aws

Classic Load Balancer

Elastic Load Balancing

Elastic Load Balancing: Classic Load Balancer

Table of Contents

Classic Load Balancer란 무엇인가요?

> **Note**
>
> Classic Load Balancer는 Elastic Load Balancing 이전 세대의 로드 밸런서입니다. 현재 세대의 로드 밸런서로 마이그레이션하는 것이 좋습니다. 자세한 내용은 Classic Load Balancer 마이그레이션을 참조하세요.

Elastic Load Balancing은 둘 이상의 가용 영역에서 EC2 인스턴스, 컨테이너, IP 주소 등 여러 대상에 걸쳐 수신되는 트래픽을 자동으로 분산합니다. 등록된 대상의 상태를 모니터링하면서 상태가 양호한 대상으로만 트래픽을 라우팅합니다. Elastic Load Balancing은 수신 트래픽이 시간이 지남에 따라 변경됨에 따라 로드 밸런서를 확장합니다. 대다수의 워크로드에 맞게 자동으로 조정할 수 있습니다.

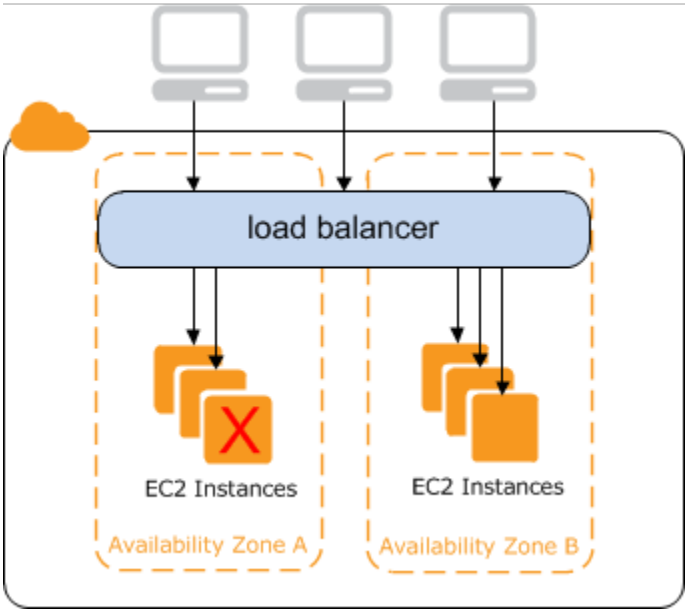
Classic Load Balancer 개요

로드 밸런서는 여러 가용 영역에서 여러 EC2 인스턴스에 수신 애플리케이션 트래픽을 분산합니다. 이렇게 하면 애플리케이션의 내결함성이 향상됩니다. Elastic Load Balancing은 비정상 인스턴스를 탐지하고 정상 인스턴스에만 트래픽을 라우팅합니다.

로드 밸런서는 클라이언트에 대한 단일 접점 역할을 수행합니다. 이렇게 하면 애플리케이션의 가용성이 향상됩니다. 애플리케이션에 대한 요청의 전체적인 흐름을 방해하지 않고 필요에 따라 로드 밸런서에서 인스턴스를 추가 및 제거할 수 있습니다. 애플리케이션에 대한 트래픽이 시간에 따라 변화하므로 Elastic Load Balancing은 로드 밸런서를 확장합니다. Elastic Load Balancing은 대다수의 워크로드에 맞게 자동으로 조정할 수 있습니다.

리스너는 사용자가 구성한 프로토콜 및 포트를 사용하여 클라이언트의 연결 요청을 확인하고, 사용자가 구성한 프로토콜 및 포트 번호를 사용하여 하나 이상의 등록된 인스턴스에 요청을 전달합니다. 로드 밸런서에 하나 이상의 리스너를 추가할 수 있습니다.

로드 밸런서만 정상적인 인스턴스에 요청을 보내도록 등록된 인스턴스의 상태를 모니터링하는 데 사용되는 상태 확인을 구성할 수 있습니다.



등록된 인스턴스가 각각의 가용 영역에서 요청 로드를 처리할 수 있도록 하려면 로드 밸런서에 등록된 각각의 영역에 있는 인스턴스의 수가 거의 동일해야 합니다. 예를 들어, 인스턴스가 가용 영역 us-west-2a에 10개가 있고 us-west-2b에 2개가 있는 경우 요청은 두 가용 영역에 고르게 분산됩니다. 따라서 us-west-2b에 있는 두 개의 인스턴스는 us-west-2a에 있는 10개의 인스턴스와 동일한 양의 트래픽을 처리해야 합니다. 대신, 각 가용 영역에서 6개의 인스턴스를 가질 수 있습니다.

기본적으로 로드 밸런서는 사용자의 로드 밸런서에 대해 활성화된 가용 영역에 트래픽을 고르게 분산합니다. 활성화된 모든 가용 영역에서 등록된 모든 인스턴스에 트래픽을 고르게 분산하려면 로드 밸런서에서 교차 영역 로드 밸런싱을 사용하도록 설정하십시오. 하지만 내결함성을 높이기 위해 각 가용 영역에서 인스턴스 수를 대략적으로 동일하게 유지하는 것이 좋습니다.

자세한 내용은 Elastic Load Balancing 사용 설명서의 Elastic Load Balancing 작동 방식을 참조하세요.

이점

Application Load Balancer 대신 Classic Load Balancer를 사용하면 다음과 같은 이점이 있습니다.

- TCP 및 SSL 지원
- 애플리케이션 생성 쿠키를 사용하여 고정 세션 지원

각 유형의 로드 밸런서가 지원하는 기능에 대한 자세한 내용은 Elastic Load Balancing 제품 비교를 참조하세요.

시작하는 방법

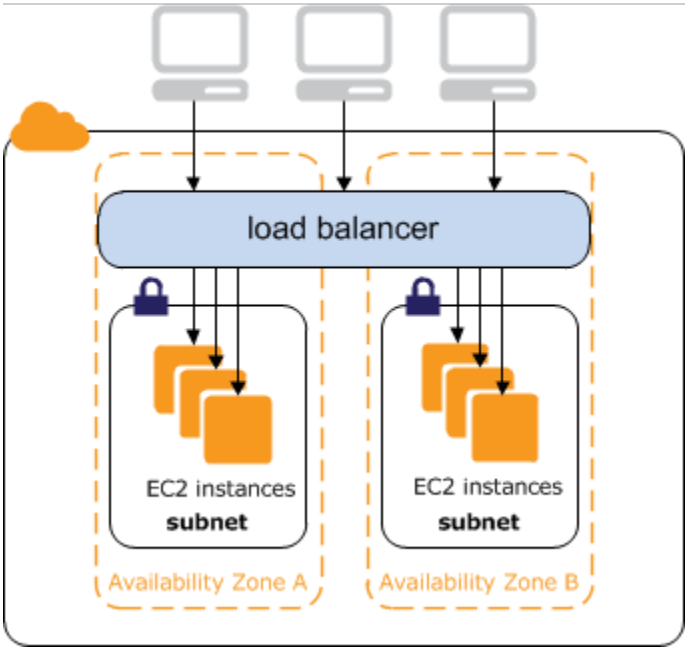
- Classic Load Balancer를 생성하고 EC2 인스턴스에 이를 등록하는 방법을 알아보려면 인터넷 경계 Classic Load Balancer 생성을(를) 참조하세요.
- HTTPS 로드 밸런서를 생성하여 EC2 인스턴스를 등록하는 방법을 알아보려면 HTTPS 리스너를 통해 Classic Load Balancer를 생성을(를) 참조하세요.
- Classic Load Balancer가 지원하는 다양한 기능의 사용 방법을 알아보려면 Classic Load Balancer 구성 섹션을 참조하세요.

가격 책정

로드 밸런서에서는 사용한 만큼만 지불하면 됩니다. 자세한 정보는 Elastic Load Balancing 요금을 참조하세요.

인터넷 경계 Classic Load Balancer

Classic Load Balancer를 생성할 때 로드 밸런서를 내부 로드 밸런서 또는 인터넷 경계 로드 밸런서로 생성할 수 있습니다. 인터넷 경계 로드 밸런서는 공개적으로 확인이 가능한 DNS 이름을 가지고 있으므로 인터넷을 통해 클라이언트의 요청을 로드 밸런서에 등록된 EC2 인스턴스로 라우팅할 수 있습니다.



내부 로드 밸런서의 DNS 이름은 노드의 프라이빗 IP 주소로 공개적으로 확인이 가능합니다. 따라서 내부 로드 밸런서는 로드 밸런서를 위한 VPC에 액세스하여 클라이언트의 요청만 라우팅할 수 있습니다. 자세한 내용은 내부 로드 밸런서 섹션을 참조하세요.

내용

로드 밸런서의 퍼블릭 DNS 이름

로드 밸런서를 생성하면 클라이언트가 요청을 전송하기 위해 사용하는 퍼블릭 DNS 이름을 받습니다. DNS 서버는 로드 밸런서의 DNS 이름을 내부 로드 밸런서에 대한 로드 밸런서 노드의 퍼블릭 IP 주소에 포함합니다. 각 로드 밸런서는 프라이빗 IP 주소를 사용하여 백엔드 인스턴스에 연결합니다.

콘솔에 다음 형식의 퍼블릭 DNS 이름이 표시됩니다.

```
name-1234567890.region.elb.amazonaws.com
```

인터넷 경계 Classic Load Balancer 생성

로드 밸런서를 생성할 때 리스너 및 상태 확인을 구성하며 백엔드 인스턴스를 등록합니다. 프런트 엔드(클라이언트에서 로드 밸런서) 연결을 위한 프로토콜 및 포트와 백엔드(로드 밸런서에서 백엔드 인스턴스) 연결을 위한 프로토콜 및 포트를 지정하여 리스너를 구성합니다. 로드 밸런서에 대해 여러 리스너를 구성할 수 있습니다.

이 자습서에서는 웹 기반 인터페이스 AWS Management Console인를 통해 Classic Load Balancer를 직접 소개합니다. 퍼블릭 HTTP 트래픽을 수신해 EC2 인스턴스로 전송하는 로드 밸런서를 만들어 보겠습니다.

HTTPS 리스너를 가진 로드 밸런서를 생성하려면 HTTPS 리스너를 통해 Classic Load Balancer를 생성 단원을 참조하십시오.

업무

- 시작하기 전 준비 사항
- 를 사용하여 Classic Load Balancer 생성 AWS Management Console

시작하기 전 준비 사항

- Virtual Private Cloud(VPC)를 생성합니다. 자세한 내용은 VPC 관련 권장 사항 단원을 참조하십시오.
- 로드 밸런서에 등록할 계획인 EC2 인스턴스를 시작합니다. 이들 인스턴스에 대한 보안 그룹이 포트 80에서 HTTP 액세스를 허용하는지 확인합니다.
- Apache 또는 IIS(인터넷 정보 서비스) 같은 웹 서버를 각 인스턴스에 설치하고 인터넷에 연결된 웹 브라우저의 주소 필드에 DNS 이름을 입력한 다음, 브라우저가 서버의 기본 페이지를 표시하는지 확인합니다.

를 사용하여 Classic Load Balancer 생성 AWS Management Console

다음 절차를 따라 Classic Load Balancer를 생성합니다. 이름과 스키마와 같은 로드 밸런서의 기본 구성 정보를 제공합니다. 그런 다음 네트워크 관련 정보 및 트래픽을 인스턴스로 라우팅하는 리스너 관련 정보를 제공합니다.

콘솔을 사용하여 Classic Load Balancer를 생성하려면

1. https://console.aws.amazon.com/ec2/에서 Amazon EC2 콘솔을 엽니다.
2. 탐색 모음에서 로드 밸런서의 리전을 선택합니다. EC2 인스턴스를 위해 선택한 리전과 동일한 리전을 선택해야 합니다.
3. 탐색 창의 Load Balancing 아래에서 로드 밸런서를 선택합니다.
4. 로드 밸런서 생성을 선택합니다.
5. Classic Load Balancer 섹션을 확장하고 생성을 선택합니다.
6. 기본 구성

   a. 로드 밸런서 이름에 로드 밸런서의 이름을 입력합니다.

      Classic Load Balancer의 이름은 해당 리전의 Classic Load Balancer 세트 내에서 고유한 이름이어야 하고, 최대 32자여야 하며, 알파벳 문자 및 하이픈만 포함해야 하고, 하이픈으로 시작하거나 끝나지 않아야 합니다.

   b. 스키마에서 인터넷 연결을 선택합니다.
7. 네트워크 매핑

   a. VPC에서는 인스턴스에 대해 선택한 것과 동일한 VPC를 선택합니다.

   b. 매핑에서 먼저 가용 영역을 선택하고, 사용 가능한 서브넷 중에서 퍼블릭 서브넷을 선택합니다. 가용 영역당 서브넷 한 개만 선택할 수 있습니다. 로드 밸런서의 가용성을 높이려면 둘 이상의 가용 영역과 서브넷을 선택합니다.
8. 보안 그룹

   - 보안 그룹에서 포트 80에서 필요한 HTTP 트래픽을 허용하도록 구성된 기존 보안 그룹을 선택합니다.
9. 리스너 및 라우팅

   a. 리스너에서 프로토콜이 HTTP이고 포트가 80인지 확인합니다.

   b. 인스턴스에서 프로토콜이 HTTP이고 포트가 80인지 확인합니다.
10. 상태 확인

    a. ping 프로토콜에서 프로토콜이 HTTP인지 확인합니다.

    b. ping 포트에서 포트가 80인지 확인합니다.

    c. ping 경로에서 경로가 /인지 확인합니다.

    d. 고급 상태 확인 설정에서 기본값을 사용합니다.

11. 인스턴스

    a. 인스턴스 추가를 선택하면 인스턴스 선택 화면이 나타납니다.

    b. 사용 가능한 인스턴스에서 현재 네트워크 설정을 기반으로 로드 밸런서에 사용 가능한 현재 인스턴스 중 선택할 수 있습니다.

    c. 원하는 대로 선택했다면 확인을 선택하여 로드 밸런서에 등록할 인스턴스를 추가합니다.

12. 속성

    • 교차 영역 로드 밸런싱 활성화, Connection Draining 활성화, 제한 시간(드레이닝 간격)의 기본값은 유지합니다.

13. 로드 밸런서 태그(선택 사항)

    a. 키 필드는 필수입니다.

    b. 값 필드는 선택 사항입니다.

    c. 다른 태그를 추가하려면 새 태그 추가를 선택하고 키 필드 및 값 필드(선택 사항)에 값을 입력합니다.

    d. 제거하려는 태그 옆에 있는 제거를 선택하여 기존 태그를 제거합니다.

14. 요약 및 생성

    a. 설정을 변경해야 하는 경우 변경해야 하는 설정 옆에 있는 편집을 선택합니다.

    b. 요약에 표시된 모든 설정이 원하는 대로 되어 있다면 로드 밸런서 생성을 선택하여 로드 밸런서 생성을 시작합니다.

    c. 마지막 생성 페이지에서 로드 밸런서 보기를 선택하여 Amazon EC2 콘솔에서 로드 밸런서를 봅니다.

15. Verify

    a. 새로운 로드 밸런서를 선택합니다.

    b. 대상 인스턴스 탭에서 상태 확인 열을 확인합니다. EC2 인스턴스 중 적어도 하나 이상이 서비스 상태가 되어야만 로드 밸런서를 테스트할 수 있습니다.

    c. 세부 정보 섹션에서 로드 밸런서 DNS 이름을 복사합니다(`my-load-balancer-1234567890.us-east-1.elb.amazonaws.com`과 유사함).

    d. 로드 밸런서 DNS 이름을 퍼블릭 인터넷에 연결된 웹 브라우저의 주소 필드에 붙여넣습니다. 로드 밸런서가 올바르게 작동 중인 경우 서버의 기본 페이지가 표시됩니다.

16. 삭제(선택 사항)

a. 로드 밸런서를 가리키는 도메인을 위한 CNAME 레코드가 있는 경우에는 새로운 위치를 가리키도록 하고 로드 밸런서를 삭제하기 전에 DNS 변경이 적용될 때까지 기다립니다.

b. https://console.aws.amazon.com/ec2/에서 Amazon EC2 콘솔을 엽니다.

c. 로드 밸런서를 선택합니다.

d. 작업, 로드 밸런서 삭제를 선택합니다.

e. 확인 메시지가 나타나면 `confirm`을 입력한 다음 삭제를 선택합니다.

f. 로드 밸런서를 삭제한 후에도 로드 밸런서에 등록된 EC2 인스턴스는 계속 실행됩니다. 계속 실행되는 일부 또는 전체 시간별로 요금이 청구됩니다. EC2 인스턴스가 더 이상 필요하지 않은 경우 추가 요금이 발생하지 않도록 중지하거나 종료할 수 있습니다.

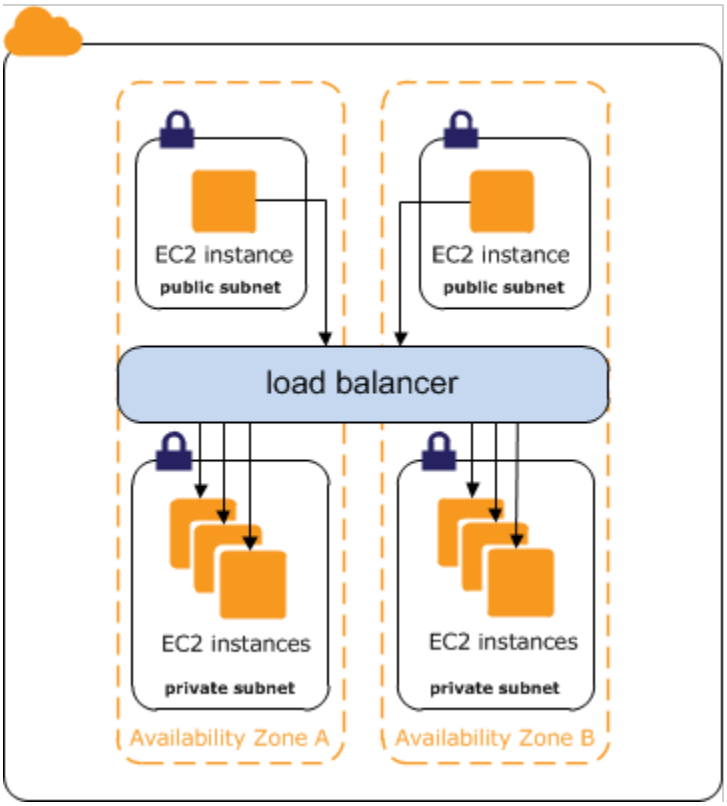
내부 Classic Load Balancer

로드 밸런서를 생성할 때 로드 밸런서를 내부 로드 밸런서 또는 인터넷 경계 로드 밸런서로 생성할지 여부를 선택해야 합니다.

인터넷 경계 로드 밸런서의 노드는 퍼블릭 IP 주소를 가집니다. 인터넷 경계 로드 밸런서의 DNS 이름은 노드의 퍼블릭 IP 주소로 공개적으로 확인이 가능합니다. 따라서 인터넷 경계 로드 밸런서는 인터넷을 통해 클라이언트의 요청을 라우팅할 수 있습니다. 자세한 내용은 인터넷 경계 Classic Load Balancer 단원을 참조하십시오.

내부 로드 밸런서의 노드는 오직 프라이빗 IP 주소만 가집니다. 내부 로드 밸런서의 DNS 이름은 노드의 프라이빗 IP 주소로 공개적으로 확인이 가능합니다. 따라서 내부 로드 밸런서는 로드 밸런서를 위한 VPC에 액세스하여 클라이언트의 요청만 라우팅할 수 있습니다.

애플리케이션에 여러 개의 티어가 있는 경우(예: 인터넷에 반드시 연결되어야 하는 웹 서버와 웹 서버에만 연결되는 데이터베이스 서버) 내부 로드 밸런서와 인터넷 경계 로드 밸런서를 모두 사용하는 아키텍처를 설계할 수 있습니다. 인터넷 경계 로드 밸런서를 생성하고 여기에 웹 서버를 등록합니다. 내부 로드 밸런서를 생성하고 여기에 데이터베이스 서버를 등록합니다. 웹 서버는 인터넷 경계 로드 밸런서에서 요청을 수신하고 데이터베이스 서버에서 내부 로드 밸런서로 요청을 전송합니다. 데이터베이스 서버는 내부 로드 밸런서에서 요청을 수신합니다.



내용

로드 밸런서의 퍼블릭 DNS 이름

내부 로드 밸런서를 생성하면 다음 형식의 퍼블릭 DNS 이름을 받습니다.

```
internal-name-123456789.region.elb.amazonaws.com
```

DNS 서버는 로드 밸런서의 DNS 이름을 내부 로드 밸런서에 대한 로드 밸런서 노드의 프라이빗 IP 주소에 포함합니다. 각 로드 밸런서 노드는 탄력적 네트워크 인터페이스를 사용하여 백엔드 인스턴스의 프라이빗 IP 주소에 연결됩니다. 영역 간 로드 밸런싱을 활성화하면 가용 영역에 관계없이 각 노드가 백엔드 인스턴스에 각각 연결됩니다. 그렇지 않으면 각 노드는 가용 영역에 있는 인스턴스에만 연결됩니다.

내부 Classic Load Balancer 생성

로드 밸런서용 VPC에 액세스 권한이 있는 클라이언트에서 EC2 인스턴스로 트래픽을 분산시키는 내부 로드 밸런서를 생성할 수 있습니다.

내용

사전 조건

- 로드 밸런서용 VPC를 아직 생성하지 않은 경우 시작하기 전에 VPC를 먼저 생성해야 합니다. 자세한 내용은 VPC 관련 권장 사항 단원을 참조하십시오.
- 내부 로드 밸런서에 등록할 계획인 EC2 인스턴스를 시작합니다. 로드 밸런서용 VPC의 프라이빗 서브넷에서 시작해야 합니다.

콘솔을 사용하여 내부 로드 밸런서 생성

다음 절차를 따라 내부 Classic Load Balancer를 생성합니다. 이름과 스키마와 같은 로드 밸런서의 기본 구성 정보를 제공합니다. 그런 다음 네트워크 관련 정보 및 트래픽을 인스턴스로 라우팅하는 리스너 관련 정보를 제공합니다.

콘솔을 사용하여 내부 Classic Load Balancer를 생성하려면

1. https://console.aws.amazon.com/ec2/에서 Amazon EC2 콘솔을 엽니다.
2. 탐색 모음에서 로드 밸런서의 리전을 선택합니다. EC2 인스턴스를 위해 선택한 리전과 동일한 리전을 선택해야 합니다.
3. 탐색 창의 Load Balancing 아래에서 로드 밸런서를 선택합니다.
4. 로드 밸런서 생성을 선택합니다.
5. Classic Load Balancer 섹션을 확장하고 생성을 선택합니다.
6. 기본 구성
   a. 로드 밸런서 이름에 로드 밸런서의 이름을 입력합니다.

      Classic Load Balancer의 이름은 해당 리전의 Classic Load Balancer 세트 내에서 고유한 이름이어야 하고, 최대 32자여야 하며, 알파벳 문자 및 하이픈만 포함해야 하고, 하이픈으로 시작하거나 끝나지 않아야 합니다.

   b. 스키마에서 내부를 선택합니다.
7. 네트워크 매핑
   a. VPC에서는 인스턴스에 대해 선택한 것과 동일한 VPC를 선택합니다.
   b. 매핑에서 먼저 가용 영역을 선택하고, 사용 가능한 서브넷 중에서 서브넷을 선택합니다. 가용 영역당 서브넷 한 개만 선택할 수 있습니다. 로드 밸런서의 가용성을 높이려면 둘 이상의 가용 영역과 서브넷을 선택합니다.
8. 보안 그룹에서 포트 80에서 필요한 HTTP 트래픽을 허용하도록 구성된 기존 보안 그룹을 선택합니다. 또는 애플리케이션이 다른 프로토콜과 포트를 사용하는 경우 새 보안 그룹을 생성할 수 있습니다.
9. 리스너 및 라우팅
   a. 리스너에서 프로토콜이 HTTP이고 포트가 80인지 확인합니다.
   b. 인스턴스에서 프로토콜이 HTTP이고 포트가 80인지 확인합니다.
10. 상태 확인

a. ping 프로토콜에서 기본값은 HTTP입니다.

b. ping 포트에서 기본값은 80입니다.

c. ping 경로에서 기본값은 /입니다.

d. 고급 상태 확인 설정에서 기본값을 사용하거나 애플리케이션에 맞는 값을 입력합니다.

11. 인스턴스

a. 인스턴스 추가를 선택하면 인스턴스 선택 화면이 나타납니다.

b. 사용 가능한 인스턴스에서 앞서 선택한 네트워크 설정을 기반으로 로드 밸런서에 사용 가능한 현재 인스턴스 중 선택할 수 있습니다.

c. 원하는 대로 선택했다면 확인을 선택하여 로드 밸런서에 등록할 인스턴스를 추가합니다.

12. 속성

- 교차 영역 로드 밸런싱 활성화, Connection Draining 활성화, 제한 시간(드레이닝 간격)의 기본값은 유지합니다.

13. 로드 밸런서 태그(선택 사항)

a. 키 필드는 필수입니다.

b. 값 필드는 선택 사항입니다.

c. 다른 태그를 추가하려면 새 태그 추가를 선택하고 키 필드 및 값 필드(선택 사항)에 값을 입력합니다.

d. 제거하려는 태그 옆에 있는 제거를 선택하여 기존 태그를 제거합니다.

14. 요약 및 생성

a. 설정을 변경해야 하는 경우 변경해야 하는 설정 옆에 있는 편집을 선택합니다.

b. 요약에 표시된 모든 설정이 원하는 대로 되어 있다면 로드 밸런서 생성을 선택하여 로드 밸런서 생성을 시작합니다.

c. 마지막 생성 페이지에서 로드 밸런서 보기를 선택하여 Amazon EC2 콘솔에서 로드 밸런서를 봅니다.

15. Verify

a. 새로운 로드 밸런서를 선택합니다.

b. 대상 인스턴스 탭에서 상태 확인 열을 확인합니다. EC2 인스턴스 중 적어도 하나 이상이 서비스 상태가 되어야만 로드 밸런서를 테스트할 수 있습니다.

c. 세부 정보 섹션에서 로드 밸런서 DNS 이름을 복사합니다(`my-load-balancer-1234567890.us-east-1.elb.amazonaws.com`과 유사함).

d. 로드 밸런서 DNS 이름을 퍼블릭 인터넷에 연결된 웹 브라우저의 주소 필드에 붙여넣습니다. 로드 밸런서가 올바르게 작동 중인 경우 서버의 기본 페이지가 표시됩니다.

16. 삭제(선택 사항)

a. 로드 밸런서를 가리키는 도메인을 위한 CNAME 레코드가 있는 경우에는 새로운 위치를 가리키도록 하고 로드 밸런서를 삭제하기 전에 DNS 변경이 적용될 때까지 기다립니다.

b. https://console.aws.amazon.com/ec2/에서 Amazon EC2 콘솔을 엽니다.

c. 로드 밸런서를 선택합니다.

d. 작업, 로드 밸런서 삭제를 선택합니다.

e. 확인 메시지가 나타나면 `confirm`을 입력한 다음 삭제를 선택합니다.

f. 로드 밸런서를 삭제한 후에도 로드 밸런서에 등록된 EC2 인스턴스는 계속 실행됩니다. 계속 실행되는 일부 또는 전체 시간별로 요금이 청구됩니다. EC2 인스턴스가 더 이상 필요하지 않은 경우 추가 요금이 발생하지 않도록 중지하거나 종료할 수 있습니다.

를 사용하여 내부 로드 밸런서 생성 AWS CLI

기본적으로 Elastic Load Balancing은 인터넷 연결 로드 밸런서를 생성합니다. 다음 절차에 따라 내부 로드 밸런서를 생성하고 EC2 인스턴스를 새로 생성한 내부 로드 밸런서에 등록합니다.

내부 로드 밸런서를 만들려면

1. 다음과 같이 `--scheme` 옵션을 `internal`로 설정한 create-load-balancer 명령을 사용합니다.

```
aws elb create-load-balancer --load-balancer-name my-internal-loadbalancer --
listeners Protocol=HTTP,LoadBalancerPort=80,InstanceProtocol=HTTP,InstancePort=80
 --subnets subnet-4e05f721 --scheme internal --security-groups sg-b9ffedd5
```

다음은 응답의 예입니다. 해당 이름은 이것이 내부 로드 밸런서라는 것을 나타냅니다.

```
{
    "DNSName": "internal-my-internal-loadbalancer-786501203.us-
west-2.elb.amazonaws.com"
}
```

2. 다음과 같이 register-instances-with-load-balancer 명령을 사용하여 인스턴스를 추가합니다.

```
aws elb register-instances-with-load-balancer --load-balancer-name my-internal-loadbalancer --instances i-4f8cf126 i-0bb7ca62
```

다음은 응답의 예입니다.

```
{
    "Instances": [
        {
            "InstanceId": "i-4f8cf126"
        },
        {
            "InstanceId": "i-0bb7ca62"
        }
    ]
}
```

3. (선택 사항) 다음 describe-load-balancers 명령을 사용하여 내부 로드 밸런서를 확인합니다.

```
aws elb describe-load-balancers --load-balancer-name my-internal-loadbalancer
```

응답에는 이것이 내부 로드 밸런서라는 것을 나타내는 DNSName 및 Scheme 필드가 포함되어 있습니다.

```
{
    "LoadBalancerDescriptions": [
        {
            ...
            "DNSName": "internal-my-internal-loadbalancer-1234567890.us-
west-2.elb.amazonaws.com",
            "SecurityGroups": [
                "sg-b9ffedd5"
            ],
            "Policies": {
                "LBCookieStickinessPolicies": [],
                "AppCookieStickinessPolicies": [],
                "OtherPolicies": []
            },
            "LoadBalancerName": "my-internal-loadbalancer",
            "CreatedTime": "2014-05-22T20:32:19.920Z",
            "AvailabilityZones": [
                "us-west-2a"
```

```
            ],
            "Scheme": "internal",
            ...
        }
    ]
}
```

Classic Load Balancer 구성

Classic Load Balancer를 생성한 후 구성을 변경할 수 있습니다. 예를 들어 로드 밸런서 속성, 서브넷 및 보안 그룹을 업데이트할 수 있습니다.

로드 밸런서 속성

연결 드레이닝

활성화할 경우, 로드 밸런서가 등록 취소된 또는 비정상 백엔드 인스턴스에 대한 트래픽을 차단하기 전에 기존 요청을 완료할 수 있게 해줍니다.

교차 영역 로드 밸런싱

활성화할 경우, 로드 밸런서는 가용 영역과 상관없이 요청 트래픽을 모든 인스턴스에 균일하게 라우팅합니다.

비동기화 완화 모드

애플리케이션에 보안 위험을 초래할 수 있는 요청을 로드 밸런서에서 처리하는 방법을 결정합니다. 가능한 값은 `monitor`, `defensive` 및 `strictest` 입니다. 기본값은 `defensive`입니다.

유휴 제한 시간

활성화할 경우, 로드 밸런서에서 지정된 시간 동안 연결을 유휴 상태(연결을 통해 데이터를 전송하지 않는 상태)로 유지하도록 허용합니다. 기본값은 60초입니다.

고정 세션

Classic Load Balancer는 기간 기반 세션과 애플리케이션 기반 세션의 고정성을 모두 지원합니다.

로드 밸런서 세부 정보

보안 그룹

로드 밸런서의 보안 그룹은 리스너 및 상태 확인 포트에서 트래픽을 허용해야 합니다.

서브넷

로드 밸런서의 기능을 추가 서브넷까지 확장할 수 있습니다.

프록시 프로토콜

활성화된 경우 인스턴스로 전송되는 연결 정보가 포함된 헤더를 추가합니다.

Tags

태그를 추가하여 로드 밸런서를 분류할 수 있습니다.

Classic Load Balancer에 대한 유휴 연결 제한 시간 구성

클라이언트가 Classic Load Balancer를 통해 생성하는 각 요청에 대해 로드 밸런서는 두 가지 연결을 유지합니다. 프런트 엔드 연결은 클라이언트와 로드 밸런서 사이에 있습니다. 백엔드 연결은 로드 밸런서와 등록된 EC2 인스턴스 사이에 있습니다. 로드 밸런서에 해당 연결에 적용되는 유휴 제한 시간 기간이 구성되어 있습니다. 유휴 제한 시간이 경과할 때까지 데이터가 전송되거나 전송 또는 수신되지 않으면 로드 밸런서는 프런트 엔드 연결을 종료합니다. 파일 업로드 같이 시간이 오래 걸리는 작업이 완료될 수 있도록 시간 여유를 두려면 유휴 제한 시간이 지나기 전에 최소 1바이트의 데이터를 전송하고 필요에 따라 유휴 제한 시간의 길이를 늘립니다.

HTTP 및 HTTPS 리스너를 사용하는 경우에는 인스턴스에 대해 HTTP 연결 유지 옵션을 활성화하는 것이 좋습니다. 연결 유지는 인스턴스의 웹 서버 설정에서 활성화할 수 있습니다. 연결 유지를 활성화하면 연결 유지 제한 시간이 만료될 때까지 로드 밸런서가 백엔드 연결을 다시 사용할 수 있습니다. 로드 밸런서가 인스턴스에 대한 연결을 종료할 책임이 있는지 확인하고 싶다면 HTTP 연결 유지 시간 동안 설정된 값이 로드 밸런서에 구성된 유휴 제한 시간 설정보다 큰지 확인합니다.

TCP 연결 유지 프로브는 페이로드에 데이터를 전송하지 않기 때문에 로드 밸런서에서 연결이 종료되지 않도록 방지합니다.

목차

콘솔을 사용하여 유휴 제한 시간 구성

기본적으로 Elastic Load Balancing은 로드 밸런서의 유휴 시간 초과를 60초로 설정합니다. 다른 유휴 제한 시간 값을 설정하려면 다음 절차를 따르십시오.

콘솔을 사용하여 로드 밸런서에서 유휴 제한 시간 설정을 구성하려면

1. https://console.aws.amazon.com/ec2/에서 Amazon EC2 콘솔을 엽니다.
2. 탐색 창의 Load Balancing 아래에서 로드 밸런서를 선택합니다.
3. 로드 밸런서 이름을 선택하여 세부 정보 페이지를 엽니다.

4. 속성 탭에서 편집을 선택합니다.
5. 로드 밸런서 속성 편집 페이지의 트래픽 구성 섹션에서 유휴 제한 시간 값을 입력합니다. 유휴 제한 시간의 범위는 1 ~ 4,000초입니다.
6. 변경 사항 저장을 선택합니다.

를 사용하여 유휴 제한 시간 구성 AWS CLI

로드 밸런서에서 유휴 제한 시간을 설정하려면 아래 modify-load-balancer-attributes 명령을 사용하세요.

```
aws elb modify-load-balancer-attributes --load-balancer-name my-loadbalancer --load-balancer-attributes "{\"ConnectionSettings\":{\"IdleTimeout\":30}}"
```

다음은 응답의 예입니다.

```
{
    "LoadBalancerAttributes": {
        "ConnectionSettings": {
            "IdleTimeout": 30
        }
    },
    "LoadBalancerName": "my-loadbalancer"
}
```

Classic Load Balancer에서 교차 영역 로드 밸런싱을 구성

교차 영역 로드 밸런싱을 사용하면 Classic Load Balancer에 대한 각각의 로드 밸런서 노드가 활성화된 모든 가용 영역에 있는 등록된 인스턴스 간에 요청을 균등하게 분산합니다. 교차 영역 로드 밸런싱이 비활성화된 경우에는 각각의 로드 밸런서 노드가 해당 가용 영역에만 있는 등록된 인스턴스 간에 요청을 균등하게 분산합니다. 자세한 내용은 Elastic Load Balancing 사용 설명서의 교차 영역 로드 밸런싱을 참조하세요.

교차 영역 로드 밸런싱을 사용하면 활성화된 각 가용 영역에 동일한 수의 인스턴스를 유지할 필요가 없으며, 애플리케이션이 보다 효과적으로 하나 이상의 인스턴스 손실을 처리할 수 있습니다. 하지만 내결함성을 높이기 위해서는 활성화된 각 가용 영역에서 인스턴스 수를 대략적으로 동일하게 유지하는 것이 좋습니다.

클라이언트가 DNS 조회를 캐싱하는 환경에서는 들어오는 요청이 가용 영역 중 하나를 선호할 수 있습니다. 교차 영역 로드 밸런싱을 사용하면 요청 로드의 이러한 불균형을 리전의 모든 가용 인스턴스로 분산시켜서 클라이언트 작동에 문제가 있을 때 그 영향을 줄일 수 있습니다.

Classic Load Balancer를 생성하면 교차 영역 로드 밸런싱에 대한 기본값은 로드 밸런서 생성 방법에 따라 달라집니다. API 또는 CLI에서는 교차 영역 로드 밸런싱이 기본적으로 비활성화되어 있습니다. AWS Management Console를 사용하면 교차 영역 로드 밸런싱을 활성화하는 옵션이 기본적으로 선택됩니다. Classic Load Balancer를 생성한 후 언제든지 교차 영역 로드 밸런싱을 활성화하거나 비활성화할 수 있습니다.

목차

교차 영역 로드 밸런싱 활성화

Classic Load Balancer에서 언제든지 교차 영역 로드 밸런싱을 활성화할 수 있습니다.

콘솔을 사용하여 교차 영역 로드 밸런싱을 활성화하려면

1. https://console.aws.amazon.com/ec2/에서 Amazon EC2 콘솔을 엽니다.
2. 탐색 창의 Load Balancing 아래에서 로드 밸런서를 선택합니다.
3. 로드 밸런서 이름을 선택하여 세부 정보 페이지를 엽니다.
4. 속성 탭에서 편집을 선택합니다.
5. 로드 밸런서 속성 편집 페이지의 가용 영역 라우팅 구성 섹션에서 교차 영역 로드 밸런싱을 활성화합니다.
6. 변경 사항 저장을 선택합니다.

를 사용하여 교차 영역 로드 밸런싱을 활성화하려면 AWS CLI

1. 다음 modify-load-balancer-attributes 명령을 사용하여 로드 밸런서의 `CrossZoneLoadBalancing` 속성을 `true`로 설정하세요.

```
aws elb modify-load-balancer-attributes --load-balancer-name my-loadbalancer --
load-balancer-attributes "{\"CrossZoneLoadBalancing\":{\"Enabled\":true}}"
```

   다음은 응답의 예입니다.

```
{
   "LoadBalancerAttributes": {
     "CrossZoneLoadBalancing": {
         "Enabled": true
       }
   },
   "LoadBalancerName": "my-loadbalancer"
 }
```

2. (선택 사항) 로드 밸런서에서 교차 영역 로드 밸런싱이 활성화되었는지 확인하려면 아래 describe-load-balancer-attributes 명령을 사용하세요.

```
aws elb describe-load-balancer-attributes --load-balancer-name my-loadbalancer
```

다음은 응답의 예입니다.

```
{
    "LoadBalancerAttributes": {
        "ConnectionDraining": {
            "Enabled": false,
            "Timeout": 300
        },
        "CrossZoneLoadBalancing": {
            "Enabled": true
        },
        "ConnectionSettings": {
            "IdleTimeout": 60
        },
        "AccessLog": {
            "Enabled": false
        }
    }
}
```

교차 영역 로드 밸런싱 비활성화

로드 밸런서에서의 언제든지 교차 영역 로드 밸런싱 옵션을 비활성화할 수 있습니다.

콘솔을 사용하여 교차 영역 로드 밸런싱을 비활성화하려면

1. https://console.aws.amazon.com/ec2/에서 Amazon EC2 콘솔을 엽니다.
2. 탐색 창의 Load Balancing 아래에서 로드 밸런서를 선택합니다.
3. 로드 밸런서 이름을 선택하여 세부 정보 페이지를 엽니다.
4. 속성 탭에서 편집을 선택합니다.
5. 로드 밸런서 속성 편집 페이지의 가용 영역 라우팅 구성 섹션에서 교차 영역 로드 밸런싱을 비활성화합니다.
6. 변경 사항 저장을 선택합니다.

교차 영역 로드 밸런싱을 비활성화하려면 로드 밸런서의 `CrossZoneLoadBalancing` 속성을 `false`로 설정합니다.

