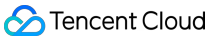
Cloud Load Balancer

사례 튜토리얼

제품 문서

저작권 고지

상표 고지



서비스 고지

본 문서는 고객에게 텐센트 클라우드(Tencent Cloud) 제품 및 서비스 전부 또는 일부에 대한 현재적 개괄적 정보를 제공 하기 위한 것으로, 특정 제품·서비스의 내용은 수시로 변경될 수 있습니다. 고객이 실제 구매한 제품·서비스의 적용 기준은 고객과 텐센트 클라우드간 체결된 상업계약에 명시된 내용이 우선하며, 별도 서면 합의가 없는 한 텐센트 클라우드는 본 문서 내용에 대해 법적 효력이 있는 명시적·묵시적 진술이나 보증을 일체 하지 않습니다.

목록:

사례 튜토리얼
CLB에 인증서 배치(양방향 인증)

최종 업데이트 날짜:: 2025-06-25 16:37:22

작업 시나리오

기존의 단방향 인증에서는 클라이언트가 서버의 신원만을 검증합니다. 하지만 보안 요구가 높은 시나리오에서는 단방향 인증만으로는 충분하지 않습니다. 양방향 인증은 클라이언트와 서버가 서로의 신원을 모두 검증해야 하며, 이는 통신 보안을 강화하고 중간자 공격, 신원 위조 및 데이터 유출을 효과적으로 방지할 수 있습니다. 단방향 인증과 양방향 인증에 대한 설명은 단방향 인증 및 양방향 인증 설명 을 참조하십시오.

전제 조건

1. 이미 CLB 인스턴스를 생성했습니다. 자세한 내용은 CLB 인스턴스 생성 을 참조하십시오.
2. 도메인 이름을 통해 접근하려면, 도메인을 보유하고 있으며 CLB 포워딩 도메인 이름 설정이 완료되어 있어야 합니다. 자세한 내용은 CLB 포워딩 도메인 이름 설정 을 참조하십시오.
3. 이미 CVM 인스턴스 rs-1 및 rs-2를 생성하여 CLB 인스턴스의 리얼 서버로 사용하였습니다. 자세한 내용은 리얼 서버 를 참조하십시오.

설정 절차

다음은 자체 서명 인증서를. HTTPS 양방향 인증 CLB 설정 과정을 아래의 순서대로 진행합니다.
1. CA 인증서: 인증 기관의 인증서로서 서버 인증서 또는 클라이언트 인증서를 발급하는 데 사용할 수 있습니다.
2. 서버 인증서: 구매 또는 자체 서명을 통해 획득할 수 있습니다.
3. 클라이언트 인증서: 획득한 CA 인증서를 이용해 클라이언트 인증서를 발급할 수 있습니다.
4. 인증서 업로드: CA 인증서를 인증서 플랫폼에 업로드하고, 구매하거나 자체 서명한 서버 인증서도 함께 업로드해야 합니다.
5. CLB 설정: HTTPS 리스너를 설정할 때 양방향 인증을 활성화해야 하며, 서버 인증서는 업로드한 인증서를 선택하고, CA 인증서는 자체 서명 생성된 CA 인증서 ca.crt를 선택합니다.
6. 클라이언트 인증서 가져오기: 요청 시 사용할 클라이언트 인증서를 클라이언트로 가져와야 합니다.

절차 1: CA 인증서 획득

1. 다음 명령어를 실행하여 CA 인증서의 개인 키 파일 ca.key를 생성합니다.

```
# CA 개인 키 생성
openssl genrsa -out ca.key 2048
```

2. 다음 명령어를 실행하여 CA 인증서의 요청 파일 ca.csr을 생성합니다.

```
# CA 인증서 요청 파일 생성
openssl req -new -key ca.key -out ca.csr
```

> ⚠ **주의:**
> 다음 매개변수는 사용자가 직접 입력해야 하며, Common Name 값은 서버 또는 클라이언트 인증서의 Common Name과 동일하지 않아야 합니다.

```
Country Name (2 letter code) [AU]:sz
State or Province Name (full name) [Some-State]:sz
Locality Name (eg, city) []:sz
Organization Name (eg, company) [Internet Widgits Pty Ltd]:sz
Organizational Unit Name (eg, section) []:sz
Common Name (e.g. server FQDN or YOUR name) []:clb
Email Address []:1.qq.com

Please enter the following 'extra' attributes
to be sent with your certificate request
A challenge password []:
An optional company name []:
```

3. 다음 명령어를 실행하여 자체 서명된 CA 인증서 ca.crt를 생성합니다.

```
# 자체 서명으로 CA 인증서 생성, 유효기간 3650일
openssl x509 -req -in ca.csr -out ca.crt -signkey ca.key -days 3650
```

성공적으로 실행되면 다음과 같이 나타납니다.

```
Certificate request self-signature ok
subject=C = sz, ST = sz, L = sz, O = sz, OU = sz, CN = clb, emailAddress = 1.qq.com
```

절차 2: 서버 인증서 획득

1. 다음 명령어를 실행하여 서버 인증서의 개인 키 파일 server.key를 생성합니다.

```
# 서버 개인 키 생성
openssl genrsa -out server.key 2048
```

2. 다음 명령어를 실행하여 서버 인증서의 요청 파일 server.csr을 생성합니다.

```
# 서버 인증서 요청 파일 생성
```

```
openssl req -new -key server.key -out server.csr
```

3. 다음 명령어를 실행하여 CA 인증서로 서버 인증서 server.crt를 발급합니다.

```
# CA 인증서로 서명하여 서버 인증서 생성, 유효기간 365일
openssl x509 -req -in server.csr -out server.crt -CA ca.crt -CAkey
ca.key -CAcreateserial -days 3650
```

절차 3: 클라이언트 인증서 획득

1. 다음 명령어를 실행하여 클라이언트 인증서의 개인 키 파일 client.key를 생성합니다.

```
# 클라이언트 개인 키 생성
openssl genrsa -out client.key 2048
```

2. 다음 명령어를 실행하여 클라이언트 인증서의 요청 파일 client.csr을 생성합니다.

```
# 클라이언트 인증서 요청 파일 생성
openssl req -new -key client.key -out client.csr
```

3. 다음 명령어를 실행하여 CA 인증서로 클라이언트 인증서 client.crt를 발급합니다.

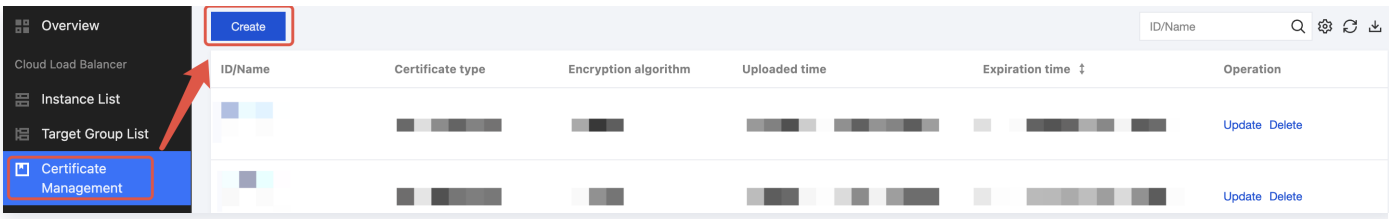
```
# CA 인증서로 서명하여 클라이언트 인증서 생성, 유효기간 365일
openssl x509 -req -in client.csr -out client.crt -CA ca.crt -CAkey
ca.key -CAcreateserial -days 3650
```

4. 다음 명령어를 실행하여 생성된 클라이언트 인증서 client.crt를 브라우저에서 인식할 수 있는 p12 형식 파일로 변환합니다.

```
# 클라이언트 인증서 형식 변환
openssl pkcs12 -export -clcerts -in client.crt -inkey client.key -out
client.p12
```

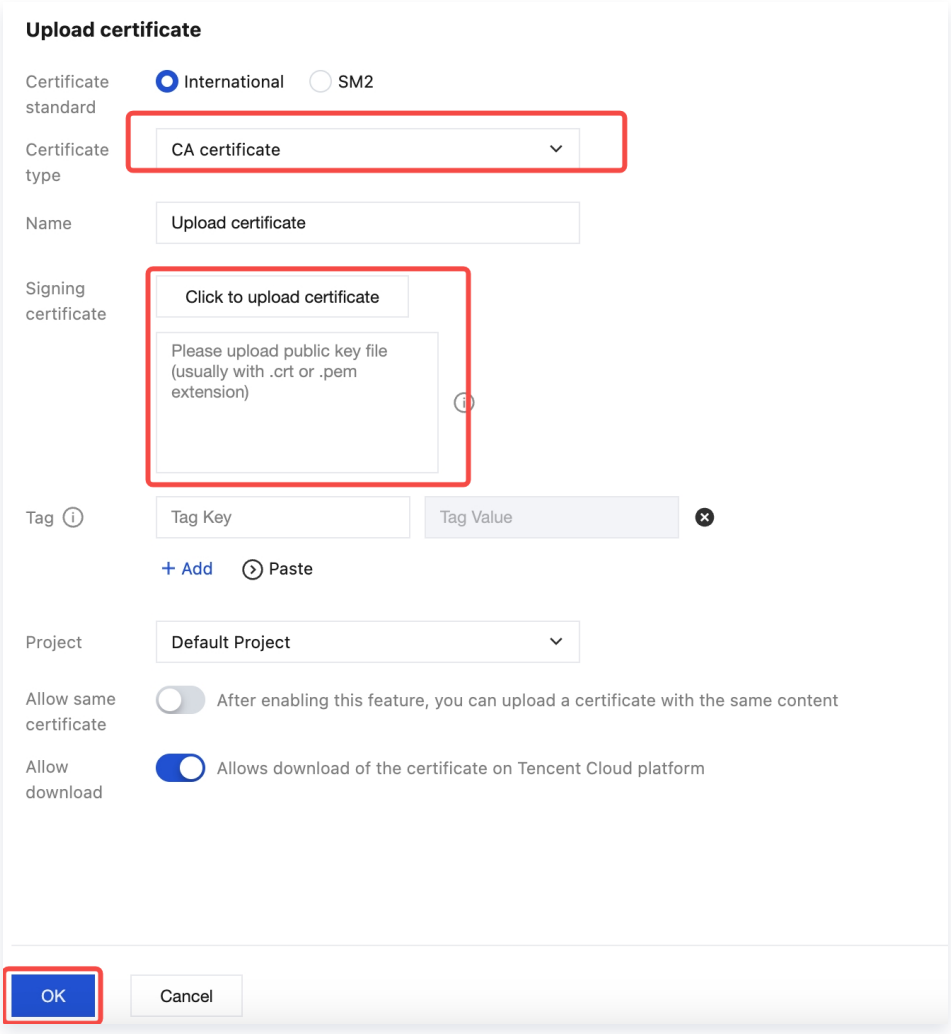
절차 4: CA 인증서 업로드

1. CLB 콘솔에 로그인합니다.
2. **인증서 관리**를 클릭하여 **새로 만들기**를 클릭합니다.



3. 인증서 업로드 페이지에서 인증서 유형은 **CA 인증서**를 선택하고, 절차 1에서 생성한 CA 인증서 ca.crt의 내용을 **서명 인증서** 구역에 복사한 후 **확인**을 클릭합니다.

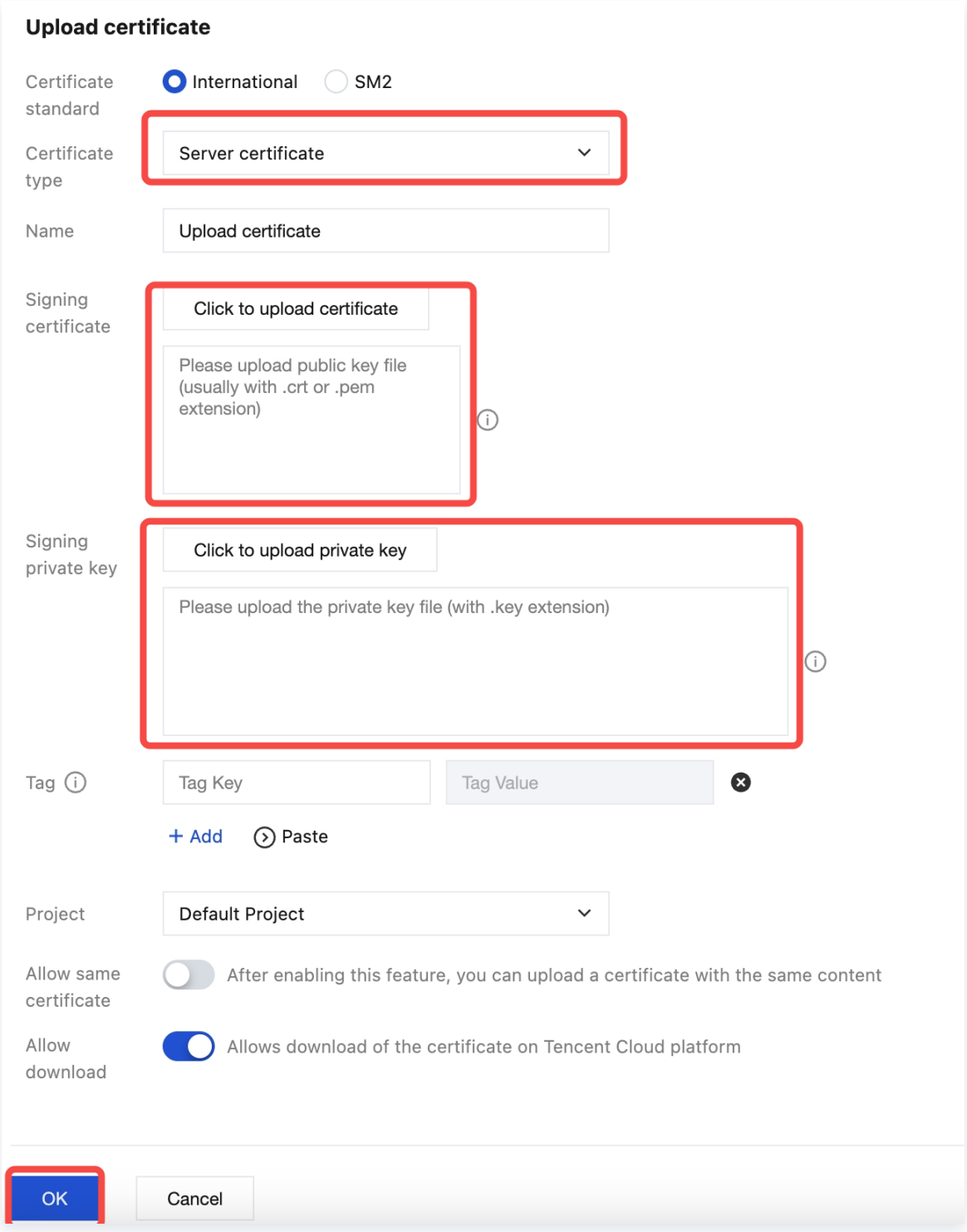
> ⚠ **주의:**
> 복사할 때 마지막 줄바꿈 문자를 삭제하여 오류를 방지하세요.



절차 5: 서버 인증서 업로드

1. CLB 콘솔에 로그인합니다.
2. **인증서 관리**를 클릭하여 **새로 만들기**를 클릭합니다.
3. 인증서 업로드 페이지에서 인증서 유형은 **서버 인증서**를 선택하고, 절차 2 에서 생성된 서버 인증서 server.crt

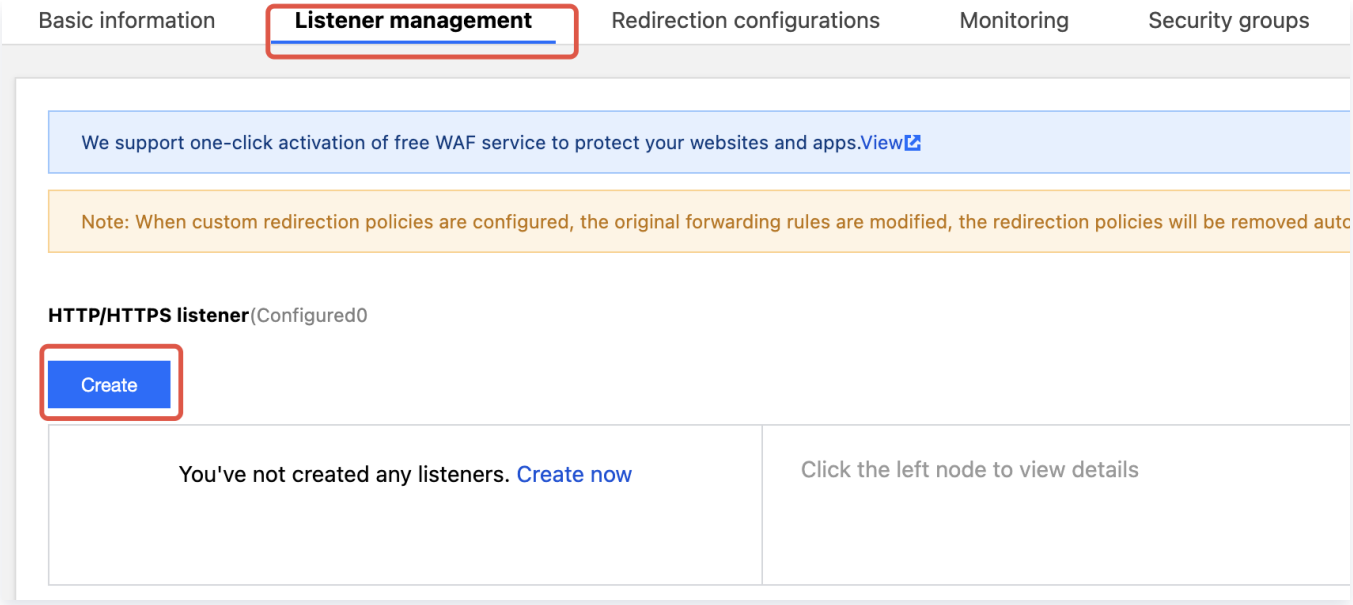
와 서버 개인 키 server.key의 내용을 서명 인증서 구역에 복사한 후 **확인**을 클릭합니다.



