherente a través de un equilibrador de carga. Existen dos tipos de rigidez: la persistencia en las sesiones y la persistencia en el grupo objetivo.

- Sesiones fijas: mantenimiento de los datos de las sesiones locales en una instancia de Amazon Elastic Compute Cloud (Amazon EC2) para simplificar la arquitectura de la aplicación o mejorar el rendimiento de la aplicación, ya que la instancia puede mantener o almacenar en caché la información del estado de la sesión de forma local. AWS Actualmente, ofrece dos tipos de sesiones fijas, que se analizan en detalle en esta guía: las cookies de aplicación y las cookies del equilibrador de carga.
- Fijación del grupo objetivo: en las implementaciones en azul o verde, es posible que tenga implementadas varias versiones de una aplicación y que desee que el cliente siga utilizando la misma versión de la aplicación durante la sesión. En este caso, puede utilizar la fidelidad del grupo de destino para enrutar todas las comunicaciones del cliente al mismo grupo de destino en lugar de a la misma instancia. EC2

Puedes usar estas dos estrategias de adherencia por separado o juntas.

En esta guía, se describen los diferentes tipos de rigidez de los balanceadores de carga y los casos de uso aplicables, para ayudarte a elegir una estrategia. La guía incluye AWS CloudFormation plantillas que ilustran cada estrategia.

Código de muestra

En esta guía se incluye un archivo.zip adjunto que incluye cuatro AWS CloudFormation plantillas que puede implementar para crear una arquitectura básica y probar cada estrategia de rigor. Le recomendamos que implemente estas plantillas en un entorno de laboratorio para probar cada enfoque.

Descar

el código de muestra

La descarga incluye estas plantillas:

- `basic.yml`— Configura un Application Load Balancer sin problemas.
- `targetgroupstickiness.yml`— Demuestra adherencia en función de los grupos objetivo.
- `stickysessionslb.yml`— Muestra sesiones fijas con cookies generadas por el balanceador de carga.
- `stickysessionsapp.yml`— Muestra sesiones fijas con cookies basadas en aplicaciones.

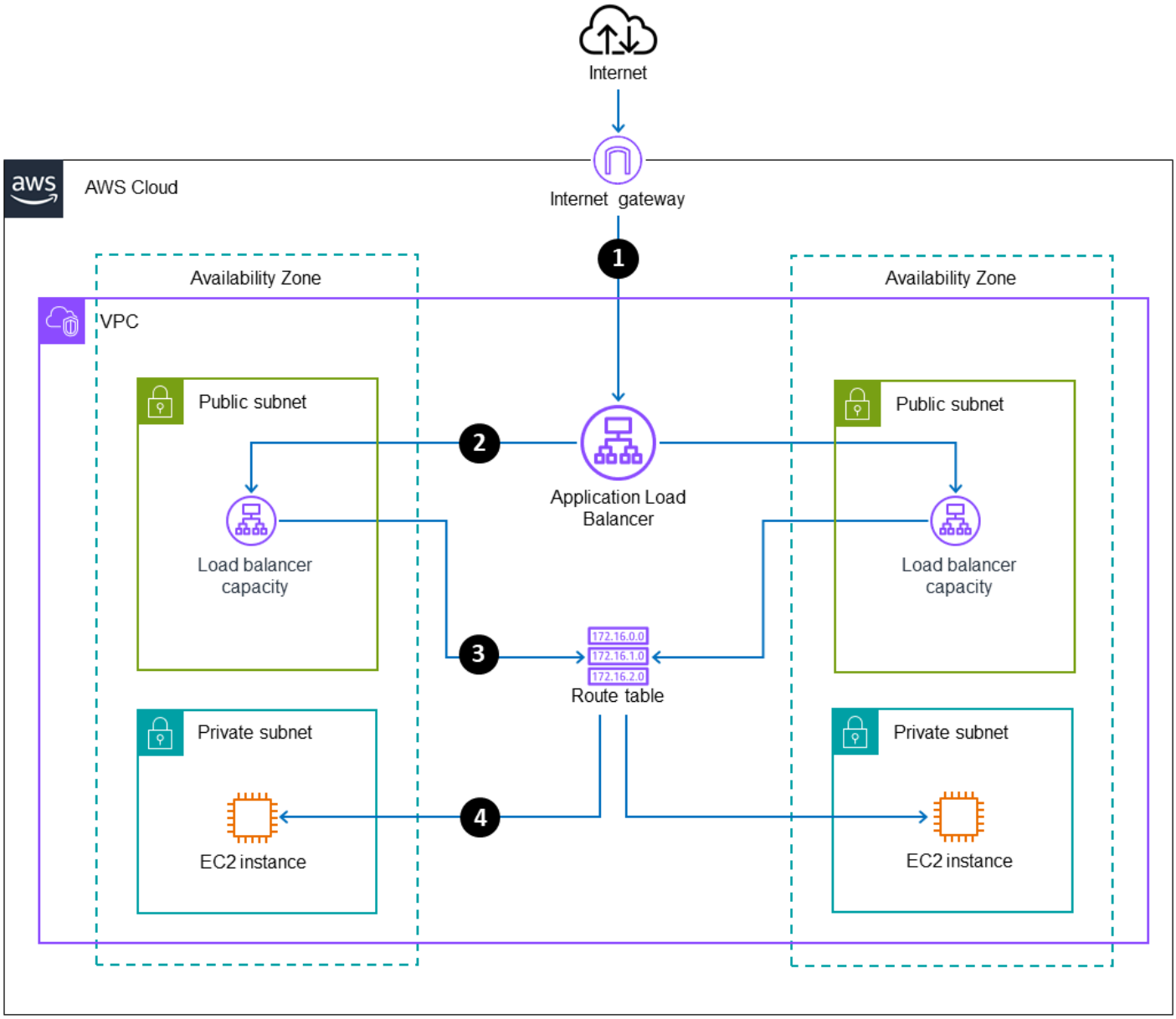
Para implementar estas plantillas, necesitará una AWS cuenta activa y acceso a la CloudFormation consola. Para step-by-steps obtener instrucciones sobre cómo implementar una CloudFormation plantilla, consulte Crear una pila en la CloudFormation documentación.