를 사용하여 교차 영역 로드 밸런싱을 비활성화하려면 AWS CLI

1. 아래 modify-load-balancer-attributes 명령을 사용하세요.

```
aws elb modify-load-balancer-attributes --load-balancer-name my-loadbalancer --
load-balancer-attributes "{\"CrossZoneLoadBalancing\":{\"Enabled\":false}}"
```

   다음은 응답의 예입니다.

```
{
   "LoadBalancerAttributes": {
     "CrossZoneLoadBalancing": {
         "Enabled": false
       }
   },
   "LoadBalancerName": "my-loadbalancer"
 }
```

2. (선택 사항) 로드 밸런서에서 교차 영역 로드 밸런싱이 비활성화되었는지 확인하려면 아래describe-load-balancer-attributes 명령을 사용하세요.

```
aws elb describe-load-balancer-attributes --load-balancer-name my-loadbalancer
```

   다음은 응답의 예입니다.

```
{
    "LoadBalancerAttributes": {
        "ConnectionDraining": {
            "Enabled": false,
            "Timeout": 300
        },
        "CrossZoneLoadBalancing": {
            "Enabled": false
        },
        "ConnectionSettings": {
            "IdleTimeout": 60
        },
        "AccessLog": {
            "Enabled": false
        }
    }
}
```

Classic Load Balancer에서 Connection Draining 구성

기존 연결이 열려 있는 상태에서 Classic Load Balancer가 등록 취소 중이거나 비정상 상태인 인스턴스로의 요청 전송을 중지하도록 하려면 Connection Draining을 사용하세요. 이렇게 하면 로드 밸런서가 등록 취소 중이거나 비정상 상태인 인스턴스로 진행 중인 요청을 완료합니다.

Connection Draining을 활성화하면 로드 밸런서가 인스턴스의 등록 취소를 보고하기 전에 연결을 유지할 수 있는 최대 시간을 지정할 수 있습니다. 최대 제한 시간 값의 범위는 1 ~ 3,600초입니다(기본은 300초). 최대 제한 시간에 도달하면 로드 밸런서는 등록 취소 중인 인스턴스로의 연결을 강제로 종료합니다.

등록을 취소하는 인스턴스에 진행 중인 요청이 없고 활성 연결이 없는 경우 Elastic Load Balancing은 등록 취소 프로세스를 즉시 완료합니다.

진행 중인 요청이 처리되는 동안 로드 밸런서는 등록 취소 중인 인스턴스의 상태를 `InService: Instance deregistration currently in progress`로 보고합니다. 등록 취소 중인 인스턴스가 진행 중인 모든 요청의 처리를 완료하거나 최대 제한 시간에 도달하면 로드 밸런서는 인스턴스 상태를 `OutOfService: Instance is not currently registered with the LoadBalancer`로 보고합니다.

인스턴스가 비정상 상태가 되면 로드 밸런서는 인스턴스 상태를 `OutOfService`로 보고합니다. 비정상 상태의 인스턴스로의 요청이 진행 중인 경우에는 해당 요청이 완료됩니다. 최대 제한 시간은 비정상 상태의 인스턴스에 대한 연결에는 적용되지 않습니다.

인스턴스가 Auto Scaling 그룹의 일부이고 로드 밸런서에 대해 Connection Draining이 활성화된 경우 Auto Scaling은 조정 이벤트 또는 상태 확인 교체로 인해 인스턴스를 종료하기 전에 진행 중인 요청이 완료되거나 최대 시간 초과가 만료될 때까지 기다립니다.

로드 밸런서가 등록 취소 중이거나 비정상 상태가 된 인스턴스에 대한 연결을 즉시 종료하도록 하고 싶은 경우에는 Connection Draining을 비활성화할 수 있습니다. Connection Draining이 비활성화되면 등록 취소 중이거나 비정상 상태인 인스턴스로 진행 중인 요청이 완료되지 않습니다.

목차

Connection Draining 활성화

언제든지 로드 밸런서에 대한 Connection Draining을 활성화할 수 있습니다.

콘솔을 사용하여 Connection Draining을 활성화하려면

1. https://console.aws.amazon.com/ec2/에서 Amazon EC2 콘솔을 엽니다.
2. 탐색 창의 Load Balancing 아래에서 로드 밸런서를 선택합니다.
3. 로드 밸런서 이름을 선택하여 세부 정보 페이지를 엽니다.
4. 속성 탭에서 편집을 선택합니다.
5. 로드 밸런서 속성 편집 페이지의 트래픽 구성 섹션에서 Connection Draining 활성화를 선택합니다.
6. (선택 사항) 제한 시간(드레이닝 간격)에 1~3,600초 범위 내에서 값을 입력합니다. 그러지 않으면 기본값인 300초가 사용됩니다.
7. 변경 사항 저장을 선택합니다.

를 사용하여 연결 드레이닝을 활성화하려면 AWS CLI

아래 modify-load-balancer-attributes 명령을 사용하세요.

```
aws elb modify-load-balancer-attributes --load-balancer-name my-loadbalancer --load-
balancer-attributes "{\"ConnectionDraining\":{\"Enabled\":true,\"Timeout\":300}}"
```

다음은 응답의 예입니다.

```
{
    "LoadBalancerAttributes": {
        "ConnectionDraining": {
            "Enabled": true,
            "Timeout": 300
        }
    },
    "LoadBalancerName": "my-loadbalancer"
}
```

Connection Draining 비활성화

언제든지 로드 밸런서에 대한 Connection Draining을 비활성화할 수 있습니다.

콘솔을 사용하여 Connection Draining을 비활성화하려면

1. https://console.aws.amazon.com/ec2/에서 Amazon EC2 콘솔을 엽니다.
2. 탐색 창의 Load Balancing 아래에서 로드 밸런서를 선택합니다.
3. 로드 밸런서 이름을 선택하여 세부 정보 페이지를 엽니다.
4. 속성 탭에서 편집을 선택합니다.
5. 로드 밸런서 속성 편집 페이지의 트래픽 구성 섹션에서 Connection Draining 활성화를 선택 취소합니다.
6. 변경 사항 저장을 선택합니다.

를 사용하여 연결 드레이닝을 비활성화하려면 AWS CLI

아래 modify-load-balancer-attributes 명령을 사용하세요.

```
aws elb modify-load-balancer-attributes --load-balancer-name my-loadbalancer --load-
balancer-attributes "{\"ConnectionDraining\":{\"Enabled\":false}}"
```

다음은 응답의 예입니다.

```
{
    "LoadBalancerAttributes": {
        "ConnectionDraining": {
            "Enabled": false,
            "Timeout": 300
        }
    },
    "LoadBalancerName": "my-loadbalancer"
}
```

Classic Load Balancer를 위한 고정 세션 구성

기본적으로 Classic Load Balancer는 로드가 가장 적은 등록 인스턴스에 각 요청을 독립적으로 라우팅합니다. 한편, 고정 세션 기능(세션 어피니티라고도 함)을 사용해 로드 밸런서가 사용자의 세션을 특정 인스턴스에 바인딩하도록 할 수 있습니다. 이렇게 하면 세션 중에 사용자로부터 들어오는 모든 요청이 동일한 인스턴스로 전송됩니다.

고정 세션 관리에서 핵심은 로드 밸런서가 얼마나 오랫동안 동일한 인스턴스로 사용자 요청을 일관되게 라우팅하도록 하느냐입니다. 애플리케이션이 자체 세션 쿠키를 가지고 있는 경우에는 세션 쿠키가 애플리케이션의 세션 쿠키에 지정된 기간을 따르도록 Elastic Load Balancing을 구성할 수 있습니다. 애플리케이션이 자체 세션 쿠키를 가지고 있지 않은 경우에는 자체 고정 기간을 지정하여 세션 쿠키를 생성하도록 Elastic Load Balancing을 구성할 수 있습니다.

Elastic Load Balancing은 인스턴스로 세션을 매핑하는 데 사용되는 AWSELB라는 쿠키를 생성합니다.

요구 사항

- HTTP/HTTPS 로드 밸런서입니다.
- 각 가용 영역에 있는 하나 이상의 정상 상태 인스턴스입니다.

호환성

- 쿠키의 경로 속성에 대한 RFC는 밑줄을 허용합니다. 한편 Elastic Load Balancing URI는 밑줄 문자를 `%5F`로 인코딩합니다. Internet Explorer 7 같은 일부 브라우저들이 밑줄을 `%5F`로 인코딩된 URI로 예상하기 때문입니다. 현재 작동 중인 브라우저에 영향을 미칠 가능성이 있기 때문에 Elastic Load Balancing은 밑줄 문자를 계속 URI로 인코딩합니다. 예를 들어 쿠키가 `path=/my_path` 속성을 가지고 있으면 Elastic Load Balancing은 `path=/my%5Fpath`로 전송된 요청에서 이 속성을 변경합니다.

- 기간 기반의 고정 쿠키에서는 `secure` 플래그나 `HttpOnly` 플래그를 설정할 수 없습니다. 그러나 이들 쿠키는 민감한 데이터를 포함하고 있지 않습니다. 애플리케이션이 제어하는 세션 고정 쿠키에서 `secure` 플래그나 `HttpOnly` 플래그를 설정하면 AWSELB 쿠키에서도 그렇게 설정이 됩니다.
- 애플리케이션 쿠키의 `Set-Cookie` 필드에 후행 세미콜론이 있으면 로드 밸런서는 이 쿠키를 무시합니다.

목차

기간 기반 세션 고정

로드 밸런서는 각 리스너에 대한 요청에서 인스턴스를 추적하기 위해 특별한 쿠키인 AWSELB를 사용합니다. 로드 밸런서는 요청을 받으면 가장 먼저요청에 쿠키가 있는지 여부를 확인합니다. 쿠키가 있으면 해당 요청이 쿠키에 지정된 인스턴스에 전송됩니다. 쿠키가 없는 경우에는 로드 밸런서가 기존 로드 밸런싱 알고리즘을 기반으로 인스턴스를 선정합니다. 동일한 사용자의 후속 요청이 계속 해당 인스턴스에 바인딩되도록 쿠키가 응답에 삽입됩니다. 고정 정책 구성에서 각 쿠키의 유효 기간을 설정하는 쿠키 만료 시한을 정의합니다. 로드 밸런서는 쿠키 만료 시간을 새로 고침하지 않으며 쿠키가 사용 전에 만료되었는지 여부를 확인하지 않습니다. 쿠키가 만료된 후에는 세션이 더 이상 고정 상태가 아닙니다. 클라이언트는 쿠키가 만료되면 쿠키 저장소에서 쿠키를 제거해야 합니다.

CORS(Cross-Origin Resource Sharing) 요청의 경우 고정을 활성화하려면 `SameSite=None; Secure`가 필요합니다. 이 경우 Elastic Load Balancing은 두 번째 고정 쿠키인 AWSELBCORS를 생성합니다. 이 쿠키에는 원래 고정 쿠키와 동일한 정보와 `SameSite` 속성이 포함되어 있습니다. 클라이언트는 두 쿠키를 모두 수신합니다.

인스턴스가 실패하거나 비정상 상태가 되면 로드 밸런서는 해당 인스턴스로의 요청 라우팅을 중지하고 기존 로드 밸런싱 알고리즘을 기반으로 정상 상태의 인스턴스를 새로 선정합니다. 마치 쿠키가 없고 세션이 더 이상 고정 상태가 아닌 것처럼 새 인스턴스로 요청이 라우팅됩니다.

클라이언트가 다른 백엔드 포트를 가진 리스너로 전환되면 고정성이 손실됩니다.

콘솔을 사용하여 로드 밸런서에 대한 기간 기반 고정 세션을 활성화하려면

1. https://console.aws.amazon.com/ec2/에서 Amazon EC2 콘솔을 엽니다.
2. 탐색 창의 Load Balancing 아래에서 로드 밸런서를 선택합니다.
3. 로드 밸런서 이름을 선택하여 세부 정보 페이지를 엽니다.

4. 리스너 탭에서 리스너 관리를 선택합니다.
5. 리스너 관리 페이지에서 업데이트할 리스너를 찾고 쿠키 고정에서 편집을 선택합니다.
6. 쿠키 고정 설정 편집 팝업에서 로드 밸런서에서 생성됨을 선택합니다.
7. (선택 사항) 만료 기간에 쿠키 만료 기간(초)을 입력합니다. 만료 기간을 지정하지 않은 경우에는 브라우저 세션 기간 동안 고정 세션이 지속됩니다.
8. 변경 사항 저장을 선택하여 팝업 창을 닫습니다.
9. 변경 사항 저장을 선택하여 로드 밸런서 세부 정보 페이지로 돌아갑니다.

를 사용하여 로드 밸런서에 대한 기간 기반 고정 세션을 활성화하려면 AWS CLI

1. 쿠키 만료 기간을 60초로 설정해서 로드 밸런서가 쿠키 고정 정책을 생성하도록 하려면 아래 create-lb-cookie-stickiness-policy 명령을 사용하세요.

```
aws elb create-lb-cookie-stickiness-policy --load-balancer-name my-loadbalancer --
policy-name my-duration-cookie-policy --cookie-expiration-period 60
```

2. 지정된 로드 밸런서에서 세션 고정을 활성화하려면 아래 set-load-balancer-policies-of-listener 명령을 사용하세요.

```
aws elb set-load-balancer-policies-of-listener --load-balancer-name my-loadbalancer
 --load-balancer-port 443 --policy-names my-duration-cookie-policy
```

> **Note**
>
> `set-load-balancer-policies-of-listener` 명령은 지정된 로드 밸런서 포트에 연결된 현재의 정책 세트를 대체합니다. 이 명령을 사용할 때마다 활성화할 모든 정책의 목록을 표시하도록 `--policy-names` 옵션을 지정합니다.

3. (선택 사항) 정책이 활성화되었는지 확인하려면 아래 describe-load-balancers 명령을 사용하세요.

```
aws elb describe-load-balancers --load-balancer-name my-loadbalancer
```

응답에는 다음과 같이 지정된 포트의 리스너에 대해 정책이 활성화되었음을 보여주는 정보가 포함되어 있습니다.

```
{
    "LoadBalancerDescriptions": [
```

```
        {
            ...
            "ListenerDescriptions": [
                {
                    "Listener": {
                        "InstancePort": 443,
                        "SSLCertificateId": "arn:aws:iam::123456789012:server-
certificate/my-server-certificate",
                        "LoadBalancerPort": 443,
                        "Protocol": "HTTPS",
                        "InstanceProtocol": "HTTPS"
                    },
                    "PolicyNames": [
                        "my-duration-cookie-policy",
                        "ELBSecurityPolicy-TLS-1-2-2017-01"
                    ]
                },
                ...
            ],
            ...
            "Policies": {
                "LBCookieStickinessPolicies": [
                 {
                        "PolicyName": "my-duration-cookie-policy",
                        "CookieExpirationPeriod": 60
                    }

                ],
                "AppCookieStickinessPolicies": [],
                "OtherPolicies": [
                    "ELBSecurityPolicy-TLS-1-2-2017-01"
                ]
            },
            ...
        }
    ]
}
```

애플리케이션 제어 세션 고정

로드 밸런서는 특별한 쿠키를 이용하여 최초 요청을 처리한 인스턴스에 세션을 연결하지만, 정책 구성에 지정된 애플리케이션 쿠키의 사용 기간이 적용됩니다. 로드 밸런서는 애플리케이션 응답에 새 애플

리케이션 쿠키가 포함되어 있는 경우에만 새 고정 쿠키를 삽입합니다. 로드 밸런서 고정 쿠키는 요청이 있을 때마다 업데이트되지 않습니다. 애플리케이션 쿠키가 명백하게 제거되거나 만료되면 새 애플리케이션 쿠키가 발급될 때까지 세션에서 고정 상태가 중지됩니다.

백엔드 인스턴스에서 설정한 속성(`path`, `port`, `domain`, `secure`, `httponly`, `discard`, `max-age`, `expires`, `version`, `comment`, `commenturl` 및 `samesite`)이 쿠키의 클라이언트로 전송됩니다.

인스턴스가 실패하거나 비정상 상태가 되면 로드 밸런서는 해당 인스턴스로의 요청 라우팅을 중지하고 기존 로드 밸런싱 알고리즘을 기반으로 정상 상태의 인스턴스를 새로 선정합니다. 로드 밸런서는 새로운 정상 상태 인스턴스에 "고정"된 것으로 세션을 처리하고 실패한 인스턴스가 작동을 재개한 경우라도 해당 인스턴스로 계속해서 요청을 라우팅합니다.

콘솔을 사용하여 애플리케이션 제어 세션 고정을 활성화하려면

1. https://console.aws.amazon.com/ec2/에서 Amazon EC2 콘솔을 엽니다.
2. 탐색 창의 Load Balancing 아래에서 로드 밸런서를 선택합니다.
3. 로드 밸런서 이름을 선택하여 세부 정보 페이지를 엽니다.
4. 리스너 탭에서 리스너 관리를 선택합니다.
5. 리스너 관리 페이지에서 업데이트할 리스너를 찾고 쿠키 고정에서 편집을 선택합니다.
6. 애플리케이션에서 생성을 선택합니다.
7. [Cookie Name]에 애플리케이션 쿠키의 이름을 입력합니다.
8. 변경 사항 저장을 선택합니다.

를 사용하여 애플리케이션 제어 세션 고정을 활성화하려면 AWS CLI

1. 로드 밸런서가 쿠키 고정 정책을 생성하도록 하려면 아래 create-app-cookie-stickiness-policy 명령을 사용하세요.

```
aws elb create-app-cookie-stickiness-policy --load-balancer-name my-loadbalancer --
policy-name my-app-cookie-policy --cookie-name my-app-cookie
```

2. 로드 밸런서에서 세션 고정을 활성화하려면 아래 set-load-balancer-policies-of-listener 명령을 사용하세요.

```
aws elb set-load-balancer-policies-of-listener --load-balancer-name my-loadbalancer
 --load-balancer-port 443 --policy-names my-app-cookie-policy
```

> **Note**
>
> `set-load-balancer-policies-of-listener` 명령은 지정된 로드 밸런서 포트에 연결된 현재의 정책 세트를 대체합니다. 이 명령을 사용할 때마다 활성화할 모든 정책의 목록을 표시하도록 `--policy-names` 옵션을 지정합니다.

3. (선택 사항) 고정 정책이 활성화되었는지 확인하려면 아래 describe-load-balancers 명령을 사용하세요.

```
aws elb describe-load-balancers --load-balancer-name my-loadbalancer
```

4. 응답에는 다음과 같이 지정된 포트의 리스너에 대해 정책이 활성화되었음을 보여주는 정보가 포함되어 있습니다.

```
{
    "LoadBalancerDescriptions": [
        {
            ...
            "ListenerDescriptions": [
                {
                    "Listener": {
                        "InstancePort": 443,
                        "SSLCertificateId": "arn:aws:iam::123456789012:server-
certificate/my-server-certificate",
                        "LoadBalancerPort": 443,
                        "Protocol": "HTTPS",
                        "InstanceProtocol": "HTTPS"
                    },
                    "PolicyNames": [
                        "my-app-cookie-policy",
                        "ELBSecurityPolicy-TLS-1-2-2017-01"
                    ]
                },
                {
                    "Listener": {
                        "InstancePort": 80,
                        "LoadBalancerPort": 80,
                        "Protocol": "TCP",
                        "InstanceProtocol": "TCP"
                    },
                    "PolicyNames": []
```

```
                }
            ],
            ...
            "Policies": {
                "LBCookieStickinessPolicies": [],
                "AppCookieStickinessPolicies": [
                {
                        "PolicyName": "my-app-cookie-policy",
                        "CookieName": "my-app-cookie"
                    }

                ],
                "OtherPolicies": [
                    "ELBSecurityPolicy-TLS-1-2-2017-01"
                ]
            },
            ...
        }
    ]
}
```

Classic Load Balancer에 대한 동기화 해제 완화 모드 구성

Desync Mitigation Mode는 HTTP Desync로 인한 문제로부터 애플리케이션을 보호합니다. 로드 밸런서는 위협 수준에 따라 각 요청을 분류하고 안전한 요청을 허용한 다음 지정한 완화 모드에서 지정한 대로 위험을 완화합니다. Desync Mitigation Mode는 Monitor, Defensive 또는 Strictest 모드입니다. 기본값은 Defensive 모드입니다. 이 모드는 애플리케이션의 가용성을 유지하면서 HTTP Desync에 대한 지속적인 완화를 제공합니다. 애플리케이션에서 RFC 7230을 준수하는 요청만 수신하도록 Strictest 모드로 전환할 수 있습니다.

http_desync_guardian 라이브러리는 HTTP Desync 공격을 방지하기 위해 HTTP 요청을 분석합니다. 자세한 내용은 github의 HTTP Desync Guardian을 참조하십시오.

목차

> **Tip**
>
> 이 구성은 Classic Load Balancer에만 적용됩니다. Application Load Balancer에 적용되는 자세한 내용은 Application Load Balancers에 대한 동기화 해제 완화 모드를 참조하세요.

분류

분류는 다음과 같습니다.

- 규정 준수 - 요청이 RFC 7230을 준수하며 알려진 보안 위협이 없습니다.
- 허용 가능 - 요청이 RFC 7230을 준수하지 않지만 알려진 보안 위협이 없습니다.
- 모호 - 요청이 RFC 7230을 준수하지 않지만 다양한 웹 서버와 프록시가 다르게 처리할 수 있으므로 위험을 초래합니다.
- 심각 - 요청이 높은 보안 위험을 초래합니다. 로드 밸런서는 요청을 차단하고 클라이언트에 대해 400 응답을 제공하고 클라이언트 연결을 종료합니다.

다음 목록에서는 각 분류에 대한 문제를 설명합니다.

허용 가능

- 헤더에 비 ASCII 또는 제어 문자가 포함되어 있습니다.
- 요청 버전에 잘못된 값이 포함되어 있습니다.
- GET 또는 HEAD 요청에 대한 값이 0인 Content-Length 헤더가 있습니다.
- 요청 URI에 URL이 인코딩되지 않은 공백이 포함되어 있습니다.

모호

- 요청 URI에 제어 문자가 포함되어 있습니다.
- 요청에 Transfer-Encoding 헤더와 Content-Length 헤더가 모두 포함되어 있습니다.
- 값이 동일한 Content-Length 헤더가 여러 개 있습니다.
- 헤더가 비어 있거나 공백만 있는 줄이 있습니다.
- 일반적인 텍스트 정규화 기술을 사용하여 Transfer-Encoding 또는 Content-Length로 정규화할 수 있는 헤더가 있습니다.
- GET 또는 HEAD 요청에 대한 Content-Length 헤더가 있습니다.

- GET 또는 HEAD 요청에 대한 Transfer-Encoding 헤더가 있습니다.

심각

- 요청 URI에 null 문자 또는 캐리지 리턴이 포함되어 있습니다.
- Content-Length 헤더에 구문 분석할 수 없거나 유효한 숫자가 아닌 값이 포함되어 있습니다.
- 헤더에 null 문자 또는 캐리지 리턴이 포함되어 있습니다.
- Transfer-Encoding 헤더에 잘못된 값이 포함되어 있습니다.
- 요청 메서드의 형식이 잘못되었습니다.
- 요청 버전의 형식이 잘못되었습니다.
- 값이 서로 다른 Content-Length 헤더가 여러 개 있습니다.
- 여러 Transfer-Encoding이 있습니다: chunked 헤더.

요청이 RFC 7230을 준수하지 않는 경우 로드 밸런서는 `DesyncMitigationMode_NonCompliant_Request_Count` 지표를 증가시킵니다. 자세한 내용은 Classic Load Balancer 지표 단원을 참조하십시오.

Modes

다음 표에서는 Classic Load Balancer가 모드 및 분류를 기준으로 요청을 처리하는 방법에 대해 설명합니다.

Classification	Monitor 모드	Defensive 모드	Strictest 모드
규정 준수	Allowed	허용됨	Allowed
허용 가능	Allowed	Allowed	차단됨
모호	Allowed	허용[1]	차단됨
심각	Allowed	차단됨	차단됨

[1] 요청을 라우팅하지만 클라이언트 연결과 대상 연결을 종료합니다.

Desync Mitigation Mode 수정

콘솔을 사용하여 Desync Mitigation Mode를 업데이트하려면

1. https://console.aws.amazon.com/ec2/에서 Amazon EC2 콘솔을 엽니다.
2. 탐색 창의 Load Balancing 아래에서 로드 밸런서를 선택합니다.
3. 로드 밸런서 이름을 선택하여 세부 정보 페이지를 엽니다.
4. 속성 탭에서 편집을 선택합니다.
5. 로드 밸런서 속성 편집 페이지의 트래픽 구성에서 방어적 - 권장,가장 엄격함 또는 모니터링을 선택합니다.
6. 변경 사항 저장을 선택합니다.

를 사용하여 비동기화 완화 모드를 업데이트하려면 AWS CLI

`elb.http.desyncmitigationmode` 속성을 `monitor`, `defensive` 또는 `strictest`로 설정하여 modify-load-balancer-attributes 명령을 사용하세요.

```
aws elb modify-load-balancer-attributes --load-balancer-name my-load-balancer --load-
balancer-attributes file://attribute.json
```

다음은 `attribute.json`의 내용입니다.

```
{
    "AdditionalAttributes": [
        {
            "Key": "elb.http.desyncmitigationmode",
            "Value": "strictest"
        }
    ]
}
```

Classic Load Balancer에 프록시 프로토콜 구성

프록시 프로토콜은 연결을 요청하는 소스에서 연결이 요청된 대상으로 연결 정보를 전달하는 데 사용되는 인터넷 프로토콜입니다. Elastic Load Balancing은 사람이 읽을 수 있는 형식의 헤더를 가진 프록시 프로토콜 버전 1을 사용합니다.

기본적으로 프런트 엔드 및 백 엔드 연결 모두에서 TCP(Transmission Control Protocol)를 사용하면 Classic Load Balancer는 요청 헤더를 변경하지 않고 인스턴스로 요청을 전달합니다. 프록시 프로토콜을 활성화하면 사람이 읽을 수 있는 형식의 헤더가 소스 IP 주소, 대상 IP 주소, 포트 번호와 같은 연결 정보를 포함하는 요청 헤더에 추가됩니다. 이렇게 하면 헤더가 요청의 일부로 인스턴스에 전송됩니다.

> **Note**
>
> AWS Management Console 는 프록시 프로토콜 활성화를 지원하지 않습니다.

내용

프록시 프로토콜 헤더

프록시 프로토콜 헤더는 백엔드 연결에 TCP를 사용하는 로드 밸런서를 가지고 있을 때 클라이언트의 IP 주소를 식별하는 데 도움을 줍니다. 로드 밸런서가 클라이언트와 인스턴스 간의 트래픽을 가로채기 때문에 인스턴스의 액세스 로그에 발원 클라이언트가 아닌 로드 밸런서의 IP 주소가 포함됩니다. 요청의 첫 번째 줄을 분석해서 클라이언트의 IP 주소와 포트 번호를 알아낼 수 있습니다.

IPv6를 위한 헤더의 프록시 주소가 로드 밸런서의 퍼블릭 IPv6 주소가 됩니다. 이 IPv6 주소는 로드 밸런서의 DNS 이름에서 확인할 수 있는 IP 주소와 일치하며 `ipv6` 또는 `dualstack`으로 시작됩니다. 클라이언트가 IPv4에 연결되면 헤더의 프록시 주소는 로드 밸런서의 프라이빗 IPv4 주소가 되고, DNS 조회를 통해 확인할 수 없습니다.

프록시 프로토콜 줄은 캐리지 리턴과 라인 피드("\r\n")로 끝나는 단일 줄이며 다음의 형식을 가지고 있습니다.

```
PROXY_STRING + single space + INET_PROTOCOL + single space + CLIENT_IP + single space +
 PROXY_IP + single space + CLIENT_PORT + single space + PROXY_PORT + "\r\n"
```

예제: IPv4

다음은 IPv4를 위한 프록시 프로토콜 줄의 예제입니다.

```
PROXY TCP4 198.51.100.22 203.0.113.7 35646 80\r\n
```

프록시 프로토콜을 활성화하기 위한 사전 조건

시작하기 전에 다음을 수행하십시오.

- 로드 밸런서가 프록시 프로토콜이 활성화된 프록시 서버를 지원하지 않는지 확인합니다. 프록시 서버와 로드 밸런서 모두에서 프록시 프로토콜이 활성화된 경우에는 로드 밸런서가 프록시 서버에서 이미 헤더를 가지고 있는 요청에 또 다른 헤더를 추가합니다. 인스턴스의 구성 방법에 따라 이러한 중복이 오류를 일으킬 수 있습니다.
- 인스턴스가 프록시 프로토콜 정보를 처리할 수 있는지 확인합니다.
- 리스너 설정이 프록시 프로토콜을 지원하는지 확인합니다. 자세한 내용은 Classic Load Balancer의 리스너 구성 단원을 참조하십시오.

를 사용하여 프록시 프로토콜 활성화 AWS CLI

프록시 프로토콜을 활성화하려면 `ProxyProtocolPolicyType` 유형의 정책을 생성한 다음 인스턴스 포트에서 정책을 활성화해야 합니다.

다음 단계를 통해 `ProxyProtocolPolicyType` 유형의 로드 밸런서를 위한 새 정책을 생성하고 새롭게 생성된 정책을 포트 `80`의 인스턴스로 설정한 다음, 정책이 활성화되었는지 확인합니다.

로드 밸런서에 대한 프록시 프로토콜을 활성화하려면

1. (선택 사항) Elastic Load Balancing이 지원하는 정책의 목록을 확인하려면 아래 describe-load-balancer-policy-types 명령을 사용하세요.

   ```
   aws elb describe-load-balancer-policy-types
   ```

   응답에는 지원되는 정책 유형의 이름과 이에 대한 설명이 포함되어 있습니다. 다음은 `ProxyProtocolPolicyType` 유형에 대한 출력을 보여줍니다.

   ```
   {
       "PolicyTypeDescriptions": [
           ...
           {
               "PolicyAttributeTypeDescriptions": [
                   {
   ```

```
                    "Cardinality": "ONE",
                    "AttributeName": "ProxyProtocol",
                    "AttributeType": "Boolean"
                }
            ],
            "PolicyTypeName": "ProxyProtocolPolicyType",
            "Description": "Policy that controls whether to include the IP address
 and port of the originating
request for TCP messages. This policy operates on TCP/SSL listeners only"
        },
        ...
    ]
}
```

2. 프록시 프로토콜을 활성화하는 정책을 생성하려면 아래 create-load-balancer-policy 명령을 사용하세요.

```
aws elb create-load-balancer-policy --load-balancer-name my-loadbalancer --policy-
name my-ProxyProtocol-policy --policy-type-name ProxyProtocolPolicyType --policy-
attributes AttributeName=ProxyProtocol,AttributeValue=true
```

3. 새로 생성된 정책을 지정된 포트에서 활성화하려면 아래 set-load-balancer-policies-for-backend-server 명령을 사용하세요. 이 명령이 현재 활성화된 정책 세트를 대체합니다. 따라서 `--policy-names` 옵션을 통해 목록에 추가 중인 정책(예: `my-ProxyProtocol-policy`)과 현재 활성화된 정책(예: `my-existing-policy`) 모두를 지정해야 합니다.

```
aws elb set-load-balancer-policies-for-backend-server --load-balancer-name my-
loadbalancer --instance-port 80 --policy-names my-ProxyProtocol-policy my-existing-
policy
```

4. (선택 사항) 프록시 프로토콜이 활성화되어 있는지 확인하려면 아래 describe-load-balancers 명령을 사용하세요.

```
aws elb describe-load-balancers --load-balancer-name my-loadbalancer
```

응답에는 다음과 같이 `my-ProxyProtocol-policy` 정책이 포트 `80`에 연결되었음을 보여주는 정보가 포함되어 있습니다.

```
{
    "LoadBalancerDescriptions": [
        {
```

```
            ...
            "BackendServerDescriptions": [
                {
                    "InstancePort": 80,
                    "PolicyNames": [
                        "my-ProxyProtocol-policy"
                    ]
                }
            ],
            ...
        }
    ]
}
```

를 사용하여 프록시 프로토콜 비활성화 AWS CLI

인스턴스에 연결된 정책을 비활성화한 다음, 나중에 활성화를 할 수 있습니다.

프록시 프로토콜 정책을 비활성화하려면

1. `--policy-names` 옵션에서 이를 생략하고 활성화된 상태를 유지하는 다른 정책(예: `my-existing-policy`)을 포함시켜 프록시 프로토콜 정책을 비활성화하려면 아래 set-load-balancer-policies-for-backend-server 명령을 사용하세요.

   ```
   aws elb set-load-balancer-policies-for-backend-server --load-balancer-name my-loadbalancer --instance-port 80 --policy-names my-existing-policy
   ```

   활성화할 다른 정책이 없는 경우에는 다음과 같이 `--policy-names` 옵션을 통해 빈 문자열을 지정합니다.

   ```
   aws elb set-load-balancer-policies-for-backend-server --load-balancer-name my-loadbalancer --instance-port 80 --policy-names "[]"
   ```

2. (선택 사항) 정책이 비활성화되었는지 확인하려면 아래 describe-load-balancers 명령을 사용하세요.

   ```
   aws elb describe-load-balancers --load-balancer-name my-loadbalancer
   ```

   응답에는 다음과 같이 정책에 연결된 포트가 없음을 보여주는 정보가 포함되어 있습니다.

```
{
    "LoadBalancerDescriptions": [
        {
            ...
            "BackendServerDescriptions": [],
            ...
        }
    ]
}
```

Classic Load Balancer에 대한 태그 지정

태그는 용도, 소유자, 환경 등 다양한 방식으로 로드 밸런서를 분류할 수 있도록 해줍니다.

각 Classic Load Balancer에 여러 태그를 추가할 수 있습니다. 태그 키는 각 로드 밸런서에 대해 고유해야 합니다. 로드 밸런서에 이미 연결된 키를 통해 태그를 추가하면 해당 태그의 값이 업데이트됩니다.

태그 사용을 마치면 로드 밸런서에서 이를 제거할 수 있습니다.

목차

태그 제한

태그에 적용되는 기본 제한은 다음과 같습니다.

- 리소스당 최대 태그 수 - 50개
- 최대 키 길이 - 유니코드 문자 127자
- 최대 값 길이 - 유니코드 문자 255자
- 태그 키와 값은 대소문자를 구분합니다. 허용되는 문자는 UTF-8로 표현할 수 있는 문자, 공백 및 숫자와 특수 문자 + - = . _ : / @입니다. 선행 또는 후행 공백을 사용하면 안 됩니다.
- 태그 이름 또는 값에 `aws:` 접두사는 AWS 사용하도록 예약되어 있으므로 사용하지 마십시오. 이 접두사가 지정된 태그 이름이나 값은 편집하거나 삭제할 수 없습니다. 이 접두사가 지정된 태그는 리소스당 태그 수 제한에 포함되지 않습니다.

태그 추가

언제라도 로드 밸런서에 태그를 추가할 수 있습니다.

콘솔을 사용하여 태그를 추가하려면

1. https://console.aws.amazon.com/ec2/에서 Amazon EC2 콘솔을 엽니다.
2. 탐색 창의 Load Balancing 아래에서 로드 밸런서를 선택합니다.
3. 로드 밸런서 이름을 선택하여 세부 정보 페이지를 엽니다.
4. 태그 탭에서 태그 관리를 선택합니다.
5. 태그 관리 페이지에서 각 태그에 대해 새 태그 생성을 선택한 다음 키와 값을 지정합니다.
6. 태그 추가를 완료하면 변경 사항 저장을 선택합니다.

를 사용하여 태그를 추가하려면 AWS CLI

지정된 태그를 추가하려면 아래 add-tags 명령을 사용하세요.

```
aws elb add-tags --load-balancer-name my-loadbalancer --tag "Key=project,Value=lima"
```

태그 제거

태그 사용을 마칠 때마다 로드 밸런서에서 태그를 제거할 수 있습니다.

콘솔을 사용하여 태그를 제거하려면

1. https://console.aws.amazon.com/ec2/에서 Amazon EC2 콘솔을 엽니다.
2. 탐색 창의 Load Balancing 아래에서 로드 밸런서를 선택합니다.
3. 로드 밸런서 이름을 선택하여 세부 정보 페이지를 엽니다.
4. 태그 탭에서 태그 관리를 선택합니다.
5. 태그 관리 페이지에서 제거할 각 태그 옆의 제거를 선택합니다.
6. 태그 제거를 완료하면 변경 사항 저장을 선택합니다.

를 사용하여 태그를 제거하려면 AWS CLI

지정된 키를 통해 태그를 제거하려면 아래 remove-tags 명령을 사용하세요.

```
aws elb remove-tags --load-balancer-name my-loadbalancer --tag project
```

Classic Load Balancer에서 서브넷 구성

로드 밸런서에 서브넷을 추가하면 Elastic Load Balancing이 해당 가용 영역에서 로드 밸런서 노드를 생성합니다. 로드 밸런서 노드는 클라이언트에서 오는 트래픽을 허용하고, 하나 이상의 가용 영역에서 정상 상태의 등록 인스턴스로 요청을 전달합니다. 최소 두 개의 가용 영역에 대해 가용 영역당 하나의 서브넷을 추가하는 것이 좋습니다. 이렇게 하면 로드 밸런서의 가용성이 향상됩니다. 로드 밸런서에 대한 서브넷을 언제든지 수정할 수 있습니다.

인스턴스와 동일한 가용 영역에서 서브넷을 선택합니다. 로드 밸런서가 인터넷 경계 로드 밸런서인 경우에는 퍼블릭 서브넷을 선택해야만 백엔드 인스턴스가 로드 밸런서에서 오는 트래픽을 수신할 수 있습니다(백엔드 인스턴스가 프라이빗 서브넷에 있다 하더라도). 로드 밸런서가 내부 로드 밸런서인 경우에는 프라이빗 서브넷을 선택하는 것이 좋습니다. 로드 밸런서용 서브넷에 대한 자세한 내용은 VPC 관련 권장 사항를 참조하십시오.

서브넷을 추가하려면 가용 영역의 인스턴스를 로드 밸런서에 등록한 다음, 해당 가용 영역의 서브넷을 로드 밸런서에 연결합니다. 자세한 내용은 Classic Load Balancer에 인스턴스 등록 단원을 참조하십시오.

서브넷을 추가하고 나면 로드 밸런서가 해당 가용 영역의 등록 인스턴스로 요청을 라우팅하기 시작합니다. 기본적으로 로드 밸런서는 서브넷에서 가용 영역 전반에 균등하게 요청을 라우팅합니다. 서브넷에서 가용 영역의 등록 인스턴스에 균등하게 요청을 라우팅할 수 있도록 교차 영역 로드 밸런싱을 활성화합니다. 자세한 내용은 Classic Load Balancer에서 교차 영역 로드 밸런싱을 구성 단원을 참조하십시오.

정상 상태의 등록 인스턴스가 없거나 등록 인스턴스에서 문제 해결 또는 업데이트를 원할 경우에 일시적으로 로드 밸런서에서 서브넷을 제거하고 싶을 수 있습니다. 서브넷을 제거하고 나면 로드 밸런서는 이 가용 영역의 등록 인스턴스로의 요청 라우팅은 중지하되, 나머지 서브넷에서 가용 영역의 등록 인스턴스로 계속해서 요청을 라우팅합니다. 서브넷을 제거해도 해당 서브넷의 인스턴스들은 로드 밸런서에 등록된 상태로 남아 있다는 점에 유의하세요. 원할 경우 인스턴스의 등록을 취소할 수 있습니다. 자세한 내용은 Classic Load Balancer에 인스턴스 등록 단원을 참조하십시오.

내용

요구 사항

로드 밸런서에 대한 서브넷을 업데이트할 때는 다음 요구 사항을 반드시 충족해야 합니다.

- 로드 밸런서가 언제나 서브넷을 하나 이상 가지고 있어야 합니다.
- 가용 영역당 서브넷을 한 개 이상 추가할 수 있습니다.
- 로컬 영역 서브넷은 추가할 수 없습니다.