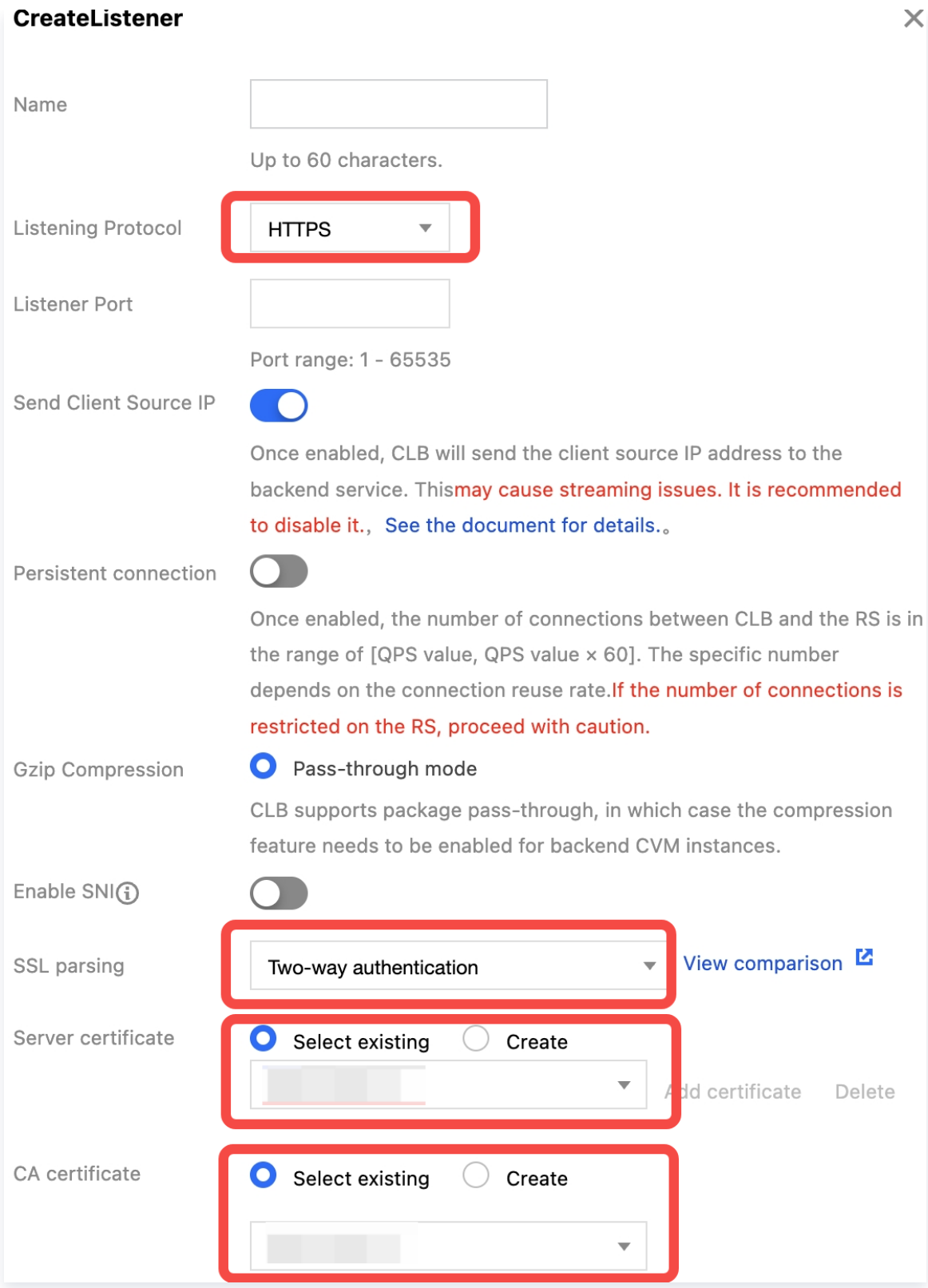
절차 6: HTTPS 양방향 인증 리스너 설정

방법 1: SNI 비활성화

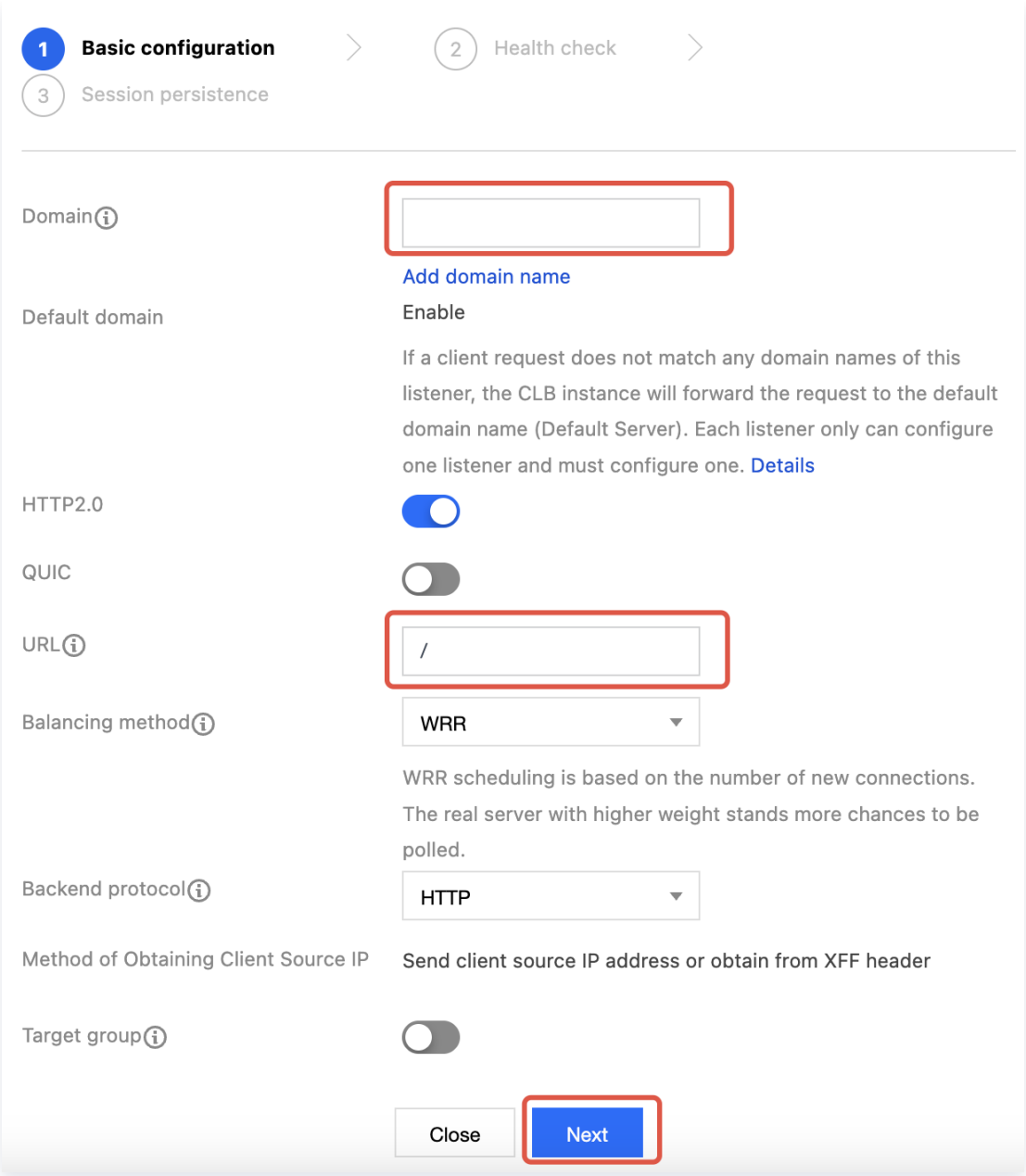
1. CLB 콘솔에 로그인하며, **인스턴스** 페이지에서 대상 CLB 인스턴스를 찾아 인스턴스 ID를 클릭한 후, **리스너 관리** 페이지에서 **새로 만들기**를 클릭합니다.



2. 설정 화면에서 수신 프로토콜을 HTTPS로 선택하고 지정된 포트를 입력합니다. SNI 비활성화 상태에서 양방향 인증 방식을 선택하고, 획득한 서버 인증서와 CA 인증서를 업로드합니다. 설정 정보를 확인한 후 **제출**을 클릭합니다.



3. 더하기(+) 버튼을 클릭하여 포워딩 규칙을 생성합니다; CLB 도메인 이름과 URL 경로를 입력합니다; 밸런싱 방식과 백엔드 프로토콜을 선택하여 기본 설정을 완료합니다.

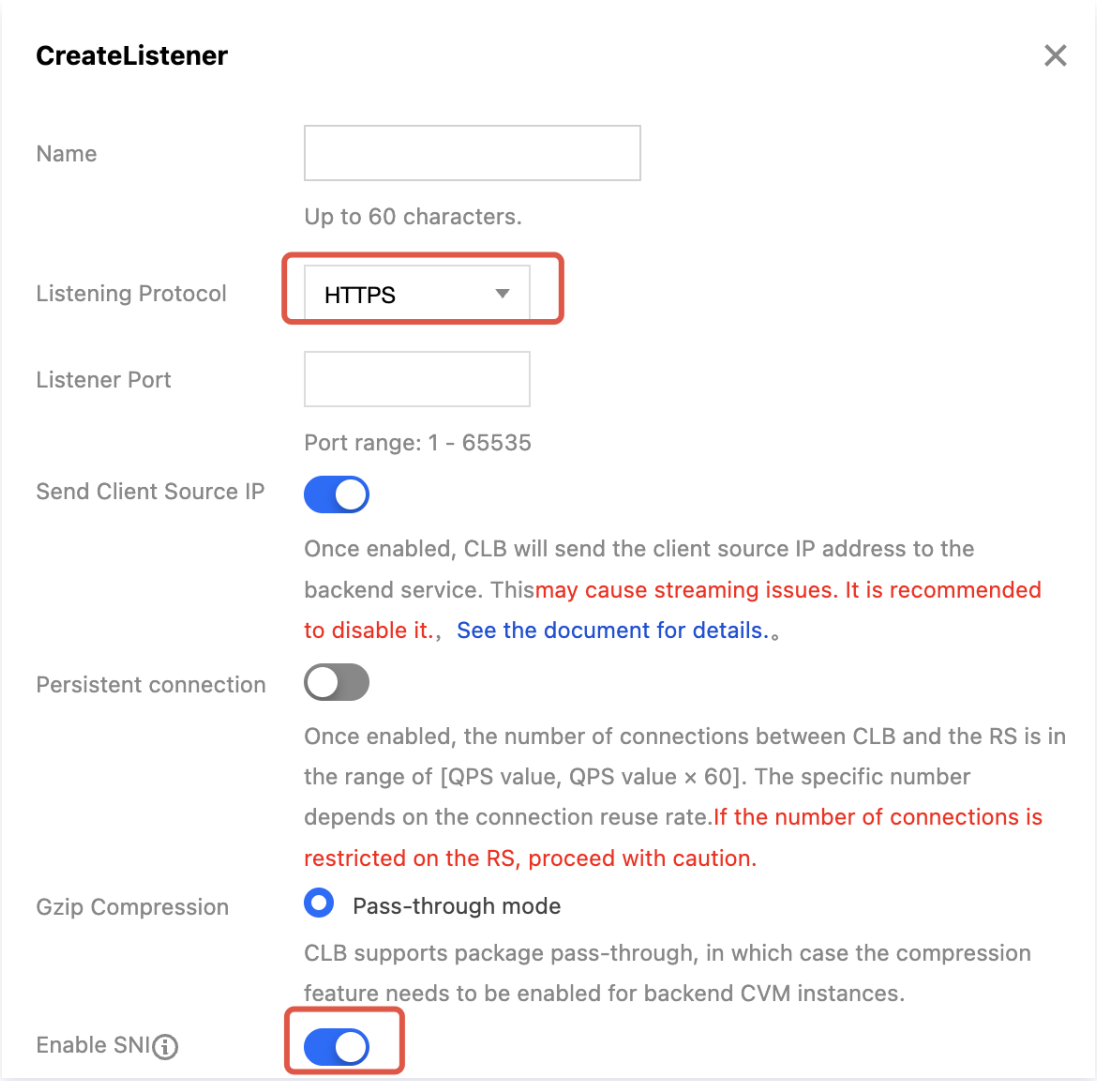


> **설명:**
> HTTPS 수신 프로토콜을 선택하면 클라이언트에서 CLB로 접근할 때 HTTPS를 사용하며, CLB에서 리얼 서버로 포워딩할 때는 HTTP 또는 HTTPS를 선택할 수 있습니다.

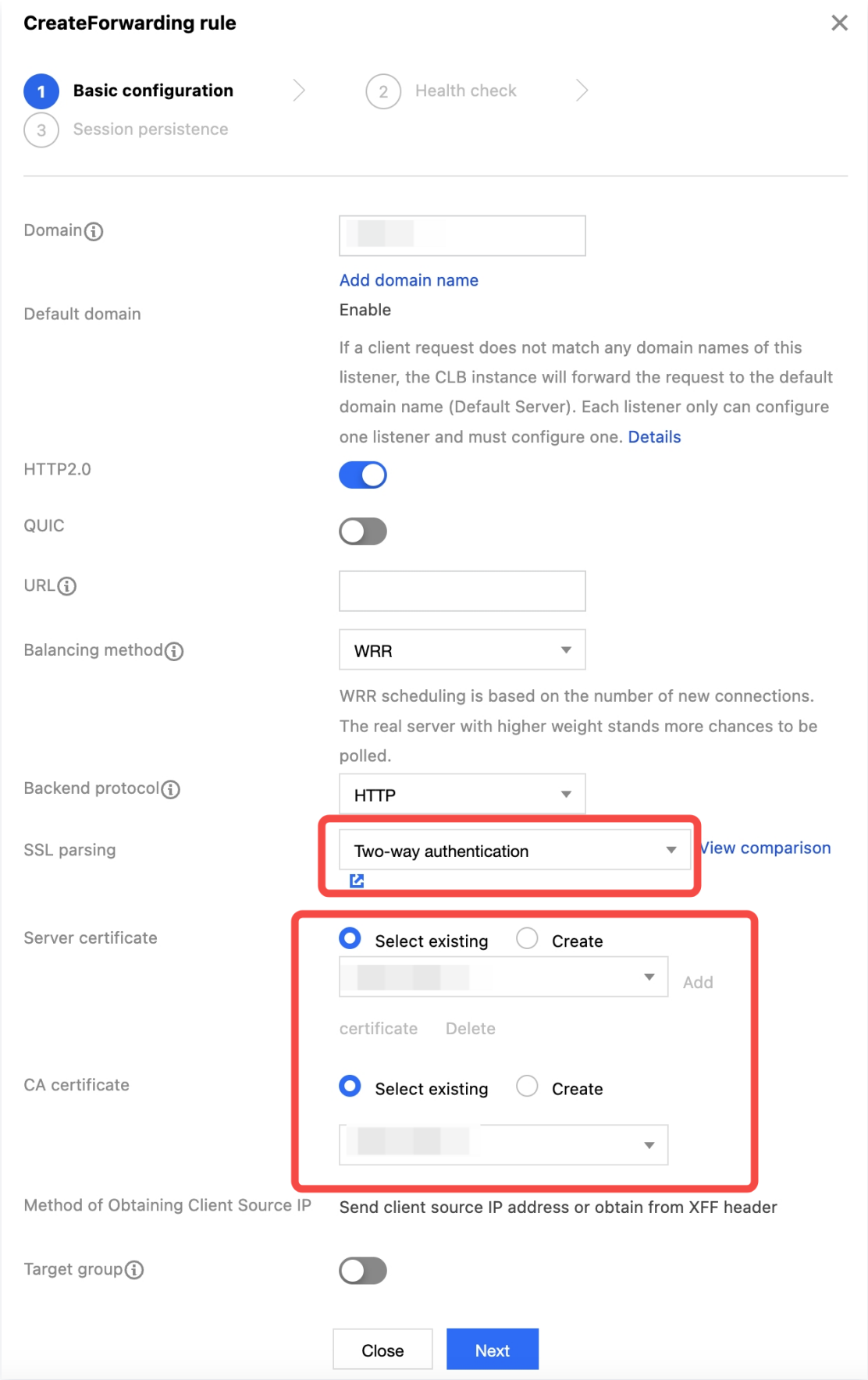
4. 상태 점검 포트를 설정하고 선택에 따라 세션 지속성을 설정합니다.
5. 규칙을 펼친 후 **바인딩**을 클릭하여 생성한 rs-1, rs-2 리얼 서버를 선택합니다.

방법 2: SNI 활성화

1. 설정 화면에서 수신 프로토콜을 **HTTPS**로 선택하고 지정된 포트를 입력합니다. SNI를 활성화한 후 **제출**을 클릭합니다.



2. 더하기(+) 버튼을 클릭하여 포워딩 규칙을 생성합니다. CLB 도메인 이름과 URL 경로를 입력하며, 밸런싱 방식과 백엔드 프로토콜을 선택하여 기본 설정을 완료합니다. 해당 도메인 이름에 해당하는 서버 인증서와 자체 서명한 CA 인증서를 입력합니다.



3. 이후 절차는 SNI 비활성화 설정 방식과 동일합니다. SNI에 대한 자세한 내용은 CLB 인스턴스에 SNI 다중 인증서 바인딩 지원을 참조하십시오.

절차 7: 클라이언트 인증서 가져오기

방법 1: 브라우저 방식

1. 발급된 클라이언트 인증서 client.p12를 다운로드하여 로컬에 가져옵니다.
2. 클라이언트 인증서를 더블 클릭하고 인증서 가져오기 안내에 따라 클라이언트 인증서 설치를 완료합니다.

방법 2: 명령줄 방식

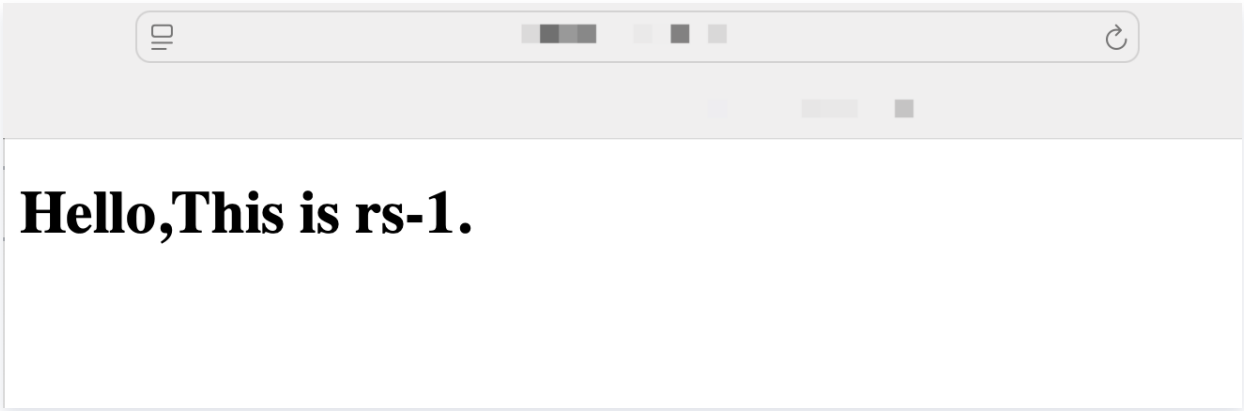
1. 클라이언트 인증서 client.crt와 클라이언트 개인 키 파일 client.key를 새 디렉토리에 복사합니다.
2. 명령줄에서 명령어를 실행하여 지정된 디렉토리의 클라이언트 인증서를 가져옵니다.

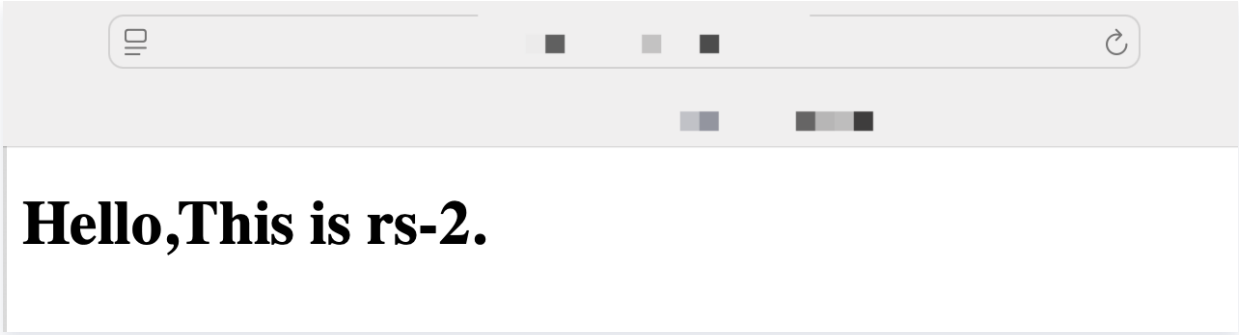
절차 8: 양방향 인증 검증

방법 1: 브라우저 방식

1. 브라우저에 CLB가 바인딩된 도메인을 입력합니다. 도메인이 바인딩되지 않은 경우 IP 포트에 접근할 수 있습니다. 접근 시 가져온 클라이언트 인증서를 선택하세요.

2. 브라우저를 새로 고침하여 클라이언트 요청이 rs-1 및 rs-2 서버 간에 전환되는 것을 확인하면 검증 성공입니다.





방법 2: 명령줄 방식

1. shell 인터페이스에서 다음 명령어를 입력하고, 인증서 주소, 키 주소 및 접근하는 CLB 주소를 정확히 입력합니다.

```
curl --cert client.crt --key client.key --cacert ca.crt
https://xxx.xxx.xxx
```

2. 해당하는 올바른 응답 코드가 출력되면 검증 성공입니다.

```
<!DOCTYPE html>
<html lang="en">
  <head>
    <!-- ... existing code ... -->
  </head>
  <body>
    <header>
      <h1>Hello,This is rs-1.</h1>
    </header>
    <!-- ... existing code ... -->
  </body>
```

```
<!DOCTYPE html>
<html lang="en">
  <head>
    <!-- ... existing code ... -->
  </head>
  <body>
    <header>
      <h1>Hello,This is rs-2.</h1>
    </header>
    <!-- ... existing code ... -->
  </body>
```

HTTPS 포워딩 구성

최종 업데이트 날짜:: 2024-01-04 20:09:01

1. CLB(Cloud Load Balancer) 기능 설명

프로토콜 스택과 서버를 심층적으로 최적화함으로써 Tencent Cloud CLB는 HTTPS 성능을 크게 향상시킵니다. 한편, Tencent Cloud는 국제 인증 기관과의 협업을 통해 인증서 비용을 크게 절감합니다. CLB는 다음과 같은 측면에서 귀하의 비즈니스에 상당한 이점을 제공할 수 있습니다.

