Choisir une stratégie de rigidité pour votre équilibreur de charge

AWS Conseils prescriptifs

AWS Conseils prescriptifs: Choisir une stratégie de rigidité pour votre équilibreur de charge

Table of Contents

Choisir une stratégie d'adhérence pour votre équilibreur de charge

Ryan Griffin, Amazon Web Services ()AWS

Juillet 2024 (historique du document)

La rigidité est un terme utilisé pour décrire la fonctionnalité d'un équilibreur de charge permettant d'acheminer à plusieurs reprises le trafic d'un client vers une seule destination, au lieu d'équilibrer le trafic entre plusieurs destinations. Par exemple, le trafic du client A peut être acheminé en continu vers un serveur spécifique, afin que le serveur puisse conserver les données d'état de session. Si le trafic du client A est acheminé vers deux serveurs distincts, il se peut que chaque serveur ne dispose pas d'informations importantes mises à la disposition de l'autre serveur.

Par conséquent, il est souvent nécessaire de maintenir une connexion client cohérente par le biais d'un équilibreur de charge. Il existe deux types de viscosité : les sessions persistantes et l'adhérence du groupe cible.

- Sessions persistantes : gestion des données de session locales dans une instance Amazon Elastic Compute Cloud (Amazon EC2) afin de simplifier l'architecture des applications ou d'améliorer les performances des applications, car l'instance peut conserver ou mettre en cache les informations d'état de session localement. AWS propose actuellement deux types de sessions persistantes, que ce guide décrit en détail : les cookies d'application et les cookies d'équilibrage de charge.
- Persistance du groupe cible : dans les déploiements bleu/vert, plusieurs versions d'une application peuvent être déployées et vous pouvez souhaiter que le client continue à utiliser la même version de l'application au cours de sa session. Dans ce cas, vous pouvez utiliser la rigidité du groupe cible pour acheminer toutes les communications du client vers le même groupe cible plutôt que vers la même EC2 instance.

Vous pouvez utiliser ces deux stratégies d'adhérence séparément ou ensemble.

Ce guide décrit les différents types d'adhérence des équilibreurs de charge et les cas d'utilisation applicables, afin de vous aider à choisir une stratégie. Le guide inclut des AWS CloudFormation modèles illustrant chaque stratégie.

Exemple de code

Ce guide fournit un fichier .zip joint qui inclut quatre AWS CloudFormation modèles que vous pouvez déployer pour créer une architecture de base et tester chaque stratégie de fidélisation. Nous vous recommandons de déployer ces modèles dans un environnement de laboratoire afin de tester chaque approche.

Téléch

un exemple de code

Le téléchargement inclut les modèles suivants :

- `basic.yml`— Configure un Application Load Balancer sans difficulté.
- `targetgroupstickiness.yml`— Fait preuve d'adhérence en fonction des groupes cibles.
- `stickysessionslb.yml`— Démontre les sessions persistantes avec des cookies générés par l'équilibreur de charge.
- `stickysessionsapp.yml`— Démontre les sessions persistantes avec des cookies basés sur des applications.