로드 밸런서에서 서브넷을 추가 및 제거하기 위한 별도의 API가 있기 때문에 이러한 요구 사항을 충족하기 위해 현재 서브넷을 새로운 서브넷으로 바꿀 때는 작업 순서를 세심하게 고려해야 합니다. 또한 로드 밸런서에 대한 모든 서브넷을 바꿔야 하는 경우에는 또 다른 가용 영역에서 나온 서브넷을 일시적으로 추가해야 합니다. 예를 들어 로드 밸런서가 단일 가용 영역을 가지고 있고 현재 서브넷을 또 다른 서브넷으로 바꿔야 하는 경우에는 두 번째 가용 영역에서 나온 서브넷을 먼저 추가해야 합니다. 그런 다음 원래 가용 영역에서 서브넷을 제거하고(서브넷을 한 개 이하로 변경하지 않음), 원래 가용 영역에서 새로운 서브넷을 추가한 다음(가용 영역당 하나 이상으로 서브넷 수를 초과하지 않음), 두 번째 가용 영역에서 서브넷을 제거할 수 있습니다(오직 교체가 필요한 경우).

콘솔을 사용하여 서브넷 구성

다음 절차에 따라 콘솔을 사용하여 서브넷을 추가하거나 제거합니다.

콘솔을 사용하여 서브넷을 구성하려면

1. https://console.aws.amazon.com/ec2/에서 Amazon EC2 콘솔을 엽니다.
2. 탐색 창의 Load Balancing 아래에서 로드 밸런서를 선택합니다.
3. 로드 밸런서 이름을 선택하여 세부 정보 페이지를 엽니다.
4. 네트워크 매핑 탭에서 서브넷 편집을 선택합니다.
5. 서브넷 편집 페이지의 네트워크 매핑 섹션에서 필요에 따라 서브넷을 추가하거나 제거합니다.
6. 작업을 마쳤으면 변경 사항 저장을 선택합니다.

CLI를 사용하여 서브넷 구성

다음 예제를 따라 AWS CLI를 사용하여 서브넷을 추가하거나 제거합니다.

CLI를 사용하여 로드 밸런서에 서브넷을 추가하려면

로드 밸런서에 두 개의 서브넷을 추가하려면 아래 attach-load-balancer-to-subnets 명령을 사용하세요.

```
aws elb attach-load-balancer-to-subnets --load-balancer-name my-load-balancer --
subnets subnet-dea770a9 subnet-fb14f6a2
```

응답에는 해당 로드 밸런서에 대한 모든 서브넷이 나열됩니다. 예제:

```
{
    "Subnets": [
        "subnet-5c11033e",
        "subnet-dea770a9",
        "subnet-fb14f6a2"
    ]
}
```

를 사용하여 서브넷을 제거하려면 AWS CLI

지정된 로드 밸런서에서 지정된 서브넷을 제거하려면 아래 detach-load-balancer-from-subnets 명령을 사용하세요.

```
aws elb detach-load-balancer-from-subnets --load-balancer-name my-loadbalancer --
subnets subnet-450f5127
```

응답에는 해당 로드 밸런서에 대한 나머지 서브넷들이 나열됩니다. 예제:

```
{
    "Subnets": [
        "subnet-15aaab61"
    ]
}
```

Classic Load Balancer 보안 그룹 구성

AWS Management Console 를 사용하여 로드 밸런서를 생성할 때 기존 보안 그룹을 선택하거나 새 보안 그룹을 생성할 수 있습니다. 기존 보안 그룹을 선택하는 경우에는 반드시 로드 밸런서를 위한 리스너 포트 및 상태 확인 포트로의 양방향 트래픽을 허용해야 합니다. 보안 그룹을 생성하기로 한 경우에는 콘솔이 이들 포트에서 모든 트래픽을 허용하는 규칙을 자동으로 추가합니다.

[기본이 아닌 VPC] AWS CLI 또는 API를 사용하여 기본이 아닌 VPC에서 로드 밸런서를 생성하지만 보안 그룹을 지정하지 않으면 로드 밸런서가 VPC의 기본 보안 그룹과 자동으로 연결됩니다.

[기본 VPC] AWS CLI 또는 API를 사용하여 기본 VPC에서 로드 밸런서를 생성하는 경우 로드 밸런서에 대한 기존 보안 그룹을 선택할 수 없습니다. 대신에 Elastic Load Balancing이 로드 밸런서를 위해 지정된 포트에서 모든 트래픽을 허용하는 규칙을 보안 그룹에 제공합니다. Elastic Load Balancing은 default_elb_*id* 형식의 이름으로 AWS 계정당 이러한 보안 그룹을 하나만 생성합니다(예: `default_elb_fc5fbed3-0405-3b7d-a328-ea290EXAMPLE`). 이후에 기본 VPC에서 생성되는 로드 밸런서는 이 보안 그룹도 사용합니다. 새로운 로드 밸런서의 리스너 포트와 상태 확인 포트에서 트래픽을 허용하는지 보안 그룹 규칙을 확인해야 합니다. 로드 밸런서를 삭제해도 이 보안 그룹은 자동 삭제되지 않습니다.

기존 로드 밸런서에 리스너를 추가하는 경우에는 보안 그룹이 새로운 리스너 포트에서 양방향 트래픽을 허용하는지 확인해야 합니다.

목차

로드 밸런서 보안 그룹을 위한 권장 규칙

로드 밸런서를 위한 보안 그룹은 인스턴스와의 통신을 허용해야 합니다. 권장 규칙은 로드 밸런서의 유형(인터넷 경계 또는 내부)에 따라 다릅니다.

인터넷 경계 로드 밸런서

다음 표에는 인터넷 경계 로드 밸런서를 위한 권장 인바운드 규칙이 나와 있습니다.

소스	프로토콜	포트 범위	Comment
0.0.0.0/0	TCP	*###*	로드 밸런서 리스너 포트에서 모든 인바운드 트래픽을 허용

다음 표에는 인터넷 경계 로드 밸런서를 위한 권장 아웃바운드 규칙이 나와 있습니다.

Destination	프로토콜	포트 범위	Comment
#### ## ##	TCP	*#### ###*	인스턴스 리스너 포트의 인스턴스로 아웃바운드 트래픽을 허용합니다

Destination	프로토콜	포트 범위	Comment
#### ## ##	TCP	*## ##*	상태 확인 포트의 인스턴스로 아웃바운드 트래픽을 허용합니다

내부 로드 밸런서

다음 표에는 내부 로드 밸런서를 위한 권장 인바운드 규칙이 나와 있습니다.

소스	프로토콜	포트 범위	Comment
VPC CIDR	TCP	*###*	로드 밸런서 리스너 포트에서 VPC CIDR에서 오는 인바운드 트래픽을 허용

다음 표에는 내부 로드 밸런서를 위한 권장 아웃바운드 규칙이 나와 있습니다.

Destination	프로토콜	포트 범위	Comment
#### ## ##	TCP	*#### ###*	인스턴스 리스너 포트의 인스턴스로 아웃바운드 트래픽을 허용합니다
#### ## ##	TCP	*## ##*	상태 확인 포트의 인스턴스로 아웃바운드 트래픽을 허용합니다

콘솔을 사용하여 보안 그룹 할당

다음 절차를 사용하여 로드 밸런서에 연결된 보안 그룹을 변경합니다.

콘솔을 사용하여 로드 밸런서에 할당된 보안 그룹을 업데이트하려면

1. https://console.aws.amazon.com/ec2/에서 Amazon EC2 콘솔을 엽니다.
2. 탐색 창의 Load Balancing 아래에서 로드 밸런서를 선택합니다.
3. 로드 밸런서 이름을 선택하여 세부 정보 페이지를 엽니다.
4. 보안 탭에서 편집을 선택합니다.

5. 보안 그룹 편집 페이지의 보안 그룹에서 필요에 따라 보안 그룹을 추가 또는 제거합니다.

   최대 5개의 보안 그룹을 추가할 수 있습니다.

6. 작업을 마쳤으면 변경 사항 저장을 선택합니다.

를 사용하여 보안 그룹 할당 AWS CLI

다음 apply-security-groups-to-load-balancer 명령을 사용하여 보안 그룹과 로드 밸런서를 연결합니다. 지정된 보안 그룹은 이전에 연결된 보안 그룹을 재정의합니다.

```
aws elb apply-security-groups-to-load-balancer --load-balancer-name my-loadbalancer --
security-groups sg-53fae93f
```

다음은 응답의 예입니다.

```
{
  "SecurityGroups": [
     "sg-53fae93f"
  ]
}
```

Classic Load Balancer에서 네트워크 ACL 구성

VPC의 기본 네트워크 액세스 제어 목록(ACL)은 인바운드 트래픽과 아웃바운드 트래픽을 모두 허용합니다. 사용자 지정 네트워크 ACL을 생성하는 경우에는 로드 밸런서 및 인스턴스의 통신을 허용하는 규칙을 추가해야 합니다.

로드 밸런서 서브넷을 위한 권장 규칙은 로드 밸런서의 유형(인터넷 경계 또는 내부)에 따라 다릅니다.

인터넷 경계 로드 밸런서

다음은 인터넷 경계 로드 밸런서를 위한 권장 인바운드 규칙입니다.

소스	프로토콜	포트 범위	Comment
0.0.0.0/0	TCP	*###	로드 밸런서 리스너 포트에서 모든 인바운드 트래픽을 허용

소스	프로토콜	포트 범위	Comment
VPC CIDR	TCP	1024-65535	휘발성 포트에서 VPC CIDR에서 오는 인바운드 트래픽을 허용

다음은 인터넷 경계 로드 밸런서를 위한 권장 아웃바운드 규칙입니다.

Destination	프로토콜	포트 범위	Comment
VPC CIDR	TCP	*#### ###*	인스턴스 리스너 포트에서 모든 아웃바운드 트래픽을 허용
VPC CIDR	TCP	*## ##*	상태 확인 포트에서 모든 아웃바운드 트래픽을 허용
0.0.0.0/0	TCP	1024-65535	휘발성 포트에서 모든 아웃바운드 트래픽을 허용

내부 로드 밸런서

다음은 내부 로드 밸런서를 위한 권장 인바운드 규칙입니다.

소스	프로토콜	포트 범위	Comment
VPC CIDR	TCP	*###*	로드 밸런서 리스너 포트에서 VPC CIDR에서 오는 인바운드 트래픽을 허용
VPC CIDR	TCP	1024-65535	휘발성 포트에서 VPC CIDR에서 오는 인바운드 트래픽을 허용

다음은 내부 로드 밸런서를 위한 권장 아웃바운드 규칙입니다.

Destination	프로토콜	포트 범위	Comment
VPC CIDR	TCP	*#### ###*	인스턴스 리스너 포트에서 VPC CIDR로 가는 아웃바운드 트래픽을 허용
VPC CIDR	TCP	*## ##*	상태 확인 포트에서 VPC CIDR로 가는 아웃바운드 트래픽을 허용
VPC CIDR	TCP	1024-65535	휘발성 포트에서 VPC CIDR로 가는 아웃바운드 트래픽을 허용

Classic Load Balancer에 대한 사용자 지정 도메인 이름을 구성합니다.

각 Classic Load Balancer는 기본 DNS(도메인 이름 시스템) 이름을 수신합니다. 이 DNS 이름에는 로드 밸런서가 생성되는 AWS 리전의 이름이 포함됩니다. 예를 들어 미국 서부(오레곤) 리전에서 `my-loadbalancer`라는 이름의 로드 밸런서를 생성하면 로드 밸런서가 `my-loadbalancer-1234567890.us-west-2.elb.amazonaws.com` 같은 DNS 이름을 수신합니다. 인스턴스에서 웹 사이트를 액세스하려면 웹 브라우저의 주소 필드에 DNS 이름을 붙여넣습니다. 그러나 이 DNS 이름은 고객들이 쉽게 기억하고 사용할 수 없습니다.

로드 밸런서에서 기본 DNS 이름 대신 `www.example.com` 같이 친숙한 DNS 이름을 사용하려면 사용자 지정 도메인 이름을 생성하고 이를 로드 밸런서의 DNS 이름에 연결할 수 있습니다. 클라이언트가 이러한 사용자 지정 도메인 이름을 사용해 요청을 하면 DNS 서버는 이를 로드 밸런서의 DNS 이름으로 해석합니다.

목차

사용자 지정 도메인 이름을 로드 밸런서 이름에 연결

먼저, 아직 등록을 하지 않았다면 도메인 이름부터 등록합니다. 국제인터넷주소관리기구(ICANN)가 인터넷 도메인 이름을 관리합니다. 도메인 이름 등록을 관리하는 ICANN 인증 조직인 도메인 이름 등록

대행자를 이용해 도메인 이름을 등록합니다. 등록 대행자 웹 사이트는 도메인 이름 등록에 대한 자세한 지침과 요금 정보를 제공합니다. 자세한 내용은 다음 자료를 참조하세요.

- Amazon Route 53을 사용하여 도메인 이름을 등록하려면 Amazon Route 53 개발자 안내서의 Route 53을 사용하여 도메인 이름 등록을 참조하세요.
- 인증된 등록 대행자의 목록은 List of Accredited Registrar를 참조하세요.

다음으로 CNAME 레코드를 생성하여 쿼리를 로드 밸런서로 라우팅하려면 도메인 등록 대행자와 같은 DNS 서비스를 사용하면 됩니다. 자세한 내용은 DNS 서비스에 대한 설명서를 참조하십시오.

또는 DNS 서비스로 Route 53을 사용할 수도 있습니다. 도메인을 위해 인터넷에서 트래픽을 라우팅하는 방법이 포함된 호스팅 영역과 로드 밸런서로 도메인 이름에 대한 쿼리를 라우팅하는 별칭 리소스 레코드 세트를 생성합니다. Route 53은 별칭 레코드 세트에서 DNS 쿼리에 요금을 부과하지 않기 때문에 도메인의 zone apex(예: `example.com`)에서 별칭 리소스 세트를 사용하여 로드 밸런서로 DNS 쿼리를 라우팅할 수 있습니다. 기존 도메인의 DNS 서비스를 Route 53로 전송하는 방법에 대한 자세한 내용은 Amazon Route 53 개발자 안내서의 DNS 서비스로 Route 53을 구성 섹션을 참조하세요.

Route 53을 사용하여 도메인에 대한 호스팅 영역과 별칭 레코드를 생성하려면 자세한 내용은 Amazon Route 53 개발자 안내서의 로드 밸런서로 트래픽 라우팅을 참조하세요.

로드 밸런서에 대한 Route 53 DNS 장애 조치 사용

Route 53을 사용하여 로드 밸런서에 DNS 요청을 라우팅하는 경우, Route 53을 사용하여 로드 밸런서에 대한 DNS 장애 조치를 구성할 수도 있습니다. 장애 조치 구성에서 Route 53은 로드 밸런서에서 등록 EC2 인스턴스의 상태를 확인하여 가용 여부를 결정합니다. 로드 밸런서에 정상 상태의 EC2 인스턴스가 등록되어 있지 않거나 로드 밸런서 자체가 정상 상태가 아니면 Route 53은 정상 상태 로드 밸런서나 Amazon S3의 정적 웹 사이트 같은 또 다른 가용 리소스로 트래픽을 라우팅합니다.

예를 들어 `www.example.com`에 대한 웹 사이트가 있고 서로 다른 리전에 상주하는 두 개의 로드 밸런서에서 중복 인스턴스를 실행하고 싶다고 가정합시다. 한 리전의 로드 밸런서에 트래픽을 주로 라우팅하고 다른 리전의 로드 밸런서는 장애 시 백업으로 사용하고 싶을 수 있습니다. DNS 장애 조치를 구성하면 주 및 보조(백업) 로드 밸런서를 지정할 수 있습니다. Route 53은 주 로드 밸런서가 사용 가능한 상태일 때는 여기로 트래픽을 라우팅하고, 그러지 않으면 보조 로드 밸런서로 라우팅합니다.

대상 상태 사용

- Classic Load Balancer의 별칭 레코드에서 대상 상태 평가가 `Yes`로 설정된 경우 Route 53은 `alias target` 값으로 지정된 리소스의 상태를 평가합니다. Classic Load Balancer 경우 Route 53은 로드 밸런서와 연결된 인스턴스 상태 확인을 사용합니다.

- Classic Load Balancer에 등록된 인스턴스 중 하나 이상이 정상인 경우 Route 53은 별칭 레코드를 정상 상태로 표시합니다. 그러면 Route 53은 라우팅 정책에 따라 레코드를 반환합니다. 장애 조치 라우팅 정책을 사용하는 경우 Route 53는 기본 레코드를 반환합니다.
- Classic Load Balancer에 등록된 모든 인스턴스가 비정상이면 Route 53은 별칭 레코드를 비정상으로 표시합니다. 그러면 Route 53은 라우팅 정책에 따라 레코드를 반환합니다. 장애 조치 라우팅 정책을 사용하는 경우 Route 53은 보조 레코드를 반환합니다.

자세한 내용은 Amazon Route 53 개발자 안내서의의 DNS 장애 조치 구성을 참조하세요.

로드 밸런서에서 사용자 지정 도메인 이름의 연결 해제

호스팅 영역에서 먼저 리소스 레코드 세트를 삭제한 다음 호스팅 영역을 삭제하면 로드 밸런서 인스턴스에서 사용자 지정 도메인 이름의 연결을 해제할 수 있습니다. 자세한 내용은 Amazon Route 53 개발자 안내서의 레코드 편집 및 퍼블릭 호스팅 영역 삭제를 참조하세요.

Classic Load Balancer의 리스너

Elastic Load Balancing을 사용하기 전에 먼저 Classic Load Balancer에 대해 하나 이상의 리스너를 구성해야 합니다. 리스너는 연결 요청을 확인하는 프로세스입니다. 리스너는 프런트 엔드(클라이언트에서 로드 밸런서) 연결을 위한 프로토콜 및 포트와 백엔드(로드 밸런서에서 백엔드 인스턴스) 연결을 위한 프로토콜 및 포트로 구성됩니다.

Elastic Load Balancing은 다음의 프로토콜을 지원합니다.

- HTTP
- HTTPS (보안 HTTP)
- TCP
- SSL (보안 TCP)

HTTPS 프로토콜은 SSL 프로토콜을 사용하여 HTTP 계층에서 보안 연결을 설정합니다. 또한 SSL 프로토콜을 사용하여 TCP 계층에서 보안 연결을 설정할 수 있습니다.

프런트 엔드 연결이 TCP 또는 SSL을 사용하는 경우에는 백엔드 연결이 TCP 또는 SSL을 사용할 수 있습니다. 프런트 엔드 연결이 HTTP 또는 HTTPS를 사용하는 경우에는 백엔드 연결이 HTTP 또는 HTTPS를 사용할 수 있습니다.

백엔드 인스턴스는 포트 1-65535에 대해 수신 대기를 할 수 있습니다.

로드 밸런서는 1~65535 포트들에 대해 수신 대기를 할 수 있습니다.

내용

프로토콜

전형적인 웹 애플리케이션을 위한 통신은 하드웨어 및 소프트웨어 계층을 거칩니다. 각 계층은 특정한 통신 기능을 제공합니다. 통신 기능에 대한 제어권은 한 계층에서 다음 계층으로 순차적으로 전달됩니

다. 개방형 시스템 간 상호 연결(OSI)은 이러한 계층에서 통신용 표준 형식을 실행하기 위한 모델 프레임워크를 정의하는데, 이를 프로토콜이라고 합니다. 자세한 내용은 Wikipedia의 OSI 모델을 참조하십시오.

Elastic Load Balancing을 사용하려면 계층 4와 계층 7에 대한 기본적인 이해가 필요합니다. 계층 4는 로드 밸런서를 통한 클라이언트와 백엔드 인스턴스 간 TCP(Transmission Control Protocol) 연결을 설명하는 전송 계층입니다. 계층 4는 로드 밸런서에서 구성이 가능한 최저 수준입니다. 계층 7은 클라이언트에서 로드 밸런서로의, 그리고 로드 밸런서에서 백엔드 인스턴스로의 HTTP(Hypertext Transfer Protocol) 및 HTTPS(보안 HTTP) 연결을 설명하는 애플리케이션 계층입니다.

Secure Sockets Layer(SSL) 프로토콜은 인터넷 같이 보안성이 낮은 네트워크에서 기밀 데이터를 암호화하는 데 주로 사용됩니다. SSL 프로토콜은 클라이언트와 백엔드 서버 간에 보안 연결을 설정하여 클라이언트와 서버 간에 전달되는 모든 데이터를 안전하고 완전하게 보호합니다.

TCP/SSL 프로토콜

프런트 엔드 및 백엔드 연결 모두에서 TCP(계층 4)를 사용하면 로드 밸런서는 헤더를 변경하지 않고 백엔드 인스턴스로 요청을 전달합니다. 요청을 수신한 후 로드 밸런서는 리스너 구성에 지정된 포트에서 백엔드 인스턴스에 대한 TCP 연결을 엽니다.

로드 밸런서가 클라이언트와 백엔드 인스턴스 간의 트래픽을 가로채기 때문에 백엔드 인스턴스의 액세스 로그에 발원 클라이언트가 아닌 로드 밸런서의 IP 주소가 포함됩니다. 프록시 프로토콜을 활성화해서 소스 IP 주소, 대상 IP 주소, 포트 번호와 같은 클라이언트의 연결 정보를 포함하는 헤더를 추가할 수 있습니다. 이렇게 하면 헤더가 요청의 일부로 백엔드 인스턴스에 전송됩니다. 요청의 첫 번째 라인을 분석해서 연결 정보를 알아낼 수 있습니다. 자세한 내용은 Classic Load Balancer에 프록시 프로토콜 구성 단원을 참조하십시오.

이 구성을 사용하면 세션 고정 또는 X-Forwarded 헤더에서 쿠키가 수신되지 않습니다.

HTTP/HTTPS 프로토콜

프런트 엔드 및 백엔드 연결 모두에서 HTTP(계층 7)를 사용하면 로드 밸런서는 백엔드 인스턴스로 요청을 전송하기 앞서 요청의 헤더를 분석합니다.

Elastic Load Balancing은 HTTP/HTTPS 로드 밸런서 뒤에 있는 정상 상태의 모든 등록 인스턴스에서 하나 이상의 TCP 연결을 열어서 유지합니다. 이 경우 설정된 연결이 HTTP/HTTPS 요청을 수신할 준비가 항상 되어 있어야 합니다.

HTTP 요청 및 HTTP 응답은 헤더 필드를 사용하여 HTTP 메시지에 대한 정보를 전송합니다. Elastic Load Balancing은 `X-Forwarded-For` 헤더를 지원합니다. 로드 밸런서가 클라이언트와 서버 간의 트

래픽을 가로채기 때문에 서버 액세스 로그에 로드 밸런서의 IP 주소만 포함됩니다. 클라이언트의 IP 주소를 확인하려면 `X-Forwarded-For` 요청 헤더를 사용하십시오. 자세한 내용은 X-Forwarded-For 단원을 참조하십시오.

HTTP/HTTPS를 사용하면 로드 밸런서에서 고정 세션을 활성화할 수 있습니다. 고정 세션은 특정 백엔드 인스턴스에 사용자 세션을 바인딩합니다. 이렇게 하면 세션 중에 사용자로부터 들어오는 모든 요청이 동일한 애플리케이션 인스턴스로 전송됩니다. 자세한 내용은 Classic Load Balancer를 위한 고정 세션 구성 단원을 참조하십시오.

로드 밸런서에서 모든 HTTP 확장이 지원되는 것은 아닙니다. 예상치 못한 메서드, 응답 코드 또는 기타 비표준 HTTP 1.0/1.1 구현으로 인해 로드 밸런서가 요청을 종료할 수 없는 경우에는 TCP 리스너를 사용해야 할 수 있습니다.

HTTPS/SSL 리스너

다음 보안 기능을 통해 로드 밸런서를 생성할 수 있습니다.

SSL 서버 인증서

프런트 엔드 연결에서 HTTPS 또는 SSL을 사용할 때는 로드 밸런서에 X.509 인증서(SSL 서버 인증서)를 반드시 배포해야 합니다. 로드 밸런서는 백엔드 인스턴스로 전송하기 전에 클라이언트의 요청을 해독합니다(일명 SSL 종료). 자세한 내용은 Classic Load Balancer를 위한 SSL/TLS 인증서 단원을 참조하십시오.

로드 밸런서가 SSL 종료를 처리하기를 원치 않는 경우(일명 SSL 오프로드)에는 프런트 엔드 및 백엔드 연결 모두에서 TCP를 사용하고 요청을 처리하는 등록 인스턴스에 인증서를 배포할 수 있습니다.

SSL 협상

Elastic Load Balancing은 클라이언트와 로드 밸런서 간에 연결을 설정할 때 SSL 협상에 사용되는 사전 정의된 SSL 협상 구성을 제공합니다. SSL 협상 구성은 광범위한 클라이언트와 호환되며 암호 집합이라는 강력한 암호화 알고리즘을 사용합니다. 그러나 네트워크 상의 모든 데이터를 암호화하고 특정 암호만 허용해야 하는 경우가 있을 수 있습니다. 일부 보안 규정 준수 표준(예: PCI, SOX 등)은 보안 표준 준수를 위해 클라이언트에서 특정한 프로토콜 및 암호 세트를 요구할 수 있습니다. 이 경우에는 특정 요구 사항에 맞춰 사용자 지정 SSL 협상 구성을 생성할 수 있습니다. 암호 및 프로토콜은 30초 이내에 적용됩니다. 자세한 내용은 Classic Load Balancer를 위한 SSL 협상 구성 단원을 참조하십시오.

백엔드 서버 인증

백엔드 연결에서 HTTPS 또는 SSL을 사용하는 경우에는 등록 인스턴스에 대한 인증을 활성화할 수 있습니다. 그런 다음, 인증 프로세스를 사용해 인스턴스가 암호화된 통신만 허용하도록 하고 등록 인스턴스가 올바른 퍼블릭 키를 갖도록 할 수 있습니다.

자세한 내용은 백엔드 서버 인증 구성을 참조하십시오.

Classic Load Balancer의 리스너 구성

다음 표에서는 Classic Load Balancer의 HTTP 및 HTTPS 리스너에 사용할 수 있는 구성을 설명합니다.

사용 사례:	프런트 엔드 프로토콜	프런트 엔드 옵션	백엔드 프로토콜	백엔드 옵션	참고
기본 HTTP 로드 밸런서	HTTP	NA	HTTP	NA	• X-Forwarded 헤더 지원
SSL 암호 해독을 오프로드하기 위해 Elastic Load Balancing을 사용하는 보안 웹 사이트 또는 애플리케이션	HTTPS	SSL 협상	HTTP	NA	• X-Forwarded 헤더 지원 • 로드 밸런서에서 SSL 인증서 배포가 필요
종단 간 암호화를 사용하는 보안 웹 사이트 또는 애플리케이션	HTTPS	SSL 협상	HTTPS	백엔드 인증	• X-Forwarded 헤더 지원 • 로드 밸런서와 등록 인스턴스에서 SSL 인

사용 사례:	프런트 엔드 프로토콜	프런트 엔드 옵션	백엔드 프로토콜	백엔드 옵션	참고
					증서 배포가 필요

다음 표에서는 Classic Load Balancer의 TCP 및 SSL 리스너에 사용할 수 있는 구성을 설명합니다.

사용 사례:	프런트 엔드 프로토콜	프런트 엔드 옵션	백엔드 프로토콜	백엔드 옵션	참고
기본 TCP 로드 밸런서	TCP	NA	TCP	NA	• 프록시 프로토콜 헤더 지원
SSL 암호 해독을 오프로드하기 위해 Elastic Load Balancing을 사용하는 보안 웹 사이트 또는 애플리케이션	SSL	SSL 협상	TCP	NA	• 로드 밸런서에서 SSL 인증서 배포가 필요 • 프록시 프로토콜 헤더 지원
Elastic Load Balancing에서 종단 간 암호화를 사용하는 보안 웹 사이트 또는 애플리케이션	SSL	SSL 협상	SSL	백엔드 인증	• 로드 밸런서와 등록 인스턴스에서 SSL 인증서 배포가 필요 • 백엔드 SSL 연결에서는 SNI 헤더를 삽입하지 않습니다.

사용 사례:	프런트 엔드 프로토콜	프런트 엔드 옵션	백엔드 프로토콜	백엔드 옵션	참고
					• 프록시 프로토콜 헤더를 지원하지 않습니다.

HTTP 헤더 및 Classic Load Balancer

HTTP 요청 및 HTTP 응답은 헤더 필드를 사용하여 HTTP 메시지에 대한 정보를 전송합니다. 헤더 필드는 콜론으로 구분된 이름-값 쌍이며 CR(캐리지 리턴) 및 LF(줄 바꿈)로 구분됩니다. HTTP 헤더 필드의 표준 집합은 RFC 2616, 메시지 헤더에 정의되어 있습니다. 애플리케이션에서 널리 사용되는 비표준 HTTP 헤더도 제공되며 자동으로 추가됩니다. 일부 비표준 HTTP 헤더는 `X-Forwarded` 접두사를 가지고 있습니다. Classic Load Balancer는 다음 `X-Forwarded` 헤더를 지원합니다.

HTTP 연결에 대한 자세한 내용은 Elastic Load Balancing 사용 설명서의 라우팅 요청을 참조하세요.

사전 조건

- 리스너 설정에서 X-Forwarded 헤더가 지원되는지 확인합니다. 자세한 내용은 Classic Load Balancer의 리스너 구성 단원을 참조하십시오.
- 클라이언트 IP 주소를 기록하도록 웹 서버를 구성합니다.

X-Forwarded 헤더

- X-Forwarded-For
- X-Forwarded-Proto
- X-Forwarded-Port

X-Forwarded-For

`X-Forwarded-For` 요청 헤더는 자동으로 추가되어 HTTP 또는 HTTPS 로드 밸런서를 사용할 때 클라이언트의 IP 주소를 식별하는 데 도움을 줍니다. 로드 밸런서가 클라이언트와 서버 간의 트래픽을 가로채기 때문에 서버 액세스 로그에 로드 밸런서의 IP 주소만 포함됩니다. 클라이언트의 IP 주소를 확인하려면 `X-Forwarded-For` 요청 헤더를 사용하십시오. Elastic Load Balancing은 `X-Forwarded-`

For 요청 헤더에 클라이언트의 IP 주소를 저장하고 이 헤더를 서버로 전달합니다. X-Forwarded-For 요청 헤더가 요청에 포함되지 않은 경우, 로드 밸런서는 클라이언트 IP 주소를 요청 값으로 사용하여 하나를 생성합니다. 그렇지 않으면, 로드 밸런서가 클라이언트 IP 주소를 기존 헤더에 추가하고 헤더를 서버로 전달합니다. X-Forwarded-For요청 헤더에는 쉼표로 구분된 여러 IP 주소가 포함될 수 있습니다. 가장 왼쪽 주소는 요청이 처음 만들어진 클라이언트 IP입니다. 그 다음에는 체인에서 후속 프록시 식별자가 옵니다.

X-Forwarded-For 요청 헤더의 형식은 다음과 같습니다.

```
X-Forwarded-For: client-ip-address
```

다음은 IP 주소가 203.0.113.7인 클라이언트의 X-Forwarded-For 요청 헤더입니다.

```
X-Forwarded-For: 203.0.113.7
```

다음은 IPv6 주소가 X-Forwarded-For인 클라이언트의 2001:DB8::21f:5bff:febf:ce22:8a2e 요청 헤더입니다.

```
X-Forwarded-For: 2001:DB8::21f:5bff:febf:ce22:8a2e
```

X-Forwarded-Proto

X-Forwarded-Proto 요청 헤더는 클라이언트가 로드 밸런서 연결에 사용한 프로토콜(HTTP 또는 HTTPS)을 식별하는 데 도움을 줍니다. 서버 액세스 로그에는 서버와 로드 밸런서 간에 사용된 프로토콜만 포함되어 있으며, 클라이언트와 로드 밸런서 간에 사용된 프로토콜에 대한 정보는 포함되어 있지 않습니다. 클라이언트와 로드 밸런서 간에 사용된 프로토콜을 확인하려면 X-Forwarded-Proto 요청 헤더를 사용하십시오. Elastic Load Balancing은 X-Forwarded-Proto 요청 헤더에 클라이언트와 로드 밸런서 간에 사용된 프로토콜을 저장하고 서버로 헤더를 전달합니다.

애플리케이션이나 웹 사이트는 X-Forwarded-Proto 요청 헤더에 저장된 프로토콜을 사용하여 해당 URL로 응답이 리디렉션 되도록 합니다.

X-Forwarded-Proto 요청 헤더의 형식은 다음과 같습니다.

```
X-Forwarded-Proto: originatingProtocol
```

다음 예제에는 HTTPS 요청으로서 클라이언트에서 시작된 요청에 대한 X-Forwarded-Proto 요청 헤더가 포함되어 있습니다.

```
X-Forwarded-Proto: https
```

X-Forwarded-Port

`X-Forwarded-Port` 요청 헤더는 클라이언트가 로드 밸런서 연결에 사용한 대상 포트를 식별하는 데 도움을 줍니다.

Classic Load Balancer를 위한 HTTPS 리스너

암호화된 연결(SSL 오프로드라고도 함)을 위해 SSL/TLS 프로토콜을 사용하는 로드 밸런서를 생성할 수 있습니다. 이 기능을 사용하면 로드 밸런서와 HTTPS 세션을 시작하는 클라이언트 간에, 그리고 로드 밸런서와 EC2 인스턴스 간 연결에 대해 트래픽 암호화가 가능합니다.

Elastic Load Balancing은 보안 정책이라고 하는 SSL(보안소켓 계층) 협상 구성을 사용하여 클라이언트와 로드 밸런서 간의 연결을 협상합니다. 프런트 엔드 연결에서 HTTPS/SSL을 사용하면 사전 정의 보안 정책이나 사용자 지정 보안 정책을 사용할 수 있습니다. 로드 밸런서에 SSL 인증서를 반드시 배포해야 합니다. 로드 밸런서는 이 인증서를 사용해 연결을 종료하고 인스턴스로 전송하기 전에 클라이언트의 요청을 해독합니다. 로드 밸런서는 백엔드 연결에서 정적 암호 그룹을 사용합니다. 인스턴스에서 인증을 활성화하는 옵션을 선택할 수 있습니다.

Classic Load Balancer는 Server Name Indication(SNI)를 지원하지 않습니다. 대신에 다음 대안 중 하나를 사용할 수 있습니다.

- 로드 밸런서에 한 개의 인증서를 배포하고 각 추가 웹사이트에 주체 대체 이름(SAN)을 추가합니다. SAN을 사용하면 하나의 인증서를 사용하여 여러 호스트 이름을 보호할 수 있습니다. 인증서별로 지원되는 SAN의 수와 SAN을 추가 및 삭제하는 방법에 대한 자세한 내용은 인증서 공급자에게 문의하십시오.
- 프런트 엔드 및 백엔드 연결을 위한 포트 443에서 TCP 리스너를 사용하십시오. 로드 밸런서는 요청을 그대로 전달하므로 EC2 인스턴스에서 HTTPS 종료를 처리할 수 있습니다.

Classic Load Balancer는 상호 TLS 인증(mTLS)을 지원하지 않습니다. mTLS 지원을 위해 TCP 수신기를 생성합니다. 로드 밸런서는 요청을 그대로 전달하므로 EC2 인스턴스에서 mTLS를 구현할 수 있습니다.

내용

Classic Load Balancer를 위한 SSL/TLS 인증서

프런트 엔드 리스너에서 HTTPS(SSL 또는 TLS)를 사용하는 경우에는 로드 밸런서에 SSL/TLS 인증서를 반드시 배포해야 합니다. 로드 밸런서는 이 인증서를 사용해 연결을 종료하고 인스턴스로 전송하기 전에 클라이언트의 요청을 해독합니다.

SSL 및 TLS 프로토콜은 X.509 인증서(SSL/TLS 서버 인증서)를 사용해 클라이언트와 백엔드 애플리케이션 모두를 인증합니다. X.509 인증서는 인증 기관(CA)에서 발급한 디지털 형태의 신분증으로, ID 정보, 유효 기간, 퍼블릭 키, 일련 번호, 발급자의 디지털 서명이 포함되어 있습니다.

AWS Certificate Manager 또는 OpenSSL과 같은 SSL 및 TLS 프로토콜을 지원하는 도구를 사용하여 인증서를 생성할 수 있습니다. 로드 밸런서를 위한 HTTPS 리스너를 생성 또는 업데이트할 때 이 인증서를 지정합니다. 로드 밸런서와 함께 사용할 인증서를 생성할 때 도메인 이름을 지정해야 합니다.

로드 밸런서와 함께 사용할 인증서를 생성할 때 도메인 이름을 지정해야 합니다. 인증서의 도메인 이름은 사용자 지정 도메인 이름 레코드와 일치해야 합니다. 일치하지 않는 경우 TLS 연결을 확인할 수 없으므로 트래픽이 암호화되지 않습니다.

인증서에 `www.example.com`과 같은 정규화된 도메인 이름(FQDN) 또는 `example.com`과 같은 apex 도메인 이름을 지정해야 합니다. 별표(*)를 와일드카드로 사용하여 동일한 도메인 내에서 여러 사이트 이름을 보호할 수도 있습니다. 와일드카드 인증서를 요청할 때 별표(*)는 도메인 이름의 맨 왼쪽에 와야 하며 하나의 하위 도메인 수준만 보호할 수 있습니다. 예를 들어 `*.example.com`은 `corp.example.com` 및 `images.example.com`은 보호하지만 `test.login.example.com`을 보호할 수는 없습니다. 또한 `*.example.com`은 `example.com`의 하위 도메인만 보호하고 베어 또는 apex 도메인(`example.com`)은 보호하지 못합니다. 와일드카드 이름은 주체 필드와 인증서의 주체 대체 이름 확장에 표시됩니다. 퍼블릭 인증서 요청에 대한 자세한 내용은 AWS Certificate Manager 사용 설명서의 퍼블릭 인증서 요청을 참조하세요.

를 사용하여 SSL/TLS 인증서 생성 또는 가져오기 AWS Certificate Manager

AWS Certificate Manager (ACM)을 사용하여 로드 밸런서에 대한 인증서를 생성하거나 가져오는 것이 좋습니다. ACM은 Elastic Load Balancing과 통합하여 로드 밸런서에 인증서를 배포합니다. 로드 밸런서에 인증서를 배포하려면 인증서가 로드 밸런서와 같은 리전에 있어야 합니다. 자세한 내용은 AWS Certificate Manager 사용 설명서의 퍼블릭 인증서 요청 또는 인증서 가져오기를 참조하세요.

사용자가 AWS Management Console을 사용하여 로드 밸런서에 인증서를 배포하려면 ACM `ListCertificates` API 작업에 대한 액세스를 허용해야 합니다. 자세한 내용은 AWS Certificate Manager 사용 설명서의 인증서 목록 조회를 참조하세요.

> **Important**
>
> ACM과의 통합을 통해 로드 밸런서에 4096비트 RSA 키 또는 EC 키가 포함된 인증서를 설치할 수 없습니다. 로드 밸런서와 함께 사용하기 위해서는 4096비트 RSA 키 또는 EC 키가 포함된 인증서를 IAM에 업로드해야 합니다.

IAM을 사용하여 SSL/TLS 인증서 확인

ACM을 사용하지 않는 경우 OpenSSL과 같은 SSL/TLS 도구를 사용하여 인증서 서명 요청(CSR)을 생성하고 CA가 서명한 CSR을 가져와서 인증서를 만들고 IAM에 인증서를 업로드할 수 있습니다. 자세한 내용은 IAM 사용 설명서에서 서버 인증서 작업을 참조하세요.

Classic Load Balancer를 위한 SSL 협상 구성

Elastic Load Balancing은 보안 정책이라고 하는 Secure Socket Layer(SSL) 협상 구성을 사용해 클라이언트와 로드 밸런서 간의 연결을 협상합니다. 보안 정책은 A security policy is a combination of SSL 프로토콜, SSL 암호 및 Server Order Preference 옵션의 조합입니다. 로드 밸런서를 위한 SSL 연결 구성에 대한 자세한 내용은 Classic Load Balancer의 리스너를 참조하십시오.

목차

보안 정책

보안 정책은 클라이언트와 로드 밸런서 간의 SSL 협상 동안 어떤 암호와 프로토콜이 지원되는지 결정합니다. 사전 정의 또는 사용자 정의 보안 정책을 사용하도록 Classic Load Balancer를 구성할 수 있습니다.

AWS Certificate Manager (ACM)에서 제공하는 인증서에는 RSA 퍼블릭 키가 포함되어 있습니다. 따라서 ACM가 제공하는 인증서를 사용할 경우 보안 정책에 RSA를 사용하는 암호 그룹을 반드시 포함하고 있어야 하며, 그렇지 않으면 TLS 연결에 실패합니다.

