Escolhendo uma estratégia de aderência para seu balanceador de carga

AWS Recomendações

AWS Recomendações: Escolhendo uma estratégia de aderência para seu balanceador de carga

Table of Contents

Escolhendo uma estratégia de aderência para seu balanceador de carga

Ryan Griffin, Amazon Web Services ()AWS

Julho de 2024 (histórico do documento)

A aderência é um termo usado para descrever a funcionalidade de um balanceador de carga para rotear repetidamente o tráfego de um cliente para um único destino, em vez de equilibrar o tráfego em vários destinos. Por exemplo, o tráfego do cliente A pode ser roteado continuamente para um servidor específico, para que o servidor possa manter os dados do estado da sessão. Se o tráfego do cliente A for roteado para dois servidores distintos, cada servidor pode estar perdendo informações importantes que estão disponíveis para o outro servidor.

Portanto, muitas vezes é necessário manter uma conexão consistente com o cliente por meio de um balanceador de carga. Existem dois tipos de aderência: sessões fixas e aderência ao grupo-alvo.

- Sessões fixas — Manter os dados da sessão local em uma instância do Amazon Elastic Compute Cloud EC2 (Amazon) para simplificar a arquitetura do aplicativo ou melhorar o desempenho do aplicativo, porque a instância pode manter ou armazenar em cache as informações do estado da sessão localmente. AWS atualmente oferece dois tipos de sessões fixas, que este guia discute em detalhes: cookies de aplicativos e cookies de balanceamento de carga.
- Estabilidade do grupo-alvo — Em implantações em azul/verde, você pode ter várias versões de um aplicativo implantadas e talvez queira que o cliente continue usando a mesma versão do aplicativo durante a sessão. Nesse caso, você pode usar a aderência do grupo-alvo para rotear todas as comunicações do cliente para o mesmo grupo-alvo em vez da mesma EC2 instância.

Você pode usar essas duas estratégias de aderência separadamente ou em conjunto.

Este guia descreve os diferentes tipos de aderência do balanceador de carga e os casos de uso aplicáveis, para ajudar você a escolher uma estratégia. O guia inclui AWS CloudFormation modelos que ilustram cada estratégia.

Código de exemplo

Este guia fornece um arquivo.zip anexo que inclui quatro AWS CloudFormation modelos que você pode implantar para criar uma arquitetura básica e testar cada estratégia de aderência.

Recomendamos que você implante esses modelos em um ambiente de laboratório para testar cada abordagem.

Baixe o código de amostra

O download inclui esses modelos:

- `basic.yml`— Configura um Application Load Balancer sem aderência.
- `targetgroupstickiness.yml`— Demonstra aderência com base em grupos-alvo.
- `stickysessionslb.yml`— Demonstra sessões fixas com cookies gerados pelo balanceador de carga.
- `stickysessionsapp.yml`— Demonstra sessões persistentes com cookies baseados em aplicativos.