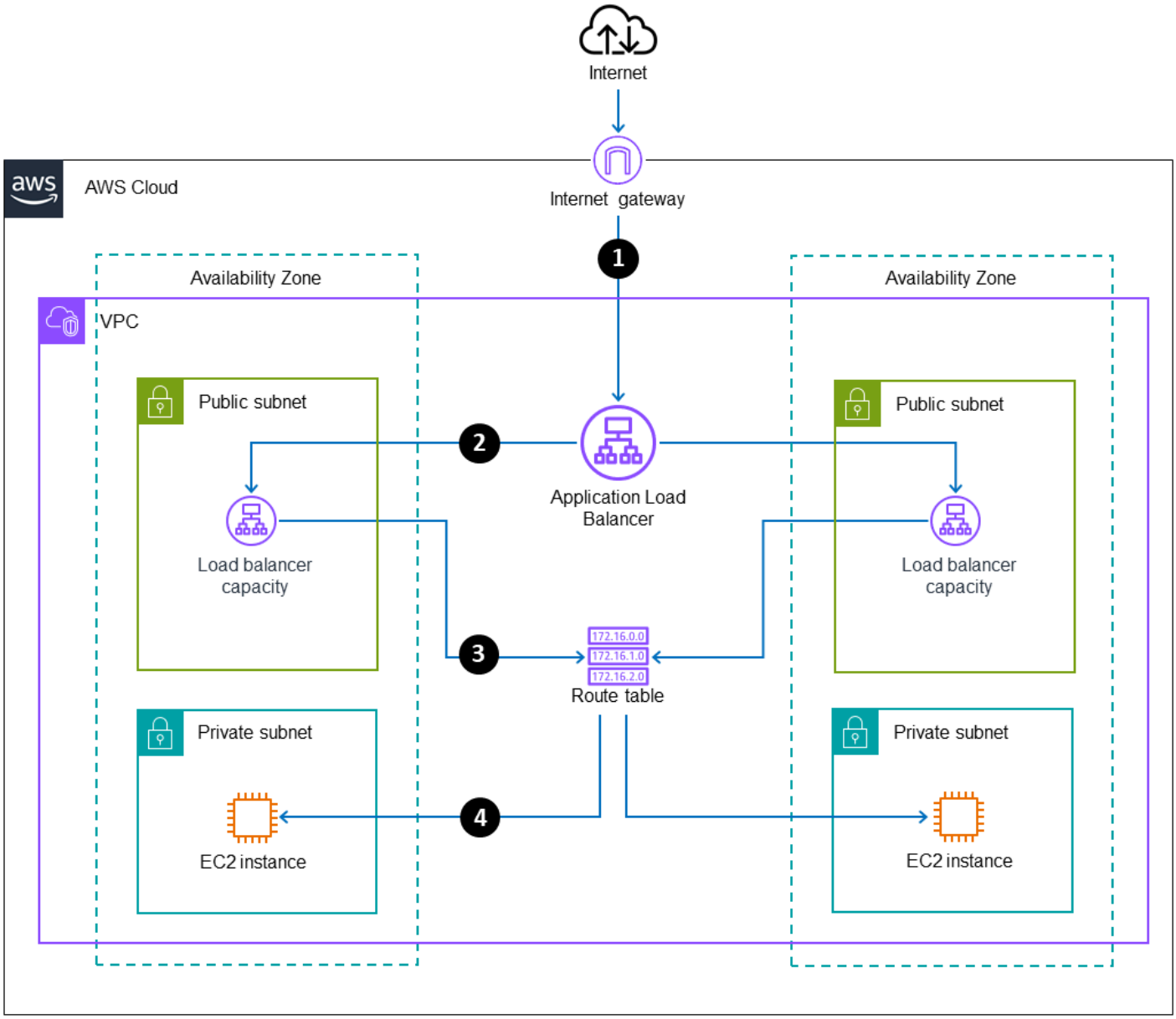
Pour déployer ces modèles, vous aurez besoin d'un AWS compte actif : et d'un accès à la CloudFormation console. Pour step-by-steps obtenir des instructions sur le déploiement d'un CloudFormation modèle, consultez la section Création d'une pile dans la CloudFormation documentation.

Sous-réseaux et routage de l'équilibreur de charge

Commençons par une illustration de la manière dont le trafic circule lorsque vous configurez un Application Load Balancer faisant face à Internet vers des instances Amazon Elastic Compute Cloud (Amazon EC2) dans un sous-réseau privé. Cette architecture reflète les meilleures pratiques lors du déploiement d'un Application Load Balancer ouvert sur Internet.

Trajectoire du trafic entrant

Le schéma suivant illustre les sous-réseaux du cloud privé virtuel (VPC) et le routage associés au flux de trafic entrant, le trafic de retour étant supprimé du diagramme pour des raisons de clarté.

