Cloud Load Balance Practical Tutorial

Copyright Notice

Trademark Notice

Service Notice

This document provides an overview of the as-is details of Tencent Cloud's products and services in their entirety or part. The descriptions of certain products and services may be subject to adjustments from time to time.

The commercial contract concluded by you and Tencent Cloud will provide the specific types of Tencent Cloud products and services you purchase and the service standards. Unless otherwise agreed upon by both parties, Tencent Cloud does not make any explicit or implied commitments or warranties regarding the content of this document.

Contact Us

We are committed to providing personalized pre-sales consultation and technical after-sale support. Don't hesitate to contact us at 4009100100 or 95716 for any inquiries or concerns.

Contents

Practical Tutorial
- Enabling Gzip Compression & Testing
- Deploying a Certificate to Load Balancer
- HTTPS Forwarding Configurations
- Obtaining Real Client IPs
  - Real Servers Obtaining the Real Client IP Address Through CLB
  - Obtaining Real Client IPs in IPv4 CLB Scenarios
  - Obtaining Real Client IPs via TOA in Hybrid Cloud Deployment
- Implementing HA Across Multiple AZs
- Load Balancing Algorithm Selection and Weight Configuration Examples
- Configuring WAF protection for CLB listening domain names

Practical Tutorial
Enabling Gzip Compression & Testing

Last updated: 2023-09-05 18:24:48

In **public network Cloud Load Balancer instances and Cloud Load Balancer instances with a static public IP address**, the **HTTP/HTTPS protocols** support enabling Gzip compression by default. Activating Gzip compression for web pages effectively reduces data volume during network transmission and enhances the browsing speed on the client's browser. While using this feature, please be mindful of the following considerations:

Supports and Limits

- **You must also enable Gzip compression on the backend CVM instances of CLB.**
  For common Nginx service containers, you must enable Gzip compression in their configuration files (nginx.conf by default) and restart the service.

```
gzip on;
```

- **Currently, Cloud Load Balancer supports the following file types. You can specify the file types for compression in the Gzip_types configuration option.**

```
application/atom+xml application/javascript application/json application/rss+xml
application/vnd.ms-fontobject application/x-font-ttf application/x-web-app-
manifest+json application/xhtml+xml application/xml font/opentype image/svg+xml
image/x-icon text/css text/plain text/x-component;
```

> **Note:**
> You must enable Gzip compression for the above file types in the business software of the backend CVM instances of CLB.

- **The client requests must carry the compression request identifier.**
  To enable compression, the client requests must also carry the following identifier:

```
Accept-Encoding: gzip,deflate,sdch
```

Example of enabling Gzip compression on CVM instances

Example of CVM runtime environment: Debian 6

1. Use Vim to open the Nginx configuration file based on the user path:

```
vim /etc/nginx/nginx.conf
```

2. Find the following code:

```
gzip on;
gzip_min_length 1k;
gzip_buffers 4 16k;
gzip_http_version 1.1;
gzip_comp_level 2;
gzip_types text/html application/json;
```

Description of the above code syntax:

- Gzip: Specifies whether to enable or disable the Gzip module.
  Syntax: `gzip on/off`
  Scopes: http, server, location
- gzip_min_length: Specifies the minimum number of bytes that a page can be compressed to. The number of bytes can be obtained from the Content-Length in the HTTP header. The default value is 1k.
  Syntax: `gzip_min_length length`
  Scopes: http, server, location
- gzip_buffers: configures the number and size of buffers for processing Gzip compressed files. 16k represents a unit of 16k, and memory is allocated at four times the original data size in 16k units.
  Syntax: `gzip_buffers number size`
  Scopes: http, server, location
- gzip_http_version: represents the minimum HTTP version that can utilize Gzip compression. Setting it to HTTP/1.0 indicates the lowest version required for Gzip functionality, thus making it backward compatible with HTTP/1.1. As Tencent Cloud now fully supports HTTP/1.1, there is no need to make any changes.
  Syntax: `gzip_http_version 1.0 | 1.1;`
  Scopes: http, server, location
- gzip_comp_level: specifies the Gzip compression ratio with a value range of 1-9. Value 1 is the smallest compression ratio with the fastest processing speed, while value 9 is the greatest compression ratio with the slowest processing speed (fast transmission with high CPU consumption).
  Syntax: `gzip_comp_level 1..9`
  Scopes: http, server, location
- gzip_types: compresses files based on their MIME types. By default, "text/html" type files are compressed. Additionally, Gzip in Nginx does not compress static resource files such as JavaScript and images by default. You can specify the MIME types to be compressed using gzip_types, and files with non-specified MIME types will not be compressed. **For example, to compress JSON format data, you need to add the application/json data type to this statement.**
  Supported types include:

  ```
  text/html text/plain text/css application/x-javascript text/javascript
  application/xml
  ```