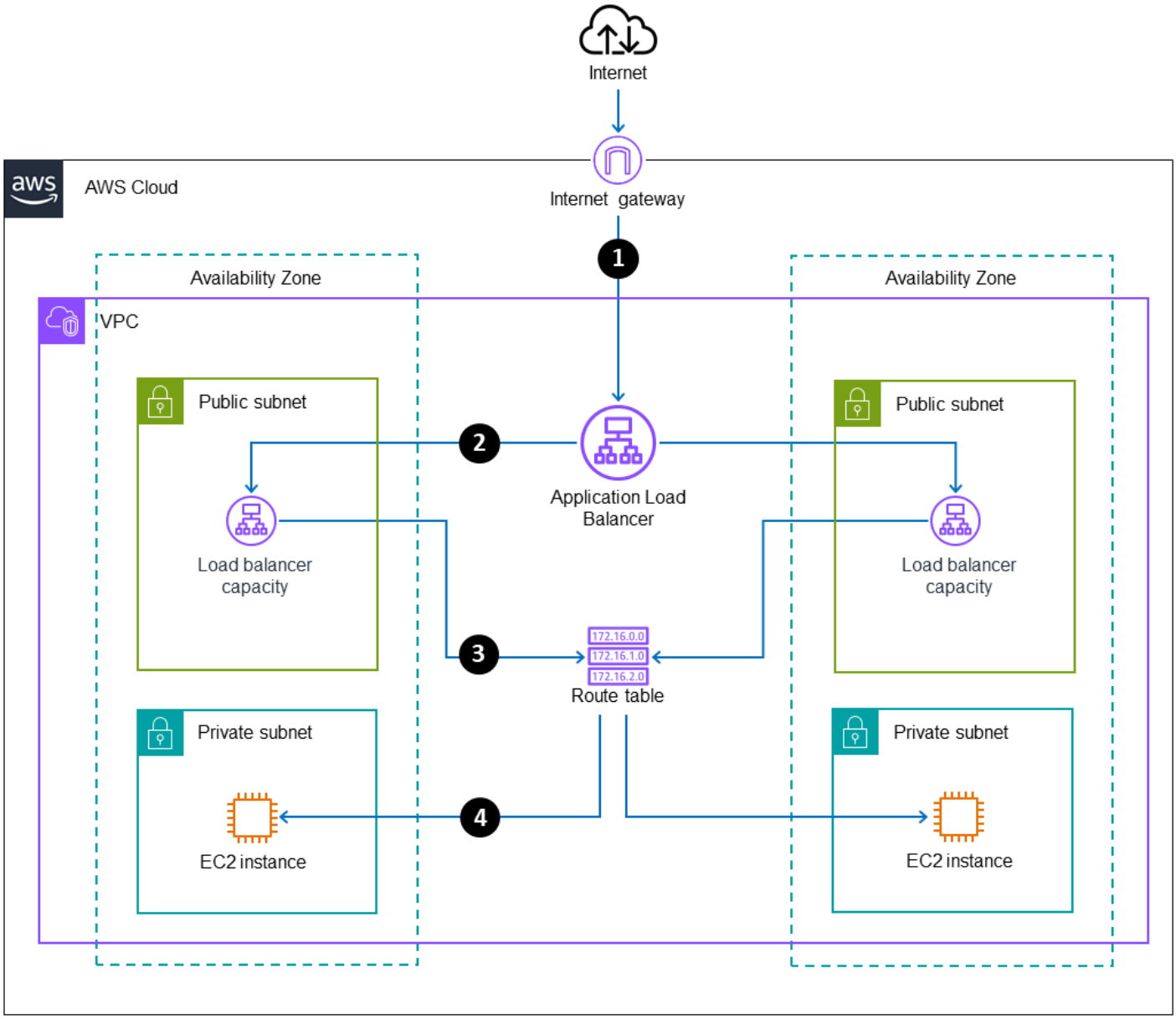
Para implantar esses modelos, você precisará de uma AWS conta ativa: e acesso ao CloudFormation console. Para step-by-steps obter instruções sobre como implantar um CloudFormation modelo, consulte Como criar uma pilha na CloudFormation documentação.

Sub-redes e roteamento do balanceador de carga

Vamos começar com uma ilustração de como o tráfego flui quando você configura um Application Load Balancer voltado para a Internet para instâncias do Amazon Elastic Compute Cloud (Amazon EC2) em uma sub-rede privada. Essa arquitetura reflete as melhores práticas ao implantar um Application Load Balancer aberto à Internet.

Caminho de tráfego de entrada

O diagrama a seguir ilustra as sub-redes e o roteamento da nuvem privada virtual (VPC) associados ao fluxo de tráfego de entrada, com o tráfego de retorno removido do diagrama para maior clareza.