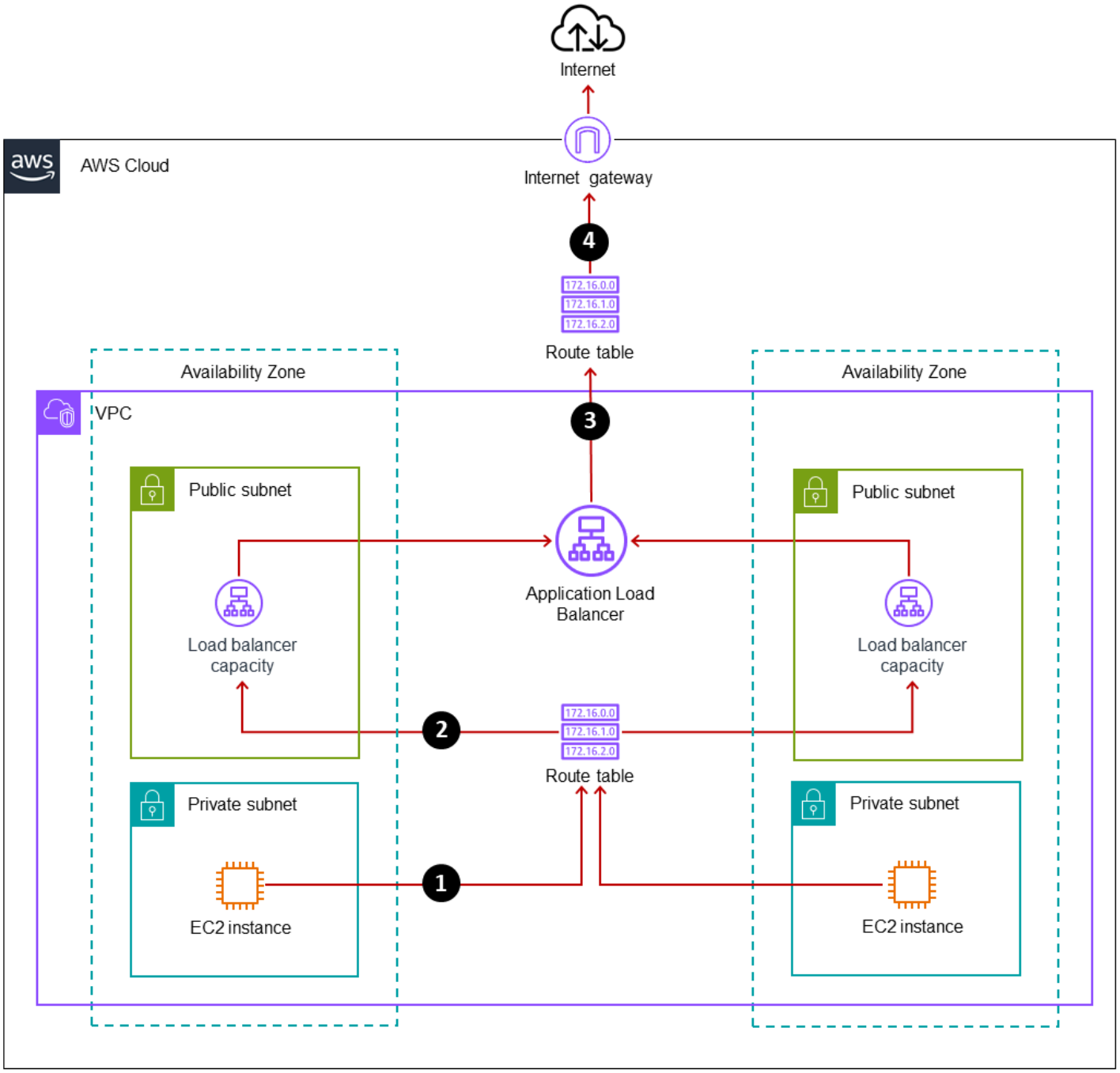


1. Le trafic provenant d'Internet est acheminé vers le nom DNS de l'Application Load Balancer.
2. L'Application Load Balancer est associé à deux sous-réseaux publics dans le scénario illustré. Le service Elastic Load Balancing crée une capacité d'équilibrage de charge dans chaque zone de disponibilité activée et envoie le trafic via la zone de disponibilité afin de déterminer la logique de routage appropriée. L'Application Load Balancer utilise sa logique interne pour déterminer le groupe cible et l'instance vers lesquels acheminer le trafic.
3. L'Application Load Balancer achemine la demande vers l'instance EC2 via un nœud associé au sous-réseau public dans la même zone de disponibilité. (Ces nœuds sont configurés, gérés et dimensionnés par le service Elastic Load Balancing et ne sont pas visibles pour les utilisateurs.)

4. La table de routage achemine le trafic localement au sein du VPC, entre le sous-réseau public et le sous-réseau privé, et vers l'instance EC2.

Trajectoire de retour

Le schéma suivant illustre les sous-réseaux VPC et le routage associés au chemin de retour vers Internet, le trafic entrant étant supprimé du diagramme pour plus de clarté.