1. HTTPS 사용은 Client의 액세스 속도에 영향을 미치지 않습니다.
2. 클러스터에 있는 단일 서버의 SSL 암호화 및 암호 해독 성능은 고성능 CPU보다 최소 3.5배 높은 초당 최대 6.5W cps의 전체 핸드셰이크를 유지할 수 있습니다. 이를 통해 서버 비용이 절감되고 비즈니스 피크 및 트래픽 급증 시 서비스 기능이 크게 향상되며 컴퓨팅 기반 공격 방지 기능이 강화됩니다.
3. 여러 프로토콜의 오프로딩 및 변환이 지원되어 다양한 클라이언트 프로토콜에 적응하는 비즈니스의 스트레스를 줄입니다. 비즈니스 백엔드는 HTTP2, SPDY, SSL 3.0 및 TLS 1.2와 같은 다른 프로토콜을 사용하기 위해 HTTP1.1만 지원하면 됩니다.
4. 원스톱 SSL 인증서 신청, 모니터링, 교체 서비스를 제공합니다. Tencent Cloud는 두 개의 선도적인 글로벌 인증 기관인 Comodo 및 SecureSite와 협력하여 인증서 신청 프로세스를 크게 간소화하고 신청 비용을 절감합니다.
5. Anti-CC 및 WAF 기능을 제공하여 느린 HTTP 공격, 고빈도 표적 공격, SQL 인젝션 및 웹 사이트 트로이 목마와 같은 애플리케이션 레이어 공격을 효과적으로 제거합니다.

2. HTTP 및 HTTPS 헤더 식별자

CLB는 HTTPS에 대한 프록시 역할을 합니다. HTTP 및 HTTPS 요청은 모두 CLB에서 백엔드 CVM 인스턴스로 포워딩할 때 HTTP 요청이 됩니다. 이 경우 프런트엔드 요청이 HTTP인지 HTTPS인지 구분할 수 없습니다.
CLB는 요청을 리얼 서버로 포워딩할 때 X-Client-Proto를 header에 삽입합니다.
X-Client-Proto: http(프런트엔드의 HTTP 요청)
X-Client-Proto: https(프런트엔드의 HTTPS 요청)

3. 시작하기

최종 사용자가 브라우저에서 `www.example.com`을 직접 입력할 때 HTTPS를 통해 안전하게 액세스할 수 있도록 `https://example.com` 웹 사이트를 구성해야 한다고 가정합니다.
이 경우 최종 사용자가 입력한 `www.example.com` 엑세스 요청은 아래와 같이 포워딩됩니다.

1. 요청은 HTTP를 통해 전송되고 VIP를 통해 CLB 수신기의 포트 80에 액세스합니다. 그러면 리얼 서버의 8080번 포트로 포워딩됩니다.
2. 리얼 서버의 Nginx에서 rewrite를 구성하면 요청이 포트 8080을 통해 전달되고 `https://example.com` 페이지에 다시 작성됩니다.
3. 그런 다음 브라우저는 `https://example.com` 요청을 해당 HTTPS 사이트로 다시 보냅니다. 요청은 VIP

를 통해 CLB 수신기의 포트 443에 액세스한 다음 리얼 서버의 포트 80으로 포워딩됩니다.

이제 요청 포워딩 프로세스가 완료되었습니다.

이 작업은 브라우저 사용자의 HTTP 요청을 보다 안전한 HTTPS 요청으로 다시 작성하고 사용자가 감지할 수 없습니다. 위의 요청 포워딩 작업을 구현하기 위해 리얼 서버를 다음과 같이 구성할 수 있습니다.

```
server {

    listen 8080;
    server_name example.qcloud.com;

    location / {

        #! customized_conf_begin;
        client_max_body_size 200m;
        rewrite ^/.(.*) https://$host/$1 redirect;

}
}
```

또는 새 버전의 Nginx에서 권장되는 301 리디렉션 메소드를 사용하여 Nginx HTTP 페이지를 HTTPS 페이지로 리디렉션합니다.

```
server {
      listen       80;
      server_name    example.qcloud.com;
      return       301 https://$server_name$request_uri;
}

server {
      listen       443 ssl;
     server_name    example.qcloud.com;
    [....]
}
```

리얼 클라이언트 IP 가져오기
CLB로 리얼 서버의 클라이언트 실제 IP 획득

최종 업데이트 날짜:: 2025-07-09 12:05:56

CLB로 클라이언트 실제 IP 획득 설명

CLB의 4계층 리스너(TCP/UDP/TCP SSL/QUIC)는 별도의 설정 없이 리얼 서버가 클라이언트의 실제 IP를 직접 획득할 수 있도록 지원합니다. 기본 설정에서는 리얼 서버가 획득한 소스 IP가 곧 클라이언트의 실제 IP입니다. 하지만, CLB와 리얼 서버 사이에 하나 이상의 NAT 게이트웨이가 존재하는 경우, 리얼 서버는 클라이언트의 실제 소스 IP를 획득할 수 없습니다. 이러한 상황에서는 CLB의 4계층 리스너에서 Proxy Protocol을 활성화할 수 있으며, Proxy Protocol을 통해 클라이언트의 실제 소스 IP를 리얼 서버로 전달할 수 있습니다.

주의:
- 이 기능을 사용하려면 리얼 서버 또한 Proxy Protocol을 활성화해야 하며, 그래야만 클라이언트의 실제 IP 주소를 획득할 수 있습니다. 리얼 서버가 Proxy Protocol을 분석할 수 없는 경우 Proxy Protocol 설정을 활성화하면 서비스 이상이 발생할 수 있습니다.
- 해당 기능은 온라인 무중단 전환을 지원하지 않으며, Proxy Protocol로 전환 시에는 서비스 중단 후 업그레이드가 필요하므로 신중히 설정하십시오.
- CLB는 Proxy Protocol v2만 지원합니다. Proxy Protocol v2는 TCP 및 UDP 등 다양한 전송 프로토콜을 지원합니다. 자세한 내용은 The PROXY protocol 을 참조하십시오.

- 해당 기능은 표준 계정 유형에서만 지원되며, 기존 계정 유형은 지원하지 않습니다. 계정 유형 확인 방법은 계정 유형 판단 을 참조하십시오.
- IPv4 및 IPv6 인스턴스의 TCP/UDP/TCP SSL/QUIC 리스너에서만 해당 기능을 지원합니다.
- IPv6 CLB의 TCP/UDP 리스너에서 Proxy Protocol 설정 기능은 현재 내부 테스트 중이며, 필요한 경우 티켓 제출 을 통해 신청해 주세요.

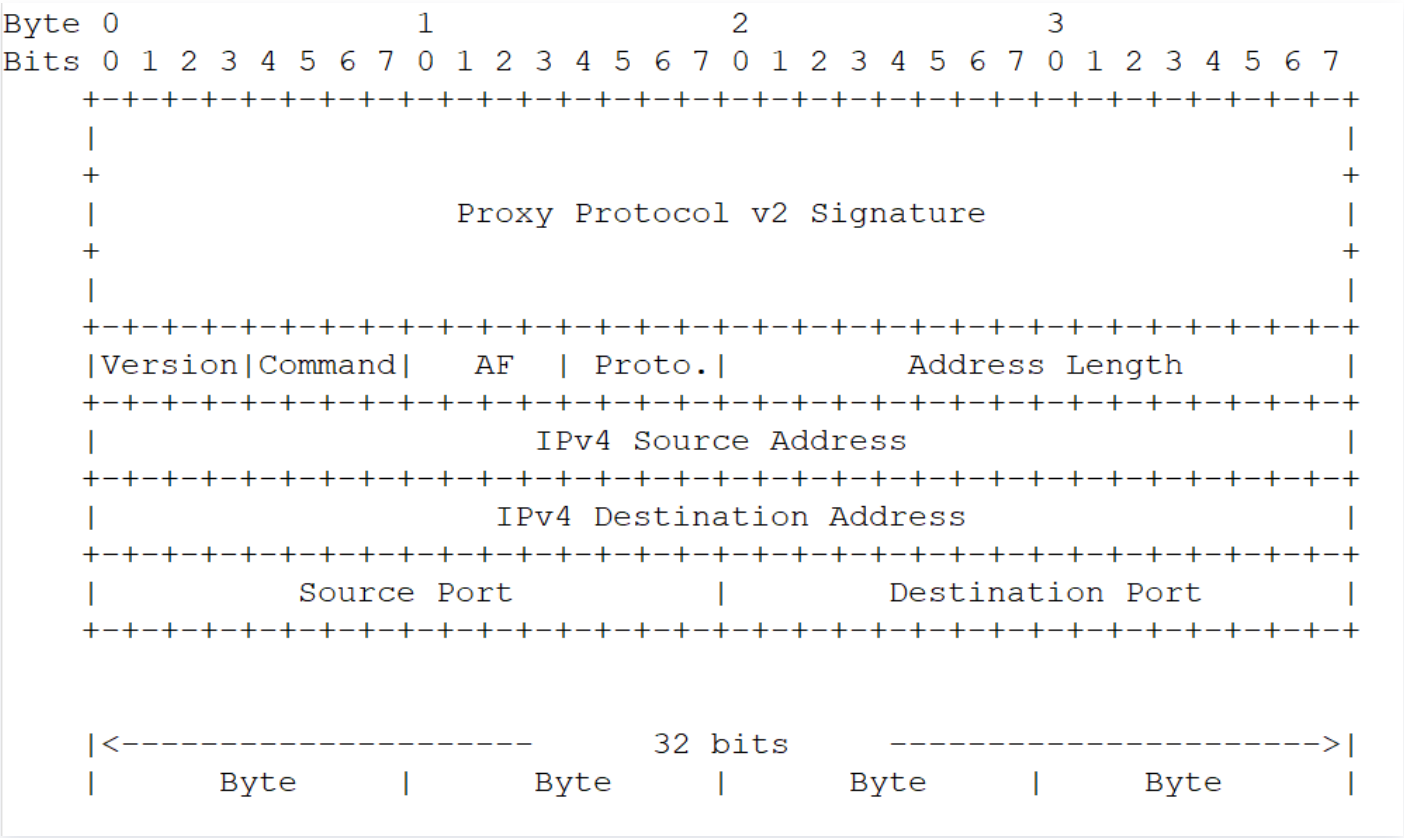
Proxy Protocol 설명

Proxy Protocol을 활성화하면 프록시 서버가 요청을 리얼 서버로 포워딩할 때 클라이언트의 원래 네트워크 연결 정보를 요청 헤더에 포함시켜 전송합니다. 리얼 서버는 Proxy Protocol 헤더를 분석하여 클라이언트의 실제 IP 주소, 포트, 전송 프로토콜 등의 정보를 획득할 수 있습니다.
Proxy Protocol을 통해 리얼 서버는 클라이언트의 원래 네트워크 연결 정보를 정확히 획득할 수 있어 보다 정밀한 로그 기록, 접근 제어, 트래픽 모니터링 등의 작업이 가능합니다.

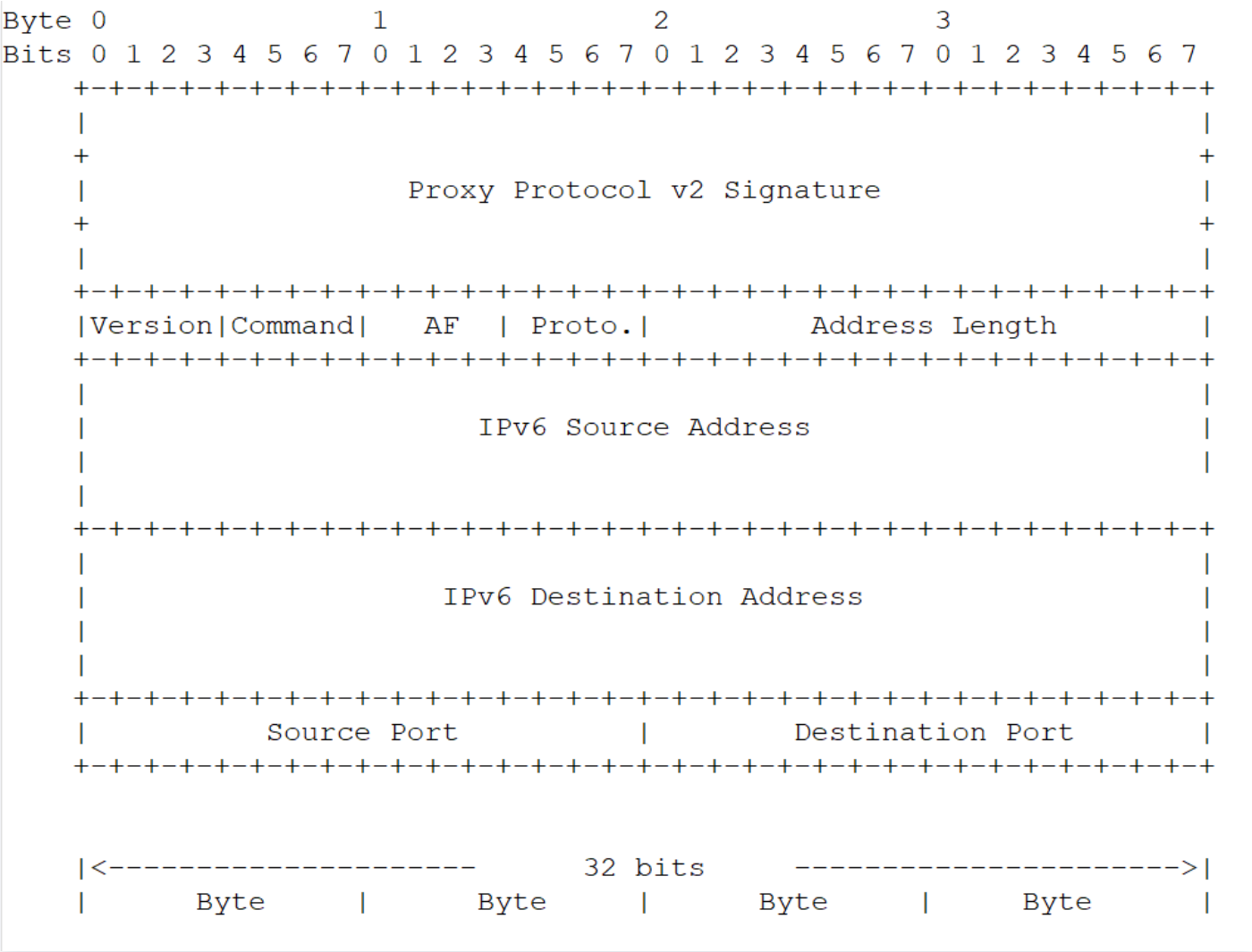
Proxy Protocol V2

Proxy Protocol V2는 바이너리 형식을 사용하며 TCPv4, TCPv6, UDPv4, UDPv6 프로토콜을 지원합니다. 형식은 다음과 같습니다.

IPv4 형식



IPv6 형식



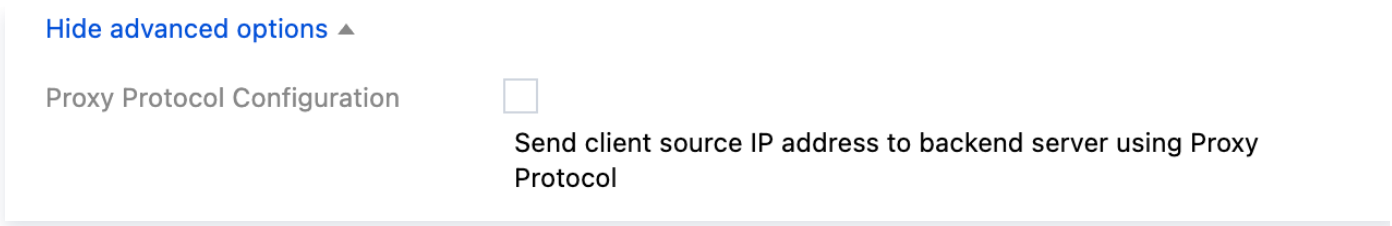
전제 조건

- Proxy Protocol을 활성화하기 전, 리얼 서버가 Proxy Protocol v2를 지원하는지 반드시 확인해야 합니다. 그렇지 않으면 새로운 연결이 실패할 수 있습니다.
- 본 문서는 IPv4 CLB의 TCP 리스너를 예시로 설명합니다.

작업 절차

절차 1: TCP 리스너에서 Proxy Protocol 설정 활성화

1. CLB 콘솔에 로그인한 후, 왼쪽 네비게이션 바에서 **인스턴스 관리**를 클릭합니다.
2. CLB 인스턴스 목록 페이지 왼쪽 상단에서 리전을 선택한 후, 인스턴스 목록 오른쪽의 작업 열에서 **리스너 설정**을 클릭합니다.
3. TCP/UDP/TCP SSL/QUIC 리스너 목록에서 대상 리스너 상세 정보를 클릭하여 ProxyProtocol 설정이 **활성화됨**인지 확인합니다. 활성화되지 않은 경우, 리스너를 편집하고 고급 옵션에서 ProxyProtocol 설정을 선택한 후 저장합니다.



절차 2: 리얼 서버에서 Proxy Protocol 활성화

다음은 CentOS 7.9 운영체제와 Nginx 1.20.1 버전을 기준으로 한 설정 예시입니다. 사용 환경에 따라 적절히 조정해 주세요.

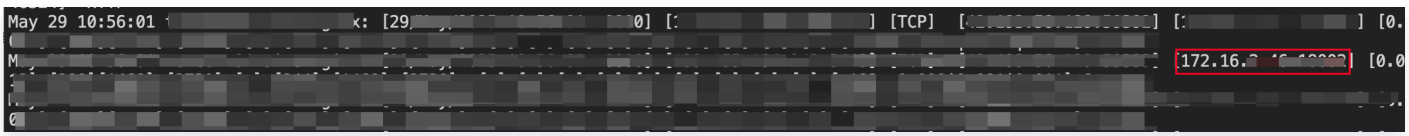
1. 리얼 서버에 로그인한 후, nginx –t 명령을 실행하여 설정 파일 경로를 확인합니다. 기본 경로는 /etc/nginx/nginx.conf이며, 사용 환경에 따라 다를 수 있습니다.
2. 설정 파일에서 Proxy Protocol 관련 항목을 수정하고 저장합니다. 수정 항목 예시는 다음과 같습니다.

```
http {
# $proxy_protocol_addr 변수 설정, 클라이언트 실제 IP 기록
log_format  main  '$proxy_protocol_addr - $remote_addr- $remote_user
[$time_local] "$request" '
                  '$status $body_bytes_sent "$http_referer" '
                  '"$http_user_agent" "$http_x_forwarded_for"';
 # 리스너 포트 80 예시, proxy_protocol 키워드 추가
 server {    listen 80   proxy_protocol;
 #...
   }
 }
```

3. sudo nginx –s reload 명령을 실행하여 Nginx 설정 파일을 다시 로드합니다.

절차 3: 리얼 서버는 클라이언트 실제 IP 획득 여부 확인

리얼 서버가 Nginx일 경우, Nginx 로그를 통해 클라이언트의 실제 IP가 제대로 획득되었는지 확인할 수 있습니다.
Nginx 로그 파일 기본 경로: /var/log/nginx/access.log
각 로그 라인의 $proxy_protocol_addr 값이 클라이언트의 실제 IP입니다.



IPv4 CLB를 통해 리얼 클라이언트 IP 가져오기

최종 업데이트 날짜:: 2024-01-04 20:11:24

CLB(Cloud Load Balancer)에서 클라이언트 리얼 IP 가져오기 관련 참고 사항

모든 레이어 4(TCP/UDP/TCP SSL) 및 레이어 7(HTTP/HTTPS) CLB 서비스는 추가 구성 없이 백엔드 CVM 인스턴스에서 직접 실제 클라이언트 IP 가져오기를 지원합니다.

- 레이어 4 CLB의 경우 백엔드 CVM 인스턴스에서 얻은 원본 IP는 클라이언트 IP입니다.
- 레이어 7 CLB의 경우 `X-Forwarded-For` 또는 `remote_addr` 필드를 사용하여 클라이언트 IP를 직접 가져올 수 있습니다. 레이어 7 CLB의 액세스 로그는 Configuring Access Logs를 참고하십시오.

> **설명:**
> - 레이어 4 CLB의 경우 백엔드 CVM 인스턴스에서 추가 구성 없이 클라이언트 IP를 직접 가져올 수 있습니다.
> - SNAT가 활성화된 다른 레이어 7 로드 밸런싱 서비스의 경우 백엔드 CVM 인스턴스를 구성한 다음 X-Forwarded-For를 사용하여 실제 클라이언트 IP를 가져와야 합니다.