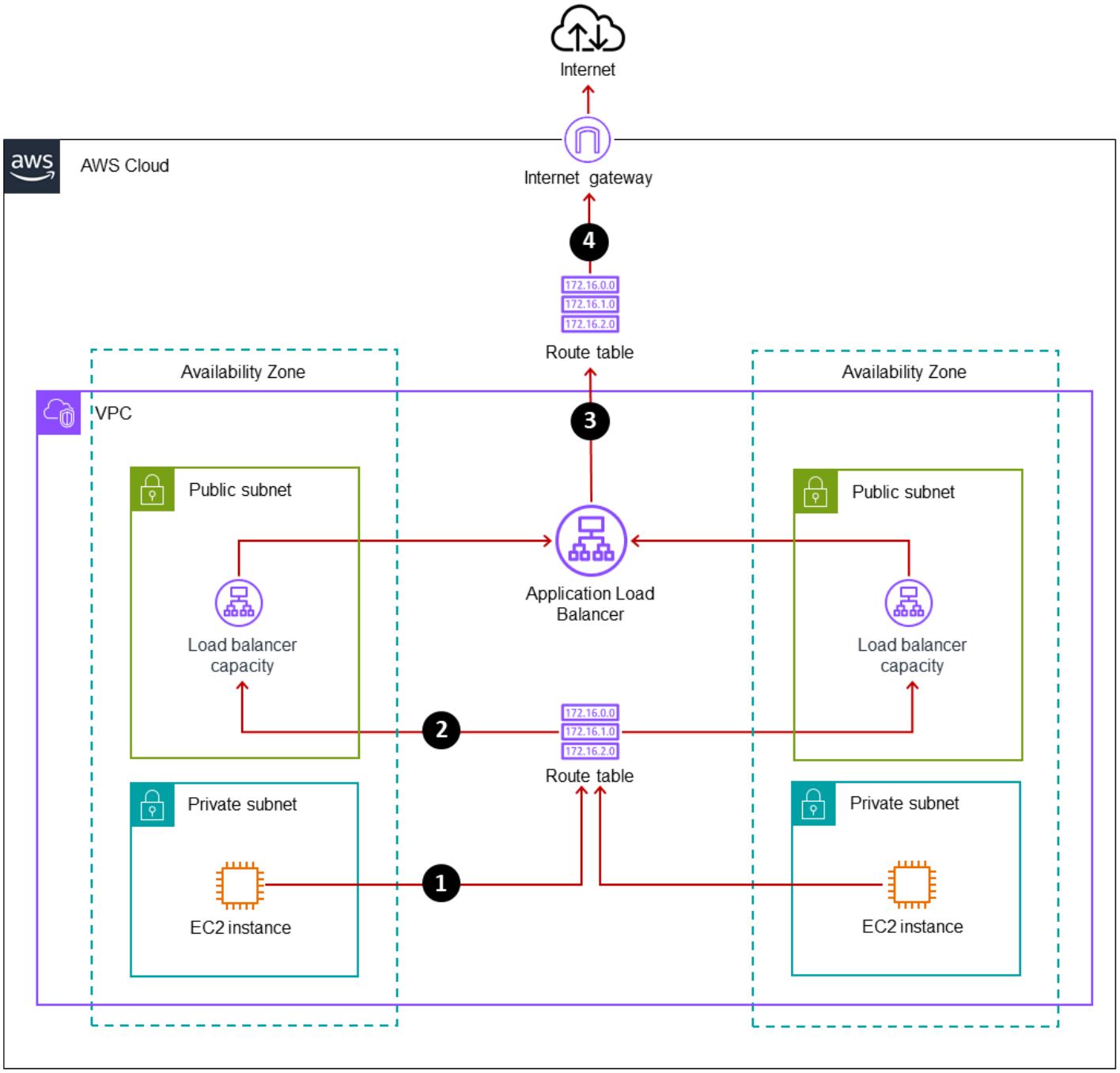


1. O tráfego da Internet flui para o nome DNS do Application Load Balancer.
2. O Application Load Balancer está associado a duas sub-redes públicas no cenário ilustrado. O serviço Elastic Load Balancing cria capacidade de balanceador de carga em cada zona de disponibilidade ativada e envia tráfego pela zona de disponibilidade para determinar a lógica de roteamento apropriada. O Application Load Balancer usa sua lógica interna para determinar para qual grupo-alvo e instância direcionar o tráfego.
3. O Application Load Balancer encaminha a solicitação para a instância do EC2 por meio de um nó associado à sub-rede pública na mesma zona de disponibilidade. (Esses nós são configurados, gerenciados e escalados pelo serviço Elastic Load Balancing e não são visíveis para os usuários.)

4. A tabela de rotas roteia o tráfego localmente dentro da VPC, entre a sub-rede pública e a sub-rede privada, e para a instância do EC2.

Caminho de tráfego de retorno

O diagrama a seguir ilustra as sub-redes VPC e o roteamento associados ao caminho de tráfego de volta para a Internet, com o tráfego de entrada removido do diagrama para maior clareza.