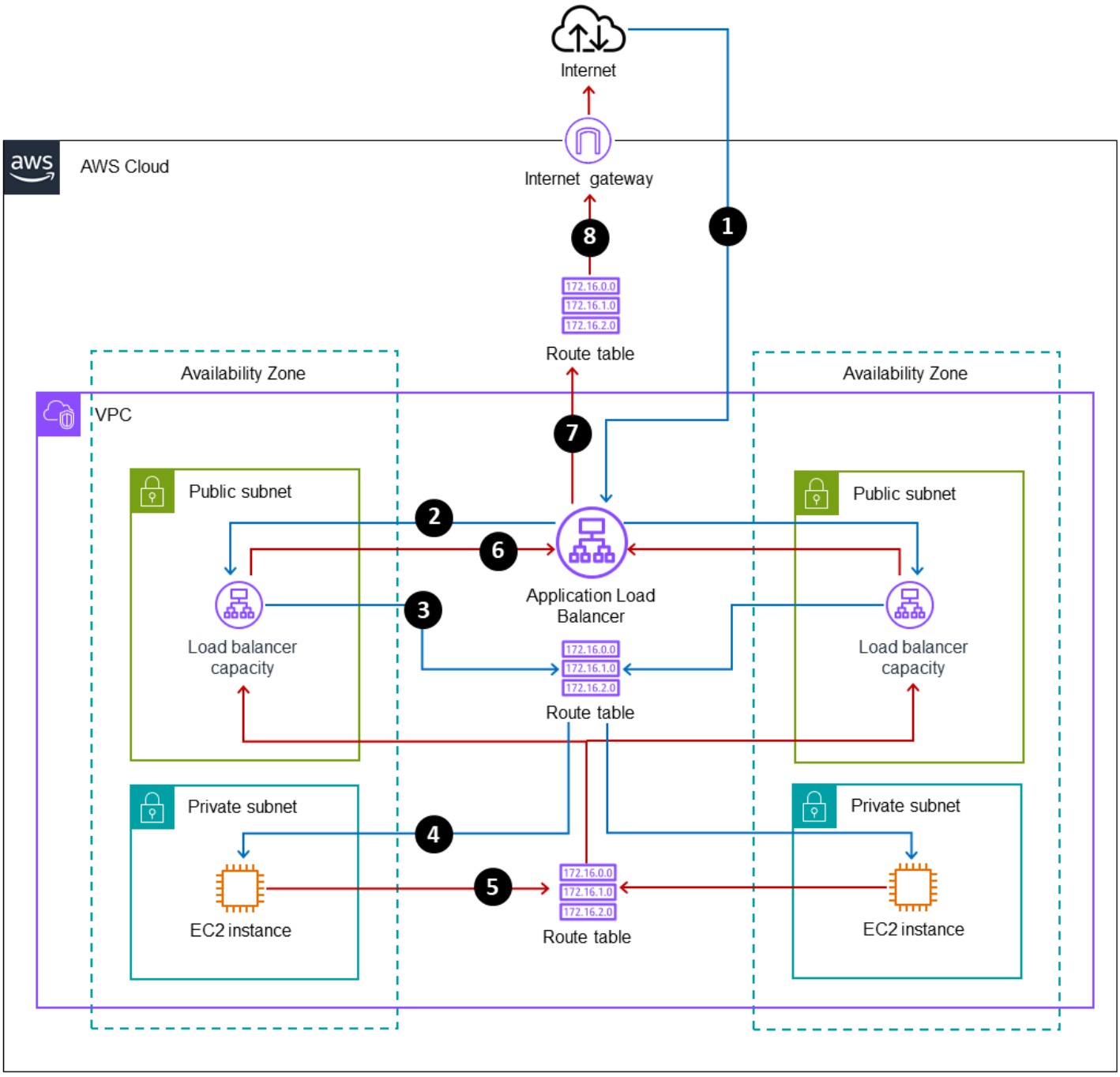


1. L'instance EC2 du sous-réseau privé achemine le trafic sortant via la table de routage.
2. La table de routage possède une route locale vers le sous-réseau public. Il atteint la capacité de l'Application Load Balancer sur laquelle le trafic est entré, dans le sous-réseau public correspondant, en suivant le chemin de retour par lequel le trafic est entré.
3. L'Application Load Balancer achemine le trafic vers l'extérieur via son interface publique.

4. La table de routage du sous-réseau public possède un itinéraire par défaut pointant vers une passerelle Internet, qui redirige le trafic vers Internet.

Schéma complet du flux de trafic

Le schéma suivant combine les flux de trafic entrant et de retour pour fournir une illustration complète du routage de l'équilibreur de charge.