사전 정의 보안 정책

가장 최근의 사전 정의 보안 정책의 이름에는 출시된 연과 월에 따른 버전 정보가 포함되어 있습니다. 예를 들어 기본 사전 정의 보안 정책은 `ELBSecurityPolicy-2016-08`입니다. 사전 정의 보안 정책이 새로 발표될 때마다 이를 사용하도록 구성을 업데이트할 수 있습니다.

사전 정의 보안 정책에서 지원되는 프로토콜 및 암호에 대한 자세한 내용은 Classic Load Balancer에 대해 미리 정의된 SSL 보안 정책를 참조하십시오.

사용자 지정 보안 정책

필요한 암호와 프로토콜을 갖춘 사용자 지정 협상 구성을 생성할 수 있습니다. 예를 들어 일부 보안 규정 준수 표준(예: PCI 및 SOC)은 보안 표준 준수를 위해 특정한 프로토콜 및 암호 세트를 요구할 수 있습니다. 이러한 경우에는 이러한 표준을 충족하는 사용자 지정 보안 정책을 생성할 수 있습니다.

사용자 지정 보안 정책을 생성하는 방법에 대한 자세한 내용은 Classic Load Balancer의 SSL 협상 구성 업데이트를 참조하십시오.

SSL 프로토콜

SSL 프로토콜은 클라이언트와 서버 간에 보안 연결을 설정하여 클라이언트와 로드 밸런서 간에 전달되는 모든 데이터를 안전하게 보호합니다.

Secure Sockets Layer(SSL)와 Transport Layer Security(TLS)는 인터넷 같이 보안성이 낮은 네트워크에서 기밀 데이터를 암호화하는 데 사용되는 암호화 프로토콜입니다. TLS 프로토콜은 SSL 프로토콜의 최신 버전입니다. Elastic Load Balancing 설명서에서는 SSL과 TLS 프로토콜 모두를 SSL 프로토콜이라고 지칭하고 있습니다.

권장 프로토콜

미리 정의된 보안 정책인 ELBSecurityPolicy-TLS-1-2-2017-01에 사용되는 TLS 1.2를 사용하는 것이 좋습니다. 사용자 지정 보안 정책에 TLS 1.2를 사용할 수도 있습니다. 기본 보안 정책은 TLS 1.2와 이전 버전의 TLS를 모두 지원하므로 ELB SecurityPolicy-TLS-1-2-2017-01보다 보안 수준이 낮습니다.

지원 중단된 프로토콜

이전에 사용자 지정 정책에서 SSL 2.0 프로토콜을 활성화한 경우에는 보안 정책을 미리 정의된 보안 정책 중 하나로 업데이트하는 것이 좋습니다.

Server Order Preference

Elastic Load Balancing은 클라이언트와 로드 밸런서 간의 연결 협상을 위해 Server Order Preference 옵션을 지원합니다. SSL 연결 협상이 이루어지는 동안 클라이언트와 로드 밸런서는 각각이 지원하는 암호 및 프로토콜 목록을 선호도 순으로 표시합니다. 기본적으로 클라이언트의 목록에서 로드 밸런서의 암호 중 하나와 일치하는 첫 번째 암호가 SSL 연결을 위해 선택됩니다. 로드 밸런서가 Server Order Preference를 지원하도록 구성이 되어 있으면 로드 밸런서가 클라이언트의 암호 목록에 있는 첫 번째 암호를 선택합니다. 따라서 로드 밸런서가 SSL 연결에 어떤 암호를 사용할 것인지 결정할 수 있습니다. Server Order Preference를 활성화하지 않은 경우에는 클라이언트가 표시한 암호의 순서가 클라이언트와 로드 밸런서 간의 연결 협상에 사용됩니다.

SSL 암호

SSL 암호는 코딩된 메시지를 생성하기 위해 암호화 키를 사용하는 암호화 알고리즘입니다. SSL 프로토콜은 여러 개의 SSL 암호를 사용해 인터넷 상의 데이터를 암호화합니다.

AWS Certificate Manager (ACM)에서 제공하는 인증서에는 RSA 퍼블릭 키가 포함되어 있습니다. 따라서 ACM가 제공하는 인증서를 사용할 경우 보안 정책에 RSA를 사용하는 암호 그룹을 반드시 포함하고 있어야 하며, 그렇지 않으면 TLS 연결에 실패합니다.

Elastic Load Balancing은 클래식 로드 밸런서와 함께 사용할 수 있도록 다음 암호를 지원합니다. 사전 정의 SSL 정책에 따라 이 암호의 하위 집합을 평가합니다. 이들 암호는 모두 사용자 지정 정책에 사용할 수 있습니다. 기본 보안 정책에 포함된 암호들(별표(*)가 있는 암호들)만 사용하는 것이 좋습니다. 여타의 많은 암호들은 안전하지 않기 때문에 사용에 따른 위험은 사용자가 부담해야 합니다.

암호(Ciphers)

- ECDHE-ECDSA-AES128-GCM-SHA256 *
- ECDHE-RSA-AES128-GCM-SHA256 *
- ECDHE-ECDSA-AES128-SHA256 *
- ECDHE-RSA-AES128-SHA256 *
- ECDHE-ECDSA-AES128-SHA *
- ECDHE-RSA-AES128-SHA *
- DHE-RSA-AES128-SHA
- ECDHE-ECDSA-AES256-GCM-SHA384 *
- ECDHE-RSA-AES256-GCM-SHA384 *
- ECDHE-ECDSA-AES256-SHA384 *

- ECDHE-RSA-AES256-SHA384 *
- ECDHE-RSA-AES256-SHA *
- ECDHE-ECDSA-AES256-SHA *
- AES128-GCM-SHA256 *
- AES128-SHA256 *
- AES128-SHA *
- AES256-GCM-SHA384 *
- AES256-SHA256 *
- AES256-SHA *
- DHE-DSS-AES128-SHA
- CAMELLIA128-SHA
- EDH-RSA-DES-CBC3-SHA
- DES-CBC3-SHA
- ECDHE-RSA-RC4-SHA
- RC4-SHA
- ECDHE-ECDSA-RC4-SHA
- DHE-DSS-AES256-GCM-SHA384
- DHE-RSA-AES256-GCM-SHA384
- DHE-RSA-AES256-SHA256
- DHE-DSS-AES256-SHA256
- DHE-RSA-AES256-SHA
- DHE-DSS-AES256-SHA
- DHE-RSA-CAMELLIA256-SHA
- DHE-DSS-CAMELLIA256-SHA
- CAMELLIA256-SHA
- EDH-DSS-DES-CBC3-SHA
- DHE-DSS-AES128-GCM-SHA256
- DHE-RSA-AES128-GCM-SHA256
- DHE-RSA-AES128-SHA256
- DHE-DSS-AES128-SHA256

- DHE-RSA-CAMELLIA128-SHA
- DHE-DSS-CAMELLIA128-SHA
- ADH-AES128-GCM-SHA256
- ADH-AES128-SHA
- ADH-AES128-SHA256
- ADH-AES256-GCM-SHA384
- ADH-AES256-SHA
- ADH-AES256-SHA256
- ADH-CAMELLIA128-SHA
- ADH-CAMELLIA256-SHA
- ADH-DES-CBC3-SHA
- ADH-DES-CBC-SHA
- ADH-RC4-MD5
- ADH-SEED-SHA
- DES-CBC-SHA
- DHE-DSS-SEED-SHA
- DHE-RSA-SEED-SHA
- EDH-DSS-DES-CBC-SHA
- EDH-RSA-DES-CBC-SHA
- IDEA-CBC-SHA
- RC4-MD5
- SEED-SHA
- DES-CBC3-MD5
- DES-CBC-MD5
- RC2-CBC-MD5
- PSK-AES256-CBC-SHA
- PSK-3DES-EDE-CBC-SHA
- KRB5-DES-CBC3-SHA
- KRB5-DES-CBC3-MD5
- PSK-AES128-CBC-SHA

- PSK-RC4-SHA
- KRB5-RC4-SHA
- KRB5-RC4-MD5
- KRB5-DES-CBC-SHA
- KRB5-DES-CBC-MD5
- EXP-EDH-RSA-DES-CBC-SHA
- EXP-EDH-DSS-DES-CBC-SHA
- EXP-ADH-DES-CBC-SHA
- EXP-DES-CBC-SHA
- EXP-RC2-CBC-MD5
- EXP-KRB5-RC2-CBC-SHA
- EXP-KRB5-DES-CBC-SHA
- EXP-KRB5-RC2-CBC-MD5
- EXP-KRB5-DES-CBC-MD5
- EXP-ADH-RC4-MD5
- EXP-RC4-MD5
- EXP-KRB5-RC4-SHA
- EXP-KRB5-RC4-MD5

* 기본 보안 정책 ELBSecurityPolicy-2016-08에 포함된 암호들입니다.

백엔드 연결을 위한 암호 그룹

Classic Load Balancer는 백엔드 연결에 정적 암호 그룹을 사용합니다. Classic Load Balancer와 등록된 인스턴스가 연결을 협상할 수 없는 경우 다음 암호 중 하나를 포함하세요.

- AES256-GCM-SHA384
- AES256-SHA256
- AES256-SHA
- CAMELLIA256-SHA
- AES128-GCM-SHA256
- AES128-SHA256

- AES128-SHA
- CAMELLIA128-SHA
- RC4-SHA
- DES-CBC3-SHA
- DES-CBC-SHA
- DHE-DSS-AES256-GCM-SHA384
- DHE-RSA-AES256-GCM-SHA384
- DHE-RSA-AES256-SHA256
- DHE-DSS-AES256-SHA256
- DHE-RSA-AES256-SHA
- DHE-DSS-AES256-SHA
- DHE-RSA-CAMELLIA256-SHA
- DHE-DSS-CAMELLIA256-SHA
- DHE-DSS-AES128-GCM-SHA256
- DHE-RSA-AES128-GCM-SHA256
- DHE-RSA-AES128-SHA256
- DHE-DSS-AES128-SHA256
- DHE-RSA-AES128-SHA
- DHE-DSS-AES128-SHA
- DHE-RSA-CAMELLIA128-SHA
- DHE-DSS-CAMELLIA128-SHA
- EDH-RSA-DES-CBC3-SHA
- EDH-DSS-DES-CBC3-SHA
- EDH-RSA-DES-CBC-SHA
- EDH-DSS-DES-CBC-SHA

Classic Load Balancer에 대해 미리 정의된 SSL 보안 정책

HTTPS/SSL 리스너에 대한 사전 정의 보안 정책 중 하나를 선택할 수 있습니다.
`ELBSecurityPolicy-TLS` 정책 중 하나를 사용하여 특정한 TLS 프로토콜 버전을 비활성화해야 하

는 규정 준수 및 보안 표준을 충족할 수 있습니다. 또는 사용자 지정 보안 정책을 생성할 수 있습니다. 자세한 내용은 SSL 협상 구성 업데이트 단원을 참조하십시오.

RSA 기반 및 DSA 기반 암호는 SSL 인증서를 생성하는 데 사용되는 서명 알고리즘마다 고유합니다. 보안 정책에서 지원되는 암호들을 기반으로 하는 서명 알고리즘을 사용해서 SSL 인증서를 생성하십시오.

Server Order Preference가 활성화된 정책을 선택한 경우에는 로드 밸런서가 여기에 지정된 순서대로 암호를 사용하여 클라이언트와 로드 밸런서 간에 연결 협상을 합니다. 그렇지 않으면 로드 밸런서가 클라이언트에 표시된 순서대로 암호를 사용합니다.

아래 섹션에서는 활성화된 SSL 프로토콜 및 SSL 암호를 포함하여 Classic Load Balancer에 대해 가장 최근에 사전 정의된 보안 정책을 설명합니다. describe-load-balancer-policies 명령을 사용하여 사전 정의된 정책을 설명할 수 있습니다.

> **Tip**
>
> 이 정보는 Classic Load Balancer에만 적용됩니다. 다른 로드 밸런서에 적용되는 자세한 내용은 Application Load Balancer에 대한 보안 정책 및 Network Load Balancer에 대한 보안 정책을 참조하세요.

내용

정책별 프로토콜

다음 표에서는 각 보안 정책이 지원하는 TLS 프로토콜을 설명합니다.

보안 정책	TLS 1.2	TLS 1.1	TLS 1.0
ELBSecurityPolicy-TLS-1-2-2017-01	예	아니요	아니요

보안 정책	TLS 1.2	TLS 1.1	TLS 1.0
ELBSecurityPolicy-TLS-1-1-2017-01	예	예	아니요
ELBSecurityPolicy-2016-08	예	예	예
ELBSecurityPolicy-2015-05	예	예	예
ELBSecurityPolicy-2015-03	예	예	예
ELBSecurityPolicy-2015-02	예	예	예

정책별 암호

다음 표에서는 각 보안 정책이 지원하는 암호를 설명합니다.

보안 정책	암호(Ciphers)
ELBSecurityPolicy-TLS-1-2-2017-01	• ECDHE-ECDSA-AES128-GCM-SHA256 • ECDHE-RSA-AES128-GCM-SHA256 • ECDHE-ECDSA-AES128-SHA256 • ECDHE-RSA-AES128-SHA256 • ECDHE-ECDSA-AES256-GCM-SHA384 • ECDHE-RSA-AES256-GCM-SHA384 • ECDHE-ECDSA-AES256-SHA384 • ECDHE-RSA-AES256-SHA384 • AES128-GCM-SHA256 • AES128-SHA256 • AES256-GCM-SHA384 • AES256-SHA256
ELBSecurityPolicy-TLS-1-1-2017-01	• ECDHE-ECDSA-AES128-GCM-SHA256 • ECDHE-RSA-AES128-GCM-SHA256 • ECDHE-ECDSA-AES128-SHA256

보안 정책	암호(Ciphers)
	• ECDHE-RSA-AES128-SHA256 • ECDHE-ECDSA-AES128-SHA • ECDHE-RSA-AES128-SHA • ECDHE-ECDSA-AES256-GCM-SHA384 • ECDHE-RSA-AES256-GCM-SHA384 • ECDHE-ECDSA-AES256-SHA384 • ECDHE-RSA-AES256-SHA384 • ECDHE-ECDSA-AES256-SHA • ECDHE-RSA-AES256-SHA • AES128-GCM-SHA256 • AES128-SHA256 • AES128-SHA • AES256-GCM-SHA384 • AES256-SHA256 • AES256-SHA

보안 정책	암호(Ciphers)
ELBSecurityPolicy-2016-08	• ECDHE-ECDSA-AES128-GCM-SHA256 • ECDHE-RSA-AES128-GCM-SHA256 • ECDHE-ECDSA-AES128-SHA256 • ECDHE-RSA-AES128-SHA256 • ECDHE-ECDSA-AES128-SHA • ECDHE-RSA-AES128-SHA • ECDHE-ECDSA-AES256-GCM-SHA384 • ECDHE-RSA-AES256-GCM-SHA384 • ECDHE-ECDSA-AES256-SHA384 • ECDHE-RSA-AES256-SHA384 • ECDHE-ECDSA-AES256-SHA • ECDHE-RSA-AES256-SHA • AES128-GCM-SHA256 • AES128-SHA256 • AES128-SHA • AES256-GCM-SHA384 • AES256-SHA256 • AES256-SHA

보안 정책	암호(Ciphers)
ELBSecurityPolicy-2015-05	• ECDHE-ECDSA-AES128-GCM-SHA256 • ECDHE-RSA-AES128-GCM-SHA256 • ECDHE-ECDSA-AES128-SHA256 • ECDHE-RSA-AES128-SHA256 • ECDHE-ECDSA-AES128-SHA • ECDHE-RSA-AES128-SHA • ECDHE-ECDSA-AES256-GCM-SHA384 • ECDHE-RSA-AES256-GCM-SHA384 • ECDHE-ECDSA-AES256-SHA384 • ECDHE-RSA-AES256-SHA384 • ECDHE-ECDSA-AES256-SHA • ECDHE-RSA-AES256-SHA • AES128-GCM-SHA256 • AES128-SHA256 • AES128-SHA • AES256-GCM-SHA384 • AES256-SHA256 • AES256-SHA • DES-CBC3-SHA

보안 정책	암호(Ciphers)
ELBSecurityPolicy-2015-03	• ECDHE-ECDSA-AES128-GCM-SHA256 • ECDHE-RSA-AES128-GCM-SHA256 • ECDHE-ECDSA-AES128-SHA256 • ECDHE-RSA-AES128-SHA256 • ECDHE-ECDSA-AES128-SHA • ECDHE-RSA-AES128-SHA • ECDHE-ECDSA-AES256-GCM-SHA384 • ECDHE-RSA-AES256-GCM-SHA384 • ECDHE-ECDSA-AES256-SHA384 • ECDHE-RSA-AES256-SHA384 • ECDHE-ECDSA-AES256-SHA • ECDHE-RSA-AES256-SHA • AES128-GCM-SHA256 • AES128-SHA256 • AES128-SHA • AES256-GCM-SHA384 • AES256-SHA256 • AES256-SHA • DHE-RSA-AES128-SHA • DHE-DSS-AES128-SHA • DES-CBC3-SHA

보안 정책	암호(Ciphers)
ELBSecurityPolicy-2015-02	• ECDHE-ECDSA-AES128-GCM-SHA256 • ECDHE-RSA-AES128-GCM-SHA256 • ECDHE-ECDSA-AES128-SHA256 • ECDHE-RSA-AES128-SHA256 • ECDHE-ECDSA-AES128-SHA • ECDHE-RSA-AES128-SHA • ECDHE-ECDSA-AES256-GCM-SHA384 • ECDHE-RSA-AES256-GCM-SHA384 • ECDHE-ECDSA-AES256-SHA384 • ECDHE-RSA-AES256-SHA384 • ECDHE-ECDSA-AES256-SHA • ECDHE-RSA-AES256-SHA • AES128-GCM-SHA256 • AES128-SHA256 • AES128-SHA • AES256-GCM-SHA384 • AES256-SHA256 • AES256-SHA • DHE-RSA-AES128-SHA • DHE-DSS-AES128-SHA

암호별 정책

다음 표에서는 각 암호를 지원하는 보안 정책을 설명합니다.

암호 이름	보안 정책	암호 그룹
OpenSSL – ECDHE-ECDSA-AES128-GCM-SHA256	• ELBSecurityPolicy-TLS-1-2-2017-01 • ELBSecurityPolicy-TLS-1-1-2017-01 • ELBSecurityPolicy-2016-08	c02b

암호 이름	보안 정책	암호 그룹
IANA – TLS_ECDHE_ECDSA_WITH_AES_128_GCM_SHA256	• ELBSecurityPolicy-2015-05 • ELBSecurityPolicy-2015-03 • ELBSecurityPolicy-2015-02	
OpenSSL – ECDHE-RSA-AES128-GCM-SHA256 IANA – TLS_ECDHE_RSA_WITH_AES_128_GCM_SHA256	• ELBSecurityPolicy-TLS-1-2-2017-01 • ELBSecurityPolicy-TLS-1-1-2017-01 • ELBSecurityPolicy-2016-08 • ELBSecurityPolicy-2015-05 • ELBSecurityPolicy-2015-03 • ELBSecurityPolicy-2015-02	c02f
OpenSSL – ECDHE-ECDSA-AES128-SHA256 IANA – TLS_ECDHE_ECDSA_WITH_AES_128_CBC_SHA256	• ELBSecurityPolicy-TLS-1-2-2017-01 • ELBSecurityPolicy-TLS-1-1-2017-01 • ELBSecurityPolicy-2016-08 • ELBSecurityPolicy-2015-05 • ELBSecurityPolicy-2015-03 • ELBSecurityPolicy-2015-02	c023
OpenSSL – ECDHE-RSA-AES128-SHA256 IANA – TLS_ECDHE_RSA_WITH_AES_128_CBC_SHA256	• ELBSecurityPolicy-TLS-1-2-2017-01 • ELBSecurityPolicy-TLS-1-1-2017-01 • ELBSecurityPolicy-2016-08 • ELBSecurityPolicy-2015-05 • ELBSecurityPolicy-2015-03 • ELBSecurityPolicy-2015-02	c027
OpenSSL – ECDHE-ECDSA-AES128-SHA IANA – TLS_ECDHE_ECDSA_WITH_AES_128_CBC_SHA	• ELBSecurityPolicy-TLS-1-1-2017-01 • ELBSecurityPolicy-2016-08 • ELBSecurityPolicy-2015-05 • ELBSecurityPolicy-2015-03 • ELBSecurityPolicy-2015-02	c009

암호 이름	보안 정책	암호 그룹
OpenSSL – ECDHE-RSA-AES128-SHA IANA – TLS_ECDHE_RSA_WITH_AES_128_CBC_SHA	• ELBSecurityPolicy-TLS-1-1-2017-01 • ELBSecurityPolicy-2016-08 • ELBSecurityPolicy-2015-05 • ELBSecurityPolicy-2015-03 • ELBSecurityPolicy-2015-02	c013
OpenSSL – ECDHE-ECDSA-AES256-GCM-SHA384 IANA – TLS_ECDHE_ECDSA_WITH_AES_256_GCM_SHA384	• ELBSecurityPolicy-TLS-1-2-2017-01 • ELBSecurityPolicy-TLS-1-1-2017-01 • ELBSecurityPolicy-2016-08 • ELBSecurityPolicy-2015-05 • ELBSecurityPolicy-2015-03 • ELBSecurityPolicy-2015-02	c02c
OpenSSL – ECDHE-RSA-AES256-GCM-SHA384 IANA – TLS_ECDHE_RSA_WITH_AES_256_GCM_SHA384	• ELBSecurityPolicy-TLS-1-2-2017-01 • ELBSecurityPolicy-TLS-1-1-2017-01 • ELBSecurityPolicy-2016-08 • ELBSecurityPolicy-2015-05 • ELBSecurityPolicy-2015-03 • ELBSecurityPolicy-2015-02	c030
OpenSSL – ECDHE-ECDSA-AES256-SHA384 IANA – TLS_ECDHE_ECDSA_WITH_AES_256_CBC_SHA384	• ELBSecurityPolicy-TLS-1-2-2017-01 • ELBSecurityPolicy-TLS-1-1-2017-01 • ELBSecurityPolicy-2016-08 • ELBSecurityPolicy-2015-05 • ELBSecurityPolicy-2015-03 • ELBSecurityPolicy-2015-02	c024

암호 이름	보안 정책	암호 그룹
OpenSSL – ECDHE-RSA-AES256-SHA384 IANA – TLS_ECDHE_RSA_WITH_AES_256_CBC_SHA384	• ELBSecurityPolicy-TLS-1-2-2017-01 • ELBSecurityPolicy-TLS-1-1-2017-01 • ELBSecurityPolicy-2016-08 • ELBSecurityPolicy-2015-05 • ELBSecurityPolicy-2015-03 • ELBSecurityPolicy-2015-02	c028
OpenSSL – ECDHE-ECDSA-AES256-SHA IANA – TLS_ECDHE_RSA_WITH_AES_256_CBC_SHA	• ELBSecurityPolicy-TLS-1-1-2017-01 • ELBSecurityPolicy-2016-08 • ELBSecurityPolicy-2015-05 • ELBSecurityPolicy-2015-03 • ELBSecurityPolicy-2015-02	c014
OpenSSL – ECDHE-RSA-AES256-SHA IANA – TLS_ECDHE_ECDSA_WITH_AES_256_CBC_SHA	• ELBSecurityPolicy-TLS-1-1-2017-01 • ELBSecurityPolicy-2016-08 • ELBSecurityPolicy-2015-05 • ELBSecurityPolicy-2015-03 • ELBSecurityPolicy-2015-02	c00a
OpenSSL – AES128-GCM-SHA256 IANA – TLS_RSA_WITH_AES_128_GCM_SHA256	• ELBSecurityPolicy-TLS-1-2-2017-01 • ELBSecurityPolicy-TLS-1-1-2017-01 • ELBSecurityPolicy-2016-08 • ELBSecurityPolicy-2015-05 • ELBSecurityPolicy-2015-03 • ELBSecurityPolicy-2015-02	9c

암호 이름	보안 정책	암호 그룹
OpenSSL – AES128-SHA256 IANA – TLS_RSA_WITH_AES_128_CBC_SHA256	• ELBSecurityPolicy-TLS-1-2-2017-01 • ELBSecurityPolicy-TLS-1-1-2017-01 • ELBSecurityPolicy-2016-08 • ELBSecurityPolicy-2015-05 • ELBSecurityPolicy-2015-03 • ELBSecurityPolicy-2015-02	3c
OpenSSL – AES128-SHA IANA – TLS_RSA_WITH_AES_128_CBC_SHA	• ELBSecurityPolicy-TLS-1-1-2017-01 • ELBSecurityPolicy-2016-08 • ELBSecurityPolicy-2015-05 • ELBSecurityPolicy-2015-03 • ELBSecurityPolicy-2015-02	2f
OpenSSL – AES256-GCM-SHA384 IANA – TLS_RSA_WITH_AES_256_GCM_SHA384	• ELBSecurityPolicy-TLS-1-2-2017-01 • ELBSecurityPolicy-TLS-1-1-2017-01 • ELBSecurityPolicy-2016-08 • ELBSecurityPolicy-2015-05 • ELBSecurityPolicy-2015-03 • ELBSecurityPolicy-2015-02	9d
OpenSSL – AES256-SHA256 IANA – TLS_RSA_WITH_AES_256_CBC_SHA256	• ELBSecurityPolicy-TLS-1-2-2017-01 • ELBSecurityPolicy-TLS-1-1-2017-01 • ELBSecurityPolicy-2016-08 • ELBSecurityPolicy-2015-05 • ELBSecurityPolicy-2015-03 • ELBSecurityPolicy-2015-02	3d

암호 이름	보안 정책	암호 그룹
OpenSSL – AES256-SHA IANA – TLS_RSA_WITH_AES_256_CBC_SHA	• ELBSecurityPolicy-TLS-1-1-2017-01 • ELBSecurityPolicy-2016-08 • ELBSecurityPolicy-2015-05 • ELBSecurityPolicy-2015-03 • ELBSecurityPolicy-2015-02	35
OpenSSL – DHE-RSA-AES128-SHA IANA – TLS_DHE_RSA_WITH_AES_128_CBC_SHA	• ELBSecurityPolicy-2015-03 • ELBSecurityPolicy-2015-02	33
OpenSSL – DHE-DSS-AES128-SHA IANA – TLS_DHE_DSS_WITH_AES_128_CBC_SHA	• ELBSecurityPolicy-2015-03 • ELBSecurityPolicy-2015-02	32
OpenSSL – DES-CBC3-SHA IANA – TLS_RSA_WITH_3DES_EDE_CBC_SHA	• ELBSecurityPolicy-2015-05 • ELBSecurityPolicy-2015-03	0a

HTTPS 리스너를 통해 Classic Load Balancer를 생성

로드 밸런서는 클라이언트로부터 요청을 가져와서 로드 밸런서에 등록된 EC2 인스턴스에 이들을 분배합니다.

HTTP (80) 및 HTTPS (443) 포트 모두에서 로드 밸런서 리스너를 생성할 수 있습니다. HTTPS 리스너가 포트 80의 인스턴스로 요청을 전송하도록 지정한 경우에는 로드 밸런서가 요청을 종료하고 로드 밸런서에서 인스턴스로의 통신이 암호화되지 않습니다. HTTPS 리스너가 포트 443의 인스턴스로 요청을 전송하는 경우에는 로드 밸런서에서 인스턴스로의 통신이 암호화됩니다.

로드 밸런서가 인스턴스와의 통신을 위해 암호화된 연결을 사용하는 경우에는 선택에 따라 인스턴스 인증을 활성화할 수 있습니다. 따라서 퍼블릭 키가 통신 용도로 로드 밸런서에 지정한 키와 일치하는 경우에만 로드 밸런서가 인스턴스와 통신을 하도록 할 수 있습니다.

기존 로드 밸런서에 HTTPS 리스너를 추가하는 방법에 대한 자세한 내용은 Classic Load Balancer를 위한 HTTPS 리스너 구성를 참조하십시오.

내용

사전 조건

시작하기 전에 다음 사전 조건을 충족해야 합니다.

- VPC 관련 권장 사항의 단계를 수행합니다.
- 로드 밸런서에 등록할 계획인 EC2 인스턴스를 시작합니다. 이들 인스턴스를 위한 보안 그룹은 로드 밸런서에서 나오는 트래픽을 허용해야 합니다.
- EC2 인스턴스는 HTTP 상태 코드 200을 통해 상태 확인 대상에 응답해야 합니다. 자세한 내용은 Classic Load Balancer의 인스턴스 상태 확인 단원을 참조하십시오.
- EC2 인스턴스에서 연결 유지 옵션을 활성화할 계획이라면 연결 유지 설정을 최소한 로드 밸런서의 유휴 제한 시간 설정으로 설정하는 것이 좋습니다. 로드 밸런서가 인스턴스에 대한 연결을 종료할 책임이 있는지 확인하고 싶다면 연결 유지 시간 동안 인스턴스에 설정된 값이 로드 밸런서의 유휴 제한 시간 설정보다 큰지 확인합니다. 자세한 내용은 Classic Load Balancer에 대한 유휴 연결 제한 시간 구성 단원을 참조하십시오.
- 보안 리스너를 생성하는 경우에는 로드 밸런서에 SSL 서버 인증서를 반드시 배포해야 합니다. 로드 밸런서는 이 인증서를 사용해 연결을 종료하고 인스턴스로 전송하기 전에 요청을 해독합니다. SSL 인증서가 없는 경우에는 하나를 생성할 수 있습니다. 자세한 내용은 Classic Load Balancer를 위한 SSL/TLS 인증서 단원을 참조하십시오.

콘솔을 사용하여 HTTPS 로드 밸런서 생성

이 예제에서는 로드 밸런서용으로 2개의 리스너를 구성합니다. 첫 번째 리스너는 포트 80에서 HTTP 요청을 수락하고 HTTP를 사용하여 포트 80의 인스턴스에 요청을 전송합니다. 두 번째 리스너는 포트 443에서 HTTPS 요청을 수락하고 포트 80에서 HTTP를 사용해 (또는 백엔드 인스턴스 인증을 구성하고 싶은 경우에는 포트 43에서 HTTPS를 사용해) 인스턴스에 이들을 전송합니다.

리스너는 연결 요청을 확인하는 프로세스입니다. 리스너는 프런트 엔드(클라이언트에서 로드 밸런서) 연결을 위한 프로토콜 및 포트와 백엔드(로드 밸런서에서 인스턴스) 연결을 위한 프로토콜 및 포트로

구성됩니다. Elastic Load Balancing에서 지원되는 포트, 프로토콜 및 리스너 구성에 대한 자세한 내용은 Classic Load Balancer의 리스너 섹션을 참조하세요.

콘솔을 사용하여 보안 Classic Load Balancer를 생성하려면

1. https://console.aws.amazon.com/ec2/에서 Amazon EC2 콘솔을 엽니다.
2. 탐색 모음에서 로드 밸런서의 리전을 선택합니다. EC2 인스턴스를 위해 선택한 리전과 동일한 리전을 선택해야 합니다.
3. 탐색 창의 Load Balancing 아래에서 로드 밸런서를 선택합니다.
4. 로드 밸런서 생성을 선택합니다.
5. Classic Load Balancer 섹션을 확장하고 생성을 선택합니다.
6. 기본 구성
   a. 로드 밸런서 이름에 로드 밸런서의 이름을 입력합니다.

      Classic Load Balancer의 이름은 해당 리전의 Classic Load Balancer 세트 내에서 고유한 이름이어야 하고, 최대 32자여야 하며, 알파벳 문자 및 하이픈만 포함해야 하고, 하이픈으로 시작하거나 끝나지 않아야 합니다.

   b. 스키마에서 인터넷 연결을 선택합니다.
7. 네트워크 매핑
   a. VPC에서는 인스턴스에 대해 선택한 것과 동일한 VPC를 선택합니다.
   b. 매핑에서 먼저 가용 영역을 선택하고, 사용 가능한 서브넷 중에서 퍼블릭 서브넷을 선택합니다. 가용 영역당 서브넷 한 개만 선택할 수 있습니다. 로드 밸런서의 가용성을 높이려면 둘 이상의 가용 영역과 서브넷을 선택합니다.
8. 보안 그룹
   - 보안 그룹에서 포트 80에서 필요한 HTTP 트래픽과 포트 443에서 필요한 HTTPS 트래픽을 허용하도록 구성된 기존 보안 그룹을 선택합니다.

     보안 그룹이 없는 경우 필요한 규칙을 사용하여 새 보안 그룹을 생성할 수 있습니다.
9. 리스너 및 라우팅
   a. 기본 리스너를 기본 설정으로 두고 리스너 추가를 선택합니다.
   b. 새 리스너의 리스너에서 HTTPS를 프로토콜로 선택하면 포트가 443으로 업데이트됩니다. 기본적으로 인스턴스는 포트 80에서 HTTP 프로토콜을 사용합니다.

c. 백엔드 인증이 필요한 경우 인스턴스 프로토콜을 HTTPS로 변경합니다. 이렇게 해도 인스턴스 포트가 443으로 업데이트됩니다.

10. 보안 리스너 설정

프런트 엔드 리스너에서 HTTPS 또는 SSL을 사용할 때는 로드 밸런서에 SSL 인증서를 반드시 배포해야 합니다. 로드 밸런서는 이 인증서를 사용해 연결을 종료하고 인스턴스로 전송하기 전에 클라이언트의 요청을 해독합니다. 또한 보안 그룹을 반드시 지정해야 합니다. Elastic Load Balancing이 SSL 협상 구성이 사전 정의되어 있는 보안 정책을 제공하거나 사용자가 보안 정책을 자체적으로 지정할 수 있습니다. 백엔드 연결에서 HTTPS/SSL을 설정한 경우에는 인스턴스 인증을 설정할 수 있습니다.

a. 보안 정책의 경우 항상 최신 사전 정의 보안 정책을 사용하거나 사용자 지정 정책을 생성하는 것이 좋습니다. Update the SSL Negotiation Configuration을 참조하세요.

b. 기본 SSL/TLS 인증서에서 다음 옵션을 사용할 수 있습니다.

- 를 사용하여 인증서를 생성하거나 가져온 경우 ACM에서를 AWS Certificate Manager선택한 다음 인증서 선택에서 인증서를 선택합니다.
- IAM을 사용하여 인증서를 가져온 경우 IAM에서를 선택하고 인증서 선택에서 인증서를 선택합니다.
- 가져올 인증서가 있지만 리전에서 ACM을 이용할 수 없는 경우 가져오기를 선택하고 IAM으로를 선택합니다. 인증서 이름 필드에 인증서의 이름을 입력합니다. 인증서 프라이빗 키에서 프라이빗 키 파일(PEM 인코딩)의 콘텐츠를 복사해 붙여넣습니다. 인증서 본문에서 퍼블릭 키 인증서 파일(PEM 인코딩)의 콘텐츠를 복사해 붙여 넣습니다. 자체 서명 인증서를 사용하고 있지 않고 브라우저가 인증서를 묵시적으로 수락하는 것이 중요하지 않다면 인증서 체인(Certificate Chain)에 인증서 체인 파일(PEM 인코딩)의 콘텐츠를 복사해 붙여넣습니다.

c. 암호화된 연결을 사용해 인스턴스와 통신을 하도록 HTTPS 리스너를 구성한 경우에는 백엔드 인증 인증서에서 인스턴스 인증을 설정할 수 있습니다.

> **Note**
>
> 백엔드 인증 인증서 섹션이 보이지 않는 경우 리스너 및 라우팅으로 돌아가 HTTPS를 인스턴스의 프로토콜로 선택합니다.

i. Certificate name에 퍼블릭 키 인증서의 이름을 입력합니다.

ii. 인증서 본문(PEM 인코딩)에서 인증서의 콘텐츠를 복사해 붙여넣습니다. 퍼블릭 키가 이 키와 일치하는 경우에만 로드 밸런서가 인스턴스와 통신을 합니다.

iii. 다른 인증서를 추가하려면 새 백엔드 인증서 추가를 선택합니다. 5개까지 가능합니다.

11. 상태 확인

    a. ping 대상 섹션에서 ping 프로토콜과 ping 포트를 선택합니다. EC2 인스턴스는 지정된 ping 포트에서 트래픽을 수락해야 합니다.

    b. ping 포트에서 포트가 80인지 확인합니다.

    c. ping 경로에서 기본값을 슬래시 하나(/)로 대체합니다. 이렇게 하면 Elastic Load Balancing에 index.html와 같은 웹 서버의 기본 홈 페이지에 상태 확인 쿼리를 보내도록 지시합니다.

    d. 고급 상태 확인 설정에서 기본값을 사용합니다.

12. 인스턴스

    a. 인스턴스 추가를 선택하면 인스턴스 선택 화면이 나타납니다.

    b. 사용 가능한 인스턴스에서 앞서 선택한 네트워크 설정을 기반으로 로드 밸런서에 사용 가능한 현재 인스턴스 중 선택할 수 있습니다.

    c. 원하는 대로 선택했다면 확인을 선택하여 로드 밸런서에 등록할 인스턴스를 추가합니다.

13. 속성

    • 교차 영역 로드 밸런싱 활성화, Connection Draining 활성화, 제한 시간(드레이닝 간격)의 기본값은 유지합니다.

14. 로드 밸런서 태그(선택 사항)

    a. 키 필드는 필수입니다.

    b. 값 필드는 선택 사항입니다.

    c. 다른 태그를 추가하려면 새 태그 추가를 선택하고 키 필드 및 값 필드(선택 사항)에 값을 입력합니다.

    d. 제거하려는 태그 옆에 있는 제거를 선택하여 기존 태그를 제거합니다.

15. 요약 및 생성

    a. 설정을 변경해야 하는 경우 변경해야 하는 설정 옆에 있는 편집을 선택합니다.

    b. 요약에 표시된 모든 설정이 원하는 대로 되어 있다면 로드 밸런서 생성을 선택하여 로드 밸런서 생성을 시작합니다.

    c. 마지막 생성 페이지에서 로드 밸런서 보기를 선택하여 Amazon EC2 콘솔에서 로드 밸런서를 봅니다.

16. Verify

    a. 새로운 로드 밸런서를 선택합니다.

    b. 대상 인스턴스 탭에서 상태 확인 열을 확인합니다. EC2 인스턴스 중 적어도 하나 이상이 서비스 상태가 되어야만 로드 밸런서를 테스트할 수 있습니다.

    c. 세부 정보 섹션에서 로드 밸런서 DNS 이름을 복사합니다(`my-load-balancer-1234567890.us-east-1.elb.amazonaws.com`과 유사함).

    d. 로드 밸런서 DNS 이름을 퍼블릭 인터넷에 연결된 웹 브라우저의 주소 필드에 붙여넣습니다. 로드 밸런서가 올바르게 작동 중인 경우 서버의 기본 페이지가 표시됩니다.

17. 삭제(선택 사항)

    a. 로드 밸런서를 가리키는 도메인을 위한 CNAME 레코드가 있는 경우에는 새로운 위치를 가리키도록 하고 로드 밸런서를 삭제하기 전에 DNS 변경이 적용될 때까지 기다립니다.

    b. https://console.aws.amazon.com/ec2/에서 Amazon EC2 콘솔을 엽니다.

    c. 로드 밸런서를 선택합니다.

    d. 작업, 로드 밸런서 삭제를 선택합니다.

    e. 확인 메시지가 나타나면 `confirm`을 입력한 다음 삭제를 선택합니다.

    f. 로드 밸런서를 삭제한 후에도 로드 밸런서에 등록된 EC2 인스턴스는 계속 실행됩니다. 계속 실행되는 일부 또는 전체 시간별로 요금이 청구됩니다. EC2 인스턴스가 더 이상 필요하지 않은 경우 추가 요금이 발생하지 않도록 중지하거나 종료할 수 있습니다.

를 사용하여 HTTPS 로드 밸런서 생성 AWS CLI

다음 지침에 따라 AWS CLI를 사용하여 HTTPS/SSL 로드 밸런서를 만드십시오.

작업

- 1단계: 리스너 구성
- 2단계: SSL 보안 정책 구성
- 3단계: 백엔드 인스턴스 인증 구성(선택 사항)
- 4단계: 상태 확인 구성(선택 사항)
- 5단계: EC2 인스턴스 등록
- 6단계: 인스턴스 확인
- 7단계: 로드 밸런서 삭제(선택 사항)

1단계: 리스너 구성

리스너는 연결 요청을 확인하는 프로세스입니다. 리스너는 프런트 엔드(클라이언트에서 로드 밸런서) 연결을 위한 프로토콜 및 포트와 백엔드(로드 밸런서에서 인스턴스) 연결을 위한 프로토콜 및 포트로 구성됩니다. Elastic Load Balancing에서 지원되는 포트, 프로토콜 및 리스너 구성에 대한 자세한 내용은 Classic Load Balancer의 리스너 섹션을 참조하세요.