Subredes y enrutamiento del balanceador de carga

Comencemos con una ilustración de cómo fluye el tráfico cuando se configura un Application Load Balancer orientado a Internet hacia las instancias de Amazon Elastic Compute Cloud (Amazon EC2) en una subred privada. Esta arquitectura refleja las mejores prácticas a la hora de implementar un Application Load Balancer abierto a Internet.

Ruta de tráfico entrante

El siguiente diagrama ilustra las subredes y el enrutamiento de la nube privada virtual (VPC) asociados al flujo de tráfico entrante, y el tráfico de retorno se ha eliminado del diagrama para mayor claridad.

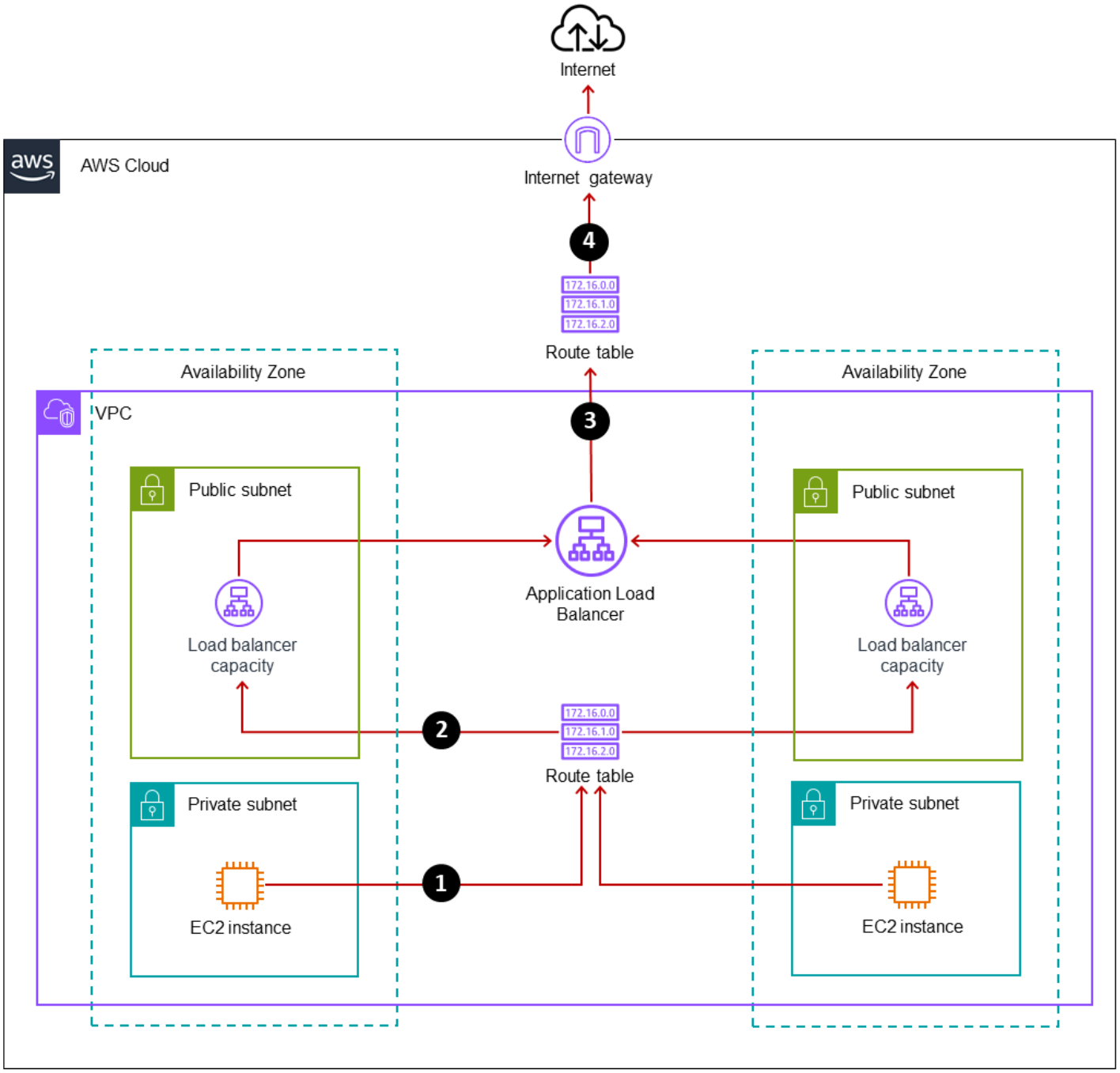


1. El tráfico de Internet fluye hacia el nombre DNS de Application Load Balancer.
2. El Application Load Balancer está asociado a dos subredes públicas en el escenario que se ilustra. El servicio Elastic Load Balancing crea una capacidad de balanceador de carga en cada zona de disponibilidad habilitada y envía el tráfico a través de la zona de disponibilidad para determinar la lógica de enrutamiento adecuada. El Application Load Balancer utiliza su lógica interna para determinar a qué grupo objetivo e instancia dirigir el tráfico.
3. El Application Load Balancer enruta la solicitud a la instancia EC2 a través de un nodo que está asociado a la subred pública de la misma zona de disponibilidad. (El servicio Elastic Load Balancing configura, administra y escala estos nodos y los usuarios no los ven).

4. La tabla de enrutamiento enruta el tráfico localmente dentro de la VPC, entre la subred pública y la subred privada, y a la instancia EC2.