1. Le trafic provenant d'Internet est acheminé vers le nom DNS de l'Application Load Balancer.
2. L'Application Load Balancer est associé à deux sous-réseaux publics dans le scénario illustré. Le service Elastic Load Balancing crée une capacité d'équilibrage de charge dans chaque zone de disponibilité activée et envoie le trafic via la zone de disponibilité afin de déterminer la logique de routage appropriée. L'Application Load Balancer utilise sa logique interne pour déterminer le groupe cible et l'instance vers lesquels acheminer le trafic.

3. L'Application Load Balancer achemine la demande vers l'instance EC2 via un nœud associé au sous-réseau public dans la même zone de disponibilité. (Ces nœuds sont configurés, gérés et dimensionnés par le service Elastic Load Balancing et ne sont pas visibles pour les utilisateurs.)
4. La table de routage achemine le trafic localement au sein du VPC, entre le sous-réseau public et le sous-réseau privé, et vers l'instance EC2.
5. L'instance EC2 du sous-réseau privé achemine le trafic sortant via la table de routage.
6. La table de routage possède une route locale vers le sous-réseau public. Il atteint la capacité de l'Application Load Balancer sur laquelle le trafic est entré, dans le sous-réseau public correspondant, en suivant le chemin de retour par lequel le trafic est entré.
7. L'Application Load Balancer achemine le trafic vers l'extérieur via son interface publique.
8. La table de routage du sous-réseau public possède un itinéraire par défaut pointant vers une passerelle Internet, qui redirige le trafic vers Internet.

Quelle stratégie d'adhérence vous convient le mieux ?

Les réponses aux questions suivantes vous aideront à déterminer la stratégie d'adhérence de votre équilibreur de charge.

En consommez-vous WebSockets ?

- Oui : WebSockets sont intrinsèquement collants, il n'est donc pas nécessaire d'utiliser de stratégie.
- Non : Passez à la question suivante.

Envisagez-vous un déploiement bleu/vert ?

- Oui : utilisez au minimum la méthode de sélection du groupe cible, mais passez à la question suivante.
- Non : Passez à la question suivante.

Avez-vous besoin qu'une partie ou la totalité du trafic d'un client soit acheminée à plusieurs reprises vers la même EC2 instance ?

- Oui : passez à la question suivante.
- Non : il n'est pas nécessaire d'utiliser des sessions persistantes.

Voulez-vous que le code de votre application ou votre client contrôle l'état, les attributs et la durée à l'aide de cookies ?

- Oui : utilisez des cookies basés sur les applications pour activer les sessions persistantes basées sur les applications.
- Non : utilisez les cookies générés par l'équilibreur de charge pour activer les sessions persistantes basées sur la durée.

Application Load Balancer sans effet collant

Lorsque vous utilisez un Application Load Balancer sans aucune forme d'adhérence, l'équilibreur de charge utilise par défaut la méthode du round robin pour déterminer l' EC2 instance vers laquelle il doit acheminer le trafic.

Modèle : utilisez le CloudFormation modèle `basic.yml` (inclus dans le fichier .zip de code d'exemple) pour tester cette fonctionnalité.

> **Note**
>
> Tous les CloudFormation modèles inclus dans ce guide déploient un VPC personnalisé, des tables de routage, des itinéraires, une passerelle Internet, un Application Load Balancer, des groupes cibles, des écouteurs EC2 et des instances, afin d'illustrer une stratégie spécifique de fidélisation de l'équilibreur de charge.

Cas d’utilisation courants

Utilisez un Application Load Balancer sans problème dans les scénarios suivants :

- Vous disposez d'une liste de cibles vers lesquelles acheminer le trafic, mais les cibles ne conservent pas l'état de session.
- Vous utilisez des serveurs Web qui ne maintiennent pas l'état des sessions.
- Vous utilisez des serveurs d'applications qui ne maintiennent pas l'état des sessions.

Étapes

> **Remarques**
>
> - Les passerelles NAT sont peu coûteuses.
> - Plusieurs EC2 instances utilisent vos heures de niveau gratuit plus rapidement qu'une seule EC2 instance.

1. Déployez le CloudFormation modèle `basic.yml` dans un environnement de laboratoire.
2. Attendez que l'état de santé des instances de votre groupe cible passe d'un état initial à un état sain.
3. Accédez à l'URL de l'Application Load Balancer dans un navigateur Web à l'aide du protocole HTTP (TCP/80).