이 예제에서는 프런트 엔드 및 백엔드 연결에 사용할 포트와 프로토콜을 지정하여 로드 밸런서용으로 2개의 리스너를 구성합니다. 첫 번째 리스너는 포트 80에서 HTTP 요청을 수락하고 HTTP를 사용하여 포트 80의 인스턴스에 요청을 전송합니다. 두 번째 리스너는 포트 443에서 HTTPS 요청을 수락하고 포트 80의 HTTP를 사용하여 인스턴스에 요청을 전송합니다.

두 번째 리스너가 프런트 엔드 연결을 위해 HTTPS를 사용하기 때문에 로드 밸런서에 SSL 서버 인증서를 반드시 배포해야 합니다. 로드 밸런서는 이 인증서를 사용해 연결을 종료하고 인스턴스로 전송하기 전에 요청을 해독합니다.

로드 밸런서를 위한 리스너를 구성하려면

1. SSL 인증서의 Amazon 리소스 이름(ARN)을 가져옵니다. 다음 예를 참조하십시오.

   ACM

   ```
   arn:aws:acm:region:123456789012:certificate/12345678-1234-1234-1234-123456789012
   ```

   IAM

   ```
   arn:aws:iam::123456789012:server-certificate/my-server-certificate
   ```

2. 두 개의 리스너를 가진 로드 밸런서를 구성하려면 아래 create-load-balancer 명령을 사용하세요.

   ```
   aws elb create-load-balancer --load-balancer-name my-load-balancer --listeners
    "Protocol=http,LoadBalancerPort=80,InstanceProtocol=http,InstancePort=80"
    "Protocol=https,LoadBalancerPort=443,InstanceProtocol=http,InstancePort=80,SSLCertificateI
    --availability-zones us-west-2a
   ```

   다음은 응답의 예입니다.

   ```
   {
     "DNSName": "my-loadbalancer-012345678.us-west-2.elb.amazonaws.com"
   }
   ```

3. (선택 사항) 로드 밸런서의 세부 정보를 확인하려면 아래 describe-load-balancers 명령을 사용하세요.

```
aws elb describe-load-balancers --load-balancer-name my-load-balancer
```

2단계: SSL 보안 정책 구성

사전 정의 보안 정책 중 하나를 선택하거나 보안 정책을 자체적으로 지정할 수 있습니다. 그렇지 않으면 Elastic Load Balancing이 기본적인 사전 정의 보안 정책, ELBSecurityPolicy-2016-08을 통해 로드 밸런서를 구성합니다. 자세한 내용은 Classic Load Balancer를 위한 SSL 협상 구성 단원을 참조하십시오.

로드 밸런서가 기본 보안 정책에 연결되어 있는지 확인하려면

아래 describe-load-balancers 명령을 사용하세요.

```
aws elb describe-load-balancers --load-balancer-name my-loadbalancer
```

다음은 응답의 예입니다. ELBSecurityPolicy-2016-08는 포트 443의 로드 밸런서와 연결되어 있습니다.

```
{
    "LoadBalancerDescriptions": [
        {
            ...
            "ListenerDescriptions": [
                {
                    "Listener": {
                        "InstancePort": 80,
                        "SSLCertificateId": "ARN",
                        "LoadBalancerPort": 443,
                        "Protocol": "HTTPS",
                        "InstanceProtocol": "HTTP"
                    },
                    "PolicyNames": [
                        "ELBSecurityPolicy-2016-08"
                    ]
                },
                {
                    "Listener": {
```

```
                    "InstancePort": 80,
                    "LoadBalancerPort": 80,
                    "Protocol": "HTTP",
                    "InstanceProtocol": "HTTP"
                },
                "PolicyNames": []
            }
        ],
        ...
    }
]
}
```

원할 경우, 기본 보안 정책을 사용하는 대신 로드 밸런서에 대한 SSL 보안 정책을 구성할 수 있습니다.

(선택 사항) 사전 정의 SSL 보안 정책을 사용하려면

1. 사전 정의 보안 정책의 이름 목록을 보려면 describe-load-balancer-policies 명령을 사용하세요.

   ```
   aws elb describe-load-balancer-policies
   ```

   사전 정의 보안 정책을 위한 구성에 대한 자세한 내용은 Classic Load Balancer에 대해 미리 정의된 SSL 보안 정책를 참조하십시오.
2. 이전 단계에서 설명한 사전 정의 보안 정책 중 하나를 사용하여 SSL 협상 정책을 생성하려면 아래 create-load-balancer-policy 명령을 사용하세요.

   ```
   aws elb create-load-balancer-policy --load-balancer-name my-loadbalancer
   --policy-name my-SSLNegotiation-policy --policy-type-name SSLNegotiationPolicyType
   --policy-attributes AttributeName=Reference-Security-
   Policy,AttributeValue=predefined-policy
   ```
3. (선택 사항) 정책이 생성되었는지 확인하려면 아래 describe-load-balancer-policies 명령을 사용하세요.

   ```
   aws elb describe-load-balancer-policies --load-balancer-name my-loadbalancer --
   policy-name my-SSLNegotiation-policy
   ```

   이 요청에는 정책에 대한 설명이 포함되어 있습니다.
4. 로드 밸런서 포트 443에서 정책을 활성화하려면 아래 set-load-balancer-policies-of-listener 명령을 사용하세요.

```
aws elb set-load-balancer-policies-of-listener --load-balancer-name my-loadbalancer
 --load-balancer-port 443 --policy-names my-SSLNegotiation-policy
```

> **Note**
>
> `set-load-balancer-policies-of-listener` 명령은 지정된 로드 밸런서 포트를 위한 현재 정책 세트를 지정된 정책 세트로 대체합니다. `--policy-names` 목록에는 활성화할 모든 정책이 반드시 포함되어 있어야 합니다. 현재 활성화된 정책을 누락하면 이 정책은 비활성화됩니다.

5. (선택 사항) 정책이 활성화되었는지 확인하려면 아래 describe-load-balancers 명령을 사용하세요.

```
aws elb describe-load-balancers --load-balancer-name my-loadbalancer
```

다음은 포트 443에서 정책이 활성화되었음을 보여주는 응답의 예입니다.

```
{
    "LoadBalancerDescriptions": [
        {
            ....
            "ListenerDescriptions": [
                {
                    "Listener": {
                        "InstancePort": 80,
                        "SSLCertificateId": "ARN",
                        "LoadBalancerPort": 443,
                        "Protocol": "HTTPS",
                        "InstanceProtocol": "HTTP"
                    },
                    "PolicyNames": [
                        "my-SSLNegotiation-policy"
                    ]
                },
                {
                    "Listener": {
                        "InstancePort": 80,
                        "LoadBalancerPort": 80,
                        "Protocol": "HTTP",
                        "InstanceProtocol": "HTTP"
                    },
```

```
                    "PolicyNames": []
                }
            ],
            ...
        }
    ]
}
```

사용자 지정 보안 정책을 생성할 때는 한 개 이상의 프로토콜과 한 개의 암호를 반드시 활성화해야 합니다. DSA 및 RSA 암호는 SSL 인증서를 생성하는 데 사용되는 서명 알고리즘마다 고유합니다. 이미 SSL 인증서를 가지고 있는 경우에는 인증서 생성에 사용된 암호를 반드시 활성화하십시오. 사용자 지정 정책의 이름은 `ELBSecurityPolicy-` 또는 `ELBSample-`로 시작할 수 없습니다. 이러한 접두사는 사전 정의 보안 정책의 이름으로 예약되어 있습니다.

(선택 사항) 사용자 지정 SSL 보안 정책을 사용하려면

1. 사용자 지정 보안 정책을 사용하여 SSL 협상 정책을 생성하려면 create-load-balancer-policy 명령을 사용하세요. 다음 예를 참조하십시오.

```
aws elb create-load-balancer-policy --load-balancer-name my-loadbalancer
 --policy-name my-SSLNegotiation-policy --policy-type-name
 SSLNegotiationPolicyType
 --policy-attributes AttributeName=Protocol-TLSv1.2,AttributeValue=true
 AttributeName=Protocol-TLSv1.1,AttributeValue=true
 AttributeName=DHE-RSA-AES256-SHA256,AttributeValue=true
 AttributeName=Server-Defined-Cipher-Order,AttributeValue=true
```

2. (선택 사항) 정책이 생성되었는지 확인하려면 아래 describe-load-balancer-policies 명령을 사용하세요.

```
aws elb describe-load-balancer-policies --load-balancer-name my-loadbalancer --
policy-name my-SSLNegotiation-policy
```

   이 요청에는 정책에 대한 설명이 포함되어 있습니다.

3. 로드 밸런서 포트 443에서 정책을 활성화하려면 아래 set-load-balancer-policies-of-listener 명령을 사용하세요.

```
aws elb set-load-balancer-policies-of-listener --load-balancer-name my-loadbalancer
 --load-balancer-port 443 --policy-names my-SSLNegotiation-policy
```

> **Note**
>
> `set-load-balancer-policies-of-listener` 명령은 지정된 로드 밸런서 포트를 위한 현재 정책 세트를 지정된 정책 세트로 대체합니다. `--policy-names` 목록에는 활성화할 모든 정책이 반드시 포함되어 있어야 합니다. 현재 활성화된 정책을 누락하면 이 정책은 비활성화됩니다.

4. (선택 사항) 정책이 활성화되었는지 확인하려면 아래 describe-load-balancers 명령을 사용하세요.

```
aws elb describe-load-balancers --load-balancer-name my-loadbalancer
```

다음은 포트 443에서 정책이 활성화되었음을 보여주는 응답의 예입니다.

```
{
    "LoadBalancerDescriptions": [
        {
            ....
            "ListenerDescriptions": [
                {
                    "Listener": {
                        "InstancePort": 80,
                        "SSLCertificateId": "ARN",
                        "LoadBalancerPort": 443,
                        "Protocol": "HTTPS",
                        "InstanceProtocol": "HTTP"
                    },
                    "PolicyNames": [
                        "my-SSLNegotiation-policy"
                    ]
                },
                {
                    "Listener": {
                        "InstancePort": 80,
                        "LoadBalancerPort": 80,
                        "Protocol": "HTTP",
                        "InstanceProtocol": "HTTP"
                    },
                    "PolicyNames": []
                }
            ],
            ...
```

```
        }
    ]
}
```

3단계: 백엔드 인스턴스 인증 구성(선택 사항)

백엔드 연결에서 HTTPS/SSL을 설정한 경우에는 선택에 따라 인스턴스 인증을 설정할 수 있습니다.

백엔드 인스턴스 인증을 설정할 때 퍼블릭 키 정책을 생성합니다. 그런 다음, 이러한 퍼블릭 키 정책을 사용하여 백엔드 인스턴스 인증 정책을 생성합니다. 마지막으로 HTTPS 프로토콜을 위한 인스턴스 포트로 백엔드 인스턴스 인증 정책을 설정합니다.

인스턴스가 로드 밸런서에 제시한 퍼블릭 키가 로드 밸런서를 위한 인증 정책의 퍼블릭 키와 일치하는 경우에만 로드 밸런서가 인스턴스와 통신합니다.

백엔드 인스턴스 인증을 구성하려면

1. 다음 명령을 사용하여 퍼블릭 키를 검색합니다.

```
openssl x509 -in your X509 certificate PublicKey -pubkey -noout
```

2. 퍼블릭 키 정책을 생성하려면 아래 create-load-balancer-policy 명령을 사용하세요.

```
aws elb create-load-balancer-policy --load-balancer-name my-loadbalancer --policy-
name my-PublicKey-policy \
--policy-type-name PublicKeyPolicyType --policy-attributes
 AttributeName=PublicKey,AttributeValue=MIICiTCCAfICCQD6m7oRw0uXOjANBgkqhkiG9w
 0BAQUFADCBiDELMAkGA1UEBhMCVVMxCzAJBgNVBAgTAldBMRAwDgYDVQQHEwdTZ
 WF0dGxlMQ8wDQYDVQQKEwZBbWF6b24xFDASBgNVBAsTC0lBTSBDb25zb2xlMRIw
 EAYDVQQDEwlUZXN0Q2lsYWMxHzAdBgkqhkiG9w0BCQEWEG5vb25lQGFtYXpvbi5
 jb20wHhcNMTEwNDI1MjA0NTIxWhcNMTIwNDI0MjA0NTIxWjCBiDELMAkGA1UEBh
 MCVVMxCzAJBgNVBAgTAldBMRAwDgYDVQQHEwdTZWF0dGxlMQ8wDQYDVQQKEwZBb
 WF6b24xFDASBgNVBAsTC0lBTSBDb25zb2xlMRIwEAYDVQQDEwlUZXN0Q2lsYWMx
 HzAdBgkqhkiG9w0BCQEWEG5vb25lQGFtYXpvbi5jb20wgZ8wDQYJKoZIhvcNAQE
 BBQADgY0AMIGJAoGBAMaK0dn+a4GmWIWJ21uUSfwfEvySWtC2XADZ4nB+BLYgVI
 k60CpiwsZ3G93vUEIO3IyNoH/f0wYK8m9TrDHudUZg3qX4waLG5M43q7Wgc/MbQ
 ITxOUSQv7c7ugFFDzQGBzZswY6786m86gpEIbb3OhjZnzcvQAaRHhdlQWIMm2nr
 AgMBAAEwDQYJKoZIhvcNAQEFBQADgYEAtCu4nUhVVxYUntneD9+h8Mg9q6q+auN
 KyExzyLwaxlAoo7TJHidbtS4J5iNmZgXL0FkbFFBjvSfpJIlJ00zbhNYS5f6Guo
 EDmFJl0ZxBHjJnyp378OD8uTs7fLvjx79LjSTbNYiytVbZPQUQ5Yaxu2jXnimvw
 3rrszlaEXAMPLE=
```

> **Note**
>
> `--policy-attributes`를 위한 퍼블릭 키 값을 지정하려면 퍼블릭 키의 첫 번째 줄과 마지막 줄("-----BEGIN PUBLIC KEY-----"를 포함하고 있는 줄과 "-----END PUBLIC KEY-----"를 포함하고 있는 줄)을 삭제하십시오. AWS CLI 는에서 공백 문자를 허용하지 않습니다`--policy-attributes`.

3. `my-PublicKey-policy`을 사용해 백엔드 인스턴스 인증 정책을 생성하려면 아래 create-load-balancer-policy 명령을 사용하세요.

```
aws elb create-load-balancer-policy --load-balancer-name my-
loadbalancer --policy-name my-authentication-policy --policy-type-
name BackendServerAuthenticationPolicyType --policy-attributes
 AttributeName=PublicKeyPolicyName,AttributeValue=my-PublicKey-policy
```

선택에 따라 여러 개의 퍼블릭 키 정책을 사용할 수 있습니다. 로드 밸런서는 한 번에 하나씩 모든 키를 시도합니다. 인스턴스가 제시한 퍼블릭 키가 이러한 퍼블릭 키 중 하나와 일치하는 경우에 인스턴스가 인증이 됩니다.

4. `my-authentication-policy`을 HTTPS를 위한 인스턴스 포트로 설정하려면 아래 set-load-balancer-policies-for-backend-server 명령을 사용하세요. 이 예제에서 인스턴스 포트는 포트 443 입니다.

```
aws elb set-load-balancer-policies-for-backend-server --load-balancer-name my-
loadbalancer --instance-port 443 --policy-names my-authentication-policy
```

5. (선택 사항) 로드 밸런서를 위한 모든 정책의 목록을 보려면 아래 describe-load-balancer-policies 명령을 사용하세요.

```
aws elb describe-load-balancer-policies --load-balancer-name my-loadbalancer
```

6. (선택 사항) 정책의 세부 정보를 확인하려면 아래 describe-load-balancer-policies 명령을 사용하세요.

```
aws elb describe-load-balancer-policies --load-balancer-name my-loadbalancer --
policy-names my-authentication-policy
```

4단계: 상태 확인 구성(선택 사항)

Elastic Load Balancing은 구성한 상태 확인에 따라 등록된 EC2 인스턴스 각각의 상태를 정기적으로 확인합니다. Elastic Load Balancing은 비정상 상태인 인스턴스를 발견하면 해당 인스턴스로의 트래픽 전송을 중단하고 정상 상태인 인스턴스로 트래픽을 라우팅합니다. 자세한 내용은 Classic Load Balancer의 인스턴스 상태 확인 단원을 참조하십시오.

로드 밸런서를 생성할 때 Elastic Load Balancing은 상태 확인을 위한 기본 설정을 사용합니다. 원할 경우, 기본 설정을 사용하는 대신 로드 밸런서에 대한 상태 확인 구성을 변경할 수 있습니다.

인스턴스에 대한 상태 확인을 구성하려면

아래 configure-health-check 명령을 사용하세요.

```
aws elb configure-health-check --load-balancer-name my-loadbalancer --health-check
 Target=HTTP:80/ping,Interval=30,UnhealthyThreshold=2,HealthyThreshold=2,Timeout=3
```

다음은 응답의 예입니다.

```
{
    "HealthCheck": {
        "HealthyThreshold": 2,
        "Interval": 30,
        "Target": "HTTP:80/ping",
        "Timeout": 3,
        "UnhealthyThreshold": 2
    }
}
```

5단계: EC2 인스턴스 등록

로드 밸런서를 생성하고 난 후에는 반드시 로드 밸런서에 EC2 인스턴스를 등록해야 합니다. 단일 가용 영역 또는 같은 리전 내 여러 가용 영역의 EC2 인스턴스를 로드 밸런서로 선택할 수 있습니다. 자세한 내용은 Classic Load Balancer를 위한 등록된 인스턴스 단원을 참조하십시오.

아래와 같이 register-instances-with-load-balancer 명령을 사용하세요.

```
aws elb register-instances-with-load-balancer --load-balancer-name my-loadbalancer --
instances i-4f8cf126 i-0bb7ca62
```

다음은 응답의 예입니다.

```
{
    "Instances": [
        {
            "InstanceId": "i-4f8cf126"
        },
        {
            "InstanceId": "i-0bb7ca62"
        }
    ]
}
```

6단계: 인스턴스 확인

등록된 인스턴스 중 하나라도 `InService` 상태가 되면 그 즉시 로드 밸런서를 사용할 수 있습니다.

새로 등록된 EC2 인스턴스의 세부 정보를 확인하려면 아래 describe-instance-health 명령을 사용하세요.

```
aws elb describe-instance-health  --load-balancer-name my-loadbalancer --
instances i-4f8cf126 i-0bb7ca62
```

다음은 응답의 예입니다.

```
{
    "InstanceStates": [
        {
            "InstanceId": "i-4f8cf126",
            "ReasonCode": "N/A",
            "State": "InService",
            "Description": "N/A"
        },
        {
            "InstanceId": "i-0bb7ca62",
            "ReasonCode": "Instance",
            "State": "OutOfService",
            "Description": "Instance registration is still in progress"
        }
    ]
}
```

인스턴스에 대한 `State` 필드가 `OutOfService`인 경우에는 인스턴스가 여전히 등록 중이기 때문일 수 있습니다. 자세한 내용은 Classic Load Balancer 문제 해결: 인스턴스 등록 단원을 참조하십시오.

하나 이상의 인스턴스 상태가 `InService`여야 로드 밸런서를 테스트할 수 있습니다. 로드 밸런서를 테스트하려면 로드 밸런서의 DNS 이름을 복사해서 인터넷에 연결된 웹 브라우저의 주소 필드에 붙여 넣습니다. 로드 밸런서가 작동 중인 경우에는 HTTP 서버의 기본 페이지가 표시됩니다.

7단계: 로드 밸런서 삭제(선택 사항)

로드 밸런서를 삭제하면 연결된 EC2 인스턴스가 자동으로 등록 취소됩니다. 로드 밸런서가 삭제되면 그 즉시 로드 밸런서에 대한 요금 발생이 중지됩니다. 그러나 EC2 인스턴스가 계속 실행되면 요금이 계속 발생합니다.

로드 밸런서를 삭제하려면 아래 delete-load-balancer 명령을 사용하세요.

```
aws elb delete-load-balancer --load-balancer-name my-loadbalancer
```

EC2 인스턴스를 중지하려면 stop-instances 명령을 사용하세요. EC2 인스턴스를 종료하려면 stop-instances 명령을 사용하세요.

Classic Load Balancer를 위한 HTTPS 리스너 구성

리스너는 연결 요청을 확인하는 프로세스입니다. 리스너는 프런트 엔드(클라이언트에서 로드 밸런서) 연결을 위한 프로토콜 및 포트와 백엔드(로드 밸런서에서 인스턴스) 연결을 위한 프로토콜 및 포트로 구성됩니다. Elastic Load Balancing에서 지원되는 포트, 프로토콜 및 리스너 구성에 대한 자세한 내용은 Classic Load Balancer의 리스너 섹션을 참조하세요.

포트 80에서 HTTP 요청을 수락하는 리스너를 가진 로드 밸런서의 경우에는 포트 443에서 HTTPS 요청을 수락하는 리스너를 추가할 수 있습니다. HTTPS 리스너가 포트 80의 인스턴스로 요청을 전송하도록 지정한 경우에는 로드 밸런서가 SSL 요청을 종료하고 로드 밸런서에서 인스턴스로의 통신이 암호화되지 않습니다. HTTPS 리스너가 포트 443의 인스턴스로 요청을 전송하는 경우에는 로드 밸런서에서 인스턴스로의 통신이 암호화됩니다.

로드 밸런서가 인스턴스와의 통신을 위해 암호화된 연결을 사용하는 경우에는 선택에 따라 인스턴스 인증을 활성화할 수 있습니다. 따라서 퍼블릭 키가 통신 용도로 로드 밸런서에 지정한 키와 일치하는 경우에만 로드 밸런서가 인스턴스와 통신을 하도록 할 수 있습니다.

새로운 HTTPS 리스너 생성에 대한 자세한 내용은 HTTPS 리스너를 통해 Classic Load Balancer를 생성를 참조하십시오.

내용

사전 조건

HTTPS 리스너를 위해 HTTPS 지원을 활성화하려면 로드 밸런서에 SSL 서버 인증서를 반드시 배포해야 합니다. 로드 밸런서는 이 인증서를 사용해 연결을 종료하고 인스턴스로 전송하기 전에 요청을 해독합니다. SSL 인증서가 없는 경우에는 하나를 생성할 수 있습니다. 자세한 내용은 Classic Load Balancer를 위한 SSL/TLS 인증서 단원을 참조하십시오.

콘솔을 사용하여 HTTPS 리스너 추가

기존 로드 밸런서에 HTTPS 리스너를 추가할 수 있습니다.

콘솔을 사용하여 로드 밸런서에 HTTPS 리스너를 추가하려면

1. https://console.aws.amazon.com/ec2/에서 Amazon EC2 콘솔을 엽니다.
2. 탐색 창의 Load Balancing 아래에서 로드 밸런서를 선택합니다.
3. 로드 밸런서 이름을 선택하여 세부 정보 페이지를 엽니다.
4. 리스너 탭에서 리스너 관리를 선택합니다.
5. 리스너 관리 페이지의 리스너 섹션에서 리스너 추가를 선택합니다.
6. 리스너 프로토콜에서 HTTPS를 선택합니다.

   > **Important**
   >
   > 인스턴스 프로토콜은 HTTP로 기본 설정되어 있습니다. 백엔드 인스턴스 인증을 설정하고 싶으면 인스턴스 프로토콜을 HTTPS로 변경합니다.

7. 보안 정책의 경우 최신 사전 정의 보안 정책을 사용하는 것이 좋습니다. 다른 사전 정의 보안 정책을 사용하거나 사용자 지정 정책을 생성해야 하는 경우에는 SSL 협상 구성 업데이트를 참조하십시오.
8. 기본 SSL 인증서에서 편집을 선택하고 다음 중 하나를 수행합니다.
   - 를 사용하여 인증서를 생성하거나 가져온 경우 ACM에서 AWS Certificate Manager를 선택하고 목록에서 인증서를 선택한 다음 변경 사항 저장을 선택합니다.

> **Note**
>
> 이 옵션은 을 지원하는 리전에서만 사용할 수 있습니다 AWS Certificate Manager

- IAM을 사용하여 인증서를 가져온 경우 IAM에서를 선택하고 목록에서 인증서를 선택한 다음 변경 사항 저장을 선택합니다.
- ACM으로 가져올 SSL 인증서가 있는 경우 가져오기와 ACM으로를 선택합니다. 인증서 프라이빗 키에서 PEM 인코딩 프라이빗 키 파일의 콘텐츠를 복사해 붙여넣습니다. 인증서 본문에서 PEM 인코딩 퍼블릭 키 인증서 파일의 콘텐츠를 복사해 붙여넣습니다. 자체 서명 인증서를 사용하고 있지 않고 브라우저가 인증서를 묵시적으로 수락하는 것이 중요하지 않다면 인증서 체인 - 선택 사항에 PEM 인코딩 인증서 체인 파일의 콘텐츠를 복사해 붙여넣습니다.
- 가져올 SSL 인증서가 있지만 이 리전에서 ACM이 지원되지 않는 경우 가져오기와 IAM으로를 선택합니다. 인증서 이름에 인증서의 이름을 입력합니다. 인증서 프라이빗 키에서 PEM 인코딩 프라이빗 키 파일의 콘텐츠를 복사해 붙여넣습니다. 인증서 본문에서 PEM 인코딩 퍼블릭 키 인증서 파일의 콘텐츠를 복사해 붙여넣습니다. 자체 서명 인증서를 사용하고 있지 않고 브라우저가 인증서를 묵시적으로 수락하는 것이 중요하지 않다면 인증서 체인 - 선택 사항에 PEM 인코딩 인증서 체인 파일의 콘텐츠를 복사해 붙여넣습니다.
- 변경 사항 저장을 선택합니다.

9. 쿠키 고정에서 기본값은 비활성화됨입니다. 이를 변경하려면 편집을 선택합니다. 로드 밸런서에서 생성을 선택한 경우 만료 기간을 지정해야 합니다. 애플리케이션에서 생성을 선택한 경우 쿠키 이름을 지정해야 합니다. 선택을 완료한 후 변경 사항 저장을 선택합니다.
10. (선택 사항) 리스너 추가를 선택하여 리스너를 추가합니다.
11. 변경 사항 저장을 선택하여 방금 구성한 리스너를 추가합니다.
12. (선택 사항) 기존 로드 밸런서에 대한 백엔드 인스턴스 인증을 설정하려면 AWS CLI 또는 API를 사용해야 합니다.이 작업은 콘솔을 사용하여 지원되지 않기 때문입니다. 자세한 내용은 백엔드 인스턴스 인증 구성을 참조하십시오.

를 사용하여 HTTPS 리스너 추가 AWS CLI

기존 로드 밸런서에 HTTPS 리스너를 추가할 수 있습니다.

를 사용하여 로드 밸런서에 HTTPS 리스너를 추가하려면 AWS CLI

1. SSL 인증서의 Amazon 리소스 이름(ARN)을 가져옵니다. 다음 예를 참조하십시오.

ACM

```
arn:aws:acm:region:123456789012:certificate/12345678-1234-1234-1234-123456789012
```

IAM

```
arn:aws:iam::123456789012:server-certificate/my-server-certificate
```

2. 포트 443에서 HTTPS 요청을 수락하고 HTTP를 사용하여 포트 80의 인스턴스로 요청을 전송하는 로드 밸런서에 리스너를 추가하려면 아래 create-load-balancer-listeners 명령을 사용하세요.

```
aws elb create-load-balancer-listeners --load-balancer-name my-load-balancer --
listeners
 Protocol=HTTPS,LoadBalancerPort=443,InstanceProtocol=HTTP,InstancePort=80,SSLCertificateId
```

백엔드 인스턴스 인증을 설정하고 싶으면 아래 명령을 사용해 포트 443에서 HTTPS 요청을 수락하고 HTTPS를 사용하여 포트 443의 인스턴스로 요청을 전송하십시오.

```
aws elb create-load-balancer-listeners --load-balancer-name my-load-balancer --
listeners
 Protocol=HTTPS,LoadBalancerPort=443,InstanceProtocol=HTTPS,InstancePort=443,SSLCertificate
```

3. (선택 사항) 로드 밸런서의 업데이트된 세부 정보를 확인하려면 아래 describe-load-balancers 명령을 사용하세요.

```
aws elb describe-load-balancers --load-balancer-name my-load-balancer
```

다음은 응답의 예입니다.

```
{
    "LoadBalancerDescriptions": [
        {
            ...
            "ListenerDescriptions": [
                {
                    "Listener": {
                        "InstancePort": 80,
                        "SSLCertificateId": "ARN",
                        "LoadBalancerPort": 443,
```

```
                    "Protocol": "HTTPS",
                    "InstanceProtocol": "HTTP"
                },
                "PolicyNames": [
                    "ELBSecurityPolicy-2016-08"
                ]
            },
            {
                "Listener": {
                    "InstancePort": 80,
                    "LoadBalancerPort": 80,
                    "Protocol": "HTTP",
                    "InstanceProtocol": "HTTP"
                },
                "PolicyNames": []
            }
        ],
        ...
    }
  ]
}
```

4. (선택 사항) 기본 보안 정책을 사용해 HTTPS 리스너를 생성했습니다. 다른 사전 정의 보안 정책이나 사용자 정의 보안 정책을 지정하고 싶으면 create-load-balancer-policy 및 set-load-balancer-policies-of-listener 명령을 사용하세요. 자세한 내용은 를 사용하여 SSL 협상 구성 업데이트 AWS CLI 단원을 참조하십시오.
5. (선택 사항) 백 엔드 인스턴스 인증을 설정하려면 set-load-balancer-policies-for-backend-server 명령을 사용하세요. 자세한 내용은 백엔드 인스턴스 인증 구성을 참조하십시오.

Classic Load Balancer를 위한 SSL 인증서 교체

HTTPS 리스너가 있다면 리스너를 생성할 때 로드 밸런서에 SSL 서버 인증서를 배포했을 것입니다. 각 인증서에는 유효 기간이 있습니다. 유효 기간이 끝나기 전에 인증서를 갱신 또는 교체해야 합니다.

에서 제공하고 로드 밸런서에 AWS Certificate Manager 배포된 인증서는 자동으로 갱신할 수 있습니다. ACM은 인증서가 만료되기 전에 갱신을 시도합니다. 자세한 내용은 AWS Certificate Manager 사용 설명서에서 관리형 갱신을 참조하세요. ACM에 인증서를 가져온 경우에는 인증서의 만료일을 반드시 모니터링해서 만료되기 전에 인증서를 갱신해야 합니다. 자세한 내용은 AWS Certificate Manager 사용 설명서에서 인증서 가져오기를 참조하세요. 로드 밸런서에 배포된 인증서가 갱신된 후에는 새로운 요청에서 갱신된 인증서가 사용됩니다.

인증서를 교체하려면 먼저 현재 인증서를 생성할 때 사용했던 것과 동일한 단계를 거쳐 새로운 인증서를 생성해야 합니다. 그래야만 인증서를 교체할 수 있습니다. 로드 밸런서에 배포된 인증서가 교체된 후에는 새로운 요청에서 새 인증서가 사용됩니다.

인증서를 갱신 또는 교체해도 로드 밸런서 노드에 이미 수신되어 정상 상태 대상으로 라우팅이 보류 중인 요청에는 영향을 미치지 않습니다.

목차

콘솔을 사용하여 SSL 인증서 교체

로드 밸런서에 배포된 인증서를 ACM이 제공한 인증서나 IAM에 업로드된 인증서로 교체할 수 있습니다.

콘솔을 사용하여 HTTPS 로드 밸런서를 위한 SSL 인증서를 교체하려면

1. https://console.aws.amazon.com/ec2/에서 Amazon EC2 콘솔을 엽니다.
2. 탐색 창의 Load Balancing 아래에서 로드 밸런서를 선택합니다.
3. 로드 밸런서 이름을 선택하여 세부 정보 페이지를 엽니다.
4. 리스너 탭에서 리스너 관리를 선택합니다.
5. 리스너 관리 페이지에서 업데이트할 리스너를 찾고, 기본 SSL 인증서에서 편집을 선택하고 다음 중 하나를 수행합니다.

   - 를 사용하여 인증서를 생성하거나 가져온 경우 ACM에서 AWS Certificate Manager를 선택하고 목록에서 인증서를 선택한 다음 변경 사항 저장을 선택합니다.

     > **Note**
     >
     > 이 옵션은 을 지원하는 리전에서만 사용할 수 있습니다 AWS Certificate Manager

   - IAM을 사용하여 인증서를 가져온 경우 IAM에서를 선택하고 목록에서 인증서를 선택한 다음 변경 사항 저장을 선택합니다.
   - ACM으로 가져올 SSL 인증서가 있는 경우 가져오기와 ACM으로를 선택합니다. 인증서 프라이빗 키에서 PEM 인코딩 프라이빗 키 파일의 콘텐츠를 복사해 붙여넣습니다. 인증서 본문에서 PEM 인코딩 퍼블릭 키 인증서 파일의 콘텐츠를 복사해 붙여넣습니다. 자체 서명 인증서를 사용

하고 있지 않고 브라우저가 인증서를 묵시적으로 수락하는 것이 중요하지 않다면 인증서 체인 - 선택 사항에 PEM 인코딩 인증서 체인 파일의 콘텐츠를 복사해 붙여넣습니다.

- 가져올 SSL 인증서가 있지만 이 리전에서 ACM이 지원되지 않는 경우 가져오기와 IAM으로를 선택합니다. 인증서 이름에 인증서의 이름을 입력합니다. 인증서 프라이빗 키에서 PEM 인코딩 프라이빗 키 파일의 콘텐츠를 복사해 붙여넣습니다. 인증서 본문에서 PEM 인코딩 퍼블릭 키 인증서 파일의 콘텐츠를 복사해 붙여넣습니다. 자체 서명 인증서를 사용하고 있지 않고 브라우저가 인증서를 묵시적으로 수락하는 것이 중요하지 않다면 인증서 체인 - 선택 사항에 PEM 인코딩 인증서 체인 파일의 콘텐츠를 복사해 붙여넣습니다.
- 변경 사항 저장을 선택합니다.

를 사용하여 SSL 인증서 교체 AWS CLI

로드 밸런서에 배포된 인증서를 ACM이 제공한 인증서나 IAM에 업로드된 인증서로 교체할 수 있습니다.

ACM에서 제공한 인증서로 SSL 인증서를 교체하려면

1. request-certificate 명령을 사용하여 새 인증서를 요청합니다.

```
aws acm request-certificate --domain-name www.example.com
```

2. 인증서를 설정하려면 아래 set-load-balancer-listener-ssl-certificate 명령을 사용하세요.

```
aws elb set-load-balancer-listener-ssl-certificate --load-balancer-
name my-load-balancer --load-balancer-port 443 --ssl-certificate-id
 arn:aws:acm:region:123456789012:certificate/12345678-1234-1234-1234-123456789012
```

IAM으로 업로드한 인증서로 SSL 인증서를 교체하려면

1. SSL 인증서가 있지만 업로드를 하지 않은 경우에는 IAM 사용 설명서의 서버 인증서 업로드를 참조하세요.
2. 인증서의 ARN을 얻으려면 아래get-server-certificate 명령을 사용하세요.

```
aws iam get-server-certificate --server-certificate-name my-new-certificate
```

3. 인증서를 설정하려면 아래 set-load-balancer-listener-ssl-certificate 명령을 사용하세요.

```
aws elb set-load-balancer-listener-ssl-certificate --load-balancer-
name my-load-balancer --load-balancer-port 443 --ssl-certificate-id
 arn:aws:iam::123456789012:server-certificate/my-new-certificate
```

Classic Load Balancer의 SSL 협상 구성 업데이트

Elastic Load Balancing은 클라이언트와 로드 밸런서 간의 연결을 협상하는 데 사용할 SSL 협상 구성이 사전 정의된 보안 정책을 제공합니다. 리스너용 HTTPS/SSL 프로토콜을 사용하고 있는 경우에는 사전 정의 보안 정책 중 하나를 사용하거나 자체적으로 보안 정책을 지정할 수 있습니다.

보안 정책에 대한 자세한 내용은 Classic Load Balancer를 위한 SSL 협상 구성를 참조하십시오. Elastic Load Balancing이 제공한 보안 정책의 구성에 대한 자세한 내용은 Classic Load Balancer에 대해 미리 정의된 SSL 보안 정책 섹션을 참조하세요.

보안 정책과 연계하지 않고 HTTPS/SSL 리스너를 생성하는 경우, Elastic Load Balancing은 로드 밸런서와 함께 기본 사전 정의인 `ELBSecurityPolicy-2016-08`를 연계합니다.

원할 경우, 사용자 지정 구성을 생성할 수 있습니다 로드 밸런서 구성을 업그레이드하기 전에 보안 정책을 테스트하는 것이 좋습니다.

다음 예제는 HTTPS/SSL 리스너를 위한 SSL 협상 구성을 업데이트하는 방법을 보여줍니다. 변경을 해도 로드 밸런서 노드에 이미 수신되어 정상 상태 대상으로 라우팅이 보류 중인 요청에는 영향을 미치지 않지만, 업데이트된 구성은 수신된 새 요청에서 사용됩니다.

목차

콘솔을 사용하여 SSL 협상 구성 업데이트

기본적으로 Elastic Load Balancing은 로드 밸런서에 최신 사전 정의 정책을 연결합니다. 새로운 사전 정의 정책이 추가되면 새로운 사전 정의 정책을 사용하도록 로드 밸런서를 업데이트하는 것이 좋습니다. 아니면 다른 사전 정의 보안 정책을 선택하거나 사용자 지정 정책을 생성할 수 있습니다.

콘솔을 사용하여 HTTPS/SSL 로드 밸런서의 SSL 협상 구성을 업데이트하려면

1. https://console.aws.amazon.com/ec2/에서 Amazon EC2 콘솔을 엽니다.

2. 탐색 창의 Load Balancing 아래에서 로드 밸런서를 선택합니다.
3. 로드 밸런서 이름을 선택하여 세부 정보 페이지를 엽니다.
4. 리스너 탭에서 리스너 관리를 선택합니다.
5. 리스너 관리 페이지에서 업데이트할 리스너를 찾고, 보안 정책의 편집에서 다음 옵션 중 하나를 사용하여 보안 정책을 선택합니다.
   - 기본 정책인 ELBSecurityPolicy-2016-08을 유지하고 변경 사항 저장을 선택합니다.
   - 기본값 이외의 사전 정의된 정책을 선택하고, 변경 사항 저장을 선택합니다.
   - 사용자 지정을 선택하고 아래와 같이 한 개 이상의 프로토콜과 한 개의 암호를 활성화합니다.
     a. [SSL Protocols]에서 활성화할 프로토콜을 한 개 이상 선택합니다.
     b. [SSL 옵션(SSL Options)]에서 [서버 순서 기본 설정(Server Order Preference)]을 선택하여 SSL 협상 시 Classic Load Balancer에 대해 미리 정의된 SSL 보안 정책에 나열된 순서를 사용합니다.
     c. [SSL Ciphers]에서 활성화할 암호를 한 개 이상 선택합니다. SSL 인증서가 이미 있는 경우에는 인증서를 생성하는 데 사용된 암호를 반드시 활성화해야 합니다. DSA 및 RSA 암호는 서명 알고리즘마다 고유하기 때문입니다.
     d. 변경 사항 저장을 선택합니다.

를 사용하여 SSL 협상 구성 업데이트 AWS CLI

기본적인 사전 정의 보안 정책인 `ELBSecurityPolicy-2016-08`, 다른 사전 정의 보안 정책 또는 사용자 지정 보안 정책을 사용할 수 있습니다.

(선택 사항) 사전 정의된 SSL 보안 정책을 사용하려면

1. Elastic Load Balancing이 제공하는 사전 정의 보안 정책의 목록을 보려면 아래 describe-load-balancer-policies 명령을 사용하세요. 사용 중인 운영 체제와 셸에 따라 사용하는 구문도 달라집니다.