다음은 일반적으로 사용되는 애플리케이션 서버 구성 스키마입니다.

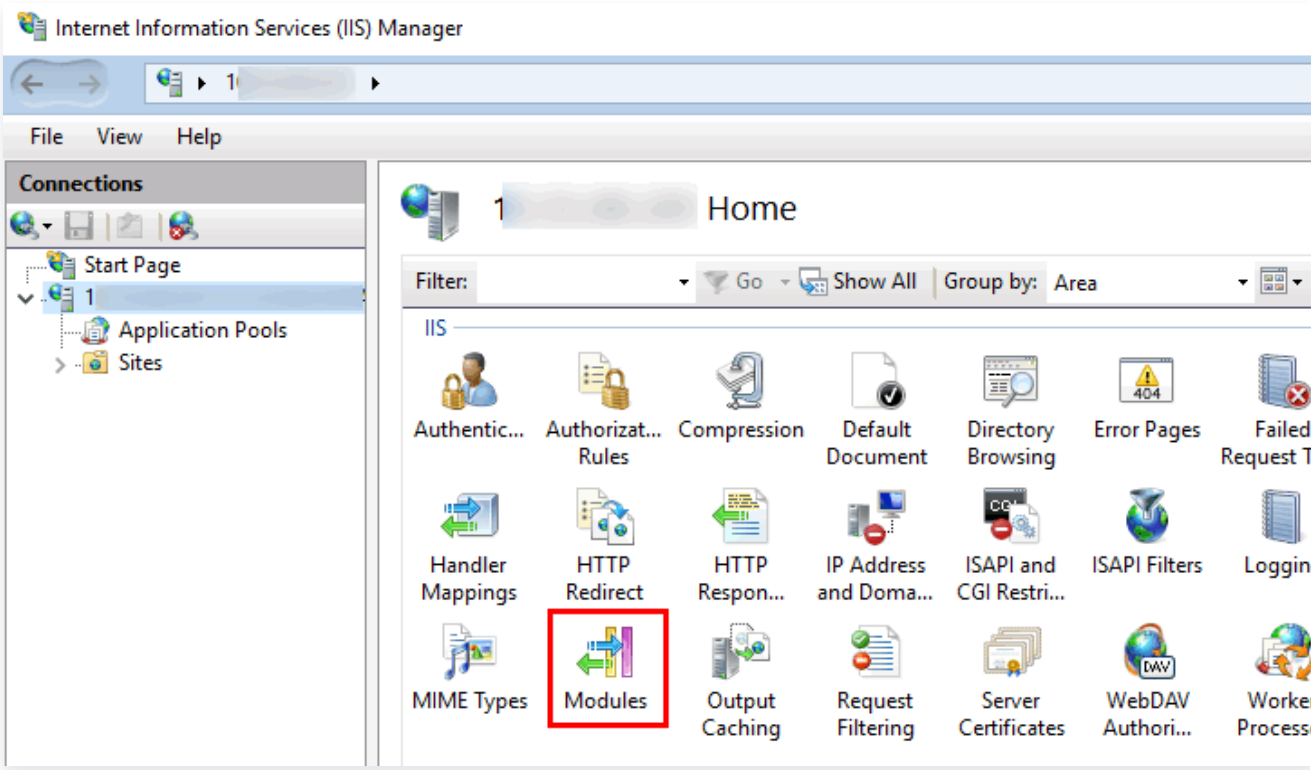
IIS 6 구성 스키마

1. F5XForwardedFor 플러그인 모듈을 다운로드하여 설치하고 서버 운영 체제 버전에 따라 `x86\Release` 또는 `x64\Release` 디렉터리의 `F5XForwardedFor.dll`을 특정 디렉터리(예시: 본문의 `C:\ISAPIFilters`)에 복사하고 IIS 프로세스에 이 디렉터리에 대한 읽기 권한이 있는지 확인합니다.
2. IIS 관리자를 열고 현재 열려 있는 웹 사이트를 찾은 다음 웹 사이트를 우클릭하고 **속성**을 선택하여 속성 페이지를 엽니다.
3. 속성 페이지에서 **ISAPI 필터**로 전환하고 **추가**를 클릭하면 필터 속성 추가/편집 창이 팝업됩니다.
4. 필터 속성 추가/편집 창에서 '필터 이름'에 'F5XForwardedFor'를 입력하고 '실행 파일'에 `F5XForwardedFor.dll`의 전체 경로를 입력한 다음 **확인**을 클릭합니다.
5. 구성을 적용하려면 IIS 서버를 다시 시작합니다.

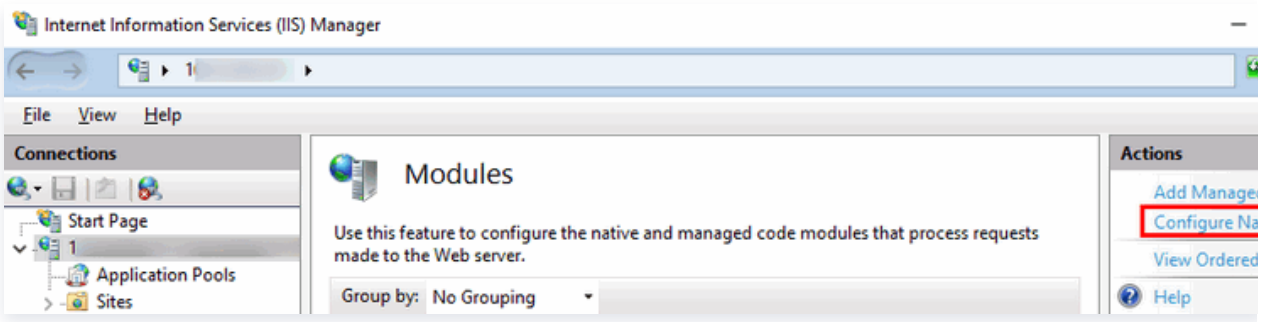
IIS 7 구성 스키마

1. F5XForwardedFor 플러그인 모듈을 다운로드하여 설치하고 서버 운영 체제 버전에 따라 `x86\Release` 또는 `x64\Release` 디렉터리의 `F5XFFHttpModule.dll` 및 `F5XFFHttpModule.ini`를 특정 디렉터리(예시: 본문의 `C:\x_forwarded_for`)에 복사하고 IIS 프로세스에 이 디렉터리에 대한 읽기 권한이 있는지 확인합니다.

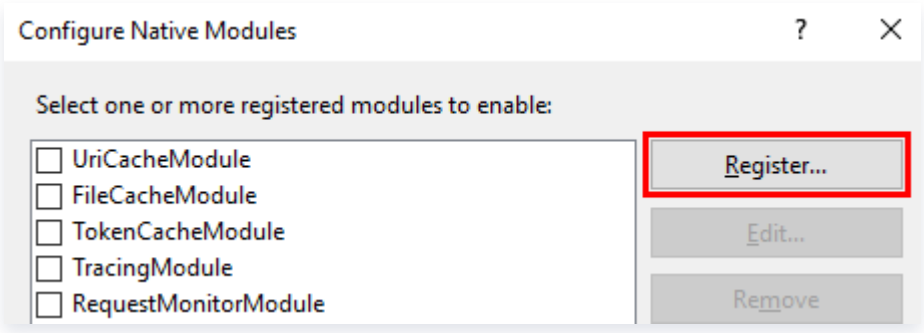
2. IIS 서버를 선택하고 **모듈**을 더블클릭합니다.



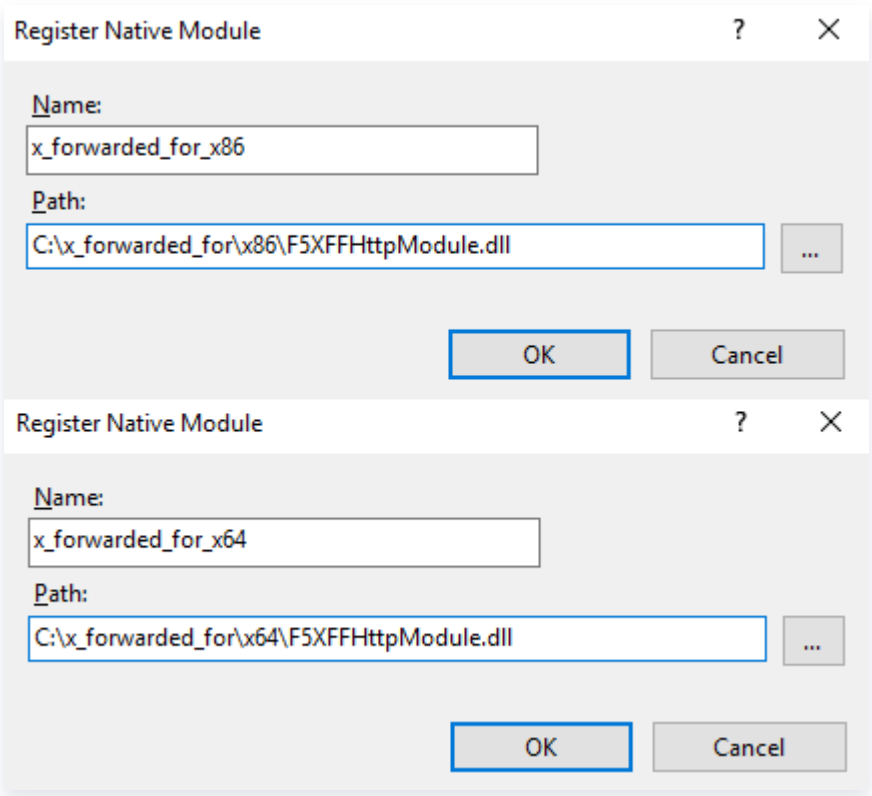
3. **네이티브 모듈 구성**을 클릭합니다.



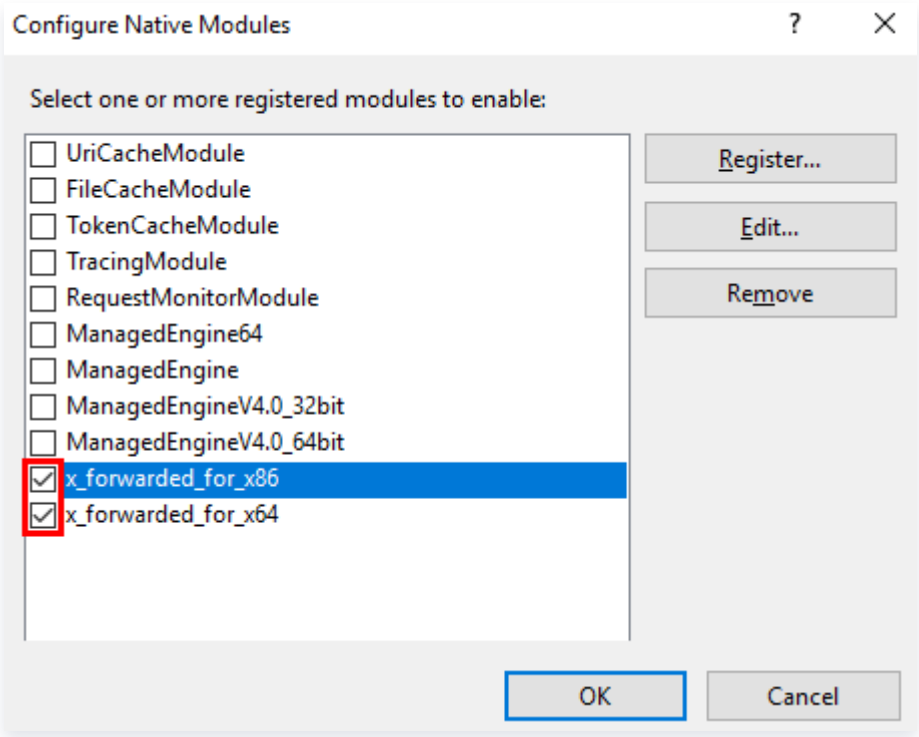
4. 팝업 창에서 **등록**을 클릭합니다.



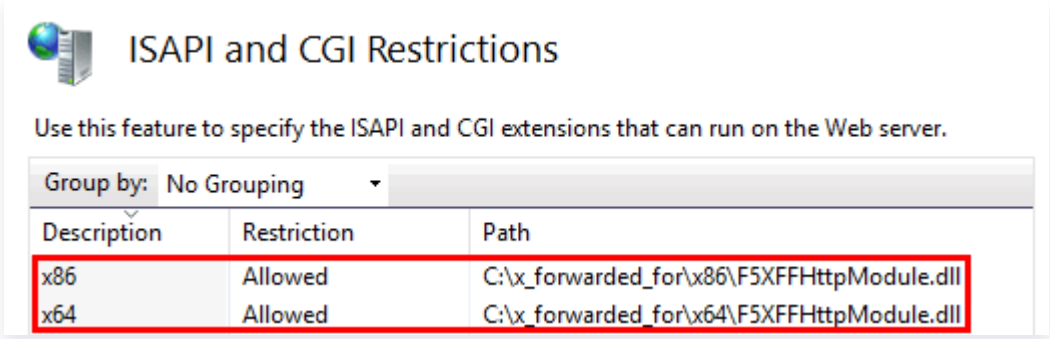
5. 다운로드한 DLL 파일을 아래와 같이 추가합니다.



6. 파일을 추가한 후 파일을 선택하고 **확인**을 클릭합니다.



7. ‘ISAPI 및 CGI 제한’에 위의 두 DLL 파일을 추가하고 제한을 허용으로 설정합니다.



8. 구성을 적용하려면 IIS 서버를 다시 시작하십시오.

Apache 구성 스키마

1. 타사 Apache 모듈 'mod_rpaf'를 설치합니다.

```
wget http://stderr.net/apache/rpaf/download/mod_rpaf-0.6.tar.gz
tar zxvf mod_rpaf-0.6.tar.gz
cd mod_rpaf-0.6
/usr/bin/apxs -i -c -n mod_rpaf-2.0.so mod_rpaf-2.0.c
```

2. 파일 끝에 다음을 추가하여 Apache 구성 `/etc/httpd/conf/httpd.conf` 를 수정합니다.

```
LoadModule rpaf_module modules/mod_rpaf-2.0.so
RPAFenable On
RPAFsethostname On
RPAFproxy_ips IP 주소(CLB에서 제공하는 공중망 IP가 아니며, 특정 IP에 대해서는
Apache 로그를 조회하고, 일반적으로 두 개의 IP 주소가 있으며 둘 다 입력해야 함)
RPAFheader X-Forwarded-For
```

3. 위의 내용을 추가한 후 Apache를 다시 시작합니다.

```
/usr/sbin/apachectl restart
```

Nginx 구성 스키마

1. Nginx를 서버로 사용할 때 http_realip_module을 사용하여 실제 클라이언트 IP를 가져올 수 있습니다. 그러나 이 모듈은 기본적으로 Nginx에 설치되어 있지 않으며 `--with-http_realip_module` 을 추가하려면 Nginx를 다시 컴파일해야 합니다.

```
yum -y install gcc pcre pcre-devel zlib zlib-devel openssl openssl-
devel
wget http://nginx.org/download/nginx-1.17.0.tar.gz
tar zxvf nginx-1.17.0.tar.gz
cd nginx-1.17.0
./configure --prefix=/path/server/nginx --with-http_stub_status_module
--without-http-cache --with-http_ssl_module --with-http_realip_module
make
make install
```

2. nginx.conf 파일을 수정합니다.

```
vi /etc/nginx/nginx.conf
```

빨간색으로 표시된 구성 필드 및 정보를 다음과 같이 수정합니다.

> **설명:**
> 여기서 `xx.xx.xx.xx` 를 리얼 IP 주소(CLB에서 제공하는 공중망 IP가 아님)로 변경해야 합니다. 특정 IP 주소에 대해 이전 Nginx 로그를 쿼리합니다. IP 주소가 여러 개인 경우 모든 IP 주소를 입력해야 합니다.

```
fastcgi connect_timeout 300;
fastcgi send_timeout 300;
fastcgi read_timeout 300;
fastcgi buffer_size 64k;
fastcgi buffers 4 64k;
fastcgi busy_buffers_size 128k;
fastcgi temp_file_write_size 128k;

set_real_ip_from xx.xx.xx.xx;
real_ip_header X-Forwarded-For;
```

3. Nginx를 다시 시작합니다.

```
service nginx restart
```

4. 실제 클라이언트 IP를 가져오려면 Nginx 액세스 로그를 조회합니다.

```
cat /path/server/nginx/logs/access.log
```

하이브리드 클라우드 배포에서 TOA를 통해 리얼 클라이언트 IP 가져오기

최종 업데이트 날짜:: 2024-01-04 20:12:38

본문은 레이어 4(TCP 전용) CLB 서비스가 하이브리드 클라우드 배포 및 NAT64 CLB 시나리오에서 TOA를 통해 실제 클라이언트 IP를 가져오는 방법을 설명합니다.

- TOA 로딩
- 리얼 서버에 적용
- (옵션)TOA 상태 모니터링

> **설명:**
> 레이어 4(TCP) CLB는 TOA를 통해 실제 클라이언트 IP를 가져올 수 있지만 레이어 4(UDP) 및 레이어 7(HTTP/HTTPS) CLB는 그렇지 않습니다.

응용 시나리오

하이브리드 클라우드 배포

Hybrid Cloud Deployment 시나리오에서 IDC와 VPC의 IP는 겹칠 수 있으므로 SNAT IP가 필요합니다. 서버의 경우 실제 클라이언트 IP는 보이지 않으며 TOA를 통해 가져와야 합니다.

NAT64 CLB

NAT64 CLB 시나리오에서 실제 IPv6 클라이언트 IP는 리얼 서버에 보이지 않는 IPv4 공중망 IP로 변환됩니다. 이 경우, 실제 클라이언트 IP는 TOA를 통해 얻을 수 있으며, 즉, TCP 패킷은 필드 TCP option에 실제 클라이언트 IP를 삽입한 후 실제 클라이언트 IP 정보를 서버로 전송하고 클라이언트는 TOA 커널 모듈의 API를 호출하여 실제 클라이언트 IP를 가져올 수 있습니다..

제한 설명

리소스 제한

- TOA 컴파일 환경의 커널 버전은 서비스 환경의 커널 버전과 일치해야 합니다.
- 컨테이너 환경에서는 TOA 커널 모듈을 호스트에 로딩해야 합니다.
- TOA 커널 모듈을 로딩하려면 root 권한이 필요합니다.

호환성 제한

- UDP 리스너는 TOA를 통해 실제 클라이언트 IP를 가져올 수 없습니다.
- 클라이언트와 리얼 서버 간에 이미 TOA 관련 작업이 있는 경우 리얼 서버는 실제 클라이언트 IP를 가져오지 못할 수 있습니다.

- TOA가 삽입된 후에는 새 연결에만 적용됩니다.
- TOA는 필드 TCP option에서 주소 추출과 같은 추가 처리를 수행해야 하므로 서버 성능이 저하될 수 있습니다.
- Tencent Cloud TOA를 다른 TOA 모듈과 함께 사용할 때 호환성 문제가 발생할 수 있습니다.
- Tencent Cloud TOA는 TencentOS에 내장되어 있으며 하이브리드 클라우드 배포 시나리오에서 실제 원본 IP를 얻는 데 사용할 수 있습니다. 서버가 TencentOS에서 실행되고 하이브리드 클라우드에 배포되는 경우 `modprobe toa` 명령을 직접 실행하여 TOA를 로딩할 수 있습니다. 서버가 Linux에서 실행되는 경우 Linux TOA를 대신 사용해야 합니다.

TOA 로딩

1. Tencent Cloud의 Linux OS 버전에 해당하는 TOA 패키지를 다운로드하고 압축을 풉니다.

centos CentOS 8.0 64 CentOS 7.6 64 CentOS 7.2 64
debian Debian 9.0 64
suse linux SUSE 12 64 SUSE 11 64
ubuntu Ubuntu 18.04.4 LTS 64 Ubuntu 16.04.7 LTS 64

2. 압축 해제 후 cd 명령을 실행하여 압축이 해제된 폴더에 액세스하고 모듈 로딩 명령을 실행합니다.

```
insmod toa.ko
```

3. 다음 명령어를 실행하여 TOA가 로딩되었는지 확인합니다. ‘to load success’ 메시지가 표시되면 로딩이 성공한 것입니다.

```
dmesg -T | grep TOA
```

4. 로딩된 후 실행 스크립트에서 `toa.ko` 파일을 로딩합니다(ko 파일은 서버가 다시 시작되면 다시 로딩해야 함).
5. (옵션)TOA가 더 이상 필요하지 않은 경우 다음 명령을 실행하여 제거합니다.

```
rmmod toa
```

6. (옵션)다음 명령을 실행하여 모듈이 제거되었는지 확인합니다. 'TOA unloaded' 메시지가 표시되면 성공적으로 제거된 것입니다.

```
dmesg -T
```

상기 OS 버전에 대한 설치 패키지를 찾을 수 없는 경우 Linux OS용 일반 소스 패키지를 다운로드하고 컴파일하여 ko 파일을 가져올 수 있습니다. 이 일반 버전은 대부분의 Linux 릴리스 버전(예시: Centos8, Centos7, Ubuntu18.04 및 Ubuntu16.04)을 지원합니다.