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

   La page Web affiche Instance 1 - TG1 ou Instance 2 - TG1.
4. Rafraîchissez la page plusieurs fois.

Résultats attendus

L'instance qui charge la page Web (Instance 1 ou Instance 2) doit changer à chaque fois, comme indiqué dans le texte de la page. La logique de l'équilibreur de charge gère la dernière cible sur plusieurs nœuds internes, ce qui peut entraîner un retard de synchronisation. Il est donc possible que vous soyez redirigé vers la même cible.

Comment ça marche

- Dans cet exemple, deux EC2 instances sont attribuées à un seul groupe cible. Un serveur Web Apache (`httpd`) est installé sur les EC2 instances, et le texte de `index.html` page de chaque EC2 instance est codé en dur pour identifier cette instance.
- L'Application Load Balancer exécute sa logique circulaire interne pour déterminer quelle EC2 instance doit recevoir le trafic.
- Chaque fois que vous rechargez la page Web, l'Application Load Balancer exécute sa logique de routage et la page affiche Instance 1 TG1 - ou Instance 2 -. TG1

Adhérence du groupe cible

Lorsque vous utilisez un Application Load Balancer compatible avec le groupe cible :

- L'Application Load Balancer utilise le poids du groupe cible pour déterminer comment équilibrer le trafic entrant entre les groupes cibles.
- Par défaut, l'Application Load Balancer utilise la méthode round robin pour acheminer les demandes vers les EC2 instances du groupe cible de destination.

Modèle : utilisez le CloudFormation modèle `targetgroupstickiness.yml` (inclus dans le fichier .zip de code d'exemple) pour tester la compatibilité avec le groupe cible.

Cas d’utilisation courants

Utilisez la rigidité du groupe cible dans les scénarios suivants :

- Plusieurs groupes cibles sont affectés à l'équilibreur de charge, et le trafic provenant d'un client doit être systématiquement acheminé vers les instances de ce groupe cible.
- Déploiements bleu/vert.

Changements de code depuis basic.yml

Une seule modification a été apportée à l'écouteur : nous avons modifié les actions par défaut de l'Application Load Balancer pour spécifier deux groupes cibles TG1 (TG2et) ayant le même poids, avec une configuration d'adhérence.

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
```

Étapes

> **Remarques**
>
> - Les passerelles NAT sont peu coûteuses.
> - Plusieurs EC2 instances utilisent vos heures de niveau gratuit plus rapidement qu'une seule EC2 instance.

1. Déployez le CloudFormation modèle `targetgroupstickiness.yml` dans un environnement de laboratoire.
2. Attendez que l'état de santé des instances de votre groupe cible passe d'un état initial à un état sain.
3. Accédez à l'URL de l'Application Load Balancer dans un navigateur Web à l'aide du protocole HTTP (TCP/80).

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

   La page Web affiche l'un des éléments suivants : Instance 1 - TG1, Instance 2 - TG1, Instance 3 - TG2 ou Instance 4 - TG2.
4. Rafraîchissez la page plusieurs fois.

Résultats attendus

> **Note**
>
> Dans cet exemple, le CloudFormation modèle configure l'adhérence pour qu'elle dure 10 secondes.

Les instances qui chargent la page Web doivent rester dans le groupe cible (TG1 ou TG2) dans les 10 secondes, comme indiqué dans le texte de la page.

Après environ 10 secondes, le caractère collant est relâché et l'ensemble d'instances du groupe cible peut changer.

Comment ça marche

- Dans cet exemple, quatre EC2 instances sont réparties entre deux groupes cibles, avec deux instances par groupe cible. Un serveur Web Apache (`httpd`) est installé sur les EC2 instances, et le texte de `index.html` page de chaque EC2 instance est codé en dur pour être distinct.
- L'Application Load Balancer crée une liaison pour la session de l'utilisateur vers le groupe cible de destination, avec une date d'expiration.
- Lorsque vous rechargez la page, l'Application Load Balancer vérifie si la liaison existe et n'a pas expiré.
  - Si la liaison a expiré ou n'existe pas, l'Application Load Balancer exécute sa logique de routage et détermine le groupe cible de destination.
  - Si la liaison n'a pas expiré, l'Application Load Balancer achemine le trafic vers le même groupe cible, mais pas nécessairement vers la même EC2 instance.

Sessions persistantes avec cookies générés par l'équilibreur de charge

Lorsque vous utilisez un Application Load Balancer avec un cookie généré par un équilibreur de charge :

- L'Application Load Balancer utilise le poids du groupe cible pour déterminer comment équilibrer le trafic entrant entre les groupes cibles.
- Par défaut, l'Application Load Balancer utilise la méthode round robin pour acheminer les demandes vers les EC2 instances du groupe cible de destination.

  Une fois que le trafic a été initialement acheminé vers une instance, le trafic suivant restera sur cette EC2 instance pendant une durée spécifiée.

Modèle : utilisez le CloudFormation modèle `stickysessionslb.yml` (inclus dans le fichier .zip de code d'exemple) pour essayer des sessions persistantes avec des cookies générés par l'équilibreur de charge.

Cas d’utilisation courants

Utilisez des sessions persistantes avec des cookies générés par l'équilibreur de charge dans les scénarios suivants :

- Serveurs Web PHP
- Serveurs qui conservent des données de session temporaires telles que les journaux, les paniers d'achat ou les conversations par chat

Changements de code depuis basic.yml

Les modifications de code pertinentes concernent la configuration du groupe cible, afin de définir le type d'adhérence `lb_cookie` et la durée à 10 secondes.

basic.yml	**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
```