   Linux

   ```
   aws elb describe-load-balancer-policies --query 'PolicyDescriptions[?
   PolicyTypeName==`SSLNegotiationPolicyType`].{PolicyName:PolicyName}' --output table
   ```

   Windows

```
aws elb describe-load-balancer-policies --query "PolicyDescriptions[?
PolicyTypeName==`SSLNegotiationPolicyType`].{PolicyName:PolicyName}" --output table
```

다음은 예제 출력입니다.

```
-----------------------------------------
|      DescribeLoadBalancerPolicies      |
+----------------------------------------+
|               PolicyName               |
+----------------------------------------+
|  ELBSecurityPolicy-2016-08             |
|  ELBSecurityPolicy-TLS-1-2-2017-01     |
|  ELBSecurityPolicy-TLS-1-1-2017-01     |
|  ELBSecurityPolicy-2015-05             |
|  ELBSecurityPolicy-2015-03             |
|  ELBSecurityPolicy-2015-02             |
|  ELBSecurityPolicy-2014-10             |
|  ELBSecurityPolicy-2014-01             |
|  ELBSecurityPolicy-2011-08             |
|  ELBSample-ELBDefaultCipherPolicy      |
|  ELBSample-OpenSSLDefaultCipherPolicy  |
+----------------------------------------+
```

정책에 어떤 암호가 활성화되어 있는지 확인하려면 아래 명령을 사용하십시오.

```
aws elb describe-load-balancer-policies --policy-names ELBSecurityPolicy-2016-08 --
output table
```

사전 정의 보안 정책을 위한 구성에 대한 자세한 내용은 Classic Load Balancer에 대해 미리 정의된 SSL 보안 정책를 참조하십시오.

2. 이전 단계에서 설명한 사전 정의 보안 정책 중 하나를 사용하여 SSL 협상 정책을 생성하려면 아래 create-load-balancer-policy 명령을 사용하세요. 예를 들어 아래 명령은 기본적인 사전 정의 보안 정책을 사용합니다.

```
aws elb create-load-balancer-policy --load-balancer-name my-loadbalancer
--policy-name my-SSLNegotiation-policy  --policy-type-name SSLNegotiationPolicyType
--policy-attributes AttributeName=Reference-Security-
Policy,AttributeValue=ELBSecurityPolicy-2016-08
```

로드 밸런서에 대한 정책 수의 한도를 초과한 경우 delete-load-balancer-policy 명령을 사용하여 사용하지 않는 정책을 삭제하세요.

3. (선택 사항) 정책이 생성되었는지 확인하려면 아래 describe-load-balancer-policies 명령을 사용하세요.

```
aws elb describe-load-balancer-policies --load-balancer-name my-loadbalancer --
policy-name my-SSLNegotiation-policy
```

이 요청에는 정책에 대한 설명이 포함되어 있습니다.

4. 로드 밸런서 포트 443에서 정책을 활성화하려면 아래 set-load-balancer-policies-of-listener 명령을 사용하세요.

```
aws elb set-load-balancer-policies-of-listener --load-balancer-name my-loadbalancer
 --load-balancer-port 443 --policy-names my-SSLNegotiation-policy
```

> **Note**
>
> set-load-balancer-policies-of-listener 명령은 지정된 로드 밸런서 포트에 대한 현재 정책 세트를 지정된 정책 세트로 교체합니다. --policy-names 목록에는 활성화할 모든 정책이 반드시 포함되어 있어야 합니다. 현재 활성화된 정책을 누락하면 이 정책은 비활성화됩니다.

5. (선택 사항) 새로운 정책이 로드 밸런서 포트에서 활성화되었는지 확인하려면 아래 describe-load-balancers 명령을 사용하세요.

```
aws elb describe-load-balancers --load-balancer-name my-loadbalancer
```

다음은 포트 443에서 정책이 활성화되었음을 보여주는 응답의 예입니다.

```
...
  {
      "Listener": {
          "InstancePort": 443,
          "SSLCertificateId": "ARN",
          "LoadBalancerPort": 443,
          "Protocol": "HTTPS",
          "InstanceProtocol": "HTTPS"
```

```
        },
        "PolicyNames": [
            "my-SSLNegotiation-policy"
        ]
    }
...
```

사용자 지정 보안 정책을 생성할 때는 한 개 이상의 프로토콜과 한 개의 암호를 반드시 활성화해야 합니다. DSA 및 RSA 암호는 SSL 인증서를 생성하는 데 사용되는 서명 알고리즘마다 고유합니다. 이미 SSL 인증서를 가지고 있는 경우에는 인증서 생성에 사용된 암호를 반드시 활성화하십시오. 사용자 지정 정책의 이름은 `ELBSecurityPolicy-` 또는 `ELBSample-`로 시작할 수 없습니다. 이러한 접두사는 사전 정의 보안 정책의 이름으로 예약되어 있습니다.

사용자 지정 SSL 보안 정책을 사용하려면

1. 사용자 지정 보안 정책을 사용하여 SSL 협상 정책을 생성하려면 create-load-balancer-policy 명령을 사용하세요. 다음 예를 참조하십시오.

   ```
   aws elb create-load-balancer-policy --load-balancer-name my-loadbalancer
    --policy-name my-SSLNegotiation-policy --policy-type-name
    SSLNegotiationPolicyType
    --policy-attributes AttributeName=Protocol-TLSv1.2,AttributeValue=true
    AttributeName=Protocol-TLSv1.1,AttributeValue=true
    AttributeName=DHE-RSA-AES256-SHA256,AttributeValue=true
    AttributeName=Server-Defined-Cipher-Order,AttributeValue=true
   ```

   로드 밸런서에 대한 정책 수의 한도를 초과한 경우 delete-load-balancer-policy 명령을 사용하여 사용하지 않는 정책을 삭제하세요.

2. (선택 사항) 정책이 생성되었는지 확인하려면 아래 describe-load-balancer-policies 명령을 사용하세요.

   ```
   aws elb describe-load-balancer-policies --load-balancer-name my-loadbalancer --
   policy-name my-SSLNegotiation-policy
   ```

   이 요청에는 정책에 대한 설명이 포함되어 있습니다.

3. 로드 밸런서 포트 443에서 정책을 활성화하려면 아래 set-load-balancer-policies-of-listener 명령을 사용하세요.

```
aws elb set-load-balancer-policies-of-listener --load-balancer-name my-loadbalancer
 --load-balancer-port 443 --policy-names my-SSLNegotiation-policy
```

> **Note**
>
> `set-load-balancer-policies-of-listener` 명령은 지정된 로드 밸런서 포트에 대한 현재 정책 세트를 지정된 정책 세트로 교체합니다. `--policy-names` 목록에는 활성화할 모든 정책이 반드시 포함되어 있어야 합니다. 현재 활성화된 정책을 누락하면 이 정책은 비활성화됩니다.

4. (선택 사항) 새로운 정책이 로드 밸런서 포트에서 활성화되었는지 확인하려면 아래 describe-load-balancers 명령을 사용하세요.

```
aws elb describe-load-balancers --load-balancer-name my-loadbalancer
```

다음은 포트 443에서 정책이 활성화되었음을 보여주는 응답의 예입니다.

```
...
  {
      "Listener": {
          "InstancePort": 443,
          "SSLCertificateId": "ARN",
          "LoadBalancerPort": 443,
          "Protocol": "HTTPS",
          "InstanceProtocol": "HTTPS"
      },
      "PolicyNames": [
          "my-SSLNegotiation-policy"
      ]
  }
...
```

Classic Load Balancer를 위한 등록된 인스턴스

Classic Load Balancer를 생성하고 난 후에는 반드시 로드 밸런서에 EC2 인스턴스를 등록해야 합니다. 단일 가용 영역 또는 같은 리전 내 여러 가용 영역의 EC2 인스턴스를 로드 밸런서로 선택할 수 있습니다. Elastic Load Balancing은 등록된 EC2 인스턴스에 대해 정기적으로 상태 확인을 수행하고, 등록된 정상 상태의 EC2 인스턴스에서 로드 밸런서의 DNS 이름으로 들어오는 요청을 자동 분산합니다.

목차

인스턴스 모범 사례

- 로드 밸런서가 리스너 포트 및 상태 확인 포트에서 인스턴스와 통신을 할 수 있는지 확인해야 합니다. 자세한 내용은 Classic Load Balancer 보안 그룹 구성 단원을 참조하십시오. 인스턴스에 대한 보안 그룹은 로드 밸런서를 위한 각 서브넷의 두 포트 모두에서 양방향으로 트래픽을 허용해야 합니다.
- Apache나 IIS(인터넷 정보 서비스) 같은 웹 서버를 로드 밸런서에 등록 예정인 모든 인스턴스에 설치합니다.
- HTTP 및 HTTPS 리스너의 경우, 로드 밸런서가 여러 클라이언트 요청에서 인스턴스에 대한 연결을 재사용할 수 있도록 EC2 인스턴스에서 연결 유지 옵션을 활성화하는 것이 좋습니다. 이렇게 하면 웹 서버에 대한 부하를 줄이고 로드 밸런서의 처리량을 늘릴 수 있습니다. 로드 밸런서가 인스턴스에 대한 연결을 책임지고 종료할 수 있으려면 연결 유지 제한 시간이 60초 이상이 되어야 합니다.
- Elastic Load Balancing은 경로 MTU(최대 연결 단위) 검색을 지원합니다. 경로 MTU 검색이 올바르게 작동할 수 있으려면 인스턴스에 대한 보안 그룹이 ICMP 조각화가 필요한(유형 3, 코드 4) 메시지를 허용하도록 해야 합니다. 자세한 내용은 Amazon EC2 사용 설명서의 경로 MTU 검색을 참조하세요.

VPC 관련 권장 사항

Virtual private cloud(VPC)

2014년 AWS 계정 이전에를 생성하지 않은 경우 각 리전에 기본 VPC가 있습니다. 로드 밸런서에 기본 VPC(있는 경우)를 사용하거나 새 VPC를 생성할 수 있습니다. 자세한 내용은 Amazon VPC 사용 설명서를 참조하세요.

로드 밸런서를 위한 서브넷

로드 밸런서가 적절하게 확장 가능하도록 로드 밸런서를 위한 각 서브넷에 최소 한 개의 /27비트 마스크(예: 10.0.0.0/27)를 가진 CIDR 블록이 있고 사용 가능한 IP 주소가 8개 이상 있는지 확인합니다. 로드 밸런서는 이러한 IP 주소를 사용하여 인스턴스에 대한 연결을 설정하고 필요한 경우 스케일 아웃합니다. IP 주소가 충분하지 않은 경우 로드 밸런서를 확장할 수 없어 용량 부족으로 인해 503 오류가 발생할 수 있습니다.

인스턴스를 시작하고자 하는 각 가용 영역에서 서브넷을 생성합니다. 사용 사례에 따라 퍼블릿 서브넷이나 프라이빗 서브넷에서, 또는 이 둘을 조합한 환경에서 인스턴스를 시작할 수 있습니다. 퍼블릭 서브넷은 인터넷 게이트웨이로 가는 경로를 가지고 있습니다. 기본 VPC는 가용 영역별로 한 개의 퍼블릭 서브넷을 기본적으로 가지고 있습니다.

로드 밸런서를 생성할 때 로드 밸런서에 퍼블릿 서브넷을 한 개 이상 추가해야 합니다. 인스턴스가 프라이빗 서브넷에 있는 경우에는 인스턴스에서 서브넷과 동일한 가용 영역에 퍼브릭 서브넷을 생성하고, 로드 밸런서에 이러한 퍼블릿 서브넷을 추가합니다.

네트워크 ACL

VPC를 위한 네트워크 ACL은 리스너 포트와 상태 확인 포트에서 양방향으로 트래픽을 허용해야 합니다. 자세한 내용은 Classic Load Balancer 인스턴스의 네트워크 ACL 단원을 참조하십시오.

Classic Load Balancer에 인스턴스 등록

EC2 인스턴스를 등록하면 로드 밸런서에 인스턴스가 추가됩니다. 로드 밸런서가 활성화된 가용 영역에서 등록된 인스턴스의 상태를 지속적으로 모니터링하여 정상 상태의 인스턴스로 트래픽을 라우팅합니다. 인스턴스에 대한 요구가 증가하면 이를 처리하기 위해 로드 밸런서에 추가로 인스턴스를 등록할 수 있습니다.

EC2 인스턴스를 등록 취소하면 로드 밸런서에서 인스턴스가 제거됩니다. 등록이 취소되는 즉시 로드 밸런서는 인스턴스로의 요청 라우팅을 중지합니다. 인스턴스에 대한 요구가 감소하거나 인스턴스를 서비스해야 하는 경우에는 로드 밸런서에서 인스턴스 등록을 취소할 수 있습니다. 등록 취소된 인스턴

스는 여전히 실행 중이지만, 더 이상 로드 밸런서에서 트래픽을 수신하지 않습니다. 준비가 되면 다시 로드 밸런서에 인스턴스를 등록할 수 있습니다.

Connection Draining이 활성화된 경우, 인스턴스 등록을 취소하면 Elastic Load Balancing이 진행 중인 요청이 완료될 때까지 기다립니다. 자세한 내용은 Classic Load Balancer에서 Connection Draining 구성 단원을 참조하십시오.

로드 밸런서가 Auto Scaling 그룹에 연결이 되어 있는 경우에는 해당 그룹의 인스턴스들이 로드 밸런서에 자동 등록됩니다. Auto Scaling 그룹에서 로드 밸런서를 분리하면 해당 그룹의 인스턴스들이 등록 취소됩니다.

Elastic Load Balancing은 IP 주소를 사용하여 로드 밸런서에 EC2 인스턴스를 등록합니다.

[EC2-VPC] 탄력적 네트워크 인터페이스(ENI)가 연결된 상태에서 인스턴스를 등록하면 로드 밸런서가 인스턴스의 주 인터페이스(eth0)의 주 IP 주소로 요청을 라우팅합니다.

내용

인스턴스 등록

준비가 되면 로드 밸런서에 인스턴스를 등록합니다. 인스턴스가 로드 밸런서에서 활성화된 가용 영역에 있는 경우, 필요한 수의 상태 확인을 통과하는 즉시 로드 밸런서에서 트래픽을 수신할 준비가 완료됩니다.

콘솔을 사용하여 인스턴스를 등록하려면

1. https://console.aws.amazon.com/ec2/에서 Amazon EC2 콘솔을 엽니다.
2. 탐색 창의 Load Balancing 아래에서 로드 밸런서를 선택합니다.
3. 로드 밸런서 이름을 선택하여 세부 정보 페이지를 엽니다.
4. 대상 인스턴스 탭에서 인스턴스 관리를 선택합니다.
5. 인스턴스 관리 페이지의 사용 가능한 인스턴스 테이블에서 로드 밸런서에 등록할 인스턴스를 선택합니다.
6. 등록이 필요한 인스턴스가 선택한 인스턴스 검토 테이블 내에 채워져 있는지 확인합니다.

7. 변경 사항 저장을 선택합니다.

를 사용하여 인스턴스를 등록하려면 AWS CLI

아래 register-instances-with-load-balancer 명령을 사용하세요.

```
aws elb register-instances-with-load-balancer --load-balancer-name my-loadbalancer --instances i-4e05f721
```

다음은 로드 밸런서에 등록된 인스턴스를 나열하는 응답의 예입니다.

```
{
    "Instances": [
        {
            "InstanceId": "i-315b7e51"
        },
        {
            "InstanceId": "i-4e05f721"
        }
    ]
}
```

로드 밸런서에 등록된 인스턴스 보기

지정된 로드 밸런서에 등록된 인스턴스 목록을 보려면 다음 describe-load-balancers 명령을 사용하세요.

```
aws elb describe-load-balancers --load-balancer-names my-load-balancer --output text --query "LoadBalancerDescriptions[*].Instances[*].InstanceId"
```

다음은 예제 출력입니다.

```
i-e905622e
i-315b7e51
i-4e05f721
```

등록된 인스턴스에 대한 로드 밸런서 결정

지정된 인스턴스가 등록된 로드 밸런서의 이름을 가져오려면 다음 describe-load-balancer 명령을 사용하세요.

```
aws elb describe-load-balancers --output text --query "LoadBalancerDescriptions[?
Instances[?InstanceId=='i-e905622e']].[LoadBalancerName]"
```

다음은 예제 출력입니다.

```
my-load-balancer
```

인스턴스 등록 취소

더 이상 용량이 필요하지 않거나 인스턴스를 서비스해야 하는 경우에는 로드 밸런서에서 인스턴스 등록을 취소할 수 있습니다.

로드 밸런서가 있는 Auto Scaling 그룹에 연결되어 있는 경우, 해당 그룹에서 인스턴스를 분리해도 로드 밸런서에서 인스턴스의 등록이 취소됩니다. 자세한 내용은 Amazon EC2 Auto Scaling 사용 설명서에서 Auto Scaling 그룹에서 EC2 인스턴스 분리를 참조하세요.

콘솔을 사용하여 인스턴스 등록을 취소하려면

1. https://console.aws.amazon.com/ec2/에서 Amazon EC2 콘솔을 엽니다.
2. 탐색 창의 Load Balancing 아래에서 로드 밸런서를 선택합니다.
3. 로드 밸런서 이름을 선택하여 세부 정보 페이지를 엽니다.
4. 대상 인스턴스 탭에서 인스턴스 관리를 선택합니다.
5. 인스턴스 관리 페이지의 사용 가능한 인스턴스 테이블에서 인스턴스의 선택을 취소하여 로드 밸런서에서 등록을 취소합니다.
6. 등록 취소가 필요한 인스턴스가 선택한 인스턴스 검토 테이블 내에 채워져 있지 않은지 확인합니다.
7. 변경 사항 저장을 선택합니다.

를 사용하여 인스턴스 등록을 취소하려면 AWS CLI

아래 deregister-instances-from-load-balancer 명령을 사용하세요.

```
aws elb deregister-instances-from-load-balancer --load-balancer-name my-loadbalancer --
instances i-4e05f721
```

다음은 로드 밸런서에 등록된 나머지 인스턴스들을 나열하는 응답의 예입니다.

```
{
    "Instances": [
        {
            "InstanceId": "i-315b7e51"
        }
    ]
}
```

Classic Load Balancer의 인스턴스 상태 확인

Classic Load Balancer는 등록된 인스턴스로 요청을 주기적으로 전송하여 상태를 확인합니다. 이러한 테스트를 바로 상태 확인이라고 합니다. 상태 확인 당시에 정상인 인스턴스의 상태는 `InService`라고 표시됩니다. 상태 확인 당시에 비정상인 인스턴스의 상태는 `OutOfService`라고 표시됩니다. 로드 밸런서는 인스턴스가 정상 상태이든 비정상 상태이든 관계 없이 등록된 모든 인스턴스에 대해 상태 확인을 수행합니다.

로드 밸런서는 정상 상태 인스턴스로만 요청을 라우팅합니다. 인스턴스가 비정상 상태라고 판단되면 로드 밸런서는 이 인스턴스로의 요청 라우팅을 중지합니다. 인스턴스가 정상 상태로 복구가 되면 로드 밸런서는 이 인스턴스로의 요청 라우팅을 재개합니다.

로드 밸런서는 Elastic Load Balancing이 제공하는 기본적인 상태 확인 구성이나 사용자가 지정한 상태 확인 구성을 사용하여 등록된 인스턴스의 상태를 확인합니다.

Auto Scaling 그룹을 Classic Load Balancer에 연결했다면, 로드 밸런서 상태 확인을 사용하여 Auto Scaling 그룹의 인스턴스 상태를 확인할 수 있습니다. Auto Scaling 그룹은 주기적으로 각 인스턴스의 상태를 확인하도록 기본 설정되어 있습니다. 자세한 내용은 Amazon EC2 Auto Scaling 사용 설명서의 Auto Scaling 그룹에 Elastic Load Balancing 상태 확인 추가를 참조하세요.

목차

상태 확인 구성

로드 밸런서는 인스턴스가 정상 상태이든 비정상 상태이든 관계 없이 등록된 모든 인스턴스에 대해 상태 확인을 수행합니다. 다음 표에서는 상태 확인 구성 필드를 설명합니다.

필드	설명
프로토콜	이 프로토콜을 사용하여 인스턴스에 연결합니다. 유효한 값: TCP, HTTP, HTTPS 및 SSL 콘솔 기본값: HTTP CLI/API 기본값: TCP
포트	이 포트를 사용하여 protocol:port 쌍 형태로 인스턴스를 연결합니다. 로드 밸런서가 구성된 응답 제한 시간 내에 지정된 포트의 인스턴스에 연결되지 못하면 해당 인스턴스는 비정상으로 간주됩니다. 프로토콜: TCP, HTTP, HTTPS 및 SSL 포트 범위: 1 ~ 65535 콘솔 기본값: HTTP:80 CLI/API 기본값: TCP:80
경로	HTTP 또는 HTTPS 요청의 대상입니다. HTTP 또는 HTTPS GET 요청은 포트 및 경로의 인스턴스로 발행됩니다. 로드 밸런서가 응답 제한 시간 내에 "200 OK" 이외의 응답을 수신하면 해당 인스턴스는 비정상으로 간주됩니다. 응답에 본문이 포함되어 있으면 애플리케이션은 0보다 크거나 같은 값으로 Content-Length 헤더를 설정하거나 'chunked'로 값을 설정해서 Transfer-Encoding을 지정해야 합니다. 기본값: /index.html
Response Timeout	상태 확인부터 응답 수신까지 기다리는 시간(초 단위)입니다.

필드	설명
	유효한 값: 2 ~ 60 기본값: 5
HealthCheck Interval	개별 인스턴스의 상태 확인 간 간격(초 단위)입니다. 유효한 값: 5 ~ 300 기본값: 30
Unhealthy Threshold	EC2 인스턴스를 비정상으로 선언하기 전까지 발생하는 연속적인 상태 확인 실패 횟수입니다. 유효한 값: 2 ~ 10 기본값: 2
Healthy Threshold	EC2 인스턴스를 정상으로 선언하기 전까지 발생하는 연속적인 상태 확인 성공 횟수입니다. 유효한 값: 2 ~ 10 기본값: 10

로드 밸런서는 지정된 포트, 프로토콜 및 경로를 사용하여 `Interval`초마다 등록된 모든 인스턴스에 상태 확인 요청을 전송합니다. 각 상태 확인 요청은 독립적이며 전체 간격 동안 지속됩니다. 인스턴스가 응답하는 데 걸리는 시간은 다음 상태 확인 요청의 간격에 영향을 미치지 않습니다. 상태 확인이 UnhealthyThresholdCount 연속 실패를 초과하면 로드 밸런서는 해당 인스턴스를 서비스에서 제외합니다. 상태 확인이 HealthyThresholdCount 연속 성공을 초과하면 로드 밸런서는 해당 인스턴스를 다시 서비스합니다.

상태 확인 간격 내에 해당 인스턴스가 200 응답 코드를 반환하면 HTTP/HTTPS 상태 확인이 지속됩니다. TCP 연결이 지속되면 TCP 상태 확인이 지속됩니다. SSL 상태 확인이 지속되면 SSL 악수가 지속됩니다.

상태 확인 구성 업데이트

로드 밸런서에서 상태 확인 구성을 언제든지 업데이트할 수 있습니다.

콘솔을 사용하여 로드 밸런서에서 상태 확인 구성을 업데이트하려면

1. https://console.aws.amazon.com/ec2/에서 Amazon EC2 콘솔을 엽니다.
2. 탐색 창의 Load Balancing 아래에서 로드 밸런서를 선택합니다.
3. 로드 밸런서 이름을 선택하여 세부 정보 페이지를 엽니다.
4. 상태 확인 탭에서 편집을 선택합니다.
5. 상태 확인 설정 편집 페이지의 상태 확인에서 필요에 따라 구성을 업데이트합니다.
6. 원하는 대로 선택했다면 변경 사항 저장을 선택합니다.

를 사용하여 로드 밸런서의 상태 확인 구성을 업데이트하려면 AWS CLI

아래 configure-health-check 명령을 사용하세요.

```
aws elb configure-health-check --load-balancer-name my-load-balancer --health-check
 Target=HTTP:80/path,Interval=30,UnhealthyThreshold=2,HealthyThreshold=2,Timeout=3
```

인스턴스의 상태 확인

등록된 인스턴스의 상태를 확인할 수 있습니다.

콘솔을 사용하여 인스턴스의 상태를 확인하려면

1. https://console.aws.amazon.com/ec2/에서 Amazon EC2 콘솔을 엽니다.
2. 탐색 창의 Load Balancing 아래에서 로드 밸런서를 선택합니다.
3. 로드 밸런서 이름을 선택하여 세부 정보 페이지를 엽니다.
4. 세부 정보 섹션에서 상태는 서비스 중인 인스턴스 수를 보여줍니다.
5. 대상 인스턴스탭의 대상 인스턴스 테이블 내에 있는 상태 열은 등록된 각 인스턴스의 특정 상태를 보여줍니다.

를 사용하여 인스턴스의 상태를 확인하려면 AWS CLI

다음 describe-instance-health 명령을 사용하세요.

```
aws elb describe-instance-health --load-balancer-name my-load-balancer
```

상태 확인 문제 해결

등록된 인스턴스는 몇 가지 이유로 로드 밸런서 상태 확인에 실패할 수 있습니다. 상태 확인에 실패하는 가장 흔한 이유는 EC2 인스턴스가 로드 밸런서에 대한 연결을 종료하거나 EC2 인스턴스에서의 응답이 시간을 초과했기 때문입니다. 잠재적인 원인과 상태 확인 실패를 해결하기 위해 취할 수 있는 조치에 대한 정보는 Classic Load Balancer 문제 해결: 상태 확인를 참조하십시오.

Classic Load Balancer 인스턴스의 보안 그룹

보안 그룹은 하나 이상의 인스턴스와 송수신이 허용되는 트래픽을 제어하는 방화벽 역할을 합니다. EC2 인스턴스를 시작할 때 하나 이상의 보안 그룹을 인스턴스에 연결할 수 있습니다. 각 보안 그룹에서 트래픽을 허용하는 규칙을 하나 이상 추가합니다. 언제든지 보안 그룹에 대한 규칙을 수정할 수 있습니다. 새 규칙은 보안 그룹에 연결된 모든 인스턴스에 자동으로 적용됩니다. 자세한 내용은 Amazon EC2 사용 설명서의 Amazon EC2 보안 그룹을 참조하세요.

인스턴스에 대한 보안 그룹은 로드 밸런서와의 통신을 허용해야 합니다. 다음 표에는 권장 인바운드 규칙이 나와 있습니다.

소스	프로토콜	포트 범위	Comment
## ### ## ##	TCP	#### ###	인스턴스 리스너 포트의 로드 밸런서에서 트래픽을 허용합니다
## ### ## ##	TCP	## ##	상태 확인 포트의 로드 밸런서에서 트래픽을 허용합니다

인바운드 ICMP 트래픽이 경로 MTU 검색을 지원하도록 허용하는 것이 좋습니다. 자세한 내용은 Amazon EC2 사용 설명서의 경로 MTU 검색을 참조하세요.

Classic Load Balancer 인스턴스의 네트워크 ACL

네트워크 액세스 제어 목록(ACL)은 서브넷 수준에서 특정 인바운드 또는 아웃바운드 트래픽을 허용하거나 거부합니다. VPC에 대한 기본 네트워크 ACL을 사용하거나 보안 그룹에 대한 규칙과 유사한 규칙을 사용하여 VPC에 대한 사용자 지정 네트워크 ACL을 생성하여 VPC에 보안 계층을 추가할 수 있습니다.

VPC에 대한 기본 네트워크 ACL(액세스 제어 목록)은 인바운드 트래픽과 아웃바운드 트래픽을 모두 허용합니다. 사용자 지정 네트워크 ACL을 생성하는 경우에는 로드 밸런서 및 인스턴스의 통신을 허용하는 규칙을 추가해야 합니다.

인스턴스 서브넷을 위한 권장 규칙은 서브넷이 프라이빗인지 퍼블릭인지에 따라 다릅니다. 다음은 프라이빗 서브넷을 위한 규칙입니다. 인스턴스가 퍼블릭 서브넷에 있는 경우에는 소스 및 대상을 VPC의 CIDR에서 `0.0.0.0/0`로 변경합니다.

다음은 권장 인바운드 규칙입니다.

소스	프로토콜	포트 범위	Comment
VPC CIDR	TCP	*#### ###*	인스턴스 리스너 포트에서 VPC CIDR에서 오는 인바운드 트래픽을 허용
VPC CIDR	TCP	*## ##*	상태 확인 포트에서 VPC CIDR에서 오는 인바운드 트래픽을 허용

다음은 권장 아웃바운드 규칙입니다.

Destination	프로토콜	포트 범위	Comment
VPC CIDR	TCP	1024-65535	휘발성 포트에서 VPC CIDR로 가는 아웃바운드 트래픽을 허용

Classic Load Balancer 모니터링

다음 기능을 사용하여 로드 밸런서를 모니터링하고 트래픽 패턴을 분석하며 로드 밸런서 및 백엔드 인스턴스의 문제를 해결할 수 있습니다.

CloudWatch 지표

Elastic Load Balancing은 로드 밸런서 및 백엔드 인스턴스에 대한 데이터 포인트를 Amazon CloudWatch에 게시합니다. CloudWatch를 사용하면 이러한 데이터 포인트에 대한 통계를 정렬된 시계열 데이터 세트로 검색할 수 있습니다. 이러한 통계를 지표라고 합니다. 이러한 지표를 사용하여 시스템이 예상대로 수행되고 있는지 확인할 수 있습니다. 자세한 내용은 Classic Load Balancer의 CloudWatch 지표 단원을 참조하십시오.

Elastic Load Balancing 액세스 로그

Elastic Load Balancing 액세스 로그는 로드 밸런서에 보낸 요청에 대한 자세한 정보를 캡처하고 사용자가 지정한 Amazon S3 버킷에 로그 파일로 저장합니다. 각 로그에는 요청을 받은 시간, 클라이언트의 IP 주소, 지연 시간, 요청 경로 및 서버 응답과 같은 세부 정보가 포함되어 있습니다. 이러한 액세스 로그를 사용하여 트래픽 패턴을 분석하고 백엔드 애플리케이션의 문제를 해결할 수 있습니다. 자세한 내용은 Classic Load Balancer 액세스 로그 단원을 참조하십시오.

CloudTrail 로그

AWS CloudTrail 를 사용하면 AWS 계정에 의해 또는 계정을 대신하여 Elastic Load Balancing API에 수행된 호출을 추적할 수 있습니다. CloudTrail은 지정하는 Amazon S3 버킷의 로그 파일에 정보를 저장합니다. 이러한 로그 파일을 사용하여 이뤄진 요청, 요청을 보낸 소스 IP 주소, 요청한 사람, 요청한 시기 등을 확인함으로써 로드 밸런서의 활동을 모니터링할 수 있습니다. 자세한 내용은 CloudTrail을 사용하여 Elastic Load Balancing에 대한 API 직접 호출 로깅을 참조하세요.

Classic Load Balancer의 CloudWatch 지표

Elastic Load Balancing은 로드 밸런서와 백엔드 인스턴스에 대해 Amazon CloudWatch에 데이터 포인트를 게시합니다. CloudWatch를 사용하면 이러한 데이터 포인트에 대한 통계를 정렬된 시계열 데이터 세트로 검색할 수 있습니다. 이러한 통계를 지표라고 합니다. 지표를 모니터링할 변수로 생각하면 데이터 요소는 시간에 따른 변수의 값을 나타냅니다. 예를 들어 지정된 기간 동안 로드 밸런서에 대한 정상 EC2 인스턴스의 총 수를 모니터링할 수 있습니다. 각 데이터 요소에는 연결된 타임스탬프와 측정 단위(선택 사항)가 있습니다.

지표를 사용하여 시스템이 예상대로 수행되고 있는지 확인할 수 있습니다. 예를 들어 CloudWatch 경보를 생성하여 지정된 지표를 모니터링할 수 있으며, 지표가 허용 범위를 벗어난다고 간주되는 경우 작업(예: 이메일 주소로 알림 전송)를 시작할 수 있습니다.

Elastic Load Balancing은 요청이 로드 밸런서를 통과하는 경우에만 지표를 CloudWatch에 보고합니다. 로드 밸런서를 통과하는 요청이 있는 경우 Elastic Load Balancing은 60초 간격으로 지표를 측정하고 전송합니다. 로드 밸런서를 통과하고 있는 요청이 없는 경우나 지표에 대한 데이터가 없는 경우에는 지표가 보고되지 않습니다.

Amazon CloudWatch에 대한 자세한 내용은 Amazon CloudWatch 사용 설명서를 참조하세요.

목차

Classic Load Balancer 지표

AWS/ELB 네임스페이스에는 다음 지표가 포함되어 있습니다.

지표	설명
BackendConnectionErrors	로드 밸런서와 등록된 인스턴스 사이에 성공적으로 구성되지 않은 연결 수. 오류 발생 시 로드 밸런서가 연결을 재시도하기 때문에 이 수가 요청 빈도를 초과할 수도 있습니다. 또한 여기에는 상태 검사와 관련된 연결 오류도 포함됩니다. 보고 기준: 0이 아닌 값이 있을 때 통계: 가장 유용한 통계는 Sum입니다. Average, Minimum 및 Maximum 통계는 로드 밸런서 노드별로 보고될 뿐 일반적으로 유용하지는 않습니다. 하지만 Minimum과 Maximum의 차이(또는 최대와 평균 비율, 평균과 저점 비율)는 로드 밸런서 노드의 이상 여부를 판단하는 데 도움이 될 수 있습니다.

지표	설명
	예: 로드 밸런서의 인스턴스가 us-west-2a에 2개, 그리고 us-west-2b에 2개 있을 때 us-west-2a의 인스턴스 1개에 연결하려는 도중 백엔드 연결 오류가 발생하였다고 가정하겠습니다. 그러면 us-west-2a의 합산에 연결 오류가 포함되지만 us-west-2b의 합산에는 연결 오류가 포함되지 않습니다. 따라서 로드 밸런서의 합산은 us-west-2a의 합산과 일치합니다.
DesyncMitigationMode_NonCompliant_Request_Count	[HTTP 리스너] RFC 7230을 준수하지 않는 요청 수입니다. 보고 기준: 0이 아닌 값이 있을 때 통계: 가장 유용한 통계는 Sum입니다.
HealthyHostCount	로드 밸런서에 등록된 정상 상태의 인스턴스 수. 새롭게 등록된 인스턴스는 첫 번째 상태 검사를 통과해야만 정상 상태로 간주됩니다. 교차 영역 로드 밸런싱이 활성화되어 있으면 LoadBalancerName 차원의 정상 인스턴스 수가 모든 가용 영역을 통틀어 계산됩니다. 그렇지 않으면 가용 영역별로 계산됩니다. 보고 기준: 등록된 인스턴스가 있을 때 통계: 가장 유용한 통계는 Average 및 Maximum입니다. 이 두 가지 통계는 로드 밸런서 노드에서 결정됩니다. 단, 일부 로드 밸런서 노드에서 정상 상태로 판단되는 인스턴스라고 해도 다른 노드에서는 일시적으로 비정상으로 판단될 수도 있습니다. 예: 로드 밸런서의 인스턴스가 us-west-2a에 2개, 그리고 us-west-2b에 2개 있을 때 us-west-2a에 비정상 인스턴스가 1개 있고, us-west-2b에는 비정상 인스턴스가 없다고 가정하겠습니다. 그러면 AvailabilityZone 차원에서는 us-west-2a에서 정상 인스턴스 1개와 비정상 인스턴스 1개를, 그리고 us-west-2b에서 정상 인스턴스 2개와 비정상 인스턴스 0개를 유지합니다.

지표	설명
HTTPCode_Backend_2XX , HTTPCode_Backend_3XX , HTTPCode_Backend_4XX , HTTPCode_Backend_5XX	[HTTP 리스너] 등록된 인스턴스에서 생성된 HTTP 응답 코드 수. 여기에 로드 밸런서에서 생성된 응답 코드 수는 포함되지 않습니다. 보고 기준: 0이 아닌 값이 있을 때 통계: 가장 유용한 통계는 Sum입니다. 단, Minimum, Maximum 및 Average는 모두 1입니다. 예: 로드 밸런서의 인스턴스가 us-west-2a에 2개, 그리고 us-west-2b에 2개 있을 때 us-west-2a의 인스턴스 1개로 요청이 전송되어 그 결과 HTTP 500 응답이 반환되었다고 가정하겠습니다. 그러면 us-west-2a의 합산에는 이 오류 응답이 포함되지만 us-west-2b의 합산에는 포함되지 않습니다. 따라서 로드 밸런서의 합산은 us-west-2a의 합산과 일치합니다.
HTTPCode_ELB_4XX	[HTTP 리스너] 로드 밸런서에서 생성된 HTTP 4XX 클라이언트 오류 코드 수. 클라이언트 오류는 요청 형식이 잘못되었거나 불완전할 때 생성됩니다. 보고 기준: 0이 아닌 값이 있을 때 통계: 가장 유용한 통계는 Sum입니다. 단, Minimum, Maximum 및 Average는 모두 1입니다. 예: 로드 밸런서에서 us-west-2a와 us-west-2b가 활성화되어 있을 때 클라이언트 요청에 잘못된 형식의 요청 URL이 포함되어 있다고 가정하겠습니다. 이때는 모든 가용 영역에 클라이언트 오류가 증가할 가능성이 높습니다. 그 결과, 로드 밸런서의 합산이 가용 영역의 값 합산과 일치하게 됩니다.

지표	설명
HTTPCode_ELB_5XX	[HTTP 리스너] 로드 밸런서에서 생성된 HTTP 5XX 서버 오류 코드 수. 여기에 등록된 인스턴스에서 생성된 응답 코드 수는 포함되지 않습니다. 이 지표는 로드 밸런서에 정상 인스턴스가 등록되어 있지 않거나, 혹은 요청 빈도가 인스턴스 용량을 초과하거나(스필오버) 또는 로드 밸런서 용량을 초과하는 경우에 보고됩니다. 보고 기준: 0이 아닌 값이 있을 때 통계: 가장 유용한 통계는 Sum입니다. 단, Minimum, Maximum 및 Average는 모두 1입니다. 예: 로드 밸런서에서 us-west-2a와 us-west-2b가 활성화되어 있을 때 us-west-2a의 인스턴스에서 지연 시간이 높아지면서 요청에 대한 응답 속도가 느려진다고 가정하겠습니다. 그러면 us-west-2a의 로드 밸런서 노드에서 서지 대기열이 채워지고 클라이언트는 503 오류를 수신하게 됩니다. 이때 us-west-2b가 계속해서 정상적으로 응답할 수 있다면 로드 밸런서의 합산은 us-west-2a의 합산과 일치합니다.

지표	설명
Latency	[HTTP 리스너] 로드 밸런서가 등록된 인스턴스에 요청을 보낸 시간부터 인스턴스가 응답 헤더를 보내기 시작할 때까지의 총 경과 시간(초)입니다. [TCP 리스너] 로드 밸런서가 등록된 인스턴스에 대한 연결을 성공적으로 설정하는 데 걸린 총 시간(초)입니다. 보고 기준: 0이 아닌 값이 있을 때 통계: 가장 유용한 통계는 Average입니다. Maximum은 평균 시간보다 더욱 오래 걸리는 요청의 유무를 확인할 때 사용합니다. Minimum은 일반적으로 사용되지 않습니다. 예: 로드 밸런서의 인스턴스가 us-west-2a에 2개, 그리고 us-west-2b에 2개 있을 때 us-west-2a의 인스턴스 1개로 전송되는 요청에서 지연 시간이 높게 발생한다고 가정하겠습니다. 이때 us-west-2a이 평균 값은 us-west-2b의 평균 값보다 높습니다.

지표	설명
RequestCount	지정한 주기(1분 또는 5분)에 완료된 요청 또는 연결 수 [HTTP 리스너] 등록된 인스턴스의 HTTP 오류 응답을 포함하여 수신 및 라우팅된 요청 수 [TCP 리스너] 등록된 인스턴스에 대한 연결 수 보고 기준: 0이 아닌 값이 있을 때 통계: 가장 유용한 통계는 Sum입니다. Minimum, Maximum 및 Average는 모두 1을 반환합니다. 예: 로드 밸런서의 인스턴스가 us-west-2a에 2개, 그리고 us-west-2b에 2개 있을 때 100개의 요청이 로드 밸런서에 전송된다고 가정하겠습니다. 이중 60개는 us-west-2a의 인스턴스로 각각 30개씩, 그리고 40개는 us-west-2b의 인스턴스로 각각 20개씩 전송됩니다. AvailabilityZone 차원에서는 us-west-2a의 요청 합산이 60개, us-west-2b의 요청 합산이 40개가 됩니다. 하지만 LoadBalancerName 차원에서는 요청 합산이 100개입니다.