> **설명:**
> Linux 커널과 Linux 릴리스 버전은 다양하므로 호환성 문제가 발생할 수 있습니다. TOA 소스 패키지를 사용하기 전에 OS에서 컴파일하는 것이 좋습니다.

1. 소스 패키지를 다운로드합니다.

> **주의:**
> OS가 Linux인 경우 Linux TOA 소스 패키지를 다운로드하십시오. TencentOS인 경우 TLinux TOA 소스 패키지를 다운로드합니다.

- Linux

```
wget "https://clb-toa-1255852779.file.myqcloud.com/tgw_toa_linux_ver.tar.gz"
```

- Tencent TLinux

```
wget "https://clb-toa-1255852779.file.myqcloud.com/tgw_toa_tlinux_ver.tar.gz"
```

2. TOA용 Linux 환경을 컴파일하려면 먼저 GCC 컴파일러, Make 툴 및 커널 개발 패키지를 설치해야 합니다.

CentOS 환경 설치 단계

```
yum install gcc
yum install make
//커널 개발 패키지를 설치합니다. 패키지 헤더 파일 및 라이브러리의 버전은 커널 버전과 일치해야 합니다.
yum install kernel-devel-`uname -r`
```

Ubuntu 및 Debian 설치 작업

```
apt-get install gcc
apt-get install make
//커널 개발 패키지를 설치합니다. 패키지 헤더 파일 및 라이브러리의 버전은 커
널 버전과 일치해야 합니다.
apt-get install linux-headers-`uname -r`
```

SUSE 환경 설치 단계

```
zypper install gcc
zypper install make
//커널 개발 패키지를 설치합니다. 패키지 헤더 파일 및 라이브러리의 버전은 커
널 버전과 일치해야 합니다.
zypper install kernel-default-devel
```

3. 소스 패키지를 컴파일하여 toa.ko 파일을 생성합니다. 컴파일 과정에서 `warning` 및 `error` 메시지가 표시되지 않으면 컴파일이 성공한 것입니다. Linux OS용 소스 패키지를 예로 들어 보겠습니다.

```
tar zxvf tgw_toa_linux_ver.tar.gz
cd tgw_toa_linux_ver//압축 해제된 tgw_toa 디렉터리로 이동
make
```

4. toa.ko 컴파일이 성공하면 2단계를 수행하여 TOA를 로딩합니다.

리얼 서버 적용

하이브리드 클라우드 배포

하이브리드 클라우드에서 리얼 서버를 적용할 때 표준 Linux 네트워크 프로그래밍 API를 호출하기만 하면 코드 변경 없이 실제 클라이언트 IP를 가져올 수 있습니다. 다음은 코드 예시를 보여줍니다.

```
struct sockaddr v4addr;
len = sizeof(struct sockaddr);
//get_peer_name 은 표준 Linux 네트워크 프로그래밍 API입니다.
if (get_peer_name(client_fd, &v4addr, &len) == 0) {
```

```
    inet_ntop(AF_INET, &(((struct sockaddr_in *)&v4addr)->sin_addr), from, sizeof(from));

printf("real client v4 [%s]:%d\n", from, ntohs(((struct sockaddr_in *)&v4addr)->sin_port));
}
```

NAT64 CLB

NAT64 CLB 시나리오에서 실제 원본 IP를 얻으려면 리얼 서버에 `toa.ko` 커널 모듈을 삽입한 후 원본 코드를 수정해야 하며 TOA는 해당 IP 주소를 리얼 서버에 전달합니다.

1. IP 주소를 저장할 데이터 구조를 정의합니다.

```
struct toa_nat64_peer {
 struct in6_addr saddr;
 uint16_t sport;
};
....
struct toa_nat64_peer client_addr;
....
```

2. TOA 변수를 정의하고 호출하여 실제 원본 IPv6 주소를 가져옵니다.

```
enum {
 TOA_BASE_CTL            = 4096,
 TOA_SO_SET_MAX          = TOA_BASE_CTL,
 TOA_SO_GET_LOOKUP       = TOA_BASE_CTL,
 TOA_SO_GET_MAX          = TOA_SO_GET_LOOKUP,
};
getsockopt(client_fd, IPPROTO_IP, TOA_SO_GET_LOOKUP, &client_addr, &len);
```

3. 원본 IP 주소를 가져옵니다.

```
real_ipv6_saddr = client_addr.saddr;
real_ipv6_sport = client_addr.sport;
```

전체 구성 예시는 다음과 같습니다.

```
//TOA 변수를 정의합니다. 'TOA_BASE_CTL'을 4096으로 설정합니다.
enum {
    TOA_BASE_CTL            = 4096,
    TOA_SO_SET_MAX          = TOA_BASE_CTL,
    TOA_SO_GET_LOOKUP       = TOA_BASE_CTL,
    TOA_SO_GET_MAX          = TOA_SO_GET_LOOKUP,
};
//IP 주소를 저장할 데이터 구조를 정의합니다.
struct toa_nat64_peer {
    struct in6_addr saddr;
    uint16_t sport;
};
//주소를 저장할 변수를 선언합니다.
struct toa_nat64_peer client_addr;
.……
//클라이언트의 파일 디스크립터를 가져옵니다. 여기서 listenfd는 서버의 수신
파일 디스크립터입니다.
client_fd = accept(listenfd, (struct sockaddr*)&caddr, &length);

// NAT64 시나리오에서 사용자의 실제 원본 IP를 가져오기 위해 호출합니다.
char from[40];
int len = sizeof(struct toa_nat64_peer);
if (getsockopt(client_fd, IPPROTO_IP, TOA_SO_GET_LOOKUP, &client_
addr, &len) == 0) {

inet_ntop(AF_INET6, &client_addr.saddr, from, sizeof(from));
    //원본 IP와 원본 port 가져오기

printf("real client [%s]:%d\n", from, ntohs(client_addr.sport));

}
```

하이브리드 클라우드에서 사용되는 NAT64 CLB

NAT64 CLB가 하이브리드 클라우드에서 사용되는 시나리오에서 실제 원본 IP를 얻으려면 리얼 서버에 `toa.ko` 커널 모듈을 삽입한 후 원본 코드를 수정해야 하며 TOA는 해당 IP 주소를 리얼 서버에 전달합니다.
전체 구성 예시는 다음과 같습니다.

```
//TOA 변수를 정의합니다. 'TOA_BASE_CTL'을 4096으로 설정합니다.
enum {
    TOA_BASE_CTL = 4096,
    TOA_SO_SET_MAX = TOA_BASE_CTL,
    TOA_SO_GET_LOOKUP = TOA_BASE_CTL,
    TOA_SO_GET_MAX = TOA_SO_GET_LOOKUP,
};

//IP 주소를 저장할 데이터 구조를 정의합니다.
struct toa_nat64_peer {
    struct in6_addr saddr;
    uint16_t sport;
};

//주소를 저장할 변수를 선언합니다.
struct toa_nat64_peer client_addr_nat64;
.......
//클라이언트의 파일 디스크립터를 가져옵니다. 여기서 listenfd는 서버의 수신
파일 디스크립터입니다.
//NAT64 시나리오에서 실제 클라이언트 IP를 가져오기 위해 호출합니다.
char from[40];
int len = sizeof(struct toa_nat64_peer);
int ret;
ret = getsockopt(client_fd, IPPROTO_IP, TOA_SO_GET_LOOKUP, &clien
t_addr_nat64, &len);
if (ret == 0) {
    inet_ntop(AF_INET6, &
(client_addr_nat64.saddr), from, sizeof(from));
    //원본 IP와 원본 Port를 가져옵니다.

printf("real client v6 [%s]:%d\n", from, ntohs(client_addr_nat64.
sport));
} else if (ret != 0) {
    struct sockaddr v4addr;
    len = sizeof(struct sockaddr);
    //원본 IP와 원본 Port를 가져옵니다.
    //하이브리드 클라우드 배포 시나리오에서 원본 IP 주소는 SNAT 뒤의 IP 주
소입니다.
```

```
    //비하이브리드 클라우드 배포 시나리오에서 원본 IP 주소는 SNAT 및
NAT64가 없는 클라이언트 IP 주소입니다.
    //이 함수의 의미는 실제 클라이언트 주소와 포트를 가져오는 것입니다.
    if (get_peer_name(client_fd, &v4addr, &len) == 0) {
        inet_ntop(AF_INET, &(((struct sockaddr_in *)&v4addr)-
>sin_addr), from, sizeof(from));

printf("real client v4 [%s]:%d\n", from, ntohs(((struct sockaddr_
in *)&v4addr)->sin_port));
    }
}
```

(옵션)TOA 상태 모니터링

실행 안정성을 보장하기 위해 이 커널 모듈을 사용하여 상태를 모니터링할 수 있습니다. toa.ko 커널 모듈을 삽입한 후 다음 방법 중 하나로 컨테이너 호스트에서 TOA 작업 상태를 모니터링할 수 있습니다.

방법1: TOA에 저장된 IPv6 주소 확인

다음 명령어를 실행하여 TOA에 저장된 IPv6 주소를 확인합니다.

- 이 명령을 실행하면 성능이 저하될 수 있습니다. 주의하여 진행하십시오.

```
cat /proc/net/toa_table
```

방법 2: TOA 메트릭 확인

다음 명령을 실행하여 TOA 메트릭을 확인하십시오.

```
cat /proc/net/toa_stats
```

```
root@VM-0-36-ubuntu:/data/[redacted]# cat /proc/net/toa_stats
                                CPU0       CPU1       CPU2       CPU3
syn_recv_sock_toa       :        803        850        870        837
syn_recv_sock_no_toa    :        830        980        867       1006
getname_toa_ok          :        761        826        813        960
getname_toa_mismatch    :          0          0          0          0
getname_toa_bypass      :       1522       1652       1626       1920
getname_toa_empty       :       3367       3751       3788       4058
ip6_address_alloc       :        803        850        870        837
ip6_address_free        :        803        851        869        837
```

모니터링 메트릭은 다음과 같이 설명됩니다.

메트릭	설명
syn_recv_sock_toa	TOA 정보와의 연결을 수신합니다.
syn_recv_sock_no_toa	TOA 정보 없이 연결을 수신합니다.
getname_toa_ok	이 수는 getsockopt를 호출하고 원본 IP를 성공적으로 가져오거나 accept를 호출하여 클라이언트 요청을 수신할 때 증가합니다.
getname_toa_mismatch	이 수는 getsockopt를 호출하고 필요한 유형과 일치하지 않는 원본 IP를 가져올 때 증가합니다. 예를 들어 클라이언트 연결에는 IPv4 원본 IP 주소가 포함되어 있는 반면 IPv6 주소를 받으면 개수가 증가합니다.
getname_toa_empty	이 카운트는 TOA를 포함하지 않는 클라이언트 파일 디스크립터에서 getsockopt 함수가 호출될 때 증가합니다.
ip6_ad	TOA가 TCP 데이터 패킷에 저장된 원본 IP와 원본 port를 가져오면 정보를 저장할 공

dress_alloc	간을 할당합니다.
ip6_address_free	연결이 릴리스되면 toa는 이전에 원본 IP 및 원본 port를 저장하는 데 사용된 메모리를 릴리스합니다. 모든 연결이 닫히면 각 CPU에 대한 ip6_address_alloc의 총 개수는 이 메트릭의 개수와 같아야 합니다.

FAQ

NAT64 CLB에 TOA를 삽입한 후 서버 프로그램을 수정해야 하는 이유는 무엇입니까?

클라이언트 IP 유형이 변경되지 않고 유지되는 NAT64 CLB 시나리오와 다른 하이브리드 클라우드 배포 시나리오에서 클라이언트 IP(IPv4)가 IPv6 주소로 변환되기 때문입니다. 따라서 서버가 IPv6 주소를 이해할 수 있도록 서버 프로그램을 수정해야 합니다.

내 OS가 Linux 릴리스 버전 또는 TLinux 커널을 기반으로 하는지 어떻게 알 수 있습니까?

- 다음 명령을 실행하여 내 커널 버전을 확인하십시오. 명령 출력에 `tlinux`가 표시되면 TLinux OS를 사용하고 있는 반면 linux는 Linux OS를 사용하고 있음을 나타냅니다.

```
uname -a
```

- 다음 명령을 사용하여 버전을 확인할 수도 있습니다. `tlinux` 또는 `tl2`가 반환되면 TLinux OS를 사용하고 있는 것입니다.

```
rpm -qa | grep kernel
```

```
root@VM_64_20_centos ~]# rpm -qa|grep kernel
kernel-devel-4.14.105-19.0010.tl2.x86_64
kernel-debuginfo-4.14.105-19.0012.tl2.x86_64
kernel-headers-4.14.105-19.0012.tl2.x86_64
kernel-tools-libs-3.10.0-1127.8.2.el7.x86_64
kernel-3.10.0-957.21.3.el7.x86_64
kernel-devel-4.14.105-19.0012.tl2.x86_64
kernel-4.14.105-19.0012.tl2.x86_64
kernel-3.10.0-1127.8.2.el7.x86_64
kernel-tools-3.10.0-1127.8.2.el7.x86_64
kernel-devel-3.10.0-957.21.3.el7.x86_64
brt-addon-kerneloops-2.1.11-57.el7.centos.x86_64
root@VM_64_20_centos ~]#
```

원본 IP를 받지 못한 경우 사전 확인은 어떻게 합니까?

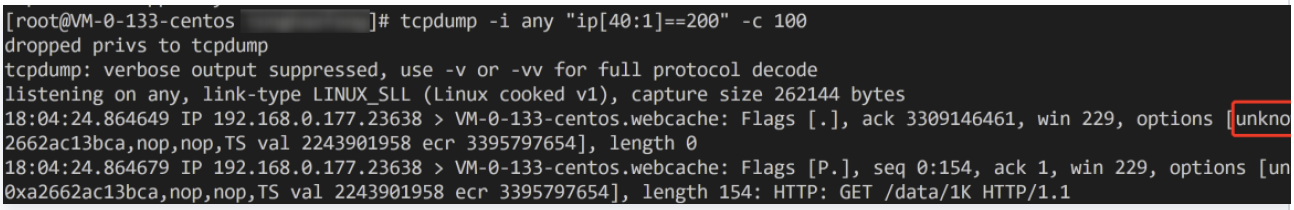
1. 다음 명령어를 실행하여 TOA가 로딩되었는지 확인합니다.

```
lsmod | grep toa
```

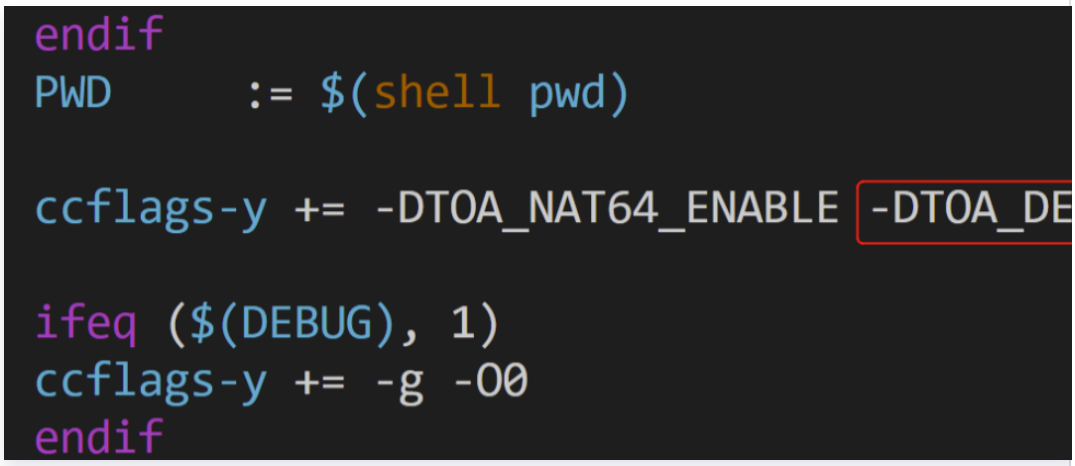
```
[root@VM-0-133-centos ]# lsmod | grep toa
toa                    278528  0
[root@VM-0-133-centos ]#
```

2. 서버 프로그램이 원본 IP를 얻기 위해 올바른 호출을 했는지 확인하십시오. 리얼 서버 적용을 참고하십시오.

3. 서버에서 TCP 패킷을 캡처하고 패킷에 원본 IP 정보가 포함되어 있는지 확인합니다.
   - tcp option 출력에 `unknown-200` 이 표시되면 TCP option 필드에 SNAT 이후의 실제 원본 IP 가 삽입되었음을 나타냅니다.
   - `unknown-253` 이 표시되면 NAT64 시나리오에 실제 원본 IPv6 주소가 삽입되었음을 나타냅니다.



4. TOA 주소가 포함된 패킷이 서버로 전송된 경우 toa.ko를 DEBUG 버전으로 컴파일하고 커널 로그를 통해 문 렉터리에서 DEBUG 컴파일 옵션을 Makefile에 추가합니다.



5. 다음 명령을 실행하여 다시 컴파일합니다.

```
make clean
make
```

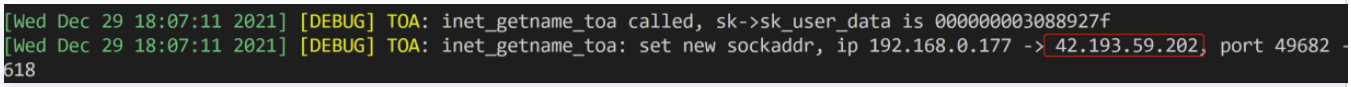
6. 다음 명령을 실행하여 원본 ko를 제거하고 최신 버전 ko를 설치합니다.

```
rmmod toa
insmod ./toa.ko
```

7. 다음 명령을 실행하여 커널 로그를 관찰합니다.

```
dmesg -Tw
```

다음 메시지가 표시되면 TOA가 정상적으로 작동하는 것입니다. 서버 프로그램이 실제 원본 IP를 얻기 위해 호출했는지 또는 API가 잘못 사용되었는지 추가로 확인할 수 있습니다.



8. 이전 단계에서 문제를 찾지 못한 경우 티켓을 제출하십시오.

로드 밸런싱 구성 모니터링 및 알람에 대한 모범 사례

최종 업데이트 날짜:: 2024-01-04 20:13:28

로드 밸런싱 CLB 업무 모니터링 시스템을 개선하도록 TCOP의 데이터 수집 및 알람 기능을 결합하여 통합 알람 메커니즘을 구축합니다. TCOP를 통해 로드 밸런싱 CLB의 리소스 사용, 성능 및 운영 상황을 종합적으로 파악할 수 있고, 관심 있는 인스턴스에 대해 모니터링 알람을 구성하여 모니터링 지표 및 사건 알람 트리거 규칙을 설정할 수 있습니다. 해당 인스턴스의 모니터링 지표에 이상이 발생한 경우, 귀하는 즉시 이상 알람을 받고 고장에 대응하여 처리할 수 있습니다.

시나리오

실행 상태가 특정 조건을 충족할 때 로드 밸런싱 CLB 인스턴스가 대상 사용자 그룹에 알람 정보를 보내도록 지정된 인스턴스 지표에 대한 알람을 생성할 수 있습니다. 발생할 수 있는 긴급 상황을 보다 편리하고 신속하게 제어하고, 운영 및 유지 관리 효율성을 향상시키며, 운영 및 유지 관리 비용을 절감합니다.
성능 용량 유형으로 업그레이드된 공중망 로드 밸런싱 CLB 인스턴스에 대한 알람 구성 방법을 표준 유형을 예로 들어 소개합니다. 성능 및 용량 사양에 대한 자세한 내용은 성능 및 용량 사양 소개를 참고하십시오.