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

Étapes

> **Remarques**
>
> - Les passerelles NAT sont peu coûteuses.
> - Plusieurs EC2 instances utiliseront vos heures de niveau gratuit plus rapidement qu'une seule EC2 instance.

1. Déployez le CloudFormation modèle `stickysessionslb.yml` dans un environnement de laboratoire.
2. Attendez que l'état de santé des instances de votre groupe cible passe d'un état initial à un état sain.
3. Accédez à l'URL de l'Application Load Balancer dans un navigateur Web à l'aide du protocole HTTP (TCP/80).

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

La page Web affiche l'un des éléments suivants : Instance 1 - TG1, Instance 2 - TG1, Instance 3 - TG2 ou Instance 4 - TG1.

4. Rafraîchissez la page plusieurs fois.

Résultats attendus

> **Note**
>
> Dans cet exemple, le CloudFormation modèle configure l'adhérence pour qu'elle dure 10 secondes.

L'instance qui charge la page Web doit rester la même pendant les 10 secondes, comme indiqué dans le texte de la page. Après environ 10 secondes, le caractère collant est relâché et l'instance de destination peut changer.

Comment ça marche

- Dans cet exemple, deux EC2 instances sont présentes dans un groupe cible. Un serveur Web Apache (`httpd`) est installé sur les EC2 instances, et le texte de `index.html` page de chaque EC2 instance est codé en dur pour être distinct.
- L'Application Load Balancer crée une liaison pour la session de l'utilisateur, qui est liée à la destination, avec une date d'expiration.
- Lorsque vous rechargez la page, l'Application Load Balancer vérifie si la liaison existe et n'a pas expiré.
  - Si la liaison a expiré ou n'existe pas, l'Application Load Balancer exécute sa logique de routage et détermine l'instance de destination.
  - Si la liaison n'a pas expiré, l'Application Load Balancer achemine le trafic vers la même instance de destination.

Sessions persistantes avec cookies basés sur les applications

Lorsque vous utilisez un Application Load Balancer avec un cookie basé sur une application :

- L'Application Load Balancer utilise le poids du groupe cible pour déterminer comment équilibrer le trafic entrant entre les groupes cibles.
- Par défaut, l'Application Load Balancer utilise la méthode round robin pour acheminer les demandes vers les EC2 instances du groupe cible de destination.
- Une fois que le trafic a été initialement acheminé vers une EC2 instance, la réponse de l'application de l' EC2 instance doit contenir un cookie d'application personnalisé, qui est renvoyé au client avec un cookie Application Load Balancer automatisé.
- Le trafic suivant restera concentré sur l' EC2 instance si le client renvoie le cookie d'application et le cookie Application Load Balancer.
- Le cookie basé sur l'application expire après non-utilisation pendant la durée configurée.

Modèle : utilisez le CloudFormation modèle `stickysessionsapp.yml` (inclus dans le fichier .zip de code d'exemple) pour essayer des sessions persistantes avec des cookies basés sur des applications.