지표	설명
SpilloverCount	서지 대기열이 가득 찼기 때문에 거부된 요청 총 수 [HTTP 리스너] 로드 밸런서가 HTTP 503 오류 코드를 반환합니다. [TCP 리스너] 로드 밸런서가 연결을 종료합니다. 보고 기준: 0이 아닌 값이 있을 때 통계: 가장 유용한 통계는 Sum입니다. Average, Minimum 및 Maximum 통계는 로드 밸런서 노드별로 보고될 뿐 일반적으로 유용하지는 않습니다. 예: 로드 밸런서에서 us-west-2a와 us-west-2b가 활성화되어 있을 때 us-west-2a의 인스턴스에서 지연 시간이 높아지면서 요청에 대한 응답 속도가 느려진다고 가정하겠습니다. 그러면 us-west-2a의 로드 밸런서 노드에서 서지 대기열이 채워지고 스필오버가 발생하게 됩니다. 이때 us-west-2b가 계속해서 정상적으로 응답할 수 있다면 로드 밸런서의 합산은 us-west-2a의 합산과 일치합니다.

지표	설명
SurgeQueueLength	정상 인스턴스 대상으로 라우팅이 보류 중인 총 요청(HTTP 리스터) 또는 연결(TCP 리스너) 수. 대기열의 최대 크기는 1,024입니다. 대기열이 가득 찼을 때는 추가 요청 또는 연결이 거부됩니다. 자세한 내용은 SpilloverCount 단원을 참조하십시오. 보고 기준: 0이 아닌 값이 있을 때. 통계: 가장 유용한 통계는 대기 중인 최대 요청 수를 의미하는 Maximum입니다. Average 통계는 Minimum 및 Maximum 통계와 함께 사용하여 대기 중인 요청의 범위를 확인할 때 유용합니다. 단, Sum 통계는 유용하지 않습니다. 예: 로드 밸런서에서 us-west-2a와 us-west-2b가 활성화되어 있을 때 us-west-2a의 인스턴스에서 지연 시간이 높아지면서 요청에 대한 응답 속도가 느려진다고 가정하겠습니다. 그러면 us-west-2a의 로드 밸런서 노드에서 서지 대기열이 채워지고, 클라이언트는 응답 시간이 길어질 가능성이 높습니다. 이렇게 지속되면 로드 밸런서는 스필오버 가능성이 커집니다(SpilloverCount 지표 참조). 이때 us-west-2b가 계속해서 정상적으로 응답할 수 있다면 로드 밸런서의 max는 us-west-2a의 max와 일치합니다.
UnHealthyHostCount	로드 밸런서에 등록된 비정상 상태의 인스턴스 수. 상태 검사에서 구성되어 있는 비정상 임계값을 초과한 인스턴스는 비정상 상태로 간주합니다. 이후 상태 검사에서 구성되어 있는 정상 임계값을 충족하는 비정상 인스턴스는 다시 정상 상태로 간주합니다. 보고 기준: 등록된 인스턴스가 있을 때 통계: 가장 유용한 통계는 Average 및 Minimum입니다. 이 두 가지 통계는 로드 밸런서 노드에서 결정됩니다. 단, 일부 로드 밸런서 노드에서 정상 상태로 판단되는 인스턴스라고 해도 다른 노드에서는 일시적으로 비정상으로 판단될 수도 있습니다. 예: HealthyHostCount 를 참조하십시오.

다음 지표를 사용하면 Classic Load Balancer를 Application Load Balancer로 마이그레이션할 경우 비용을 예측할 수 있습니다. 단, 이 지표는 참고용일 뿐 CloudWatch 경보에 사용해서는 안 됩니다. Classic Load Balancer에 다수의 리스너가 있는 경우에는 각 리스너마다 이 지표가 집계됩니다.

이러한 예측 비용은 기본 규칙 1개와 2K 크기의 인증서로 구성된 로드 밸런서를 기준으로 합니다. 크기가 4K 이상인 인증서를 사용하는 경우 비용을 다음과 같이 추정하는 것이 좋습니다. 마이그레이션 도구를 사용하여 Classic Load Balancer를 기반으로 Application Load Balancer를 생성하고 Application Load Balancer에 대한 `ConsumedLCUs` 지표를 모니터링합니다. 자세한 내용은 Elastic Load Balancing 사용 설명서의 Classic Load Balancer 마이그레이션을 참조하세요.

지표	설명
`EstimatedALBActiveConnectionCount`	클라이언트에서 로드 밸런서로, 그리고 로드 밸런서에서 대상으로 동시에 연결되는 활성 TCP 연결 예측 수.
`EstimatedALBConsumedLCUs`	Application Load Balancer에서 사용하는 예상 로드 밸런서 용량 단위(LCU) 수입니다. 시간 단위로 사용한 LCU 수만큼 요금을 지불하면 됩니다. 자세한 내용은 Elastic Load Balancing 요금을 참조하세요.
`EstimatedALBNewConnectionCount`	클라이언트에서 로드 밸런서로, 그리고 로드 밸런서에서 대상으로 새롭게 구성된 TCP 연결 예측 수
`EstimatedProcessedBytes`	Application Load Balancer에서 처리되는 바이트 예측 수.

Classic Load Balancer의 지표 차원

Classic Load Balancer 지표를 필터링하려면 다음 차원을 사용하세요.

차원	설명
`AvailabilityZone`	지정한 가용 영역을 기준으로 지표 데이터를 필터링합니다.

차원	설명
LoadBalancerName	지정한 로드 밸런서를 기준으로 지표 데이터를 필터링합니다.

Classic Load Balancer 지표에 대한 통계

CloudWatch는 Elastic Load Balancing에서 게시한 지표 데이터 포인트를 기반으로 통계를 제공합니다. 통계는 지정한 기간에 걸친 지표 데이터 집계입니다. 통계를 요청하면 지표 이름 및 차원으로 반환된 데이터 스트림이 식별됩니다. 차원이란 지표를 고유하게 식별하는 데 도움이 되는 이름/값 쌍을 말합니다. 예를 들어 특정 가용 영역에서 시작된 로드 밸런서를 지원하는 정상 상태의 모든 EC2 인스턴스에 대한 통계를 요청할 수 있습니다.

Minimum 및 Maximum 통계는 개별 로드 밸런서 노드가 보고한 최소 및 최대 값을 반영합니다. 예를 들어 로드 밸런서 노드가 2개라고 가정해 보겠습니다. 하나의 노드에는 HealthyHostCount이 2, Minimum이 10, Maximum가 6인 Average가 있으며 다른 노드에는 HealthyHostCount이 1, Minimum이 5, Maximum가 3인 Average가 있습니다. 따라서 로드 밸런서의 Minimum은 1, Maximum은 10, Average는 4입니다.

Sum 통계는 모든 로드 밸런서 노드의 집계 값입니다. 지표에는 기간별 보고서가 여러 개 있기 때문에 Sum은 RequestCount, HTTPCode_ELB_XXX, HTTPCode_Backend_XXX, BackendConnectionErrors 및 SpilloverCount 등 모든 로드 밸런서 노드에서 집계된 지표에만 적용할 수 있습니다.

SampleCount 통계는 측정된 샘플의 수입니다. 지표는 샘플링 간격 및 이벤트를 토대로 수집이 되기 때문에 일반적으로 이 통계는 유용하지 않습니다. 예를 들어 HealthyHostCount에 대해 SampleCount는 각 로드 밸런서 노드가 보고하는 샘플 수를 기반으로 하며 정상 호스트 수는 아닙니다.

백분위 수는 데이터 세트에서 값의 상대적 위치를 나타냅니다. 소수점 두 자리까지 사용하여 백분위 수를 지정할 수 있습니다(예: p95.45). 예를 들어 95 백분위는 데이터의 95%가 이 값보다 아래에 있고 5%가 위에 있다는 것을 의미합니다. 백분위 수는 종종 이상치를 격리하는 데 사용됩니다. 예를 들어 애플리케이션이 캐시에서 오는 요청의 대다수를 1-2 ms에 처리하지만, 캐시가 비어 있는 경우에는 처리에 100 - 200 ms가 걸린다고 가정해 봅시다. 최대값은 가장 느린 경우(200 ms 정도)를 반영합니다. 평균은 데이터의 분산을 나타내지 않습니다. 백분위 수는 애플리케이션 성능을 훨씬 의미 있는 방식으로 볼 수 있습니다. Auto Scaling 트리거 또는 CloudWatch 경보로 99 백분위를 사용하면 처리에 2 ms가 넘게 걸리는 요청이 전체의 1%를 넘지 않게 할 수 있습니다.

로드 밸런서에 대한 CloudWatch 지표 보기

Amazon EC2 콘솔을 사용하여 로드 밸런서에 대한 CloudWatch 지표를 볼 수 있습니다. 이 측정치들은 모니터링 그래프로 표시됩니다. 로드 밸런서가 활성 상태로 요청을 수신 중에 있으면 모니터링 그래프에 데이터 요소가 표시됩니다.

또는 CloudWatch 콘솔을 사용하여 로드 밸런서에 대한 지표를 볼 수 있습니다.

콘솔을 사용한 메트릭 확인

1. https://console.aws.amazon.com/ec2/에서 Amazon EC2 콘솔을 엽니다.
2. 탐색 창의 Load Balancing 아래에서 로드 밸런서를 선택합니다.
3. 로드 밸런서 이름을 선택하여 세부 정보 페이지를 엽니다.
4. 모니터링 탭을 선택합니다.
5. 단일 지표를 더 크게 보려면 그래프 위에 커서를 두고 `Maximize` 아이콘을 선택합니다. 다음과 같은 측정치를 사용할 수 있습니다.

   - 정상 호스트 - `HealthyHostCount`
   - 비정상 호스트 - `UnHealthyHostCount`
   - 평균 지연 시간 - `Latency`
   - 요청 - `RequestCount`
   - 백 엔드 연결 오류 - `BackendConnectionErrors`
   - 서지 대기열 길이 - `SurgeQueueLength`
   - 초과 수 - `SpilloverCount`
   - HTTP 2XX - `HTTPCode_Backend_2XX`
   - HTTP 3XX - `HTTPCode_Backend_3XX`
   - HTTP 4XX - `HTTPCode_Backend_4XX`
   - HTTP 5XX - `HTTPCode_Backend_5XX`
   - ELB HTTP 4XX - `HTTPCode_ELB_4XX`
   - ELB HTTP 5XX - `HTTPCode_ELB_5XX`
   - 예상 처리 바이트 수 - `EstimatedProcessedBytes`
   - 예상 ABL 소비 LCU - `EstimatedALBConsumedLCUs`
   - 예상 ALB 활성 연결 수 - `EstimatedALBActiveConnectionCount`
   - 예상 ALB 신규 연결 수 - `EstimatedALBNewConnectionCount`

CloudWatch 콘솔을 사용하여 지표를 보려면

1. https://console.aws.amazon.com/cloudwatch/에서 CloudWatch 콘솔을 엽니다.
2. 탐색 창에서 [지표(Metrics)]를 선택합니다.
3. [ELB] 네임스페이스를 선택합니다.
4. 다음 중 하나를 수행하십시오.
   - 지표 차원을 선택하여 로드 밸런서, 가용 영역 또는 모든 로드 밸런서의 지표를 확인합니다.
   - 모든 차원의 지표를 보려면 검색 필드에 이름을 입력합니다.
   - 단일 로드 밸런서에 대한 지표를 보려면 검색 필드에 해당되는 이름을 입력합니다.
   - 가용 영역에 대한 지표를 보려면 검색 필드에 해당되는 이름을 입력합니다.

Classic Load Balancer 액세스 로그

Elastic Load Balancing은 로드 밸런서에 전송된 요청에 대한 자세한 정보를 캡처하는 액세스 로그를 제공합니다. 각 로그에는 요청을 받은 시간, 클라이언트의 IP 주소, 지연 시간, 요청 경로 및 서버 응답과 같은 정보가 포함되어 있습니다. 이러한 액세스 로그를 사용하여 트래픽 패턴을 분석하고 문제를 해결할 수 있습니다.

액세스 로그는 Elastic Load Balancing의 옵션 기능으로, 기본적으로 비활성화되어 있습니다. 로드 밸런서에 대해 액세스 로그를 활성화하면 Elastic Load Balancing에서 로그를 캡처하여 지정한 Amazon S3 버킷에 저장합니다. 액세스 로그는 언제든지 비활성화할 수 있습니다.

각 액세스 로그 파일은 S3 버킷에 저장되기 전에 SSE-S를 사용하여 자동으로 암호화되고, 액세스할 때 해독됩니다. 어떤 조치도 취할 필요가 없습니다. 암호화 및 암호 해독이 투명하게 수행됩니다. 각 로그 파일은 고유 키로 암호화되며, 주기적으로 바뀌는 KMS 키를 사용하여 키 자체가 암호화됩니다. 자세한 내용은 Amazon S3 사용 설명서의 Amazon S3 관리형 암호화 키(SSE-S3)로 서버 측 암호화를 사용하여 데이터 보호 섹션을 참조하세요.

액세스 로그에 대한 추가 요금은 없습니다. Amazon S3에 대한 스토리지 비용은 청구되지만 Elastic Load Balancing이 Amazon S3에 로그 파일을 전송하기 위해 사용하는 대역폭에 대해서는 비용이 청구되지 않습니다. 스토리지 비용에 대한 자세한 내용은 Amazon S3 요금을 참조하세요.

목차

- Classic Load Balancer 액세스 로그 활성화
- Classic Load Balancer 액세스 로그 비활성화

액세스 로그 파일

Elastic Load Balancing은 지정한 간격으로 각 로드 밸런서 노드에 대한 로그 파일을 게시합니다. 로드 밸런서에 대한 액세스 로그를 활성화할 때 게시 간격을 5분 또는 60분으로 지정할 수 있습니다. 기본적으로 Elastic Load Balancing은 60분 간격으로 로그를 게시합니다. 간격을 5분으로 설정하면 로그는 1:05, 1:10, 1:15 등으로 게시됩니다. 간격이 5분으로 설정된 경우 로그 전달 시작이 최대 5분까지 지연되고 간격이 60분으로 설정된 경우 최대 15분까지 지연됩니다. 언제든지 게시 간격을 변경할 수 있습니다.

로드 밸런서는 같은 기간 동안 여러 개의 로그를 전달할 수 있습니다. 이는 일반적으로 사이트에 트래픽이 높고 여러 로드 밸런서 노드가 있으며 로그 게시 간격이 짧은 경우에 발생합니다.

액세스 로그의 파일 이름은 다음 형식을 사용합니다.

```
amzn-s3-demo-loadbalancer-logs[/logging-prefix]/AWSLogs/aws-account-id/
elasticloadbalancing/region/yyyy/mm/dd/aws-account-id_elasticloadbalancing_region_load-
balancer-name_end-time_ip-address_random-string.log
```

amzn-s3-demo-loadbalancer-logs

S3 버킷의 이름.

접두사

(선택 사항) 버킷의 접두사(논리적 계층 구조)입니다. 지정하는 접두사에는 문자열 AWSLogs가 포함되지 않아야 합니다. 자세한 내용은 접두사를 사용한 객체 구성을 참조하세요.

AWSLogs

AWSLogs로 시작하는 파일 이름의 일부가 지정하는 버킷 이름과 선택적 접두사 뒤에 추가됩니다.

aws-account-id

소유자의 AWS 계정 ID입니다.

리전.

로드 밸런서 및 S3 버킷을 위한 리전입니다.

yyyy/mm/dd

로그가 전달된 날짜입니다.

load-balancer-name

로드 밸런서의 이름입니다.

end-time

로깅 간격이 끝나는 날짜와 시간입니다. 예를 들어 종료 시간이 20140215T2340Z이면 게시 간격이 5분인 경우 23:35와 23:40 사이의 요청에 대한 항목이 포함됩니다.

ip-address

요청을 처리한 로드 밸런서 노드의 IP 주소입니다. 내부 로드 밸런서의 경우 프라이빗 IP 주소가 됩니다.

random-string

시스템에서 생성된 임의 문자열입니다.

다음은 접두사 'my-app'이 있는 로그 파일 이름의 예입니다.

```
s3://amzn-s3-demo-loadbalancer-logs/my-app/AWSLogs/123456789012/elasticloadbalancing/
us-west-2/2018/02/15/123456789012_elasticloadbalancing_us-west-2_my-
loadbalancer_20180215T2340Z_172.160.001.192_20sg8hgm.log
```

다음은 접두사가 없는 로그 파일 이름의 예입니다.

```
s3://amzn-s3-demo-loadbalancer-logs/AWSLogs/123456789012/elasticloadbalancing/
us-west-2/2018/02/15/123456789012_elasticloadbalancing_us-west-2_my-
loadbalancer_20180215T2340Z_172.160.001.192_20sg8hgm.log
```

원하는 기간만큼 버킷에 로그 파일을 저장할 수 있습니다. 그러나 Amazon S3 수명 주기 규칙을 정의하여 자동으로 로그 파일을 보관하거나 삭제할 수도 있습니다. 자세한 내용을 알아보려면 Amazon S3 사용 설명서의 객체 수명 주기 관리를 참조하세요.

액세스 로그 항목

Elastic Load Balancing은 백엔드 인스턴스로 전달되지 않는 요청을 포함해 로드 밸런서로 보낸 모든 요청을 기록합니다. 예를 들어 클라이언트가 잘못된 요청을 보내거나 응답할 정상 인스턴스가 없는 경우에도 요청은 계속 기록됩니다.

> **Important**
>
> Elastic Load Balancing은 최선의 노력으로 요청을 기록합니다. 모든 요청을 완벽하게 기록하기 위한 용도가 아니라 요청 특성을 이해하는 데 액세스 로그를 사용하는 것이 좋습니다.

구문

각 로그 항목에는 로드 밸런서에 대한 단일 요청의 세부 정보가 포함되어 있습니다. 로그 항목의 모든 필드는 공백으로 구분됩니다. 로그 파일의 각 항목은 다음 형식을 갖습니다.

```
timestamp elb client:port backend:port request_processing_time backend_processing_time
 response_processing_time elb_status_code backend_status_code received_bytes sent_bytes
 "request" "user_agent" ssl_cipher ssl_protocol
```

다음 표에서는 액세스 로그 항목의 필드를 설명합니다.

필드	설명
시간	로드 밸런서가 클라이언트에서 요청을 받은 시간입니다(ISO 8601 형식).
elb	로드 밸런서의 이름입니다
client:port	요청을 하는 클라이언트의 IP 주소 및 포트입니다.
backend:port	이 요청을 처리한 등록된 인스턴스의 IP 주소 및 포트입니다. 로드 밸런서가 등록된 인스턴스에 요청을 보낼 수 없거나 인스턴스가 응답을 보내기 전에 연결을 종료하면 이 값은 -로 설정됩니다. 등록된 인스턴스가 유휴 제한 시간 전에 응답하지 않으면 이 값을 -로 설정할 수도 있습니다.
request_processing_time	[HTTP 리스너] 로드 밸런서가 요청을 수신한 시간부터 등록된 인스턴스로 보낸 시간까지의 총 경과 시간(초)입니다. [TCP 리스너] 로드 밸런서가 클라이언트의 TCP/SSL 연결을 승인한 시간부터 첫 번째 데이터 바이트를 등록된 인스턴스로 보내는 데 걸린 총 경과 시간(초)입니다.

필드	설명
	로드 밸런서가 등록된 인스턴스에 요청을 디스패치할 수 없는 경우 이 값은 -1로 설정됩니다. 이는 등록된 인스턴스가 유휴 제한 시간 전에 연결을 종료하거나 클라이언트가 잘못된 요청을 보내는 경우에 발생할 수 있습니다. 또한 TCP 리스너의 경우 클라이언트가 로드 밸런서와의 연결을 설정했지만 데이터를 보내지 않으면 이러한 오류가 발생할 수 있습니다. 등록된 인스턴스가 유휴 제한 시간 전에 응답하지 않으면 이 값을 -1로 설정할 수도 있습니다.
backend_processing_time	[HTTP 리스너] 로드 밸런서가 등록된 인스턴스에 요청을 보낸 시간부터 인스턴스가 응답 헤더를 보내기 시작할 때까지의 총 경과 시간(초)입니다. [TCP 리스너] 로드 밸런서가 등록된 인스턴스에 대한 연결을 성공적으로 설정하는 데 걸린 총 시간(초)입니다. 로드 밸런서가 등록된 인스턴스에 요청을 디스패치할 수 없는 경우 이 값은 -1로 설정됩니다. 이는 등록된 인스턴스가 유휴 제한 시간 전에 연결을 종료하거나 클라이언트가 잘못된 요청을 보내는 경우에 발생할 수 있습니다. 등록된 인스턴스가 유휴 제한 시간 전에 응답하지 않으면 이 값을 -1로 설정할 수도 있습니다.

필드	설명
response_processing_time	[HTTP 리스너] 로드 밸런서가 등록된 인스턴스의 응답 헤더를 수신한 시간부터 클라이언트에 응답을 보내기 시작할 때까지의 총 경과 시간(초)입니다. 여기에는 로드 밸런서의 대기 시간과 로드 밸런서에서 클라이언트까지의 연결 확보 시간이 모두 포함됩니다. [TCP 리스너] 로드 밸런서가 등록된 인스턴스의 첫 번째 바이트를 수신한 시간부터 클라이언트에 응답을 보내기 시작할 때까지의 총 경과 시간(초)입니다. 로드 밸런서가 등록된 인스턴스에 요청을 디스패치할 수 없는 경우 이 값은 -1로 설정됩니다. 이는 등록된 인스턴스가 유휴 제한 시간 전에 연결을 종료하거나 클라이언트가 잘못된 요청을 보내는 경우에 발생할 수 있습니다. 등록된 인스턴스가 유휴 제한 시간 전에 응답하지 않으면 이 값을 -1로 설정할 수도 있습니다.
elb_status_code	[HTTP 리스너] 로드 밸런서의 응답 상태 코드입니다.
backend_status_code	[HTTP 리스너] 등록된 인스턴스의 응답 상태 코드입니다.
received_bytes	클라이언트(요청자)로부터 수신된 요청의 크기(바이트)입니다. [HTTP 리스너] 해당 값에는 요청 본문이 포함되지만 헤더는 포함되지 않습니다. [TCP 리스너] 해당 값에는 요청 본문 및 헤더가 포함됩니다.
sent_bytes	클라이언트(요청자)에게 보낸 응답의 크기(바이트)입니다. [HTTP 리스너] 해당 값에는 응답 본문이 포함되지만 헤더는 포함되지 않습니다. [TCP 리스너] 해당 값에는 요청 본문 및 헤더가 포함됩니다.

필드	설명
요청	큰 따옴표로 묶여 HTTP 메서드 + 프로토콜://호스트 헤더:포트 + 경로 + HTTP 버전 형식으로 기록된 클라이언트 요청 줄입니다. 로드 밸런서는 요청 URI를 기록할 때 클라이언트가 보낸 URL을 원본 그대로 보관합니다. 또한 액세스 로그 파일에 대한 콘텐츠 유형을 설정하지 않습니다. 이 필드를 처리하는 경우 해당 클라이언트가 URL을 보낸 방법을 고려하십시오. [TCP 리스너] URL은 각각 공백으로 구분된 세 개의 대시이며 공백으로 끝납니다("- - - ").
user_agent	[HTTP/HTTPS 리스너] 요청을 보낸 클라이언트를 식별하는 User-Agent 문자열입니다. 이 문자열은 하나 이상의 제품 식별자, 제품[/버전]으로 이루어져 있습니다. 문자열이 8 KB보다 길면 잘리게 됩니다.
ssl_cipher	[HTTPS/SSL 리스너] SSL 암호입니다. 이 값은 성공적인 협상 후에 수신되는 SSL/TLS 연결이 설정된 경우에만 기록됩니다. 그렇지 않으면 이 값은 -로 설정됩니다.
ssl_protocol	[HTTPS/SSL 리스너] SSL 프로토콜입니다. 이 값은 성공적인 협상 후에 수신되는 SSL/TLS 연결이 설정된 경우에만 기록됩니다. 그렇지 않으면 이 값은 -로 설정됩니다.

예제

HTTP 항목 예제

다음은 HTTP 리스너(포트 80에서 포트 80)를 위한 로그 항목 예제입니다.

```
2015-05-13T23:39:43.945958Z my-loadbalancer 192.168.131.39:2817 10.0.0.1:80 0.000073
 0.001048 0.000057 200 200 0 29 "GET http://www.example.com:80/ HTTP/1.1" "curl/7.38.0"
 - -
```

HTTPS 항목 예제

다음은 HTTPS 리스너(포트 443에서 포트 80)를 위한 로그 항목 예제입니다.

```
2015-05-13T23:39:43.945958Z my-loadbalancer 192.168.131.39:2817 10.0.0.1:80
 0.000086 0.001048 0.001337 200 200 0 57 "GET https://www.example.com:443/ HTTP/1.1"
 "curl/7.38.0" DHE-RSA-AES128-SHA TLSv1.2
```

TCP 항목 예제

다음은 TCP 리스너 로그 항목 예입니다(포트 8080에서 포트 80).

```
2015-05-13T23:39:43.945958Z my-loadbalancer 192.168.131.39:2817 10.0.0.1:80 0.001069
 0.000028 0.000041 - - 82 305 "- - - " "-" - -
```

SSL 항목 예제

다음은 SSL 리스너 로그 항목 예입니다(포트 8443에서 포트 80).

```
2015-05-13T23:39:43.945958Z my-loadbalancer 192.168.131.39:2817 10.0.0.1:80 0.001065
 0.000015 0.000023 - - 57 502 "- - - " "-" ECDHE-ECDSA-AES128-GCM-SHA256 TLSv1.2
```

액세스 로그 처리

웹 사이트에서 요청이 많은 경우에는 로드 밸런서가 수 기가바이트의 데이터로 로그 파일을 생성할 수 있습니다. 라인별 처리로는 이렇게 대량의 데이터를 처리할 수 없습니다. 따라서 병렬 처리 솔루션을 제공하는 분석 도구를 사용해야 할 수 있습니다. 예를 들어, 다음과 같은 분석 도구를 사용하여 액세스 로그를 분석 및 처리할 수 있습니다.

- Amazon Athena는 표준 SQL을 사용해 Amazon S3에 저장된 데이터를 간편하게 분석할 수 있는 대화식 쿼리 서비스입니다. 자세한 내용은 Amazon Athena 사용 설명서의 Classic Load Balancer 로그 쿼리를 참조하세요.
- Loggly
- Splunk
- Sumo Logic

Classic Load Balancer 액세스 로그 활성화

로드 밸런서에 대한 액세스 로그를 활성화하려면 로드 밸런서가 로그를 저장할 Amazon S3 버킷의 이름을 지정해야 합니다. Elastic Load Balancing에게 버킷에 대한 쓰기 권한을 부여하는 버킷 정책을 이 버킷에 연결해야 합니다.

작업

- 1단계: S3 버킷 생성
- 2단계: S3 버킷에 정책 연결
- 3단계: 액세스 로그 구성
- 4단계: 버킷 권한 확인
- 문제 해결

1단계: S3 버킷 생성

액세스 로그를 활성화할 때는 반드시 액세스 로그 파일에 대한 S3 버킷을 지정해야 합니다. 버킷은 다음 요구 사항을 충족해야 합니다.

요구 사항

- 버킷은 로드 밸런서와 같은 리전에 있어야 합니다. 서로 다른 계정에서 버킷과 로드 밸런서를 소유할 수 있습니다.
- 지원되는 유일한 서버 측 암호화 옵션은 Amazon S3 관리형 키(SSE-S3)입니다. 자세한 내용은 Amazon S3 관리형 암호화 키(SSE-S3) 섹션을 참조하세요.

Amazon S3 콘솔을 사용하여 S3 버킷에 폴더를 생성하려면

1. https://console.aws.amazon.com/s3/에서 Amazon S3 콘솔을 엽니다.
2. 버킷 만들기를 선택합니다.
3. [Create a bucket] 페이지에서 다음과 같이 실행합니다.
   a. [Bucket Name]에서 버킷 이름을 입력합니다. 선택한 이름은 Amazon S3에 있는 어떤 기존 버킷 이름과도 중복되지 않아야 합니다. 일부 리전에서는 버킷 이름에 대한 추가 제한이 있을 수 있습니다. 자세한 내용은 Amazon S3 사용 설명서의 버킷 할당량, 제한 및 제한 사항을 참조하세요.
   b. AWS 리전의 경우 로드 밸런서를 생성한 리전을 선택합니다.
   c. 기본 암호화에서 Amazon S3 관리형 키(SSE-S3)를 선택합니다.
   d. 버킷 생성을 선택합니다.

2단계: S3 버킷에 정책 연결

버킷에 액세스 로그를 쓰는 Elastic Load Balancing 권한을 부여하는 버킷 정책이 S3 버킷에 있어야 합니다. 버킷 정책은 버킷에 대한 액세스 권한을 정의하기 위해 액세스 정책 언어로 작성된 JSON 문의 집합입니다. 각 문에는 단일 권한에 대한 정보와 일련의 요소들이 포함되어 있습니다.

연결된 정책이 이미 있는 기존 버킷을 사용하는 Elastic Load Balancing 액세스 로그에 대한 문을 정책에 추가할 수 있습니다. 그렇게 하는 경우, 결과적인 액세스 권한 집합을 평가하여 해당 집합이 액세스 로그에 대한 버킷에 액세스해야 하는 사용자에게 적절한 권한인지 확인하는 것이 좋습니다.

버킷 정책

이 정책은 로그 전송 서비스에 권한을 부여합니다.

```
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "Service": "logdelivery.elasticloadbalancing.amazonaws.com"
      },
      "Action": "s3:PutObject",
      "Resource": "arn:aws:s3:::amzn-s3-demo-bucket/prefix/AWSLogs/123456789012/*"
    }
  ]
}
```

`Resource`에는 예제 정책에 표시된 형식을 사용하여 액세스 로그 위치의 ARN을 입력합니다. 항상 S3 버킷 ARN의 리소스 경로에는 로드 밸런서를 보유한 계정의 계정 ID를 포함해야 합니다. 이렇게 하면 지정된 계정의 로드 밸런서만 해당 S3 버킷에 액세스 로그를 기록할 수 있습니다.

지정하는 ARN은 3단계에서 액세스 로그를 활성화할 때 접두사를 포함할 계획인지 여부에 따라 다릅니다.

접두사가 있는 S3 버킷 ARN 예제

S3 버킷 이름은 amzn-s3-demo-logging-bucket이고 접두사는 logging-prefix입니다.

```
arn:aws:s3:::amzn-s3-demo-logging-bucket/logging-prefix/AWSLogs/123456789012/*
```

[AWS GovCloud (US)] 다음 예제에서는 AWS GovCloud (US) Regions에 ARN 구문을 사용합니다.

```
arn:aws-us-gov:s3:::amzn-s3-demo-logging-bucket/logging-prefix/AWSLogs/123456789012/*
```

접두사가 없는 S3 버킷 ARN 예제

S3 버킷 이름은 amzn-s3-demo-logging-bucket입니다. S3 버킷 ARN에는 접두사 부분이 없습니다.

```
arn:aws:s3:::amzn-s3-demo-logging-bucket/AWSLogs/123456789012/*
```

[AWS GovCloud (US)] 다음 예제에서는 AWS GovCloud (US) Regions에 ARN 구문을 사용합니다.

```
arn:aws-us-gov:s3:::amzn-s3-demo-logging-bucket/AWSLogs/123456789012/*
```

레거시 버킷 정책

2022년 8월 이전에 제공되던 리전의 경우, 리전별 Elastic Load Balancing 계정에 권한을 부여하는 정책이 필요했습니다. 이 레거시 정책은 여전히 지원되지만, 위에 소개된 최신 정책으로 대체할 것을 권장합니다. 표시되어 있지 않지만 레거시 버킷 정책을 계속 사용하려는 경우 그대로 사용할 수 있습니다.

참고로 `Principal`에서 지정해야 하는 Elastic Load Balancing 계정 ID는 다음과 같습니다. 이 목록에 없는 리전은 레거시 버킷 정책을 지원하지 않습니다.

- 미국 동부(버지니아 북부) – 127311923021
- 미국 동부(오하이오) – 033677994240
- 미국 서부(캘리포니아 북부) – 027434742980
- 미국 서부(오레곤) – 797873946194
- 아프리카(케이프타운) – 098369216593
- 아시아 태평양(홍콩) – 754344448648
- 아시아 태평양(자카르타) – 589379963580
- 아시아 태평양(뭄바이) – 718504428378
- 아시아 태평양(오사카) – 383597477331
- 아시아 태평양(서울) – 600734575887
- 아시아 태평양(싱가포르) – 114774131450
- 아시아 태평양(시드니) – 783225319266

- 아시아 태평양(도쿄) – 582318560864
- 캐나다(중부) – 985666609251
- 유럽(프랑크푸르트) – 054676820928
- 유럽(아일랜드) – 156460612806
- 유럽(런던) – 652711504416
- 유럽(밀라노) – 635631232127
- 유럽(파리) – 009996457667
- 유럽(스톡홀름) – 897822967062
- 중동(바레인) – 076674570225
- 남아메리카(상파울루) – 507241528517
- AWS GovCloud(미국 동부) – 190560391635
- AWS GovCloud(미국 서부) – 048591011584

보안 모범 사례

보안을 강화하려면 정확한 S3 버킷 ARN을 사용합니다.

- S3 버킷 ARN뿐만 아니라 전체 리소스 경로를 사용합니다.
- S3 버킷 ARN의 계정 ID 부분을 포함합니다.
- S3 버킷 ARN의 계정 ID 부분에 와일드카드(*)를 사용하지 마세요.

버킷 정책을 생성한 후 Amazon S3 콘솔 또는 AWS CLI 명령과 같은 Amazon S3 인터페이스를 사용하여 버킷 정책을 S3 버킷에 연결합니다.

콘솔을 사용하여 버킷 정책을 버킷에 연결하려면

1. https://console.aws.amazon.com/s3/에서 S3 콘솔을 엽니다.
2. 버킷 이름을 선택하여 세부 정보 페이지를 엽니다.
3. 권한을 선택한 다음에 버킷 정책, 편집을 선택합니다.
4. 버킷 정책을 업데이트하여 필수 권한을 부여합니다.
5. 변경 사항 저장을 선택합니다.

를 사용하여 버킷 정책을 S3 버킷에 연결하려면 AWS CLI

put-bucket-policy 명령을 사용합니다. 이 예제에서는 버킷 정책이 지정된 .json 파일에 저장되었습니다.

```
aws s3api put-bucket-policy \
    --bucket amzn-s3-demo-bucket \
    --policy file://access-log-policy.json
```

3단계: 액세스 로그 구성

다음 절차에 따라 요청 정보를 캡처하고 로그 파일을 S3 버킷에 전송하도록 액세스 로그를 구성합니다.

요구 사항

버킷은 1단계에 설명된 요구 사항을 충족해야 하며 2단계의 설명에 따라 버킷 정책을 연결해야 합니다. 접두사를 지정하는 경우 접두사에 'AWSLogs' 문자열이 포함되지 않아야 합니다.

콘솔을 사용하여 로드 밸런서에 대한 액세스 로그를 구성하는 방법

1. https://console.aws.amazon.com/ec2/에서 Amazon EC2 콘솔을 엽니다.
2. 탐색 창의 Load Balancing 아래에서 로드 밸런서를 선택합니다.
3. 로드 밸런서 이름을 선택하여 세부 정보 페이지를 엽니다.
4. 속성 탭에서 편집을 선택합니다.
5. 로드 밸런서 속성 편집 페이지의 모니터링 섹션에서 다음을 수행합니다.
    a. 액세스 로그를 활성화합니다.
    b. S3 URI로 로그 파일의 S3 URI를 입력합니다. 지정하는 URI는 접두사 사용 여부에 따라 달라집니다.
        - 접두사가 있는 URI: `s3://amzn-s3-demo-logging-bucket/logging-prefix`
        - 접두사가 없는 URI: `s3://amzn-s3-demo-logging-bucket`
    c. 로깅 간격을 `60 minutes - default`로 유지합니다.
    d. 변경 사항 저장을 선택합니다.

를 사용하여 로드 밸런서에 대한 액세스 로그를 구성하려면 AWS CLI

먼저 Elastic Load Balancing이 60분마다 로그를 캡처하여 로그용으로 생성한 S3 버킷에 전달할 수 있게 하는 .json 파일을 만듭니다.

```
{
  "AccessLog": {
    "Enabled": true,
    "S3BucketName": "amzn-s3-demo-logging-bucket",
    "EmitInterval": 60,
    "S3BucketPrefix": "my-app"
  }
}
```

다음으로, 다음과 같이 modify-load-balancer-attributes 명령에 .json 파일을 지정합니다.

```
aws elb modify-load-balancer-attributes --load-balancer-name my-loadbalancer --load-
balancer-attributes file://my-json-file.json
```

다음은 응답의 예입니다.

```
{
    "LoadBalancerAttributes": {
        "AccessLog": {
            "Enabled": true,
            "EmitInterval": 60,
            "S3BucketName": "amzn-s3-demo-logging-bucket",
            "S3BucketPrefix": "my-app"
        }
    },
    "LoadBalancerName": "my-loadbalancer"
}
```

액세스 로그에 대해 S3 버킷을 관리하려면

액세스 로그용으로 구성한 버킷을 삭제하기 전에 액세스 로그를 비활성화해야 합니다. 그렇지 않으면 이름이 같은 새 버킷과 소유 AWS 계정 하지 않은에서 생성된 필수 버킷 정책이 있는 경우 Elastic Load Balancing은 로드 밸런서에 대한 액세스 로그를이 새 버킷에 쓸 수 있습니다.

4단계: 버킷 권한 확인

로드 밸런서에 대해 액세스 로그가 활성화되면 Elastic Load Balancing에서는 S3 버킷을 검증하고 버킷 정책에서 필수 권한을 지정하는지 확인하는 테스트 파일을 생성합니다. S3 콘솔을 사용하여 테스트 파일이 생성되었는지 확인할 수 있습니다. 테스트 파일은 실제 액세스 로그 파일이 아니며, 예제 레코드가 포함되어 있지 않습니다.

S3 버킷에서 Elastic Load Balancing이 테스트 파일을 생성했는지 확인하려면

1. https://console.aws.amazon.com/s3/에서 Amazon S3 콘솔을 엽니다.
2. 액세스 로그에 대해 지정한 S3 버킷의 이름을 선택합니다.
3. 테스트 파일인 `ELBAccessLogTestFile`로 이동합니다. 위치는 접두사 사용 여부에 따라 달라집니다.

   - 접두사가 있는 위치: *amzn-s3-demo-loadbalancer-logs*/*logging-prefix*/AWSLogs/*123,456,789,012*/ELBAccessLogTestFile
   - 접두사가 없는 위치: *amzn-s3-demo-loadbalancer-logs*/AWSLogs/*123,456,789,012*/ELBAccessLogTestFile

문제 해결

bucket-name 버킷에 대한 액세스가 거부되었습니다. S3bucket 권한을 확인하세요.

이 오류가 발생한 경우, 가능한 원인은 다음과 같습니다.

- 버킷 정책이 버킷에 액세스 로그를 쓰도록 허용하는 Elastic Load Balancing 권한을 부여하지 않습니다. 리전에 올바른 버킷 정책을 사용하고 있는지 확인합니다. 액세스 로그를 활성화할 때 지정한 것과 동일한 버킷 이름을 리소스 ARN에서 사용하는지 확인하세요. 액세스 로그를 활성화할 때 접두사를 지정하지 않은 경우 리소스 ARN에 접두사가 포함되어 있지 않은지 확인합니다.
- 버킷이 지원되지 않는 서버 측 암호화 옵션을 사용합니다. 버킷이 Amazon S3 관리형 키(SSE-S3)를 사용해야 합니다.

Classic Load Balancer 액세스 로그 비활성화

언제든지 로드 밸런서에 대한 액세스 로그를 비활성화할 수 있습니다. 액세스 로그를 비활성화하면 사용자가 삭제할 때까지 액세스 로그가 Amazon S3에 남아 있습니다. 자세한 내용은 Amazon S3 사용자 안내서의 S3 버킷 작업을 참조하세요.