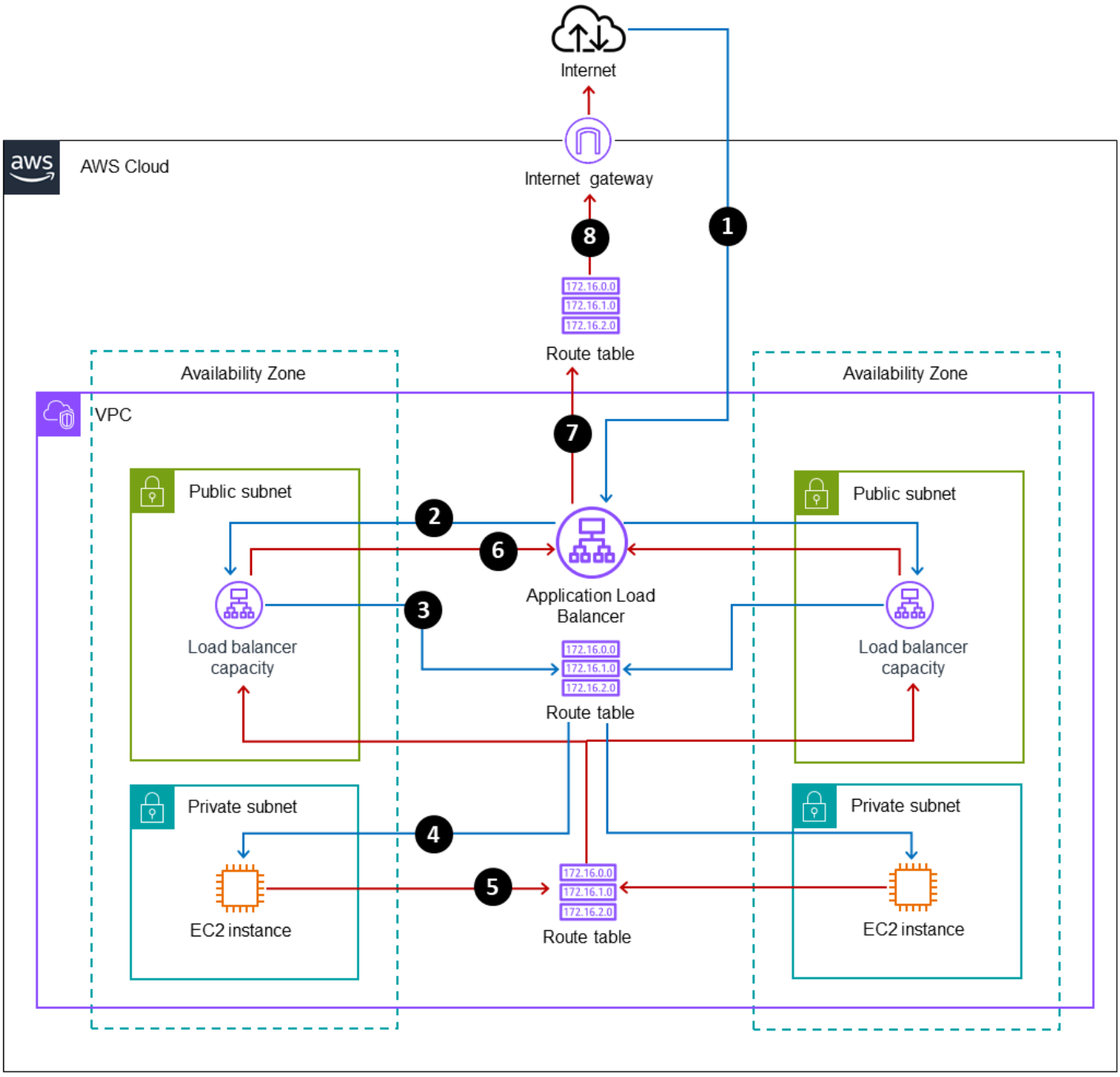


1. A instância do EC2 na sub-rede privada roteia o tráfego de saída por meio da tabela de rotas.
2. A tabela de rotas tem uma rota local para a sub-rede pública. Ele atinge a capacidade do Application Load Balancer em que o tráfego entrou, na sub-rede pública correspondente, seguindo o caminho de volta pela qual o tráfego entrou.
3. O Application Load Balancer direciona o tráfego para fora por meio de sua interface pública.