Ruta de tráfico de retorno

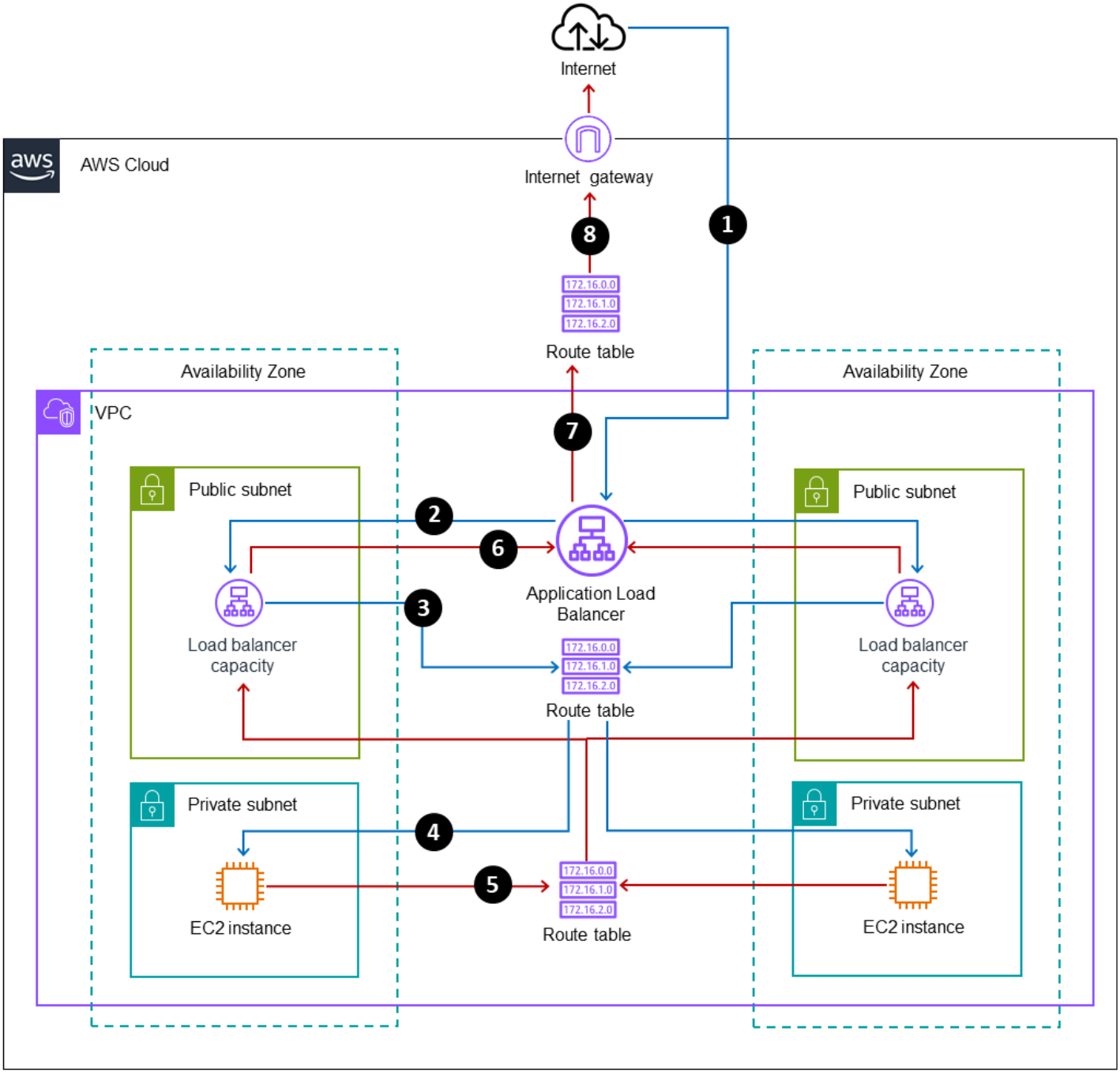
El siguiente diagrama ilustra las subredes de VPC y el enrutamiento asociados a la ruta de tráfico de regreso a Internet, y el tráfico entrante se ha eliminado del diagrama para mayor claridad.



1. La instancia EC2 de la subred privada enruta el tráfico saliente a través de la tabla de enrutamiento.
2. La tabla de enrutamiento tiene una ruta local a la subred pública. Alcanza la capacidad de Application Load Balancer en la que ingresó el tráfico, en la subred pública correspondiente, siguiendo la ruta de regreso al lugar en el que ingresó el tráfico.
3. El Application Load Balancer redirige el tráfico a través de su interfaz pública.
4. La tabla de enrutamiento de la subred pública tiene una ruta predeterminada que apunta a una puerta de enlace a Internet, que dirige el tráfico de vuelta a Internet.

Diagrama de flujo de tráfico completo

El siguiente diagrama combina los flujos de tráfico entrante y de retorno para proporcionar una ilustración completa del enrutamiento del balanceador de carga.



1. El tráfico de Internet fluye hacia el nombre DNS de Application Load Balancer.
2. El Application Load Balancer está asociado a dos subredes públicas en el escenario que se ilustra. El servicio Elastic Load Balancing crea una capacidad de balanceador de carga en cada zona de disponibilidad habilitada y envía el tráfico a través de la zona de disponibilidad para determinar la lógica de enrutamiento adecuada. El Application Load Balancer utiliza su lógica interna para determinar a qué grupo objetivo e instancia dirigir el tráfico.