전제조건

로드 밸런싱 인스턴스를 만들고 리스너를 구성했습니다. 자세한 내용은 로드 밸린싱 시작하기 을 참고하십시오.
백엔드 서버 바인딩에 성공했습니다. 자세한 내용은 백엔드 서버로 바인딩 을 참고하십시오.
이 예에 따르면 대상 인스턴스는 성능 용량 유형으로 업그레이드되어야 합니다. 자세한 내용은 성능 용량 유형 인스턴스로 업그레이드 를 참고하십시오.

기본 개념

용어	정의
알람 정책	정책 이름, 정책 유형, 알람 대상, 트리거 조건, 알림 템플릿으로 구성됩니다.
정책 유형	알람 정책 유형은 정책 분류를 식별하는 데 사용되며 특정 클라우드 제품에 해당합니다. 예를 들어, 클라우드 서버 정책을 선택하면 CPU 사용률, 디스크 사용률 등에 대한 지표 알람을 사용자 지정할 수 있습니다.
트리거 조건	트리거 조건은 지표, 비교, 임계값, 통계 세분성 및 N개의 연속 모니터링 데이터 포인트로 구성된 의미론적인 조건입니다.
모니터링 유형	모니터링 유형에는 클라우드 상품 모니터링과 프런트엔드 성능 모니터링이 포함됩니다.
알림 템플릿	원클릭으로 재사용 가능한 여러 알림 템플릿이 있어 다양한 사용 시나리오에서 알람 수신에 적합하며, 자세한 내용은 새 알림 템플릿 생성 을 참고하십시오.

지표 설명

성능 용량 유형의 인스턴스가 한도를 초과하는지 여부를 판단하는 핵심 지표는 클라이언트에서 LB로의 동시 연결 수, 클라이언트에서 LB로의 새 연결 수, 초당 요청 수, 클라이언트에서 LB로의 아웃바운드 대역폭, 클라이언트에서 LB로 인바운드 대역폭입니다. 그래서 다음 표와 같이 위 핵심 지표의 사용률 알람 지표에 주의해야 합니다. 그중에 폐기/사용률 모니터링 지표는 내부 테스트 단계이므로, 사용을 원하시면 작업 주문 시청를 하시기 바랍니다. 알람 지표에 대한 자세한 설명은 알람 지표 설명을 참조하세요.

차원	알람 정책 유형	알람 정책	알람 지표	지표 설명
인스턴스	공중망 로드 밸런싱 인스턴스	폐기/사용률 모니터링	인바운드 대역폭 사용률	통계 세분성 내에서 클라이언트가 외부 네트워크를 통해 로드 밸런싱에 액세스하는 데 사용하는 대역폭 사용률입니다.
			아웃바운드 대역폭 사용률	통계 세분성 내에서 외부 네트워크에 액세스하기 위해 로드 밸런싱에 사용되는 대역폭 사용률입니다.
			최대 연결 수 사용률	통계 세분성 내에서 주어진 특정 순간에 클라이언트에서 로드 밸런싱까지의 동시 연결 수와 성능 용량 사양의 동시 연결 수의 성능 상한을 비교한 사용률입니다.
			새 연결 수 사용률	통계 세분성 내에서 주어진 특정 순간에 클라이언트에서 로드 밸런싱까지의 새 연결 수와 성능 용량 사양의 새 연결 수의 성능 상한을 비교한 사용률입니다.
		QPS 관련 모니터링	QPS 사용률	통계 세분성 내에서 주어진 특정 순간에 로드 밸런싱된 QPS와 성능 용량 사양의 QPS 성능 상한을 비교한 사용률입니다.

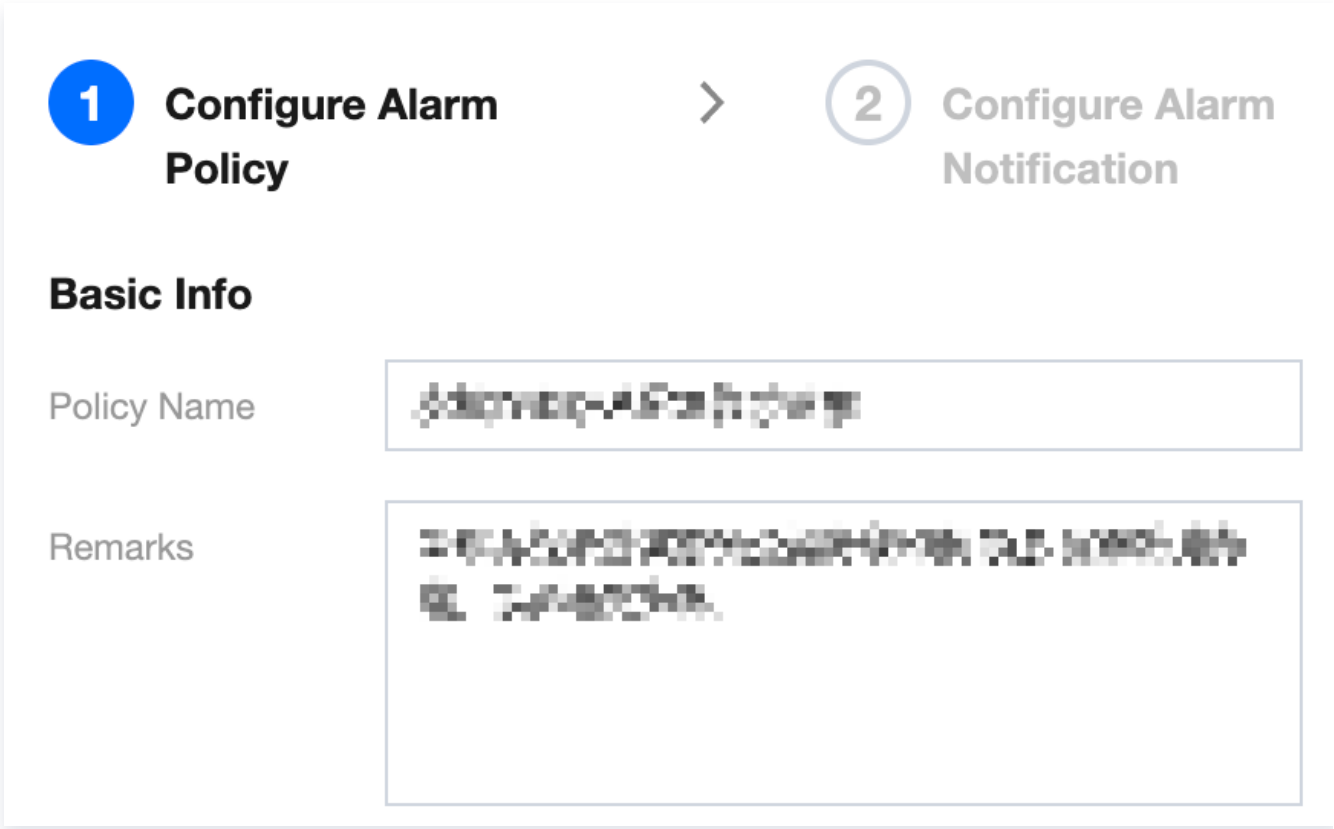
작업 단계

1. Tencent Cloud 관측 플랫폼에 로그인합니다.
2. 왼쪽 네비게이션 바에서 **알람 관리** > **알람 구성** > **알람 정책**을 클릭하여 관리 페이지로 들어갑니다.
3. **새 정책**을 클릭하고 다음 옵션을 구성합니다.

   3.1 기본정보

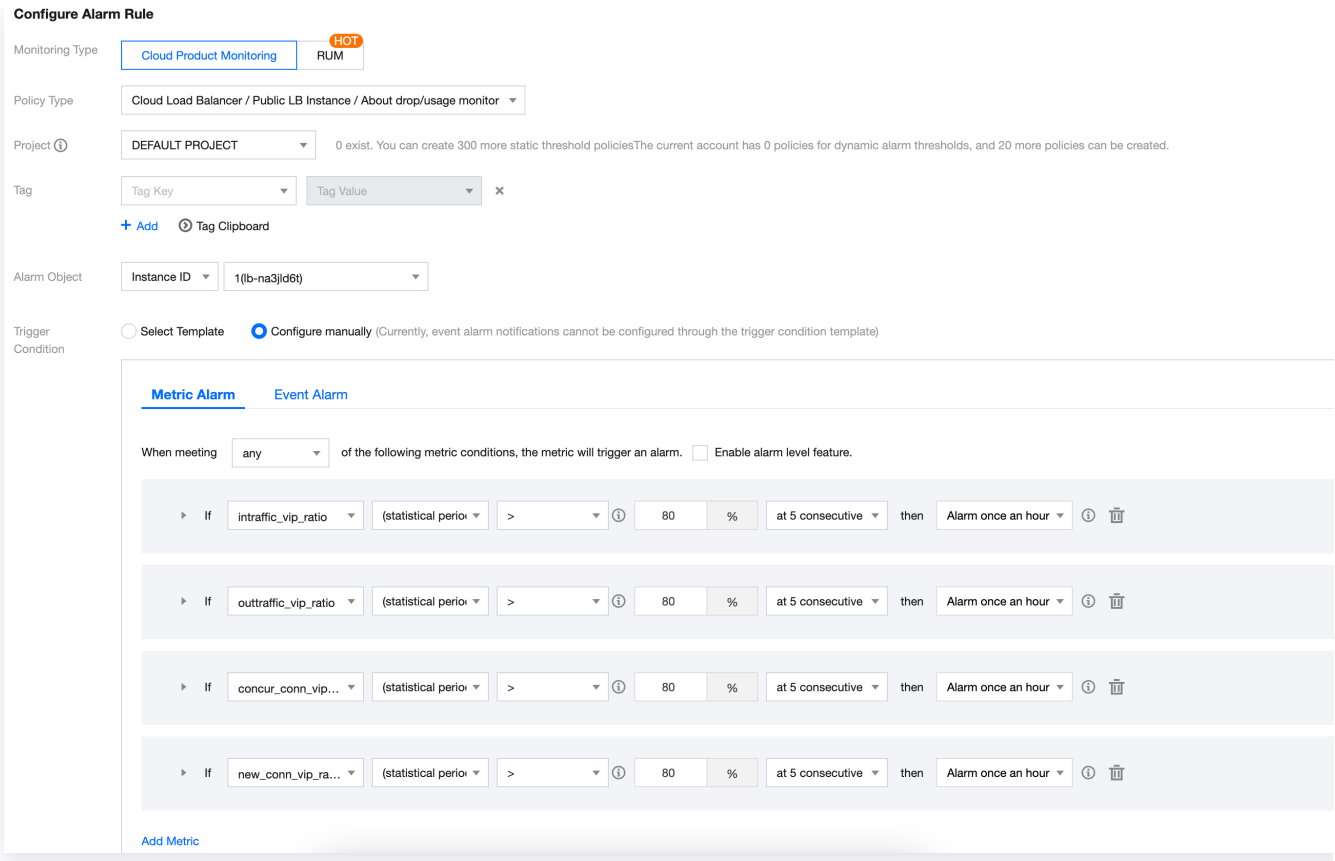
   - 정책 이름: 정책 이름을 최대 60자까지 입력합니다.

- 비고: 비고를 최대 100자까지 입력합니다.

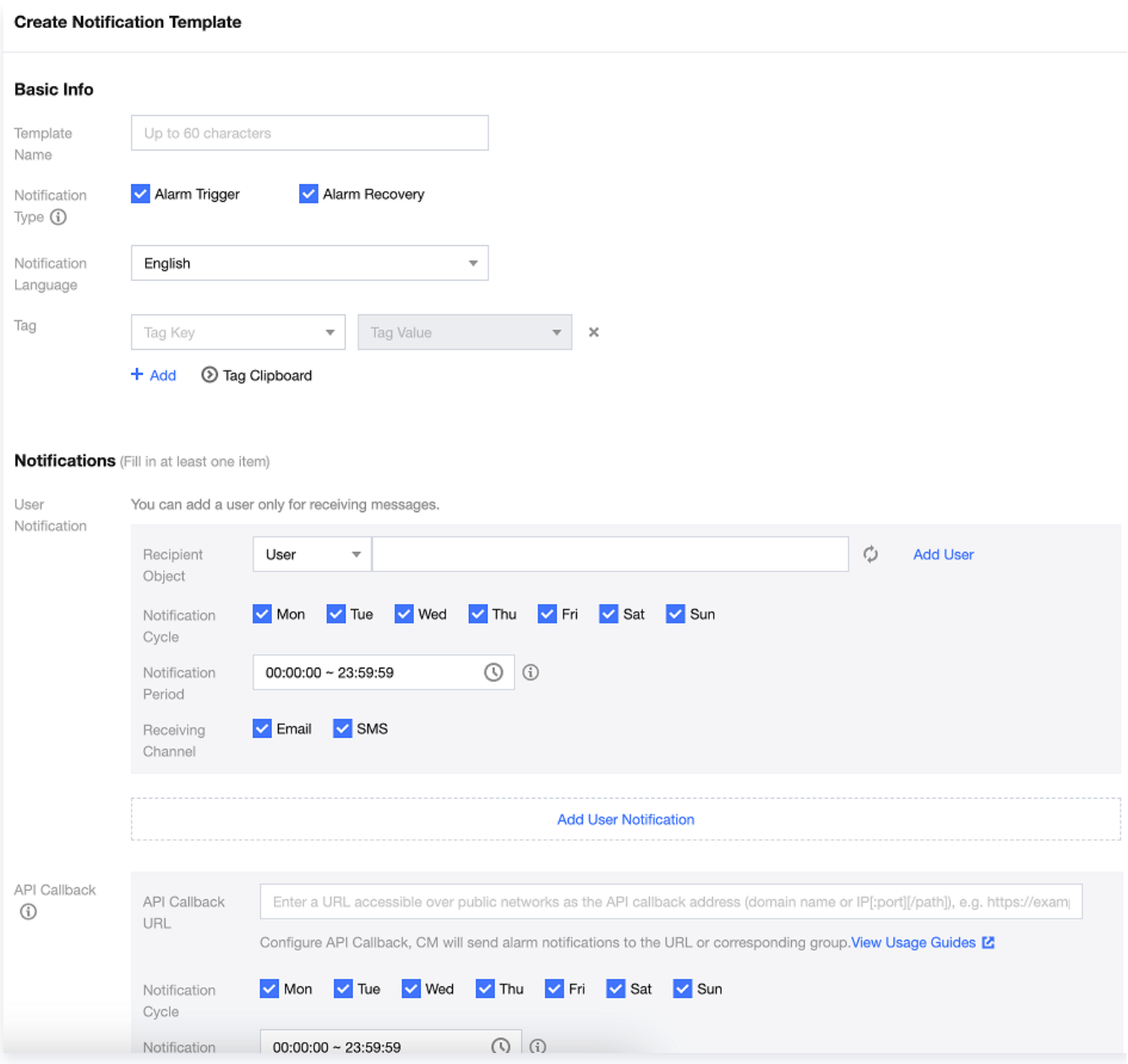


3.2 알람 규칙 구성

- 모니터링 유형: 클라우드 상품 모니터링을 선택합니다.
- 알람 정책 유형: **로드 밸런싱** > **공중망 로드 밸런싱 인스턴스** > **폐기/사용률 모니터링**을 선택합니다.
- 정책 소속 프로젝트: 정책이 속한 프로젝트를 선택합니다. 소속된 프로젝트는 알람 정책의 분류 및 권한 관리에 사용되며, 클라우드 상품 인스턴스의 프로젝트와는 강제 바인딩 관계가 없습니다.
- 레이블: 정책이 속한 레이블을 선택합니다.
- 알람 대상: 대상 인스턴스를 알람 대상으로 선택합니다.
- 트리거 조건: 알람 지표, 비교, 임계값, N개의 연속 모니터링 데이터 포인트 및 알람 빈도로 구성된 의미론적인 조건입니다.
  예를 들어 알람 지표는 **인바운드 대역폭 사용률**로, 비교 관계가 **>**로, 임계값이 **80%** 로, **연속 모니터링 데이터 포인트가 5개**로, 알람 빈도가 **1 시간 당 1회**입니다. 즉, 로드 밸런싱 인스턴스의 인바운드 대역폭 사용률이 5회 연속 80%를 초과하는 경우 알람이 트리거됩니다. 알람 빈도는 1 시간 당 1회입니다.
  아래 그림과 같이 인바운드 대역폭 사용률, 아웃바운드 대역폭 사용률, 최대 연결 수 사용률 및 새 연결 수 사용률을 선택하여 구성합니다.

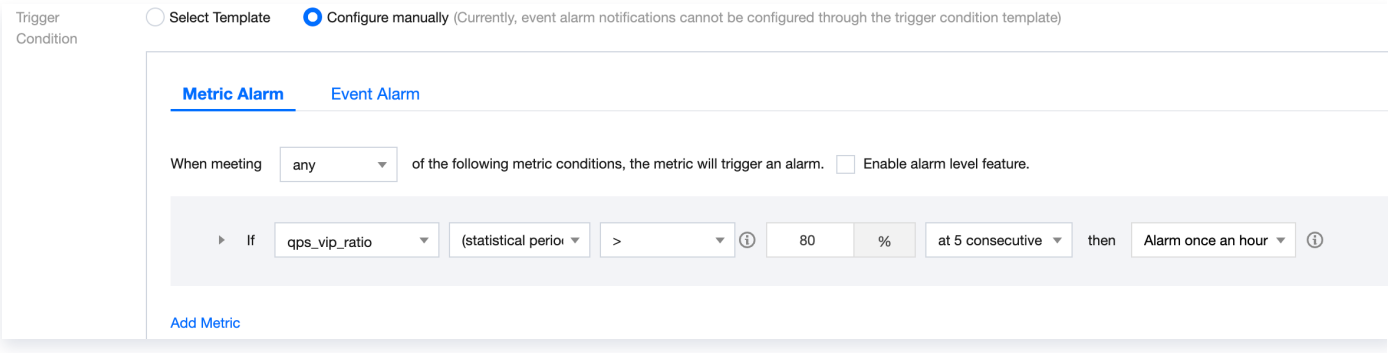


3.3 알람 구성: 알림 템플릿을 추가하고 알람 수신자, 알림 주기 및 수신 채널을 선택합니다. 알림 템플릿이 생성되지 않은 경우 **새 템플릿**을 클릭하여 생성하십시오. 자세한 내용은 새 알림 템플릿 을 참고하십시오.



4. **완성**을 클릭하여 인바운드 대역폭 사용률, 아웃바운드 대역폭 사용률, 최대 연결 수 사용률 및 새 연결 수 사용률에 대한 모니터링 알람 구성을 완료합니다. QPS 사용률 모니터링 알람의 경우 이전 단계를 참조하여 새 알

람 정책을 생성하고 정책 유형을 **로드 밸런싱 > 공중망 로드 밸런싱 인스턴스>QPS관련 모니터링**으로 수정하고 트리거 조건에 대해 다음 내용을 구성하면 됩니다.



해결 방안

위의 알람을 받았다면 이는 업무량이 증가했고 현재 표준 성능 및 용량 인스턴스 사양이 성능 상한선에 도달할 것이며 업무 요구를 충족할 수 없음을 의미합니다. 업무가 영향을 받지 않도록 성능 및 용량 인스턴스 사양 조정 으로 이동하십시오.

다중 가용존에서 HA 구현

최종 업데이트 날짜:: 2025-11-06 18:23:42

Cloud Load Balancer(CLB)의 고가용성은 시스템 아키텍쳐, 제품 구성 등 다중 차원에서 보장됩니다. 서비스 시나리오 및 요구사항에 따라 리전 간 재해 복구, 동일 리전 가용존 간 재해 복구 등 다양한 기능 솔루션을 선택할 수 있습니다.

CLB 클러스터 고가용성

Cloud Load Balancer(CLB) 인스턴스는 클러스터 배치를 채택하여 대화 동기화를 지원하고 서버 단일 노드를 제거하고 시스템 중복성을 향상시키고 서비스 안정성을 보장합니다. 모든 CLB 인스턴스는 모두 클러스터 고가용성을 구비합니다.

- 4층은 주로 Tencent에서 자체적으로 개발한 Tencent Gateway(TGW)를 기반으로 Cloud Load Balancer를 실현하며, TGW는 신뢰성이 높고 확장성이 강하고 성능이 높고 공격 방지 능력이 강한 특성을 갖고, Data Plane Development Kit(DPDK) 고성능 포워딩을 지원하며, 단일 클러스터는 억대 동시 처리, 천만대 PPS를 지원할 수 있습니다. Tencent Games, Tencent Video, WeChat, QQ 등을 포함하여 Tencent 내부의 여러 업무는 모두 TGW를 통해 서비스에 접속합니다.
- 7층은 주로 Secure Tencent Gateway(STGW)를 기반으로 Cloud Load Balancer를 실현하며, STGW는 Tencent에서 Nginx를 기반으로 자체적으로 개발한 대규모 동시 처리를 지원하는 7층 Cloud Load Balancer 서비스로 Tencent 내 대량의 7층 업무 트래픽을 수용합니다.