Cas d’utilisation courants

Utilisez des sessions persistantes avec des cookies générés par l'application lorsque vous souhaitez un contrôle supplémentaire dans les scénarios suivants :

- Serveurs Web PHP
- Serveurs qui conservent des données de session temporaires telles que les journaux, les paniers d'achat ou les conversations par chat

Changements de code depuis basic.yml

Le seul changement de code concerne la configuration du groupe cible. Nous avons ajouté une configuration d'adhérence aux attributs de l'Application Load Balancer et du groupe cible. La durée

du cookie d'application est spécifiée et le groupe cible a activé le caractère persistant des cookies d'application.

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
```

Étapes

Remarques

- Les passerelles NAT sont peu coûteuses.
- Plusieurs EC2 instances utiliseront vos heures de niveau gratuit plus rapidement qu'une seule EC2 instance.

1. Déployez le CloudFormation modèle `stickysessionslb.yml` dans un environnement de laboratoire.
2. Attendez que l'état de santé des instances de votre groupe cible passe d'un état initial à un état sain.
3. Accédez à l'URL de l'Application Load Balancer dans un navigateur Web à l'aide du protocole HTTP (TCP/80).

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

   La page Web affiche l'un des éléments suivants : Instance 1 - TG1, Instance 2 - TG1, Instance 3 - TG2 ou Instance 4 - TG2.
4. Actualisez la page plusieurs fois.

Résultats attendus

> **Note**
>
> Le CloudFormation modèle de cet exemple a configuré l'adhérence pour qu'elle dure 10 secondes. La configuration valide de la durée d'adhérence est comprise entre 1 seconde et 1 semaine.

L'instance qui charge la page Web doit rester la même, comme indiqué par le texte de la page.

La durée de conservation n'est pas actualisée, mais est basée sur l'expiration configurée dans l'Application Load Balancer pour le cookie d'application généré par l'instance. EC2

Exemple 1 : attendez 5 secondes pour actualiser la page. La même instance se chargera et le caractère collant sera actualisé pendant 10 secondes supplémentaires.

Exemple 2 : attendez plus de 10 secondes pour recharger la page. Le cookie d'application expire et vous êtes redirigé vers une autre EC2 instance. Cette nouvelle instance génère un autre cookie d'application d'une durée de 10 secondes.

Comment ça marche

- Dans cet exemple, un serveur Web Apache (`httpd`) est installé sur les EC2 instances. Le `httpd.conf` fichier est configuré pour renvoyer une `Set-Cookie` valeur statique au client (votre navigateur Web). La `Set-Cookie` valeur est codée en dur pour être. `TESTCOOKIE=<somevalue>`
- Ouvrez l'option Inspect Element de votre navigateur, choisissez l'onglet Réseau, puis choisissez la méthode Get, qui charge la page. Vous verrez un onglet Cookies.
- Le navigateur est une application cliente qui est automatiquement configurée pour renvoyer toutes les mises à jour ultérieures au serveur avec les cookies qu'il reçoit dans la `Set-Cookie` réponse du serveur.
- Lorsque vous rechargez la page, les cookies reçus lors du chargement initial de la page sont automatiquement renvoyés à l'Application Load Balancer.
  - Si le cookie a expiré (c'est-à-dire que 10 secondes se sont écoulées depuis le dernier appel), l'Application Load Balancer utilise une nouvelle logique pour déterminer l'instance vers EC2 laquelle acheminer le trafic.
  - Si le cookie n'a pas expiré, l'Application Load Balancer achemine le trafic vers la même EC2 instance.

Ressources

- Sessions permanentes pour votre Application Load Balancer (documentation Elastic Load Balancing)

Historique du document

Le tableau suivant décrit les modifications importantes apportées à ce guide. Pour être averti des mises à jour à venir, abonnez-vous à un fil RSS.

Modification	Description	Date
Section mise à jour	Mise à jour de la section sur les sous-réseaux et le routage de l'équilibreur de charge avec de nouveaux diagrammes et des clarifications.	5 juillet 2024
Mises à jour et corrections	Le code, les diagrammes et les exemples ont été mis à jour.	31 mai 2023
Sections mises à jour	Les sections Fonctionnement ont été mises à jour dans Target group stickiness et Sessions persistantes avec des cookies générés par l'équilibreur de charge afin de fournir des informations plus précises et détaillées.	18 juillet 2022
—	Publication initiale	5 août 2021

Les traductions sont fournies par des outils de traduction automatique. En cas de conflit entre le contenu d'une traduction et celui de la version originale en anglais, la version anglaise prévaudra.