3. El Application Load Balancer enruta la solicitud a la instancia EC2 a través de un nodo que está asociado a la subred pública de la misma zona de disponibilidad. (El servicio Elastic Load Balancing configura, administra y escala estos nodos y los usuarios no los ven).
4. La tabla de enrutamiento enruta el tráfico localmente dentro de la VPC, entre la subred pública y la subred privada, y a la instancia EC2.
5. La instancia EC2 de la subred privada enruta el tráfico saliente a través de la tabla de enrutamiento.
6. La tabla de enrutamiento tiene una ruta local a la subred pública. Alcanza la capacidad de Application Load Balancer en la que ingresó el tráfico, en la subred pública correspondiente, siguiendo la ruta de regreso al lugar en el que ingresó el tráfico.
7. El Application Load Balancer redirige el tráfico a través de su interfaz pública.
8. La tabla de enrutamiento de la subred pública tiene una ruta predeterminada que apunta a una puerta de enlace a Internet, que dirige el tráfico de vuelta a Internet.

¿Qué estrategia de adherencia es la adecuada para ti?

Responder a las siguientes preguntas te ayudará a determinar tu estrategia de rigidez para el balanceador de carga.

¿Lo estás usando? WebSockets

- Sí: WebSockets son intrínsecamente pegajosos, por lo que no es necesario utilizar ninguna estrategia.
- No: continúe con la siguiente pregunta.

¿Está planificando un despliegue azul/verde?

- Sí: como mínimo, utilice la fidelidad al grupo objetivo, pero continúe con la siguiente pregunta.
- No: continúe con la siguiente pregunta.

¿Necesita que parte o todo el tráfico de un cliente se enrute a la misma EC2 instancia repetidamente?

- Sí: continúe con la siguiente pregunta.
- No: no es necesario utilizar sesiones fijas.

¿Desea que el código de su aplicación o su cliente controlen los atributos de estado y la duración mediante el uso de cookies?

- Sí: utilice cookies basadas en aplicaciones para habilitar las sesiones fijas basadas en aplicaciones.
- No: usa cookies generadas por el balanceador de carga para habilitar las sesiones fijas basadas en la duración.

Aplicación: Load Balancer sin pegajosidad

Cuando utilizas un Application Load Balancer sin ningún tipo de rigidez, de forma predeterminada, el balanceador de cargas usa el método por turnos para determinar la EC2 instancia a la que debe enrutar el tráfico.

Plantilla: usa la CloudFormation plantilla `basic.yml` (incluida en el archivo.zip de código de ejemplo) para probar esta funcionalidad.

> **Note**
>
> Todas las CloudFormation plantillas incluidas en esta guía implementan una VPC personalizada, tablas de enrutamiento, rutas, una puerta de enlace a Internet, un Application Load Balancer, grupos objetivo, oyentes EC2 e instancias para ilustrar una estrategia específica de adherencia del balanceador de carga.

Casos de uso comunes

Utilice un Application Load Balancer sin complicaciones en los siguientes escenarios:

- Tiene una lista de destinos a los que dirigir el tráfico, pero los destinos no mantienen el estado de la sesión.
- Está utilizando servidores web que no mantienen el estado de la sesión.
- Está utilizando servidores de aplicaciones que no mantienen el estado de la sesión.

Pasos

> **Notas**
>
> - Las pasarelas NAT tienen un coste reducido.
> - Varias EC2 instancias consumen las horas de la capa gratuita más rápido que una sola EC2 instancia.

1. Implemente la CloudFormation plantilla `basic.yml` en un entorno de laboratorio.

2. Espere a que el estado de salud de las instancias del grupo objetivo cambie del estado inicial al correcto.
3. Navegue hasta la URL de Application Load Balancer en un navegador web mediante HTTP (TCP/80).

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

   La página web muestra la Instancia 1 (TG1o Instancia 2). TG1
4. Actualice la página varias veces.

Resultados esperados

La instancia que carga la página web (Instancia 1 o Instancia 2) debería cambiar cada vez, como se refleja en el texto de la página. La lógica del equilibrador de carga gestiona el último objetivo en varios nodos internos, lo que puede provocar un retraso en la sincronización, por lo que existe la posibilidad de que te dirijan al mismo destino.