4. A tabela de rotas da sub-rede pública tem uma rota padrão apontando para um gateway da Internet, que direciona o tráfego de volta para a Internet.

Diagrama completo do fluxo de tráfego

O diagrama a seguir combina os fluxos de tráfego de entrada e retorno para fornecer uma ilustração completa do roteamento do balanceador de carga.



1. O tráfego da Internet flui para o nome DNS do Application Load Balancer.
2. O Application Load Balancer está associado a duas sub-redes públicas no cenário ilustrado. O serviço Elastic Load Balancing cria capacidade de balanceador de carga em cada zona de disponibilidade ativada e envia tráfego pela zona de disponibilidade para determinar a lógica de roteamento apropriada. O Application Load Balancer usa sua lógica interna para determinar para qual grupo-alvo e instância direcionar o tráfego.

3. O Application Load Balancer encaminha a solicitação para a instância do EC2 por meio de um nó associado à sub-rede pública na mesma zona de disponibilidade. (Esses nós são configurados, gerenciados e escalados pelo serviço Elastic Load Balancing e não são visíveis para os usuários.)
4. A tabela de rotas roteia o tráfego localmente dentro da VPC, entre a sub-rede pública e a sub-rede privada, e para a instância do EC2.
5. A instância do EC2 na sub-rede privada roteia o tráfego de saída por meio da tabela de rotas.
6. A tabela de rotas tem uma rota local para a sub-rede pública. Ele atinge a capacidade do Application Load Balancer em que o tráfego entrou, na sub-rede pública correspondente, seguindo o caminho de volta pela qual o tráfego entrou.
7. O Application Load Balancer direciona o tráfego para fora por meio de sua interface pública.
8. A tabela de rotas da sub-rede pública tem uma rota padrão apontando para um gateway da Internet, que direciona o tráfego de volta para a Internet.

Qual estratégia de aderência é ideal para você?

Responder às perguntas a seguir ajudará você a determinar sua estratégia de aderência do balanceador de carga.

Você está usando WebSockets?

- Sim: WebSockets são inerentemente persistentes, então não há necessidade de usar nenhuma estratégia.
- Não: continue com a próxima pergunta.

Você está planejando uma implantação azul/verde?

- Sim: no mínimo, use a aderência do grupo-alvo, mas continue com a próxima pergunta.
- Não: continue com a próxima pergunta.

Você precisa que parte ou todo o tráfego de um cliente seja roteado para a mesma EC2 instância repetidamente?

- Sim: continue com a próxima pergunta.
- Não: você não precisa usar sessões fixas.

Você quer que seu código de aplicativo ou cliente controle os atributos de estado e a duração usando cookies?

- Sim: use cookies baseados em aplicativos para permitir sessões fixas baseadas em aplicativos.
- Não: use cookies gerados pelo balanceador de carga para ativar sessões fixas com base na duração.

Application Load Balancer sem aderência

Quando você usa um Application Load Balancer sem nenhuma forma de aderência, por padrão, o balanceador de carga usa o método round robin para determinar a EC2 instância para a qual ele deve rotear o tráfego.

Modelo: use o CloudFormation modelo `basic.yml` (incluído no arquivo.zip do código de amostra) para testar essa funcionalidade.

> **Note**
>
> Todos os CloudFormation modelos incluídos neste guia implantam uma VPC personalizada, tabelas de rotas, rotas, um gateway de internet, um Application Load Balancer, grupos-alvo, ouvintes EC2 e instâncias, para ilustrar uma estratégia específica de aderência do balanceador de carga.

Casos de uso comuns

Use um Application Load Balancer sem aderência nesses cenários:

- Você tem uma lista de destinos para os quais rotear o tráfego, mas os destinos não mantêm o estado da sessão.
- Você está usando servidores web que não mantêm o estado da sessão.
- Você está usando servidores de aplicativos que não mantêm o estado da sessão.

Etapas

> **Observações**
>
> - Os gateways NAT têm um custo pequeno.
> - Várias EC2 instâncias usam suas horas de nível gratuito mais rápido do que uma única EC2 instância.

1. Implante o CloudFormation modelo `basic.yml` em um ambiente de laboratório.
2. Espere até que o status de saúde das instâncias do seu grupo-alvo mude de inicial para íntegro.
3. Navegue até o URL do Application Load Balancer em um navegador da Web, usando HTTP (TCP/80).