단일 CLB 인스턴스 고가용성

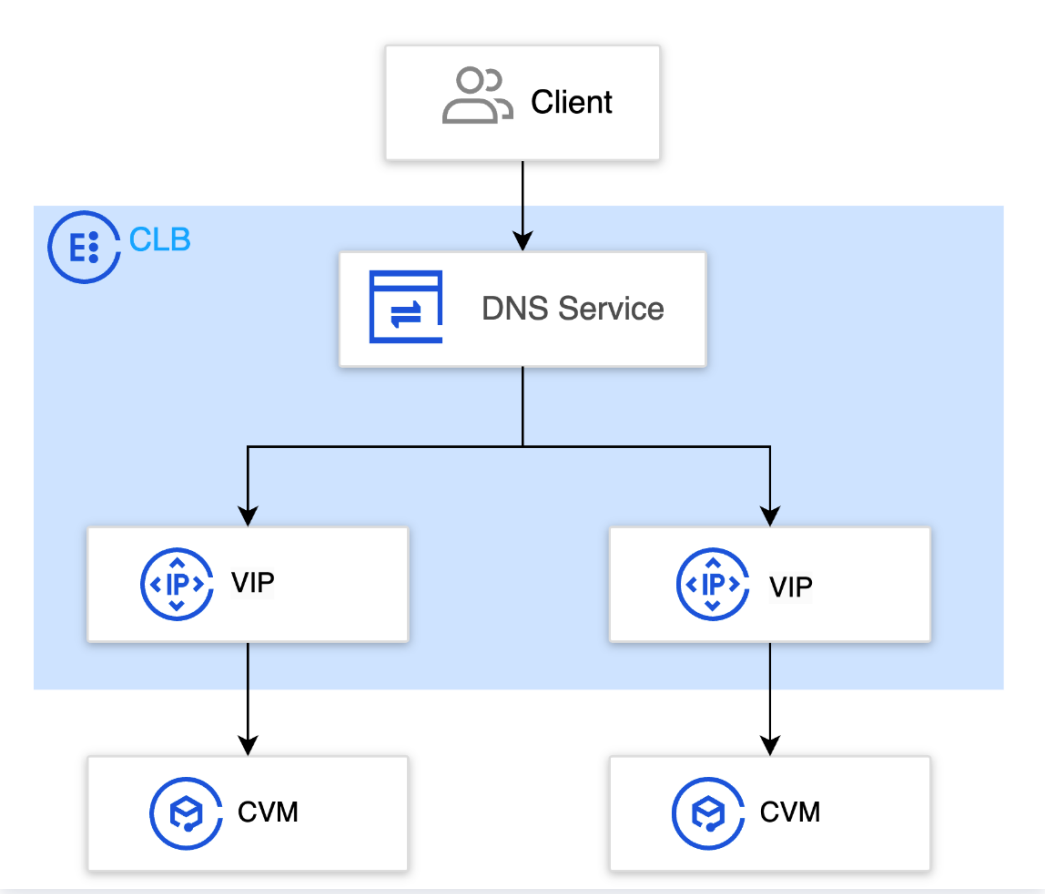
비 도메인 이름 기반 공중망 CLB

비 도메인 이름 기반 공중망 CLB는 VIP 형태로 서비스를 제공하며, SLA는 99.95%이고 VIP 소속 클러스터에는 2가지 배치 솔루션이 있습니다.

배치 모드	클러스터 재해 복구	가용존 간 재해 복구
단일 가용존	지원	지원 안 됨
다중 가용존	지원	지원, 마스터/슬레이브 가용존 모드, 마스터 가용존에 고장이 발생할 때 Cloud Load Balancer는 아주 짧은 시간 내 에 슬레이브 가용존으로 자동 전환하고 서비스를 회복합니다.

도메인 이름 기반 공중망 CLB

'도메인 이름 기반 공중망 CLB'는 상기 '비 도메인 이름 기반 공중망 CLB'에 기초하여 한 층의 DNS 서비스가 추가되었으며, SLA는 99.95%에서 99.99%로 향상되고 고장 ;VIP 자동 전환이 가능하여 고가용성이 향상됩니다. 자세한 내용은 도메인 이름 기반 공중망 CLB 출시 공고를 참고하시기 바랍니다.

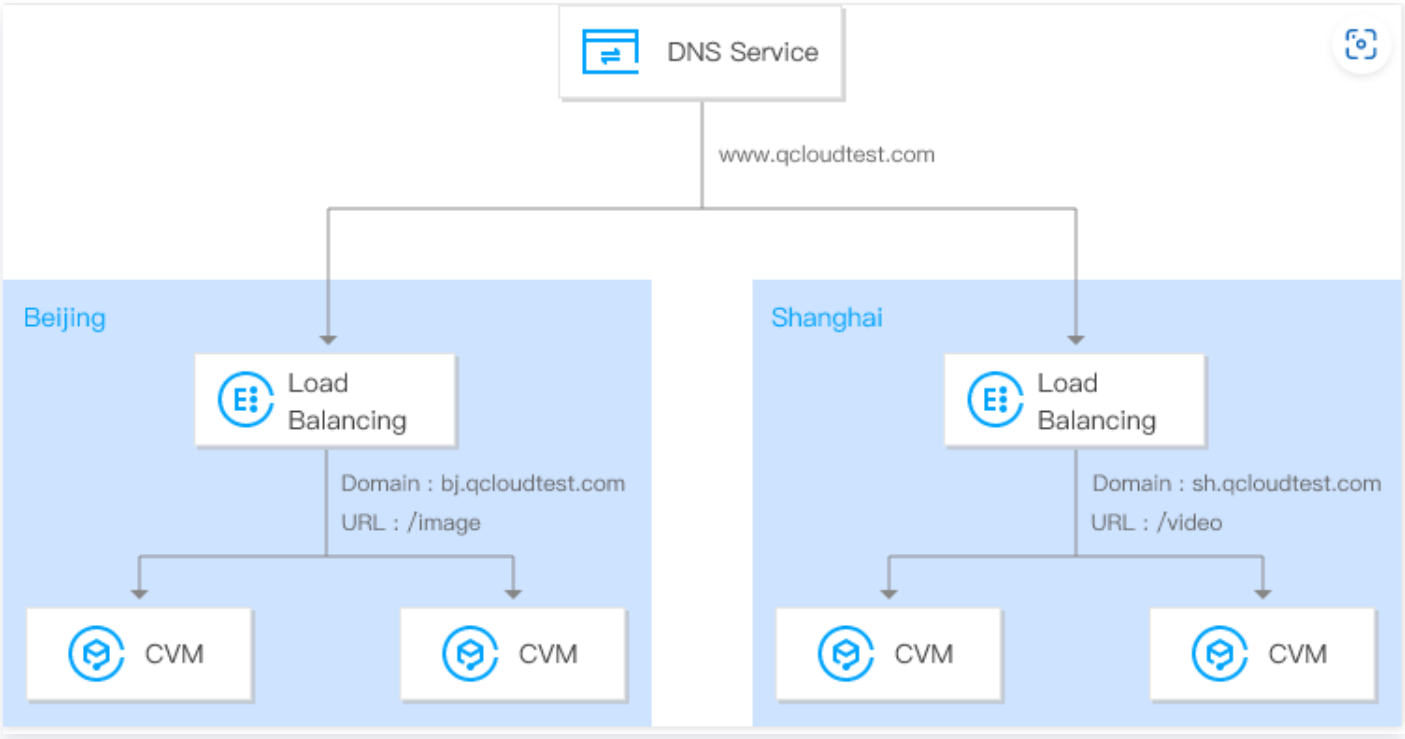


사설망 CLB

사설망 CLB는 인근 접속 아키텍처 배치를 채택합니다. 즉, 단일 CLB 인스턴스는 하나 이상의 가용존에 배포되며, 클라이언트에서 해당 CLB에 접근할 때 접근 트래픽은 지연이 가장 낮은 가용존의 클러스터를 자동으로 선택한 후 리얼 서버로 포워딩됩니다. 특정 가용존의 CLB 클러스터가 사용할 수 없는 경우, 다른 가용존의 CLB 클러스터로 전환할 수 있습니다.

다중 CLB 인스턴스 고가용성

가용성에 대한 요구가 아주 높다면 CLB 인스턴스 자체의 가용성 보장 메커니즘은 네트워크 공격, 리전 간 전환, 구성 오류 등 시나리오의 요구사항을 충족시키지 못할 수 있습니다. 다수의 CLB 인스턴스를 생성하여 DNS를 통해 접근 트래픽을 스케줄링할 수 있습니다.



다중 CLB 인스턴스 고가용성과 도메인 이름 기반 공중망 CLB 비교:

비교 항목	도메인 이름 기반 공중망 CLB	다중 CLB 인스턴스 고가용성
SLA	99.99%	99.95%
재해 복구 전환	링크 감지 및 재해 복구 전환 능력을 제공하기 때문에 단일 IP 입구 중단 문제를 걱정할 필요가 없습니다. 단일 IP 고장 발생 시 고장 IP를 자동으로 전환하여 업무에 대한 영향을 줄일 수 있습니다.	구성된 DNS 해석 및 전환 정책에 의존하므로 업무의 적시 발견 및 전환이 필요합니다.
운영 및 유지보수 관리	단일 인스턴스 구성만 필요합니다.	다수의 CLB 인스턴스 및 대응하는 DNS 해석 정책을 구성해야 합니다.
비용	비용이 저렴하고 CLB 관련 비용만 부과됩니다.	다수의 CLB 인스턴스 및 DNS 해석 등 컴포넌트를 배치해야 하기 때문에 비용이 더 높습니다.
리전	CLB 인스턴스 소속 클러스터는 단일 리전에 배치됩니다.	다중 리전의 CLB 인스턴스를 선택할 수 있습니다.

모범 사례:

- 업무가 단일 리전으로 배치되는 경우 고장 IP 자동 전환이 가능한 도메인 이름 기반 공중망 CLB 솔루션을 우선적으로 선택하는 것을 추천합니다.
- 업무가 다중 리전으로 배치되고 재해 복구 요구사항이 아주 높은 경우 다중 CLB 인스턴스 고가용성 솔루션을 선택하는 것을 추천합니다.
- 동일한 클라이언트에서 동일한 시점에 상이한 중간 노드를 통해 동일한 백엔드 서버의 동일한 포트에 접근하는 경우 스트리밍 현상이 발생할 수 있습니다. 자세한 내용은 스트리밍 문제 설명을 참고하시기 바랍니다.

백엔드 서비스 고가용성

Cloud Load Balancer(CLB)는 백엔드 서비스 비정상으로 인해 프론트엔드의 업무에 영향을 미치는 것을 방지하기 위해 헬스 체크를 통해 백엔드 서비스의 가용성을 판단함으로써 업무 전체의 가용성을 향상시킵니다.
상태 확인이 활성화되면 백엔드 서버 가중치(가중치 0을 포함)에 상관없이 Cloud Load Balancer 인스턴스는 헬스 체크를 항상 수행합니다. 인스턴스 목록 페이지의 **헬스 상태** 열에서 헬스 체크 상태를 조회하거나 리스너의 백엔드 서버 바인딩 서비스 상세정보 페이지에서 헬스 체크 상태를 조회할 수 있습니다. 헬스 체크에 관한 상세한 메커니즘은 헬스 체크 개요를 참고하시기 바랍니다.

로드 밸런싱 알고리즘 선택 및 가중치 구성 예시

최종 업데이트 날짜:: 2024-01-04 20:14:11

CLB 알고리즘의 비교 분석

가중 라운드 로빈 스케줄링 Weighted Round-Robin Scheduling

가중 라운드 로빈 스케줄링 알고리즘은 폴링을 기반으로 다른 서버에 대한 요청을 스케줄링하는 것입니다. 다른 서버의 불균형 성능 문제를 해결할 수 있습니다. 가중치를 사용하여 서버의 처리 성능을 나타내고 폴링 방식으로 가중치별로 다른 서버에 대한 요청을 스케줄링합니다. 새로운 연결 수를 기반으로 서버를 스케줄링합니다. 여기서 가중치가 더 높은 서버가 더 일찍 연결을 수신하고 폴링할 가능성이 더 높습니다. 동일한 가중치를 가진 서버는 동일한 수의 연결을 처리합니다.

- **장점**: 이 알고리즘은 단순성과 높은 실용성을 특징으로 합니다. 모든 연결의 상태를 기록할 필요가 없으므로 상태 비저장 스케줄링 알고리즘입니다.
- **단점**: 이 알고리즘은 비교적 단순하여 요청의 서비스 시간이 크게 변경되거나 각 요청에 다른 시간을 소비해야 하는 상황에는 적합하지 않습니다. 이러한 경우 서버 간에 부하 분산이 불균형하게 발생합니다.
- **적용 가능한 시나리오**: 이 알고리즘은 각 요청이 기본적으로 최고의 로드 성능으로 백엔드에서 동일한 시간을 소비하는 시나리오에 적합합니다. 일반적으로 HTTP 서비스와 같은 비지속 연결 서비스에서 사용됩니다.
- **권장 사항**: 각 요청이 기본적으로 백엔드에서 동일한 시간을 소비한다는 것을 알고 있는 경우(예시: 리얼 서버에서 처리되는 요청이 동일한 유형 또는 유사한 유형임) 가중 라운드 로빈 스케줄링을 사용하는 것이 좋습니다. 각 요청 간의 시간 차이가 작은 경우, 이 알고리즘도 순회가 필요 없고, 효율성이 높으므로 권장됩니다.

가중 최소 연결 스케줄링 Weighted Least-Connection Scheduling

- **원리**
  실제 상황에서는 각 클라이언트 요청이 서버에서 소비하는 시간이 크게 다를 수 있습니다. 작업 시간이 길어질수록 단순 라운드 로빈 또는 랜덤 부하 분산 알고리즘을 사용하는 경우 각 서버의 연결 프로세스 수가 크게 달라지고 부하 분산이 이루어지지 않을 수 있습니다. 라운드 로빈 스케줄링과 달리 최소 연결 스케줄링은 활성 연결 수를 기반으로 서버의 부하를 추정하는 동적 스케줄링 알고리즘입니다. 스케줄러는 각 서버에 현재 설정된 연결 수를 기록해야 합니다. 요청이 서버에 예약된 경우 연결 수가 1 증가합니다. 연결이 중지되거나 시간 초과되면 연결 수가 1 감소합니다. 가중 최소 연결 스케줄링 알고리즘은 최소 연결 스케줄링을 기반으로 하며 처리 성능에 따라 서버에 다른 가중치가 할당됩니다. 그 가중치에 따라 서버는 해당 수의 요청을 수신할 수 있습니다. 이 알고리즘은 최소 연결 스케줄링을 개선한 것입니다.
  1.1 리얼 서버의 가중치가 Wi(i=1...n)이고 현재 연결 수가 Ci(i=1...n)라고 가정합니다. 각 서버의 Ci/Wi 값은 순서대로 계산됩니다. 가장 작은 Ci/Wi 값을 가진 RS가 새 요청을 받는 다음 서버가 됩니다.
  1.2 동일한 Ci/Wi 값을 가진 RS의 경우 가중치 라운드 로빈 스케줄링을 기반으로 스케줄링됩니다.
- **장점**

이 알고리즘은 FTP와 같이 오랜 시간 처리가 필요한 요청에 적합합니다.

- **단점**
  API 제한으로 인해 최소 연결 및 세션 지속성을 동시에 활성화할 수 없습니다.
- **적용 시나리오**
  이 알고리즘은 백엔드에서 각 요청에 사용되는 시간이 크게 달라지는 시나리오에 적합합니다. 일반적으로 지속 연결 서비스에 사용됩니다.
- **권장 사항**
  다른 요청을 처리해야 하고 백엔드에서 서비스 시간이 크게 달라지는 경우(예시: 3ms 및 3s) 부하 분산을 달성하기 위해 가중 최소 연결 스케줄링을 사용하는 것이 좋습니다.

원본 해싱 스케줄링(ip_hash)

- **원리**
  원본 해싱 스케줄링은 요청의 원본 IP 주소를 해시 키(Hash Key)로 사용하고 정적으로 할당된 해시 테이블에서 해당 서버를 찾습니다. 사용 가능하고 오버로드되지 않은 경우 요청이 이 서버로 전송됩니다. 그렇지 않으면 null이 반환됩니다.
- **장점**
  ip_hash는 원본 IP를 기억하고 hash 테이블을 통해 client의 요청을 동일한 rs로 매핑하여 특정 세션 지속성을 달성할 수 있습니다. 세션 지속성이 지원되지 않는 시나리오에서는 ip_hash를 스케줄링에 사용할 수 있습니다.
- **권장 사항**
  이 알고리즘은 요청 원본 주소의 hash 값을 계산하고 가중치에 따라 해당 리얼 서버에 요청을 배포합니다. 이를 통해 동일한 클라이언트 IP의 요청을 동일한 서버에 배포할 수 있습니다. 이 알고리즘은 cookie를 지원하지 않는 TCP 프로토콜을 통한 부하 분산에 적합합니다.

부하 분산 알고리즘 선택 및 가중치 구성 예시

CLB의 향후 기능에서 **레이어 7 포워딩은 최소 연결 부하 분산 방법**을 지원합니다. 부하 분산 알고리즘을 선택하고 참고용으로 가중치를 구성하는 방법에 대한 예시를 제공하므로 RS 클러스터가 다양한 시나리오에서 안정적으로 비즈니스를 수행할 수 있습니다.

- 시나리오1
  동일한 구성(CPU / 메모리)을 가진 3개의 RS가 있고 이들의 가중치를 모두 10으로 설정하고, 각 RS와 client 간에 100개의 TCP 연결이 설정되었다고 가정합니다. 새로운 RS가 추가되는 경우 4번째 서버의 부하를 빠르게 증가시키고 다른 3개의 서버에 대한 부하를 줄일 수 있는 최소 연결 스케줄링 알고리즘을 사용하는 것이 좋습니다.
- 시나리오2
  Tencent Cloud 서비스를 처음 사용한다고 가정합니다. 귀하의 웹사이트는 막 구축되었으며 로드가 적습니다. 모두 액세스 레이어 서버이므로 동일한 구성의 RS를 구입하는 것이 좋습니다. 이 시나리오에서는 모든 RS의 가중치를 10으로 구성하고 가중치 기반 라운드 로빈 스케줄링 알고리즘을 사용하여 트래픽을 분산할 수 있습니다.
- 시나리오3

정적 웹 사이트에 대한 간단한 액세스 요청을 수행하는 5개의 리얼 서버가 있고 해당 서버의 컴퓨팅 성능 비율(CPU 및 메모리로 계산)이 9:3:3:3:1이라고 가정합니다. 이 시나리오에서는 RS의 가중치를 각각 90, 30, 30, 30, 10으로 구성할 수 있습니다. 정적 웹 사이트에 대한 액세스 요청은 대부분 비지속 연결 유형이므로 가중치 라운드 로빈 스케줄링 알고리즘을 사용할 수 있으므로 CLB는 RS의 성능 비율에 따라 요청을 할당할 수 있습니다.

- 시나리오4
  엄청난 양의 Web 액세스 요청을 처리하기 위해 10개의 RS가 있고 더 이상 서버를 구입하고 싶지 않다고 가정합니다. 과부하로 인해 서버 중 하나가 다시 시작되는 경우가 많습니다. 이 시나리오에서는 RS의 성능을 기반으로 기존 서버의 가중치를 구성하는 것이 좋습니다. 부하가 높은 서버는 무게가 더 작아야 합니다. 또한 최소 연결 스케줄링 알고리즘을 사용하여 서버 과부하를 피하기 위해 활성 연결이 적은 RS에 요청을 할당할 수 있습니다.
- 시나리오5
  지속 연결을 처리할 RS가 3개 있고 컴퓨팅 성능의 비율(CPU와 메모리로 계산)이 3:1:1이라고 가정합니다. 성능이 가장 좋은 서버는 더 많은 요청을 처리하지만 과부하 상태가 되는 것을 원하지 않습니다. 대신 유휴 서버에 새 요청을 할당하려고 합니다. 이 시나리오에서는 최소 연결 스케줄링 알고리즘을 사용하고 사용량이 많은 서버의 가중치를 적절하게 줄일 수 있으므로 CLB가 활성 연결이 적은 RS에 요청을 할당하여 부하 분산을 달성할 수 있습니다.
- 시나리오6
  클라이언트의 후속 요청이 동일한 서버에 할당되기를 원한다고 가정합니다. 가중 라운드 로빈 또는 가중 최소 연결 스케줄링은 동일한 클라이언트의 요청이 동일한 서버에 할당되도록 보장할 수 없습니다. 지정된 애플리케이션 서버의 요구 사항을 충족하고 클라이언트 세션의 '고정성'(또는 '연속성')을 유지하려면 ip_hash를 사용하여 트래픽을 분산하는 것이 좋습니다. 이 알고리즘은 서버 수가 변경되거나 서버를 사용할 수 없게 되지 않는 한 동일한 클라이언트의 모든 요청이 동일한 RS에 배포되도록 합니다.