Funcionamiento

- En este ejemplo, se asignan dos EC2 instancias a un único grupo objetivo. Las EC2 instancias tienen un servidor web Apache (`httpd`) instalado y el texto de la `index.html` página de cada EC2 instancia está codificado para identificarla.
- El Application Load Balancer ejecuta su lógica rotativa interna para determinar qué EC2 instancia debe recibir el tráfico.
- Cada vez que recarga la página web, Application Load Balancer ejecuta su lógica de enrutamiento y la página muestra la Instancia 1 (o Instancia 2) TG1. TG1

Fijación del grupo objetivo

Cuando utilizas un Application Load Balancer con adherencia al grupo objetivo:

- El Application Load Balancer utiliza el peso del grupo objetivo para determinar cómo equilibrar el tráfico entrante entre los grupos objetivo.
- De forma predeterminada, Application Load Balancer usa el método round robin para enrutar las solicitudes a las EC2 instancias del grupo objetivo de destino.

Plantilla: usa la CloudFormation plantilla `targetgroupstickiness.yml` (incluida en el archivo.zip de código de ejemplo) para probar la fidelidad del grupo objetivo.

Casos de uso comunes

Utilice la fidelidad del grupo objetivo en los siguientes escenarios:

- Hay varios grupos de destino asignados al balanceador de cargas y el tráfico de un cliente debe enrutarse de manera coherente a las instancias de ese grupo de destino.
- Implementaciones azul/verde.

Cambios de código con respecto a basic.yml

Se ha realizado un único cambio en el listener: modificamos las acciones predeterminadas de Application Load Balancer para especificar dos grupos objetivo `TG1` (`TG2y`) de igual peso, con una configuración estricta.

basic.yml

```
Listener1:
  Type: 'AWS::ElasticLoadBalancingV
2::Listener'
  Properties:
    LoadBalancerArn: !Ref ALB
    Protocol: HTTP
    Port: 80
```

targetgroupstickiness.yml

```
Listener1:
  Type: 'AWS::ElasticLoadBalancingV
2::Listener'
  Properties:
    LoadBalancerArn: !Ref ALB
    Protocol: HTTP
    Port: 80
```

```
    DefaultActions:
      - TargetGroupArn: !Ref TG1
        Type: forward
```

```
    DefaultActions:
      - ForwardConfig:
          TargetGroups:
            - TargetGroupArn: !Ref TG1
              Weight: 1
            - TargetGroupArn: !Ref TG2
              Weight: 1
          TargetGroupStickinessConfig
:
              DurationSeconds: 10
              Enabled: true
        Type: forward
```

Pasos

> **Notas**
>
> - Las pasarelas NAT tienen un coste reducido.
> - Varias EC2 instancias consumen las horas de la capa gratuita más rápido que una sola EC2 instancia.

1. Implemente la CloudFormation plantilla `targetgroupstickiness.yml` en un entorno de laboratorio.
2. Espere a que el estado de salud de las instancias del grupo objetivo cambie del estado inicial al correcto.
3. Navegue hasta la URL de Application Load Balancer en un navegador web mediante HTTP (TCP/80).

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

   La página web muestra una de las siguientes opciones: Instancia 1 TG1, Instancia 2 TG1, Instancia 3 o Instancia 4. TG2 TG2
4. Actualice la página varias veces.

Resultados esperados

> **Note**
>
> La CloudFormation plantilla de este ejemplo configura la adherencia para que dure 10 segundos.

Las instancias que cargan la página web deben permanecer dentro del grupo objetivo (TG1 o TG2) dentro de los 10 segundos de duración, tal y como se refleja en el texto de la página.

Al cabo de aproximadamente 10 segundos, se elimina el problema y es posible que el conjunto de instancias del grupo objetivo cambie.

Funcionamiento

- En este ejemplo, cuatro EC2 instancias se dividen en dos grupos de destino, con dos instancias por grupo de destino. Las EC2 instancias tienen un servidor web Apache (`httpd`) instalado y el texto de la `index.html` página de cada EC2 instancia está codificado de forma que sea distinto.
- El Application Load Balancer crea un enlace para la sesión del usuario con el grupo objetivo de destino, con una fecha de caducidad.
- Al volver a cargar la página, el Application Load Balancer comprueba si el enlace existe y no ha caducado.
  - Si el enlace ha caducado o no existe, Application Load Balancer ejecuta su lógica de enrutamiento y determina el grupo objetivo de destino.
  - Si el enlace no ha caducado, Application Load Balancer dirige el tráfico al mismo grupo de destino, pero no necesariamente a la misma EC2 instancia.