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

   A página da web exibe a Instância 1 - TG1 ou Instância 2 - TG1.
4. Atualize a página várias vezes.

Resultados esperados

A instância que carrega a página da web (Instância 1 ou Instância 2) deve mudar sempre, conforme refletido no texto da página. A lógica do balanceador de carga gerencia o último destino em vários nós internos, o que pode introduzir um atraso na sincronização, então existe a possibilidade de você ser roteado para o mesmo destino.

Como funciona

- Neste exemplo, duas EC2 instâncias são atribuídas a um único grupo-alvo. As EC2 instâncias têm um servidor web Apache (`httpd`) instalado e o texto da `index.html` página em cada EC2 instância é codificado para identificar essa instância.
- O Application Load Balancer executa sua lógica interna de round robin para determinar qual EC2 instância deve receber o tráfego.
- Sempre que você recarrega a página da web, o Application Load Balancer executa sua lógica de roteamento e a página exibe a Instância 1 TG1 - ou Instância 2 -. TG1

Fixação do grupo-alvo

Quando você usa um Application Load Balancer com aderência ao grupo-alvo:

- O Application Load Balancer usa o peso do grupo-alvo para determinar como equilibrar o tráfego de entrada entre os grupos-alvo.
- Por padrão, o Application Load Balancer usa o método round robin para encaminhar solicitações para as EC2 instâncias no grupo-alvo de destino.

Modelo: use o CloudFormation modelo `targetgroupstickiness.yml` (incluído no arquivo.zip do código de amostra) para testar a aderência do grupo-alvo.

Casos de uso comuns

Use a aderência do grupo-alvo nesses cenários:

- Há vários grupos-alvo atribuídos ao balanceador de carga, e o tráfego de um cliente deve ser roteado de forma consistente para instâncias dentro desse grupo-alvo.
- Implantações em azul/verde.

Alterações no código de basic.yml

Uma única alteração foi feita no ouvinte: modificamos as ações padrão do Application Load Balancer para especificar dois grupos-alvo `TG1` (`TG2e`) de igual peso, com uma configuração de aderência.

basic.yml

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

targetgroupstickiness.yml

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

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

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

Etapas

> **Observações**
>
> - Os gateways NAT têm um custo pequeno.
> - Várias EC2 instâncias usam suas horas de nível gratuito mais rápido do que uma única EC2 instância.

1. Implante o CloudFormation modelo `targetgroupstickiness.yml` em um ambiente de laboratório.
2. Espere até que o status de saúde das instâncias do seu grupo-alvo mude de inicial para íntegro.
3. Navegue até o URL do Application Load Balancer em um navegador da Web, usando HTTP (TCP/80).

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

   A página da Web exibe uma das seguintes opções: Instância 1 - TG1, Instância 2 - TG1, Instância 3 - TG2 ou Instância 4 - TG2.
4. Atualize a página várias vezes.

Resultados esperados

> **Note**
>
> O CloudFormation modelo neste exemplo configura a aderência para durar 10 segundos.

As instâncias que carregam a página da Web devem permanecer dentro do grupo-alvo (TG1 ou TG2) dentro da duração de 10 segundos, conforme refletido no texto da página.

Após aproximadamente 10 segundos, a aderência é liberada e o conjunto de instâncias do grupo-alvo pode mudar.

Como funciona

- Neste exemplo, quatro EC2 instâncias são divididas em dois grupos-alvo, com duas instâncias por grupo-alvo. As EC2 instâncias têm um servidor web Apache (`httpd`) instalado e o texto da `index.html` página em cada EC2 instância é codificado para ser distinto.
- O Application Load Balancer cria uma vinculação da sessão do usuário ao grupo-alvo de destino, com um prazo de expiração.
- Quando você recarrega a página, o Application Load Balancer verifica se a vinculação existe e se não expirou.
  - Se a associação expirou ou não existe, o Application Load Balancer executa sua lógica de roteamento e determina o grupo-alvo de destino.
  - Se a vinculação não tiver expirado, o Application Load Balancer roteará o tráfego para o mesmo grupo-alvo, mas não necessariamente para a EC2 mesma instância.