가중치를 0으로 재설정하는 것과 바인딩 해제의 차이

- 가중치를 0으로 재설정: TCP 리스너는 기존 연결을 계속 포워딩하고 UDP 리스너는 동일한 5배 연결을 포워딩하며 HTTP/HTTPS 리스너는 기존 연결을 계속 포워딩합니다.
- RS 바인딩 해제: TCP/UDP 리스너는 기존 연결 포워딩을 중지하고 HTTP/HTTPS 리스너는 기존 연결을 계속 포워딩합니다.

관련 문서

Managing Real Servers

CLB 수신 도메인 이름에 대한 WAF 보호 구성하기

최종 업데이트 날짜:: 2024-01-04 20:16:06

CLB WAF는 도메인 이름을 CLB 리스너와 바인딩하여 CLB 리스너를 통과하는 HTTP 또는 HTTPS 트래픽을 감지하고 차단할 수 있습니다. 본문은 CLB WAF를 사용하여 CLB에 추가된 도메인 이름에 Web 보안 보호를 적용하는 방법을 소개합니다.

전제 조건

- CLB WAF는 현재 베타 버전입니다. 사용하려면 신청서를 제출해야 합니다.
- HTTP 또는 HTTPS 리스너를 생성 완료하여 도메인 이름에 액세스할 수 있어야 합니다. 자세한 내용은 Getting Started with CLB를 참고하십시오.
- CLB WAF 서비스를 구입 완료해야 합니다. 자세한 내용은 Purchase Guide를 참고하십시오.

제한 조건

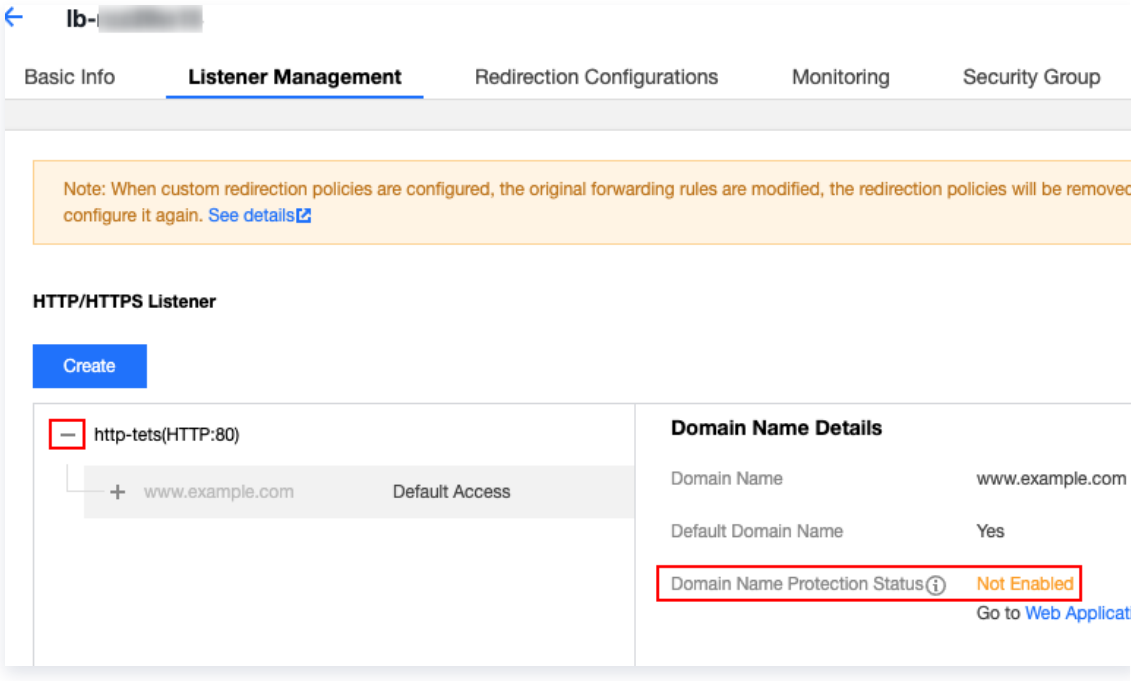
현재 IPv4 CLB 인스턴스만 CLB WAF 보호를 지원하며 이 기능은 IPv6 및 IPv6 NAT64에 사용할 수 없습니다.

작업 단계

1단계: CLB 도메인 이름 구성 확인

본문은 `www.example.com` 이라는 도메인 이름을 예로 들어 설명합니다.

1. CLB 콘솔에 로그인하고 왼쪽 사이드바에서 CLB 인스턴스 목록을 클릭하여 [인스턴스 관리] 페이지로 이동합니다.
2. '인스턴스 관리' 페이지에서 인스턴스 영역을 선택한 다음 대상 인스턴스의 오른쪽에 있는 '작업' 열에서 [리스너 구성]을 클릭합니다.
3. '리스너 관리' 탭을 선택하고 'HTTP/HTTPS 리스너' 섹션에서 대상 리스너 왼쪽의 [+] 아이콘을 클릭하여 도메

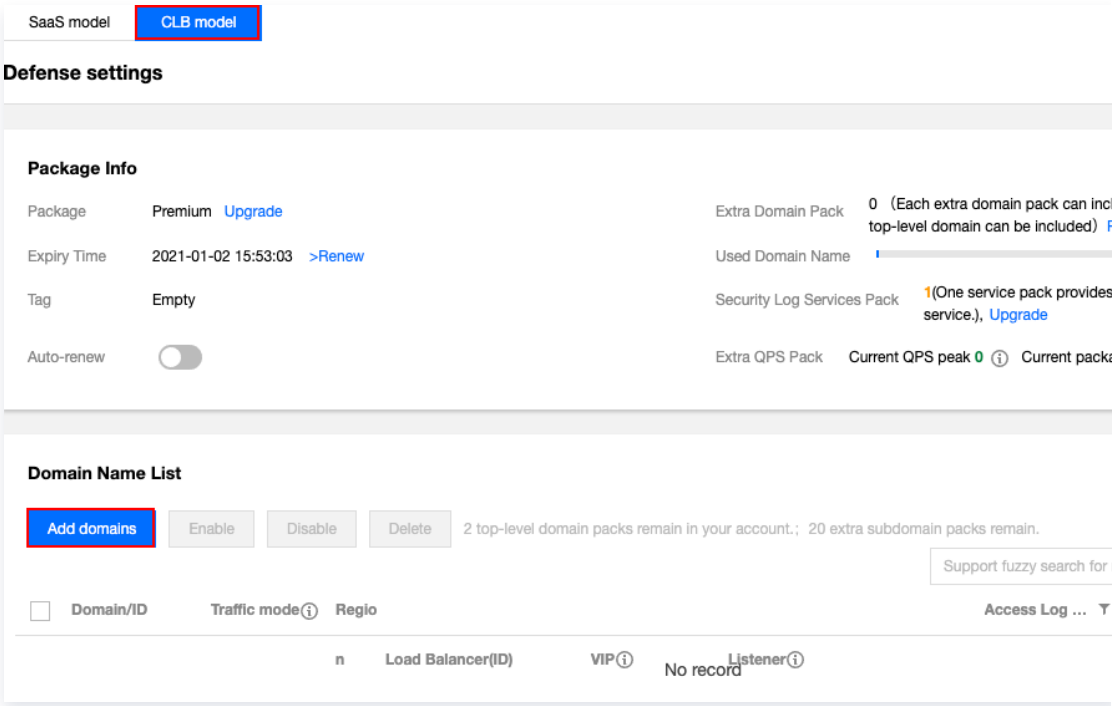


4. 다음과 일치하도록 CLB 도메인 이름 구성을 확인하십시오. CLB 인스턴스 ID: 'lb-f8lm****', 리스너 이름: 'http-test', 도메인 이름: `www.example.com` , 도메인 이름 보호 상태: '비활성화됨'(ID, 이름 및 도메인 이름은 실제 경우에 따라 다름).

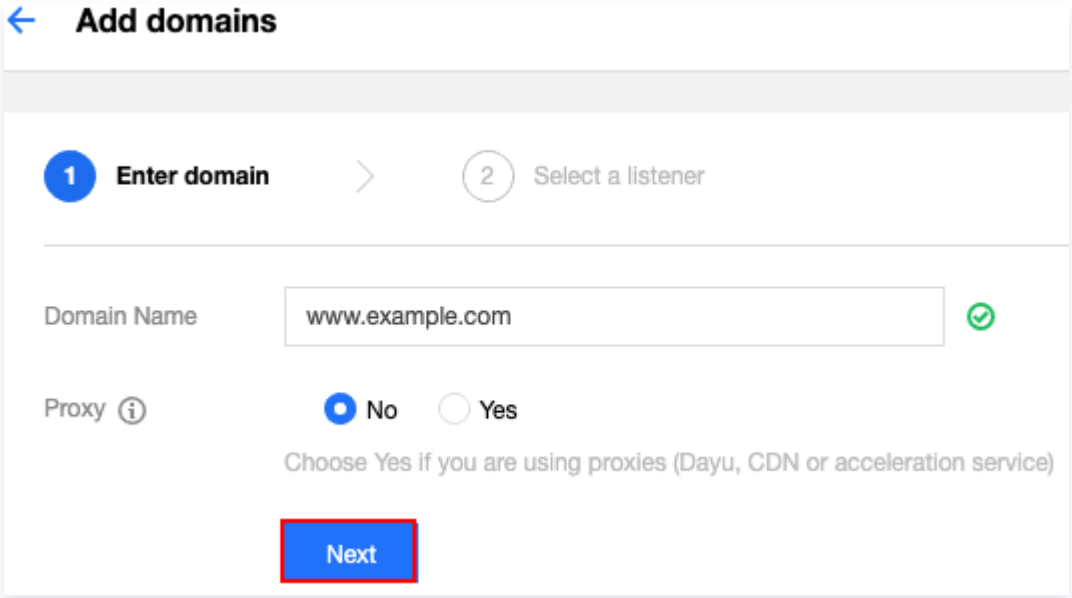
2단계: WAF 콘솔에서 도메인 이름 추가 및 CLB 인스턴스에 바인딩

CLB WAF 서비스로 도메인 이름에 보호를 적용하려면 WAF에 CLB 수신 도메인 이름을 추가하고 CLB 리스너와 바인딩해야 합니다.

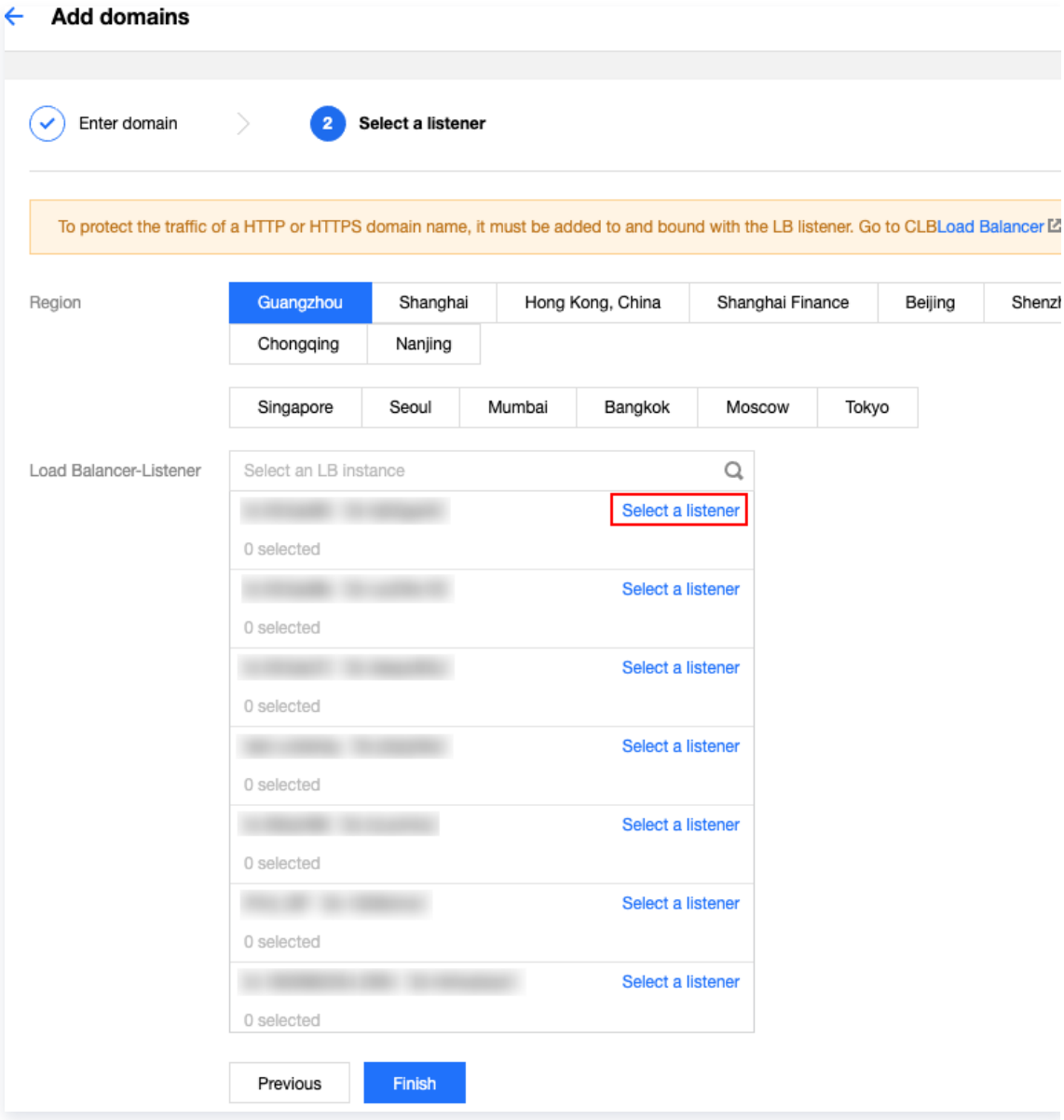
1. WAF 콘솔에 로그인하고 왼쪽 사이드바에서 [Web Application Firewall]>[보호 설정]을 선택합니다.
2. [CLB] 탭을 선택합니다.
3. [도메인 추가]를 클릭합니다.



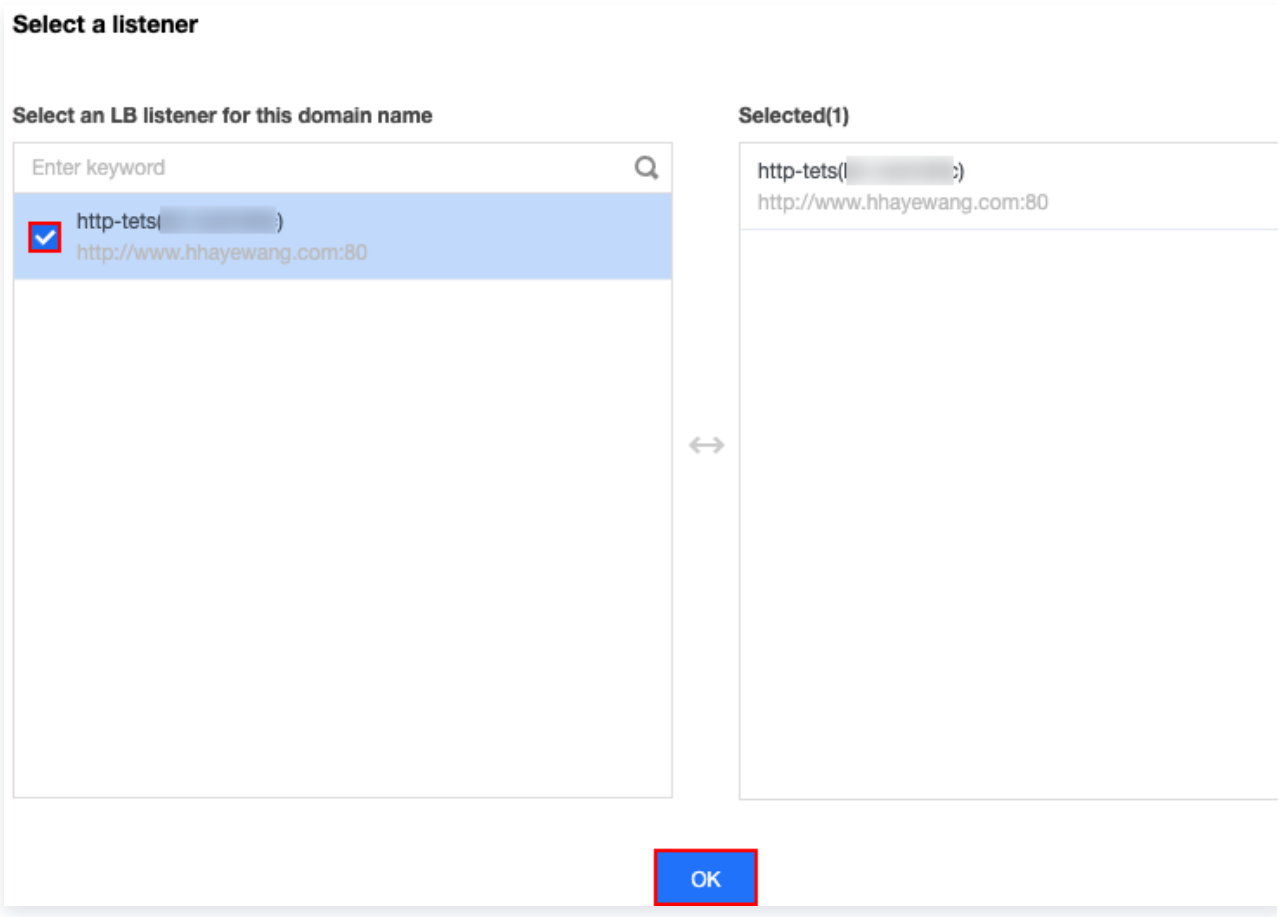
4. '도메인 이름을 입력'하고 [다음]을 클릭합니다.



5. CLB 리전을 선택한 다음 1단계: CLB 도메인 이름 구성 확인에서 도메인 이름을 선택하고 [리스너 선택]을 클릭



6. 팝업 창의 1단계: CLB 도메인 이름 구성 확인 에서 CLB 리스너를 선택한 후 [확인]을 클릭합니다.



7. ‘리스너 선택’ 단계에서 [완료]를 클릭하여 WAF에서 CLB 리스너와 도메인 이름 바인딩을 완료합니다.
8. ‘도메인 목록’ 페이지로 돌아가서 도메인 이름, 리전, 바인딩된 CLB 인스턴스 ID, 리스너 및 기타 정보를 확인합니다.

3단계: 결과 확인

1. 1단계: CLB 도메인 이름 구성 확인의 지침에 따라 도메인 이름을 확인합니다. 도메인 이름 보호가 ‘활성화’되고 트래픽 모드가 ‘미러’인 경우 도메인 이름 보호가 활성화됩니다.
   - 도메인 이름에 대한 DNS 리졸브를 구성하지 않은 경우 Step 2. Perform Local Testing을 참고하여 WAF 보호가 활성화되었는지 확인하십시오.
   - 도메인 이름에 대해 DNS 리졸브를 구성한 경우 아래 지침에 따라 WAF 보호가 활성화되어 있는지 확인하십시오.
2. 브라우저를 통해 `http://www.example.com/?test=alert(123)` 를 방문하십시오.
3. WAF 콘솔에 로그인한 다음 왼쪽 사이드바에서 [로그 서비스]>[공격 로그]를 선택합니다.
4. ‘로그 검색’ 탭을 선택하고 추가된 보호 도메인 이름 `www.example.com` 을 선택한 후 [검색]을 클릭합니다. CLB에 구성된 도메인 이름에 대한 WAF 보호는 로그 목록에 ‘XSS 공격’ 로그가 있는 경우에 유효합니다.