콘솔을 사용하여 로드 밸런서에 대한 액세스 로그 비활성화

1. https://console.aws.amazon.com/ec2/에서 Amazon EC2 콘솔을 엽니다.
2. 탐색 창의 Load Balancing 아래에서 로드 밸런서를 선택합니다.
3. 로드 밸런서 이름을 선택하여 세부 정보 페이지를 엽니다.
4. 속성 탭에서 편집을 선택합니다.

5. 로드 밸런서 속성 편집 페이지의 모니터링 섹션에서 액세스 로그를 비활성화합니다.

를 사용하여 액세스 로그를 비활성화하려면 AWS CLI

다음과 같은 modify-load-balancer-attributes 명령을 사용하여 액세스 로그를 비활성화합니다.

```
aws elb modify-load-balancer-attributes --load-balancer-name my-loadbalancer --load-balancer-attributes "{\"AccessLog\":{\"Enabled\":false}}"
```

다음은 응답의 예입니다.

```
{
    "LoadBalancerName": "my-loadbalancer",
    "LoadBalancerAttributes": {
        "AccessLog": {
            "S3BucketName": "amzn-s3-demo-loadbalancer-logs",
            "EmitInterval": 60,
            "Enabled": false,
            "S3BucketPrefix": "my-app"
        }
    }
}
```

Classic Load Balancer 문제 해결

다음 표에는 Classic Load Balancer를 사용할 때 유용하게 참조할 수 있는 문제 해결 리소스가 나와 있습니다.

API 오류

오류
CertificateNotFound: 정의되지 않음
OutofService: 일시적인 오류가 발생함

HTTP 오류

오류
HTTP 400: BAD_REQUEST
HTTP 405: METHOD_NOT_ALLOWED
HTTP 408: 요청 제한 시간
HTTP 502: 잘못된 게이트웨이
HTTP 503: 서비스 사용 불가
HTTP 504: 게이트웨이 제한 시간

응답 코드 지표

응답 코드 지표
HTTPCode_ELB_4XX
HTTPCode_ELB_5XX
HTTPCode_Backend_2XX

응답 코드 지표
HTTPCode_Backend_3XX
HTTPCode_Backend_4XX
HTTPCode_Backend_5XX

상태 확인 문제

문제
상태 확인 대상 페이지 오류
인스턴스 연결 시간 초과
퍼블릭 키 인증이 실패함
인스턴스가 로드 밸런서에서 트래픽을 수신하지 않음
인스턴스의 포트가 열려 있지 않음
Auto Scaling 그룹의 인스턴스가 ELB 상태 확인에 실패함

연결 문제

문제
클라이언트가 인터넷 경계 로드 밸런서에 연결할 수 없음
사용자 지정 도메인으로 전송된 요청은 로드 밸런서에 수신되지 않습니다.
로드 밸런서로 전송된 HTTPS 요청은 “NET: :ERR_CERT_COMMON_NAME_INVALID”를 반환합니다.

인스턴스 등록 문제

문제
EC2 인스턴스를 등록하는 데 너무 오래 걸림
유료 AMI에서 시작된 인스턴스를 등록할 수 없음

Classic Load Balancer 문제 해결: API 오류

다음은 Elastic Load Balancing API, 잠재적인 원인 및 문제를 해결하기 위해 수행할 수 있는 단계에서 반환되는 오류 메시지입니다.

오류 메시지

- CertificateNotFound: 정의되지 않음
- OutofService: 일시적인 오류가 발생함

CertificateNotFound: 정의되지 않음

원인 1: AWS Management Console을 사용하여 인증서를 생성할 때 인증서를 모든 리전에 전달하는 데 지연이 있습니다. 이러한 지연이 발생하면 로드 밸런서 생성 과정의 마지막 단계에서 오류 메시지가 표시됩니다.

솔루션 1: 약 15분간 기다린 후 다시 시도합니다. 문제가 계속되면 AWS Support 센터로 이동하여 지원을 요청합니다.

원인 2: AWS CLI 또는 API를 직접 사용하는 경우 존재하지 않는 인증서에 Amazon 리소스 이름(ARN)을 제공하면이 오류가 발생할 수 있습니다.

솔루션 2: AWS Identity and Access Management (IAM) 작업 GetServerCertificate를 사용하여 인증서 ARN을 가져오고 ARN에 올바른 값을 제공했는지 확인합니다.

OutofService: 일시적인 오류가 발생함

원인: Elastic Load Balancing 서비스 또는 기본 네트워크 내에 일시적인 내부 문제가 있습니다. 이 일시적인 문제는 Elastic Load Balancing이 로드 밸런서 및 등록된 인스턴스의 상태를 쿼리할 때에도 발생할 수 있습니다.

솔루션: API 호출을 다시 시도합니다. 문제가 계속되면 AWS Support 센터로 이동하여 지원을 요청합니다.

Classic Load Balancer 문제 해결: HTTP 오류

HTTP 메서드(동사라고도 불림)는 HTTP 요청을 수신하는 리소스에서 수행할 작업을 지정합니다. HTTP 요청에 대한 표준 메서드는 RFC 2616 Method Definitions에 정의되어 있습니다. 표준 메서드에는 GET, POST, PUT, HEAD 및 OPTIONS가 포함됩니다. 일부 웹 애플리케이션은 HTTP/1.1 메서드의 확장 메서드를 필요로 하며 때에 따라 이를 적용합니다. HTTP 확장 메서드의 일반적인 예에는 PATCH, REPORT, MKCOL, PROPFIND, MOVE 및 LOCK이 포함됩니다. Elastic Load Balancing은 모든 표준 및 비표준 HTTP 메서드를 허용합니다.

HTTP 요청 및 응답은 헤더 필드를 사용하여 HTTP 메시지에 대한 정보를 보냅니다. 헤더 필드는 콜론으로 구분된 이름-값 페어이며 CR(캐리지 리턴) 및 LF(줄 바꿈)로 구분됩니다. HTTP 헤더 필드의 표준 집합은 RFC 2616 Message Headers에 정의되어 있습니다. 자세한 내용은 HTTP 헤더 및 Classic Load Balancer 단원을 참조하십시오.

로드 밸런서는 HTTP 요청을 받으면 잘못된 형식의 요청 및 메서드 길이를 확인합니다. 로드 밸런서에 대한 HTTP 요청의 전체 메소드 길이는 127자를 초과할 수 없습니다. HTTP 요청이 두 가지 확인을 모두 통과하면 로드 밸런서는 요청을 EC2 인스턴스로 전송합니다. 요청의 메소드 필드가 잘못된 형식인 경우 로드 밸런서는 HTTP 400: BAD_REQUEST 오류로 응답합니다. 요청의 메소드 길이가 127자를 초과하는 경우 로드 밸런서는 HTTP 405: METHOD_NOT_ALLOWED 오류로 응답합니다.

EC2 인스턴스는 요청에 메소드를 구현하고 응답을 클라이언트에 다시 전송하는 방식으로 유효한 요청을 처리합니다. 인스턴스는 지원되는 메소드와 지원되지 않는 메소드를 모두 처리하도록 구성되어야 합니다.

다음은 로드 밸런서, 잠재적인 원인 및 문제를 해결하기 위해 수행할 수 있는 단계에서 반환되는 오류 메시지입니다.

오류 메시지

- HTTP 400: BAD_REQUEST
- HTTP 405: METHOD_NOT_ALLOWED
- HTTP 408: 요청 제한 시간
- HTTP 502: 잘못된 게이트웨이
- HTTP 503: 서비스 사용 불가
- HTTP 504: 게이트웨이 제한 시간

HTTP 400: BAD_REQUEST

설명: 클라이언트가 잘못된 요청을 전송했다는 것을 나타냅니다.

원인 1: 클라이언트가 HTTP 사양을 충족하지 않는 잘못된 형식의 요청을 전송했습니다. 예를 들어 요청에 URL 공백이 없어야 합니다.

원인 2: 클라이언트가 사용한 HTTP CONNECT 방법은 Elastic Load Balancing에서 지원하지 않습니다.

솔루션: 인스턴스에 직접 연결하고 클라이언트 요청의 세부 사항을 캡처합니다. 잘못된 형식의 요청에 대해 헤더와 URL을 검토합니다. 요청이 HTTP 사양에 부합하는지 확인합니다. HTTP CONNECT를 사용하지 않는지 확인하십시오.

HTTP 405: METHOD_NOT_ALLOWED

설명: 메서드 길이가 잘못되었다는 것을 나타냅니다.

원인: 요청 헤더의 메서드 길이가 127자를 초과합니다.

솔루션: 메서드 길이를 확인합니다.

HTTP 408: 요청 제한 시간

설명: 클라이언트가 요청을 취소했거나 전체 요청을 전송하지 못했다는 것을 나타냅니다.

원인 1: 네트워크 중단 또는 부분적으로 형성된 헤더 같은 잘못된 요청, 지정된 콘텐츠 크기가 전송된 실제 콘텐츠 크기와 일치하지 않는 상황 등이 있습니다.

솔루션 1: 요청을 하는 코드를 검사하고, 실제 요청을 검사할 수 있는 권한을 가진 등록된 인스턴스(또는 개발/테스트 환경)로 직접 보내 봅니다.

원인 2: 클라이언트에 대한 연결이 종료되었습니다(로드 밸런서가 응답을 보낼 수 없음).

솔루션 2: 요청을 하는 머신에서 패킷 스니퍼를 사용하여 응답을 보내기 전에 클라이언트가 연결을 종료하지 않는지 확인합니다.

HTTP 502: 잘못된 게이트웨이

설명: 로드 밸런서가 등록된 인스턴스에서 보낸 응답을 구문 분석할 수 없다는 것을 나타냅니다.

원인: 인스턴스의 응답이 잘못된 형식이거나 로드 밸런서의 문제일 수 있습니다.

솔루션: 인스턴스에서 보낸 응답이 HTTP 사양을 준수하는지 확인합니다. AWS Support 센터로 이동하여 지원을 요청합니다.

HTTP 503: 서비스 사용 불가

설명: 로드 밸런서 또는 등록된 인스턴스가 오류의 원인이라는 것을 나타냅니다.

원인 1: 요청을 처리할 로드 밸런서의 용량이 부족합니다.

솔루션 1: 이는 일시적인 문제여야 하며 몇 분 이상 지속되어서는 안 됩니다. 문제가 계속되면 AWS Support 센터로 이동하여 지원을 요청합니다.

원인 2: 등록된 인스턴스가 없습니다.

솔루션 2: 로드 밸런서가 응답하도록 구성된 모든 가용 영역에 인스턴스를 하나 이상 등록합니다. CloudWatch의 `HealthyHostCount` 지표를 보고 확인합니다. 인스턴스가 각 가용 영역에 등록되어 있는지 확인할 수 없는 경우 교차 영역 로드 밸런싱을 활성화하는 것이 좋습니다. 자세한 내용은 Classic Load Balancer에서 교차 영역 로드 밸런싱을 구성 단원을 참조하십시오.

원인 3: 정상적인 인스턴스가 없습니다.

솔루션 3: 로드 밸런서가 응답하도록 구성된 모든 가용 영역에 정상적인 인스턴스가 있는지 확인합니다. `HealthyHostCount` 지표를 보고 확인합니다.

원인 4: 서지 대기열이 가득 찼습니다.

원인 4: 인스턴스에 요청 빈도를 처리할 충분한 용량이 있는지 확인합니다. `SpilloverCount` 지표를 보고 확인합니다.

HTTP 504: 게이트웨이 제한 시간

설명: 유휴 제한 시간 내에 요청이 완료되지 않았기 때문에 로드 밸런서가 연결을 종료했다는 것을 나타냅니다.

원인 1: 애플리케이션은 구성된 유휴 제한 시간보다 응답하는 데 더 오래 걸립니다.

솔루션 1: `HTTPCode_ELB_5XX` 및 `Latency` 지표를 모니터링합니다. 이러한 지표가 증가하면 유휴 제한 시간 내에 애플리케이션이 응답하지 않을 수 있습니다. 시간이 초과된 요청에 대한 세부 정보에 대해 로드 밸런서에서 액세스 로그를 활성화하고 Elastic Load Balancing이 생성한 로그의 504 응답 코드를 검토합니다. 필요한 경우 용량을 늘리거나 구성된 유휴 제한 시간을 늘려 오래 걸리는 작업(예: 대용량 파일 업로드)을 완료할 수 있습니다. 자세한 내용은 Classic Load Balancer에 대한 유휴 연결 제한 시간 구성 및 Elastic Load Balancing 높은 지연 시간을 어떻게 해결합니까?를 참조하세요.

원인 2: 등록된 인스턴스가 Elastic Load Balancing에 대한 연결을 닫습니다.

솔루션 2: EC2 인스턴스에서 연결 유지 설정을 활성화하고 연결 유지 제한 시간이 로드 밸런서의 유휴 제한 시간 설정보다 큰지 확인합니다.

Classic Load Balancer 문제 해결: 응답 코드 지표

로드 밸런서는 클라이언트에 전송된 HTTP 응답 코드에 대한 지표를 Amazon CloudWatch로 전송하여 오류의 원인을 로드 밸런서 또는 등록된 인스턴스로 식별합니다. 로드 밸런서에 대해 CloudWatch가 반환한 지표를 사용하여 문제를 해결할 수 있습니다. 자세한 내용은 Classic Load Balancer의 CloudWatch 지표 단원을 참조하십시오.

다음은 로드 밸런서, 잠재적인 원인 및 문제를 해결하기 위해 수행할 수 있는 단계에 대해 CloudWatch가 반환한 응답 코드 지표입니다.

응답 코드 지표

- HTTPCode_ELB_4XX
- HTTPCode_ELB_5XX
- HTTPCode_Backend_2XX
- HTTPCode_Backend_3XX
- HTTPCode_Backend_4XX
- HTTPCode_Backend_5XX

HTTPCode_ELB_4XX

원인: 클라이언트의 잘못된 형식 요청 또는 요청 취소.

Solutions

- HTTP 400: BAD_REQUEST을(를) 참조하세요.
- HTTP 405: METHOD_NOT_ALLOWED을(를) 참조하세요.
- HTTP 408: 요청 제한 시간을(를) 참조하세요.

HTTPCode_ELB_5XX

원인: 로드 밸런서 또는 등록된 인스턴스가 오류의 원인이거나 로드 밸런서가 응답을 구문 분석할 수 없습니다.

Solutions

- HTTP 502: 잘못된 게이트웨이을(를) 참조하세요.
- HTTP 503: 서비스 사용 불가을(를) 참조하세요.
- HTTP 504: 게이트웨이 제한 시간을(를) 참조하세요.

HTTPCode_Backend_2XX

원인: 등록된 인스턴스의 정상적이고 성공적인 응답.

솔루션: 없음.

HTTPCode_Backend_3XX

원인: 등록된 인스턴스에서 보낸 리디렉션 응답.

솔루션:인스턴스에서 액세스 로그 또는 오류 로그를 보고 원인을 판단합니다. 응답을 보려면 인스턴스에 직접 요청을 보냅니다(로드 밸런서 우회).

HTTPCode_Backend_4XX

원인: 등록된 인스턴스에서 보낸 클라이언트 오류 응답.

솔루션:인스턴스에서 액세스 또는 오류 로그를 보고 원인을 판단합니다. 응답을 보려면 인스턴스에 직접 요청을 보냅니다(로드 밸런서 우회).

> **Note**
>
> 클라이언트가 `Transfer-Encoding: chunked` 헤더로 시작된 HTTP 요청을 취소하는 경우, 클라이언트가 요청을 취소했더라도 로드 밸런서가 요청을 인스턴스로 전달하는 문제가 알려진 바 있습니다. 이는 백엔드 오류의 원인이 될 수 있습니다.

HTTPCode_Backend_5XX

원인: 등록된 인스턴스에서 보낸 서버 오류 응답.

솔루션:인스턴스에서 액세스 로그 또는 오류 로그를 보고 원인을 판단합니다. 응답을 보려면 인스턴스에 직접 요청을 보냅니다(로드 밸런서 우회).

> **Note**
>
> 클라이언트가 `Transfer-Encoding: chunked` 헤더로 시작된 HTTP 요청을 취소하는 경우, 클라이언트가 요청을 취소했더라도 로드 밸런서가 요청을 인스턴스로 전달하는 문제가 알려진 바 있습니다. 이는 백엔드 오류의 원인이 될 수 있습니다.

Classic Load Balancer 문제 해결: 상태 확인

로드 밸런서는 Elastic Load Balancing이 제공하는 기본적인 상태 확인 구성이나 사용자가 지정한 상태 확인 구성을 사용하여 등록된 인스턴스의 상태를 확인합니다. 상태 확인 구성에는 프로토콜, ping 포트, ping 경로, 응답 시간 초과 및 상태 확인 간격과 같은 정보가 있습니다. 상태 확인 간격 내에 200 응답 코드가 반환되면 해당 인스턴스는 정상으로 간주됩니다. 자세한 내용은 Classic Load Balancer의 인스턴스 상태 확인 단원을 참조하십시오.

일부 또는 모든 인스턴스의 현재 상태가 `OutOfService`이고 설명 필드에 `Instance has failed at least the Unhealthy Threshold number of health checks consecutively` 메시지가 표시되는 경우, 인스턴스가 로드 밸런서 상태 확인에 실패한 것입니다. 다음은 살펴봐야 할 문제, 잠재적인 원인 및 문제를 해결하기 위해 수행할 수 있는 단계입니다.

문제

- 상태 확인 대상 페이지 오류
- 인스턴스 연결 시간 초과
- 퍼블릭 키 인증이 실패함
- 인스턴스가 로드 밸런서에서 트래픽을 수신하지 않음
- 인스턴스의 포트가 열려 있지 않음
- Auto Scaling 그룹의 인스턴스가 ELB 상태 확인에 실패함

상태 확인 대상 페이지 오류

문제: 지정된 ping 포트 및 ping 경로(예: HTTP:80/index.html)의 인스턴스에 발행된 HTTP GET 요청은 200이 아닌 응답 코드를 수신합니다.

원인 1: 인스턴스에 대상 페이지가 구성되어 있지 않습니다.

솔루션 1: 등록된 각 인스턴스에서 대상 페이지(예: `index.html`)를 만들고 해당 경로를 ping 경로로 지정합니다.

원인 2: 응답의 Content-Length 헤더 값이 설정되어 있지 않습니다.

솔루션 2: 응답에 본문이 포함되어 있으면 0보다 크거나 같은 값으로 Content-Length 헤더를 설정하거나 Transfer-Encoding 값을 'chunked'로 설정합니다.

원인 3: 애플리케이션이 로드 밸런서에 요청을 받거나 200 응답 코드를 반환하도록 구성되어 있지 않습니다.

솔루션 3: 인스턴스에서 애플리케이션을 확인하여 원인을 검사합니다.

인스턴스 연결 시간 초과

문제: EC2 인스턴스에 대한 로드 밸런서의 상태 확인 요청이 시간 초과되거나 간헐적으로 실패합니다.

먼저 인스턴스와 직접 연결하여 문제를 확인합니다. 인스턴스의 프라이빗 IP 주소를 사용하여 네트워크 내에서 인스턴스에 연결하는 것이 좋습니다.

TCP 연결에 대해 다음 명령을 사용합니다.

```
telnet private-IP-address-of-the-instance port
```

HTTP 또는 HTTPS 연결에 대해 다음 명령을 사용합니다.

```
curl -I private-IP-address-of-the-instance:port/health-check-target-page
```

HTTP/HTTPS 연결을 사용하고 있고 200이 아닌 응답을 받고 있는 경우 상태 확인 대상 페이지 오류 단원을 참조하십시오. 직접 연결할 수 있는 경우 다음 사항을 확인합니다.

원인 1: 인스턴스가 구성된 응답 제한 시간 내에 응답하지 않습니다.

솔루션 1: 로드 밸런서 상태 확인 구성에서 응답 제한 시간을 조정합니다.

원인 2: 인스턴스에 상당한 로드가 발생했으며 구성된 응답 제한 시간보다 응답을 하는 데 오래 걸립니다.

솔루션 2:

- CPU의 과다 사용에 대한 모니터링 그래프를 확인합니다. 자세한 내용은 Amazon EC2 사용 설명서의 특정 EC2 인스턴스에 대한 통계 가져오기를 참조하세요.
- EC2 인스턴스에 연결하여 메모리 또는 한도 같은 다른 애플리케이션 리소스의 사용률을 확인합니다.

- 필요에 따라 인스턴스를 더 추가하거나 Auto Scaling을 활성화합니다. 자세한 내용은 Amazon EC2 Auto Scaling 사용 설명서를 참조하세요.

원인 3: HTTP 또는 HTTPS 연결을 사용 중이며 ping 경로 필드에 지정된 대상 페이지(예: `HTTP:80/index.html`) 상태 확인이 수행되고 있는 경우, 대상 페이지가 응답하는 데 구성된 제한 시간보다 오래 걸릴 수 있습니다.

솔루션 3: 더 간단한 상태 확인 대상 페이지를 사용하거나 상태 확인 간격 설정을 조정합니다.

퍼블릭 키 인증이 실패함

문제: 백엔드 인증이 활성화된 HTTPS 또는 SSL 프로토콜을 사용하도록 구성된 로드 밸런서가 퍼블릭 키 인증을 실패합니다.

원인: SSL 인증서의 퍼블릭 키가 로드 밸런서에 구성된 퍼블릭 키와 일치하지 않습니다. `s_client` 명령을 사용하여 인증서 체인에 있는 서버 인증서 목록을 확인합니다. 자세한 내용은 OpenSSL 설명서의 s_client를 참조하십시오.

솔루션: SSL 인증서를 업데이트해야 할 수 있습니다. SSL 인증서가 최신 상태인 경우 로드 밸런서에 SSL 인증서를 다시 설치해 봅니다. 자세한 내용은 Classic Load Balancer를 위한 SSL 인증서 교체 단원을 참조하십시오.

인스턴스가 로드 밸런서에서 트래픽을 수신하지 않음

문제: 인스턴스에 대한 보안 그룹이 로드 밸런서의 트래픽을 차단하고 있습니다.

인스턴스에서 패킷을 캡처하여 문제를 확인합니다. 다음 명령을 사용합니다.

```
# tcpdump port health-check-port
```

원인 1: 인스턴스와 연결된 보안 그룹에서 로드 밸런서의 트래픽을 허용하지 않습니다.

솔루션 1: 로드 밸런서의 트래픽을 허용하도록 인스턴스 보안 그룹을 편집합니다. 로드 밸런서 보안 그룹의 모든 트래픽을 허용하는 규칙을 추가합니다.

원인 2: 로드 밸런서의 보안 그룹은 EC2 인스턴스에 대한 트래픽을 허용하지 않습니다.

솔루션 2: 로드 밸런서의 보안 그룹을 편집하여 서브넷 및 EC2 인스턴스에 대한 트래픽을 허용합니다.

보안 그룹 관리에 대한 자세한 내용은 Classic Load Balancer 보안 그룹 구성을(를) 참조하세요.

인스턴스의 포트가 열려 있지 않음

문제: 로드 밸런서가 EC2 인스턴스로 전송한 상태 확인이 포트 또는 방화벽에 의해 차단됩니다.

다음 명령을 사용하여 문제를 확인합니다.

```
netstat -ant
```

원인: 지정된 상태 확인 포트 또는 리스너 포트(다르게 구성된 경우)가 열려 있지 않습니다. 상태 확인을 위해 지정된 포트와 리스너 포트는 모두 열려 있어야 하며 수신 대기해야 합니다.

솔루션: 인스턴스의 상태 확인 구성에 지정된 리스너 포트 및 포트를 열고(다르게 구성된 경우) 로드 밸런서 트래픽을 수신합니다.

Auto Scaling 그룹의 인스턴스가 ELB 상태 확인에 실패함

문제: Auto Scaling 그룹의 인스턴스는 기본 Auto Scaling 상태 확인을 통과했지만 ELB 상태 확인은 통과하지 못했습니다.

원인 :Auto Scaling은 EC2 상태 확인을 사용하여 인스턴스의 하드웨어 및 소프트웨어 문제를 감지하지만, 로드 밸런서는 인스턴스에 요청을 보내고 200 응답 코드를 기다리거나 인스턴스와 TCP 연결을 설정(TCP 기반 상태 확인)하여 상태 확인을 수행합니다.

인스턴스에서 실행 중인 애플리케이션에는 로드 밸런서가 인스턴스를 서비스에서 제외시키는 것을 고려하게 하는 문제가 있기 때문에 인스턴스가 ELB 상태 확인을 실패할 수 있습니다. 이 인스턴스는 Auto Scaling 상태 확인을 통과할 수 있습니다. 즉, EC2 상태 확인을 기반으로 정상적으로 간주되기 때문에 Auto Scaling 정책에 의해 대체되지 않습니다.

솔루션: Auto Scaling 그룹에 대해 ELB 상태 확인을 사용합니다. ELB 상태 확인을 사용하면 Auto Scaling은 인스턴스 상태 확인과 ELB 상태 확인의 결과를 점검하여 인스턴스의 상태를 판단합니다. 자세한 내용은 Amazon EC2 Auto Scaling 사용 설명서의 Auto Scaling 그룹에 Elastic Load Balancing 상태 확인 추가를 참조하세요.

Classic Load Balancer 문제 해결: 클라이언트 연결

클라이언트가 인터넷 경계 로드 밸런서에 연결할 수 없음

로드 밸런서가 요청에 응답하지 않는 경우에는 다음 문제를 점검하세요.

인터넷 경계 로드 밸런서가 프라이빗 서브넷에 연결됩니다.

로드 밸런서를 위한 퍼블릭 서브넷을 지정해야 합니다. 퍼브릭 서브넷은 가상 프라이빗 클라우드(VPC)를 위한 인터넷 게이트웨이로 연결되는 경로를 가지고 있습니다.

보안 그룹이나 네트워크 ACL이 트래픽을 허용하지 않음

로드 밸런서를 위한 보안 그룹과 로드 밸런서 서브넷을 위한 모든 네트워크 ACL은 클라이언트에서의 인바운드 트래픽과 리스너 포트의 클라이언트로의 아웃바운드 트래픽을 허용해야 합니다. 자세한 내용은 Classic Load Balancer 보안 그룹 구성 단원을 참조하십시오.

사용자 지정 도메인으로 전송된 요청은 로드 밸런서에 수신되지 않습니다.

로드 밸런서가 커스텀 도메인에 보낸 요청을 받지 않는 경우에는 다음 문제를 점검하세요.

사용자 지정 도메인 이름이 로드 밸런서 IP 주소로 확인되지 않음

- 명령줄 인터페이스를 사용하여 사용자 지정 도메인 이름이 어떤 IP 주소로 변환되는지 확인합니다.
  - Linux, macOS 또는 Unix — 터미널에서 `dig` 명령을 사용할 수 있습니다. Ex.`dig example.com`
  - Windows — 명령 프롬프트에서 `nslookup` 명령을 사용할 수 있습니다. Ex.`nslookup example.com`
- 명령줄 인터페이스를 사용하여 로드 밸런서 DNS 이름이 어떤 IP 주소로 변환되는지 확인합니다.
- 두 출력의 결과를 비교합니다. IP 주소는 일치해야 합니다.

로드 밸런서로 전송된 HTTPS 요청은 “NET: :ERR_CERT_COMMON_NAME_INVALID”를 반환합니다.

HTTPS 요청이 로드 밸런서에서 `NET::ERR_CERT_COMMON_NAME_INVALID`을 받는 경우 다음과 같은 가능한 원인을 확인하세요.

- HTTPS 요청에 사용된 도메인 이름이 리스너 관련 ACM 인증서에 지정된 대체 이름과 일치하지 않습니다.
- 로드 밸런서의 기본 DNS 이름이 사용되고 있습니다. `*.amazonaws.com` 도메인에 공개 인증서를 요청할 수 없으므로 기본 DNS 이름을 사용하여 HTTPS 요청을 할 수 없습니다.

Classic Load Balancer 문제 해결: 인스턴스 등록

인스턴스를 로드 밸런서에 등록하면 로드 밸런서가 인스턴스에 요청을 보내기 시작할 때까지 여러 가지 단계가 수행됩니다.

다음은 EC2 인스턴스를 등록할 때 로드 밸런서에서 발생할 수 있는 문제, 잠재적인 원인 및 문제를 해결하기 위해 수행할 수 있는 단계입니다.

문제

EC2 인스턴스를 등록하는 데 너무 오래 걸림

문제: 등록된 EC2 인스턴스가 `InService` 상태가 되는 데 예상보다 오래 걸립니다.

원인: 인스턴스가 상태 확인에 실패할 수 있습니다. 초기 인스턴스 등록 단계가 완료되면(최대 약 30초 소요될 수 있음) 로드 밸런서가 상태 확인 요청을 보내기 시작합니다. 인스턴스는 한 번이라도 상태 확인을 통과할 때까지 `InService` 상태가 되지 않습니다.

솔루션: 인스턴스 연결 시간 초과 섹션을 참조하세요.

유료 AMI에서 시작된 인스턴스를 등록할 수 없음

문제: Elastic Load Balancing은 유료 AMI를 사용하여 시작된 인스턴스를 등록하지 않습니다.

원인: 인스턴스가 Amazon DevPay의 유료 AMI를 사용하여 시작되었을 수 있습니다.

솔루션: Elastic Load Balancing은 Amazon DevPay의 유료 AMI를 사용하여 시작된 인스턴스 등록을 지원하지 않습니다. AWS Marketplace의 유료 AMI를 사용할 수 있다는 점에 유의하세요. 에서 이미 유료 AMI를 사용하고 AWS Marketplace 있고 해당 유료 AMI에서 시작된 인스턴스를 등록할 수 없는 경우 AWS Support 센터로 이동하여 지원을 받으세요.

Classic Load Balancer에 대한 할당량

AWS 계정에는 각 AWS 서비스에 대해 이전에 제한이라고 했던 기본 할당량이 있습니다. 다르게 표시되지 않는 한, 리전별로 각 할당량이 적용됩니다.

Classic Load Balancer에 대한 할당량을 보려면 Service Quotas 콘솔을 엽니다. 탐색 창에서 AWS 서비스를 선택하고 Elastic Load Balancing을 선택합니다. Elastic Load Balancing에 대해 describe-account-limits(AWS CLI) 명령을 사용할 수도 있습니다.

할당량 증가를 요청하려면 Service Quotas 사용 설명서의 할당량 증가 요청을 참조하세요.

AWS 계정에는 Classic Load Balancer와 관련된 다음과 같은 할당량이 있습니다.

이름	기본값	조정 가능
리전당 Classic Load Balancer	20	예
Classic Load Balancer당 리스너	100	예
Classic Load Balancer당 등록된 인스턴스	1,000	예

Classic Load Balancer에 대한 문서 기록

다음 표에서는 Classic Load Balancer 릴리스를 설명합니다.

변경 사항	설명	날짜
액세스 로그 및 연결 로그에 대한 버킷 정책	이번 릴리스 이전에는 사용하는 버킷 정책이 해당 리전이 2022년 8월 이전에 사용 가능했는지 이후에 사용 가능했는지 여부에 따라 달랐습니다. 이번 릴리스부터 모든 리전에서 새로운 버킷 정책이 지원됩니다. 레거시 버킷 정책도 계속 지원됩니다.	2025년 9월 10일
Desync Mitigation Mode	Desync Mitigation Mode에 대한 지원이 추가되었습니다. 자세한 내용은 Classic Load Balancer에 대한 동기화 해제 완화 모드 구성을 참조하세요.	2020년 8월 17일
Classic Load Balancer	Application Load Balancer 및 Network Load Balancer가 도입됨에 따라 2016-06-01 API로 생성된 로드 밸런서를 이제 Classic Load Balancer라고 합니다. 이러한 로드 밸런서 유형 간의 차이점을 자세히 알아보려면 웹 사이트에서 Elastic Load Balancing 기능 부분을 참조하세요.	2016년 8월 11일
에 대한 지원 AWS Certificate Manager (ACM)	ACM에서 SSL/TLS 인증서를 요청해서 이를 로드 밸런서에 배포할 수 있습니다. 자세한 내용은 Classic Load Balancer를	2016년 1월 21일

	위한 SSL/TLS 인증서를 참조하세요.	
추가 포트 지원	로드 밸런서는 1~65535 범위의 모든 포트에서 수신 대기할 수 있습니다. 자세한 내용은 Classic Load Balancer의 리스너를 참조하세요.	2015년 9월 15일
액세스 로그 항목을 위한 추가 필드	`user_agent` , `ssl_cipher` 및 `ssl_protocol` 필드를 추가했습니다. 자세한 내용은 액세스 로그 파일을 참조하세요.	2015년 5월 18일
로드 밸런서에 대한 태그 지정 지원	이 릴리스부터 Elastic Load Balancing CLI(ELB CLI)는 여러 AWS 서비스를 관리하는 통합 도구인 AWS Command Line Interface (AWS CLI)로 대체되었습니다. ELB CLI 버전 1.0.35.0(2014년 7월 24일 출시) 이후에 발표된 새로운 기능들은 AWS CLI 에만 포함이 됩니다. 현재 ELB CLI를 사용 중인 경우 대신, AWS CLI 를 사용하는 것이 좋습니다. 자세한 내용은 AWS Command Line Interface 사용 설명서를 참조하십시오.	2014년 8월 11일
유휴 연결 제한 시간	로드 밸런서에 대한 유휴 연결 제한 시간을 구성할 수 있습니다.	2014년 7월 24일

사용자 및 그룹에게 특정 로드 밸런서나 API 작업에 대한 액세스 권한을 부여하도록 지원	사용자 및 그룹에게 특정 로드 밸런서나 API 작업에 대한 액세스 권한을 부여하도록 정책을 생성할 수 있습니다.	2014년 5월 12일
에 대한 지원 AWS CloudTrail	CloudTrail을 사용하여 ELB API, , ELB CLI 또는를 AWS 계정 사용하여에 의해 또는를 대신하여 수행된 API 호출 AWS Management Console을 캡처할 수 있습니다 AWS CLI.	2014년 4월 4일
연결 드레이닝	Connection Draining에 대한 정보를 추가했습니다. 이 기능 덕분에 기존 연결이 열려 있는 상태에서 인스턴스가 재등록 중이거나 정상 상태가 아닐 때 로드 밸런서가 등록된 인스턴스에 새로운 요청을 전송하는 것을 중지하도록 할 수 있습니다. 자세한 내용은 Classic Load Balancer에서 연결 드레이닝 구성을 참조하세요.	2014년 3월 20일
액세스 로그	로드 밸런서가 로드 밸런서에 전송된 요청에 대한 상세 정보를 캡처해서 Amazon S3 버킷에 저장하도록 할 수 있습니다. 자세한 내용은 Classic Load Balancer 액세스 로그를 참조하세요.	2014년 3월 6일

TLSv1.1-1.2 지원	HTTPS/SSL 리스너로 구성된 로드 밸런서를 위한 TLSv1.1-1.2 프로토콜 지원에 대한 정보를 추가했습니다. 이 기능 덕분에 탄력적 로드 밸런싱 역시 사전 정의된 SSL 협상 구성을 업데이트할 수 있습니다. 업데이트된 사전 정의 SSL 협상 구성에 대한 자세한 내용은 Classic Load Balancer를 위한 SSL 협상 구성을 참조하세요. 현재 SSL 협상 구성을 업데이트하는 방법에 대한 자세한 내용은 Classic Load Balancer의 SSL 협상 구성 업데이트를 참조하세요.	2014년 2월 19일
교차 영역 로드 밸런싱	로드 밸런서에서의 교차 영역 로드 밸런싱 지원에 대한 정보를 추가했습니다. 자세한 내용은 Classic Load Balancer에서 교차 영역 로드 밸런싱 구성을 참조하세요.	2013년 11월 6일
추가적인 CloudWatch 지표	Elastic Load Balancing이 보고한 추가 Cloudwatch 지표에 대한 정보를 추가했습니다. 자세한 내용은 Classic Load Balancer의 CloudWatch 지표를 참조하세요.	2013년 10월 28일
프록시 프로토콜 지원	TCP/SSL 연결을 위해 구성된 로드 밸런서에 대한 프록시 프로토콜 지원 관련 정보를 추가했습니다. 자세한 내용은 프록시 프로토콜 헤더를 참조하세요.	2013년 7월 30일

DNS 장애 조치 지원	로드 밸런서를 위한 Amazon Route 53 DNS 장애 조치에 대한 정보를 추가했습니다. 자세한 내용은 로드 밸런서에 대한 Amazon Route 53 DNS 장애 조치 사용을 참조하세요.	2013년 6월 3일
CloudWatch 지표를 확인하고 경보를 생성하기 위한 콘솔 지원	콘솔을 사용하여 지정된 로드 밸런서에 대해 CloudWatch 지표를 확인하고 경보를 생성하는 방법에 대한 정보를 추가했습니다. 자세한 내용은 Classic Load Balancer의 CloudWatch 지표를 참조하세요.	2013년 3월 28일
기본 VPC에서 EC2 인스턴스 등록 지원	기본 VPC에서 시작된 EC2 인스턴스에 대한 지원을 추가했습니다.	2013년 3월 11일
내부 로드 밸런서	이 릴리스에서는 가상 프라이빗 클라우드(VPC)의 로드 밸런서를 내부 또는 인터넷 경계로 지정할 수 있습니다. 내부 로드 밸런서는 공개적으로 확인이 가능하며 프라이빗 IP 주소로 변환이 되는 DNS 이름을 가지고 있습니다. 인터넷 경계 로드 밸런서는 공개적으로 확인이 가능하며 퍼블릭 IP 주소로 변환이 되는 DNS 이름을 가지고 있습니다. 자세한 내용은 내부 Classic Load Balancer 생성을 참조하세요.	2012년 6월 10일

리스너, 암호 설정 및 SSL 인증서를 관리할 수 있는 콘솔 지원	자세한 내용은 Classic Load Balancer를 위한 HTTPS 리스너 구성 및 Classic Load Balancer를 위한 SSL 인증서 교체를 참조하세요.	2012년 5월 18일
Amazon VPC에서 Elastic Load Balancing 지원	가상 프라이빗 클라우드(VPC)에서 로드 밸런서를 생성할 수 있는 기능 지원을 추가했습니다.	2011년 11월 21일
Amazon CloudWatch	CloudWatch를 사용하여 로드 밸런서를 모니터링할 수 있습니다. 자세한 내용은 Classic Load Balancer의 CloudWatch 지표를 참조하세요.	2011년 10월 17일
추가 보안 기능	SSL 암호, 백엔드 SSL 및 백엔드 서버 인증을 구성할 수 있습니다. 자세한 내용은 HTTPS 리스너를 통해 Classic Load Balancer를 생성을 참조하세요.	2011년 8월 30일
Zone apex 도메인 이름	자세한 내용은 Classic Load Balancer에 대한 사용자 지정 도메인 이름 구성을 참조하세요.	2011년 5월 24일

X-Forwarded-Proto 및 X-Forwarded-Port 헤더 지원	X-Forwarded-Proto 헤더는 원본 요청의 프로토콜을, X-Forwarded-Port 헤더는 원본 요청의 포트를 나타냅니다. 이들 헤더를 요청에 추가하면 고객들이 로드 밸런서로 수신되는 요청이 암호화되어 있는지 여부와 요청이 수신된 로드 밸런서의 특정 포트를 확인할 수 있습니다. 자세한 내용은 HTTP 헤더 및 Classic Load Balancer를 참조하세요.	2010년 10월 27일
HTTPS 지원	이 릴리스에서는 트래픽 암호화를 위해 SSL/TLS 프로토콜을 활용하고 애플리케이션 인스턴스에서 로드 밸런서로 SSL 처리를 오프로드할 수 있습니다. 또한 이 기능은 개별 애플리케이션 인스턴스에서 인증서를 관리하기 보다는 로드 밸런서에서 중앙집중식으로 SSL 서버 인증서를 관리할 수 있도록 해줍니다.	2010년 10월 14일
에 대한 지원 AWS Identity and Access Management (IAM)	IAM에 대한 지원이 추가되었습니다.	2010년 9월 2일
고정 세션	자세한 내용은 Classic Load Balancer를 위한 스티키 세션 구성을 참조하세요.	2010년 4월 7일
AWS SDK for Java	자바용 SDK에 대한 지원이 추가되었습니다.	2010년 3월 22일
AWS SDK for .NET	에 대한 지원이 추가되었습니다 SDK for .NET.	2009년 11월 11일

새로운 서비스	Elastic Load Balancing의 초기 공개 베타 릴리스입니다.	2009년 5월 18일

기계 번역으로 제공되는 번역입니다. 제공된 번역과 원본 영어의 내용이 상충하는 경우에는 영어 버전이 우선합니다.