Sessões fixas com cookies gerados pelo balanceador de carga

Quando você usa um Application Load Balancer com um cookie gerado pelo balanceador de carga:

- O Application Load Balancer usa o peso do grupo-alvo para determinar como equilibrar o tráfego de entrada entre os grupos-alvo.
- Por padrão, o Application Load Balancer usa o método round robin para encaminhar solicitações para as EC2 instâncias no grupo-alvo de destino.

  Depois que o tráfego for roteado inicialmente para uma instância, o tráfego subsequente permanecerá nessa EC2 instância por um período especificado.

Modelo: use o CloudFormation modelo `stickysessionslb.yml` (incluído no arquivo.zip do código de amostra) para experimentar sessões fixas com cookies gerados pelo balanceador de carga.

Casos de uso comuns

Use sessões fixas com cookies gerados pelo balanceador de carga nesses cenários:

- Servidores web PHP
- Servidores que mantêm dados temporários da sessão, como registros, carrinhos de compras ou conversas de bate-papo

Alterações no código de basic.yml

As alterações de código relevantes estão na configuração do grupo-alvo, para definir o tipo de aderência `lb_cookie` e a duração em 10 segundos.

basic.yml	**stickysessionslb.yml**
`TG1:` `  Type: 'AWS::ElasticLoadBalancingV2::TargetGroup'`	`TG1:` `  Type: 'AWS::ElasticLoadBalancingV2::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
```

Etapas

> **Observações**
>
> - Os gateways NAT têm um custo pequeno.
> - Várias EC2 instâncias usarão suas horas de nível gratuito mais rápido do que uma única EC2 instância.

1. Implante o CloudFormation modelo `stickysessionslb.yml` em um ambiente de laboratório.
2. Espere até que o status de saúde das instâncias do seu grupo-alvo mude de inicial para íntegro.
3. Navegue até o URL do Application Load Balancer em um navegador da Web, usando HTTP (TCP/80).

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

   A página da Web exibe uma das seguintes opções: Instância 1 - TG1, Instância 2 - TG1, Instância 3 - TG2 ou Instância 4 - TG1.
4. Atualize a página várias vezes.

Resultados esperados

> **Note**
>
> O CloudFormation modelo neste exemplo configura a aderência para durar 10 segundos.

A instância que carrega a página da Web deve permanecer a mesma dentro da duração de 10 segundos, conforme refletido no texto da página. Após aproximadamente 10 segundos, a aderência é liberada e a instância de destino pode mudar.

Como funciona

- Neste exemplo, duas EC2 instâncias estão presentes em um grupo-alvo. As EC2 instâncias têm um servidor web Apache (`httpd`) instalado e o texto da `index.html` página em cada EC2 instância é codificado para ser distinto.
- O Application Load Balancer cria uma vinculação para a sessão do usuário, que se vincula ao destino, com um prazo de expiração.
- Quando você recarrega a página, o Application Load Balancer verifica se a vinculação existe e se não expirou.
  - Se a vinculação tiver expirado ou não existir, o Application Load Balancer executará sua lógica de roteamento e determinará a instância de destino.
  - Se a vinculação não tiver expirado, o Application Load Balancer roteará o tráfego para a mesma instância de destino.

Sessões fixas com cookies baseados em aplicativos

Quando você usa um Application Load Balancer com um cookie baseado em aplicativo:

- O Application Load Balancer usa o peso do grupo-alvo para determinar como equilibrar o tráfego de entrada entre os grupos-alvo.
- Por padrão, o Application Load Balancer usa o método round robin para encaminhar solicitações para as EC2 instâncias no grupo-alvo de destino.
- Depois que o tráfego é roteado inicialmente para uma EC2 instância, a resposta do aplicativo da EC2 instância deve conter um cookie de aplicativo personalizado, que é enviado de volta ao cliente junto com um cookie automatizado do Application Load Balancer.
- O tráfego subsequente permanecerá na EC2 instância se o cliente devolver o cookie do aplicativo e o cookie Application Load Balancer.
- O cookie baseado no aplicativo expira após o não uso durante o período configurado.

Modelo: use o CloudFormation modelo `stickysessionsapp.yml` (incluído no arquivo.zip do código de amostra) para experimentar sessões fixas com cookies baseados em aplicativos.

Casos de uso comuns

Use sessões fixas com cookies gerados pelo aplicativo quando quiser controle adicional nesses cenários:

- Servidores web PHP
- Servidores que mantêm dados temporários da sessão, como registros, carrinhos de compras ou conversas de bate-papo

Alterações no código de basic.yml

A única alteração no código está na configuração do grupo-alvo. Adicionamos uma configuração de aderência ao Application Load Balancer e aos atributos do grupo-alvo. A duração do cookie do aplicativo é especificada e o grupo-alvo tem a aderência do cookie do aplicativo ativada.

basic.yml	**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
```

```
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
```