Sesiones fijas con cookies generadas por el balanceador de carga

Cuando utilizas un Application Load Balancer con una cookie generada por un balanceador de carga:

- El Application Load Balancer utiliza el peso del grupo objetivo para determinar cómo equilibrar el tráfico entrante entre los grupos objetivo.
- De forma predeterminada, Application Load Balancer usa el método round robin para enrutar las solicitudes a las EC2 instancias del grupo objetivo de destino.

  Una vez que el tráfico se haya enrutado inicialmente a una instancia, el tráfico subsiguiente se quedará en esa EC2 instancia durante un período específico.

Plantilla: usa la CloudFormation plantilla `stickysessionslb.yml` (incluida en el archivo.zip de código de muestra) para probar sesiones fijas con cookies generadas por el balanceador de carga.

Casos de uso comunes

Usa sesiones fijas con cookies generadas por el balanceador de carga en los siguientes escenarios:

- Servidores web PHP
- Servidores que mantienen datos de sesión temporales, como registros, carritos de compras o conversaciones de chat

El código cambia de basic.yml

Los cambios de código relevantes se encuentran en la configuración del grupo objetivo, para establecer el tipo de adherencia `lb_cookie` y la duración en 10 segundos.

basic.yml

```
TG1:
   Type: 'AWS::ElasticLoadBalancingV
2::TargetGroup'
```

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
```

```
    Properties:
      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: lb_cookie
        - Key: stickiness.lb_cook
ie.duration_seconds
Value: 10
```

Pasos

Notas

- Las pasarelas NAT tienen un coste reducido.
- Varias EC2 instancias consumirán las horas de la capa gratuita más rápido que una sola EC2 instancia.

1. Implemente la CloudFormation plantilla `stickysessionslb.yml` en un entorno de laboratorio.
2. Espere a que el estado de salud de las instancias del grupo objetivo cambie del estado inicial al correcto.
3. Navegue hasta la URL de Application Load Balancer en un navegador web mediante HTTP (TCP/80).

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

   La página web muestra una de las siguientes opciones: Instancia 1 TG1, Instancia 2 TG1, Instancia 3 o Instancia 4. TG2 TG1
4. Actualice la página varias veces.

Resultados esperados

> **Note**
>
> La CloudFormation plantilla de este ejemplo configura la adherencia para que dure 10 segundos.

La instancia que carga la página web debe permanecer igual dentro de los 10 segundos de duración, tal y como se refleja en el texto de la página. Después de aproximadamente 10 segundos, se elimina la rigidez y la instancia de destino podría cambiar.

Funcionamiento

- En este ejemplo, hay dos EC2 instancias en un grupo objetivo. Las EC2 instancias tienen un servidor web Apache (`httpd`) instalado y el texto de la `index.html` página de cada EC2 instancia está codificado de forma que sea distinto.
- El Application Load Balancer crea un enlace para la sesión del usuario, que se enlaza con el destino, con una fecha de caducidad.
- Al volver a cargar la página, el Application Load Balancer comprueba si el enlace existe y no ha caducado.
  - Si el enlace ha caducado o no existe, Application Load Balancer ejecuta su lógica de enrutamiento y determina la instancia de destino.
  - Si el enlace no ha caducado, Application Load Balancer dirige el tráfico a la misma instancia de destino.

Sesiones fijas con cookies basadas en aplicaciones

Cuando utilizas un Application Load Balancer con una cookie basada en aplicaciones:

- El Application Load Balancer utiliza el peso del grupo objetivo para determinar cómo equilibrar el tráfico entrante entre los grupos objetivo.
- De forma predeterminada, Application Load Balancer usa el método round robin para enrutar las solicitudes a las EC2 instancias del grupo objetivo de destino.
- Una vez que el tráfico se haya enrutado inicialmente a una EC2 instancia, la respuesta de la aplicación de la EC2 instancia debe contener una cookie de aplicación personalizada, que se devuelve al cliente junto con una cookie automática de Application Load Balancer.
- El tráfico posterior se quedará en la EC2 instancia si el cliente devuelve la cookie de la aplicación y la cookie Application Load Balancer.
- La cookie basada en la aplicación caduca cuando no se utiliza durante el tiempo configurado.

Plantilla: utilice la CloudFormation plantilla `stickysessionsapp.yml` (incluida en el archivo.zip de código de muestra) para probar sesiones fijas con cookies basadas en aplicaciones.

Casos de uso comunes

Utilice sesiones fijas con cookies generadas por la aplicación cuando desee tener un control adicional en estos escenarios:

- Servidores web PHP
- Servidores que mantienen datos de sesión temporales, como registros, carritos de compras o conversaciones de chat

El código cambia de basic.yml

El único cambio de código se produce en la configuración del grupo objetivo. Añadimos una configuración de adherencia a los atributos de Application Load Balancer y del grupo objetivo. Se especifica la duración de las cookies de la aplicación y el grupo objetivo tiene habilitada la persistencia de las cookies de la aplicación.

basic.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
```