Etapas

Observações

- Os gateways NAT têm um custo pequeno.
- Várias EC2 instâncias usarão suas horas de nível gratuito mais rápido do que uma única EC2 instância.

1. Implante o CloudFormation modelo `stickysessionslb.yml` em um ambiente de laboratório.
2. Espere até que o status de saúde das instâncias do seu grupo-alvo mude de inicial para íntegro.
3. Navegue até o URL do Application Load Balancer em um navegador da Web, usando HTTP (TCP/80).

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

A página da Web exibe uma das seguintes opções: Instância 1 - TG1, Instância 2 - TG1, Instância 3 - TG2 ou Instância 4 - TG2.

4. Atualize a página várias vezes.

Resultados esperados

> **Note**
>
> O CloudFormation modelo neste exemplo configurou a aderência para durar 10 segundos. A configuração válida da duração da aderência é entre 1 segundo e 1 semana.

A instância que carrega a página da Web deve permanecer a mesma, conforme indicado pelo texto da página.

A duração da aderência não é atualizada, mas é baseada na expiração configurada no Application Load Balancer para o cookie do aplicativo gerado pela instância. EC2

Exemplo 1: Aguarde 5 segundos para atualizar a página. A mesma instância será carregada e a aderência será atualizada por mais 10 segundos.

Exemplo 2: Aguarde mais de 10 segundos para recarregar a página. O cookie do aplicativo expira e você é direcionado para uma instância diferente EC2 . Essa nova instância gera outro cookie de aplicativo com duração de 10 segundos.

Como funciona

- Neste exemplo, as EC2 instâncias têm um servidor web Apache (`httpd`) instalado. O `httpd.conf` arquivo está configurado para retornar um `Set-Cookie` valor estático ao cliente (seu navegador da web). O `Set-Cookie` valor está codificado para ser. `TESTCOOKIE=<somevalue>`
- Abra a opção Inspecionar elemento do seu navegador, escolha a guia Rede e, em seguida, escolha o método Get, que carrega a página. Você verá uma guia Cookies.

- O navegador é um aplicativo cliente configurado automaticamente para retornar quaisquer atualizações subsequentes ao servidor com os cookies que recebe na `Set-Cookie` resposta do servidor.
- Quando você recarrega a página, os cookies recebidos no carregamento inicial da página são automaticamente enviados de volta ao Application Load Balancer.
  - Se o cookie tiver expirado (ou seja, 10 segundos se passaram desde que você fez a última chamada), o Application Load Balancer usa uma nova lógica para determinar para EC2 qual instância rotear o tráfego.
  - Se o cookie não tiver expirado, o Application Load Balancer direcionará o tráfego para a EC2 mesma instância.

Recursos

- Sessões fixas para seu Application Load Balancer (documentação do Elastic Load Balancing)

Histórico do documento

A tabela a seguir descreve alterações significativas feitas neste guia. Se desejar receber notificações sobre futuras atualizações, inscreva-se em um feed RSS.

Alteração	Descrição	Data
Seção atualizada	A seção de sub-redes e roteamento do balanceador de carga foi atualizada com novos diagramas e esclarecimentos.	5 de julho de 2024
Atualizações e correções	O código, os diagramas e os exemplos foram atualizados.	31 de maio de 2023
Seções atualizadas	As seções Como funciona foram atualizadas nas sessões de aderência do grupo-alvo e Sticky com cookies gerados pelo balanceador de carga para fornecer informações mais precisas e detalhadas.	18 de julho de 2022
—	Publicação inicial	5 de agosto de 2021

As traduções são geradas por tradução automática. Em caso de conflito entre o conteúdo da tradução e da versão original em inglês, a versão em inglês prevalecerá.