stickysessionsapp.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
    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
```

Pasos

Notas

- Las pasarelas NAT tienen un coste reducido.
- Varias EC2 instancias consumirán las horas de la capa gratuita más rápido que una sola EC2 instancia.

1. Implemente la CloudFormation plantilla `stickysessionslb.yml` en un entorno de laboratorio.
2. Espere a que el estado de salud de las instancias del grupo objetivo cambie del estado inicial al correcto.

3. Navegue hasta la URL de Application Load Balancer en un navegador web mediante HTTP (TCP/80).

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

   La página web muestra una de las siguientes opciones: Instancia 1 TG1, Instancia 2 TG1, Instancia 3 o Instancia 4. TG2 TG2
4. Actualice la página varias veces.

Resultados esperados

> **Note**
>
> La CloudFormation plantilla de este ejemplo ha configurado la adherencia para que dure 10 segundos. La configuración válida de duración de la adherencia es de entre 1 segundo y 1 semana.

La instancia en la que se carga la página web debe permanecer igual, como lo indica el texto de la página.

La duración de la persistencia no se actualiza, sino que se basa en la caducidad configurada en el Application Load Balancer para la cookie de aplicación que genera la instancia. EC2

Ejemplo 1: espere 5 segundos para actualizar la página. Se cargará la misma instancia y la adherencia se actualizará durante otros 10 segundos.

Ejemplo 2: espera más de 10 segundos para volver a cargar la página. La cookie de la aplicación caduca y se le redirige a una instancia diferente EC2 . Esta nueva instancia genera otra cookie de aplicación con una duración de 10 segundos.

Funcionamiento

- En este ejemplo, las EC2 instancias tienen instalado un servidor web Apache (`httpd`). El `httpd.conf` archivo está configurado para devolver un `Set-Cookie` valor estático al cliente (su navegador web). El `Set-Cookie` valor está codificado para que sea. `TESTCOOKIE=<somevalue>`

- Abre la opción Inspeccionar elemento del navegador, selecciona la pestaña Red y, a continuación, elige el método Get, que carga la página. Verás una pestaña de cookies.
- El navegador es una aplicación cliente que se configura automáticamente para devolver cualquier actualización posterior al servidor con las cookies que recibe en la `Set-Cookie` respuesta del servidor.
- Al volver a cargar la página, las cookies recibidas en la carga inicial de la página se devuelven automáticamente al Application Load Balancer.
  - Si la cookie ha caducado (es decir, han transcurrido 10 segundos desde la última llamada), Application Load Balancer utiliza una nueva lógica para determinar a EC2 qué instancia dirigir el tráfico.
  - Si la cookie no ha caducado, Application Load Balancer dirige el tráfico a la misma EC2 instancia.

Recursos

- Sesiones fijas para tu balanceador de carga de aplicaciones (documentación de Elastic Load Balancing)

Historial de documentos

En la siguiente tabla, se describen cambios significativos de esta guía. Si quiere recibir notificaciones de futuras actualizaciones, puede suscribirse a las notificaciones RSS.

Cambio	Descripción	Fecha
Se ha actualizado la sección	Se actualizó la sección de subredes y enrutamiento del balanceador de carga con nuevos diagramas y aclaraciones.	5 de julio de 2024
Actualizaciones y correcciones	Se actualizaron el código, los diagramas y los ejemplos.	31 de mayo de 2023
Secciones actualizadas	Se actualizaron las secciones Cómo funciona en las sesiones de Target group Stickess y Sticky con cookies generadas por el balanceador de carga para ofrecer información más precisa y detallada.	18 de julio de 2022
—	Publicación inicial	5 de agosto de 2021

Las traducciones son generadas a través de traducción automática. En caso de conflicto entre la traducción y la version original de inglés, prevalecerá la version en inglés.