Syntax: `gzip_types mime-type [mime-type ...]`
Scopes: http, server, location

3. To modify the configuration, save and exit the file, enter the Nginx bin file directory, and run the following command to reload Nginx:

```
./nginx -s reload
```

4. Use the following curl command to test whether Gzip compression is enabled:

```
curl -I -H "Accept-Encoding: gzip, deflate" "http://cloud.tencent.com/example/"
```

- If a result is returned, Gzip compression is enabled.
- If no result is returned, Gzip compression is not enabled.

Deploying a Certificate to Load Balancer

Last updated: 2023-09-05 18:27:10

Scenario

This document guides you on deploying an SSL certificate to Cloud Load Balance.

Preparations

You have logged in to the Certificate Management console and successfully obtained a certificate (refer to How to Apply for a Free Domain Certificate).

Instructions

> **Note:**
> Before proceeding, please ensure that you have instances in your Cloud Load Balance console. If not, create an instance first.

1. On the My Certificates > All page, click the **Issued** tab, select the certificate you want to deploy, and click the **Certificate ID** in the **Certificate Information column**.
2. Navigate to the **Certificate Details** management page, and in the **One-Click Certificate Deployment** section, click Cloud Load Balance.

> **Note:**
> Currently, the South China (Shenzhen Finance) region is not supported.

3. In the pop-up Deploy Certificate window, select **Deployment Type** as **Cloud Load Balance**, and choose the resource instance and listener resource.

> **Note:**
> If you have not created a listener for your Cloud Load Balance (CLB) instance, you can follow the instructions in Adding a Listener.

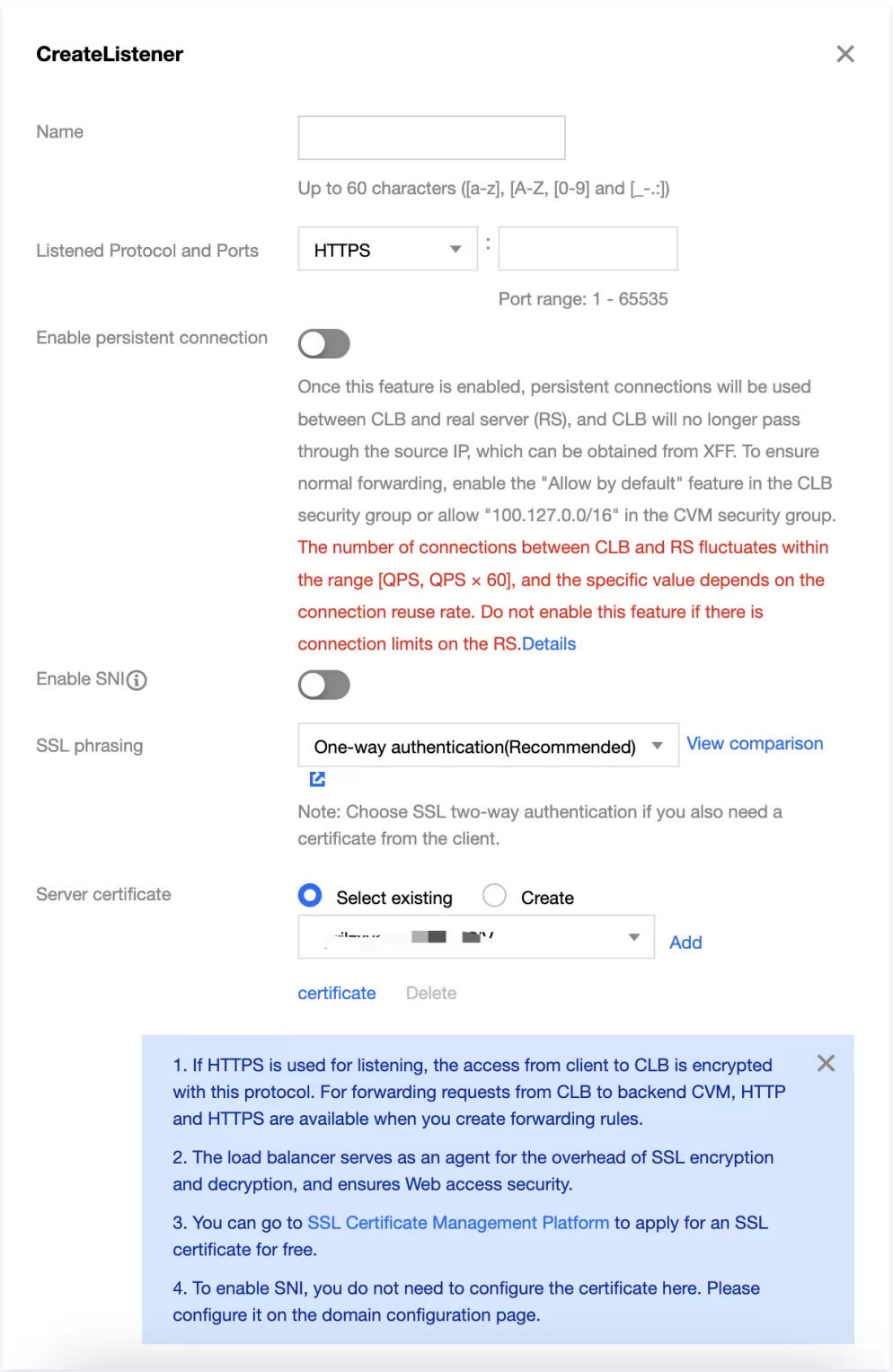
4. Click **OK** to complete the operation successfully, as shown below:

Adding a Listener

1. Log in to the Cloud Load Balance console, select the listener you need to configure, and click **Configure Listener**.
2. Navigate to the **Basic Information** page of the instance and select the **Listener Management** tab.
3. In the **HTTP/HTTPS listener** section, click **Create** to open the **Create Listener** pop-up window.
4. Switch the **listening protocol port** to HTTPS, and you can select an existing server certificate, as shown in the image below:

> **Note**

> If the **server certificate** selected here is the one to be deployed to the Cloud Load Balance instance, there is no need to perform the deployment to Cloud Load Balance operation.



5. Click **Submit** to successfully configure the listener.

Documentation

Manage Documentation

HTTPS Forwarding Configurations

Last updated: 2023-09-05 18:28:19

1. Cloud Load Balancer Capability Description

Tencent Cloud CLB Cloud Load Balancer achieves a significant improvement in HTTPS performance by deeply optimizing the protocol stack and server-side. Additionally, through international collaboration on certificates, we have substantially reduced certificate costs. Tencent Cloud CLB can bring remarkable benefits to your business in the following aspects:

1. The use of HTTPS does not affect the access speed of the client.
2. SSL encryption and decryption performance of a single server in a cluster can sustain full handshakes of up to 65,000 connections per second (CPS), which is at least 3.5 times higher than that of a high-performance CPU. This reduces server costs, greatly improves service capability during business peaks and traffic surges, and strengthens the computation-based anti-attack capability.
3. Offloading and conversion of multiple protocols are supported, which reduces the business stress in adaption to various client protocols. The business backend only needs to support HTTP/1.1 to use different protocols such as HTTP/2, SPDY, SSL 3.0 and TLS 1.2.
4. One-stop SSL certificate application, monitoring, and replacement services are provided. Tencent Cloud cooperates with Comodo and SecureSite, two leading global certificate authorities, to simplify the certificate application process and reduce application costs.
5. Anti-CC and WAF features are provided to effectively defend against various attacks at the application layer, such as slow HTTP attacks, high-traffic DDoS attacks, SQL injections, and website trojans.

2. HTTP and HTTPS Headers

CLB proxies HTTPS, handling both HTTP and HTTPS requests from clients. When forwarding requests to backend servers, the protocol between CLB and the backend service can be set to HTTP, HTTPS, or gRPC. For more information, please see Configuring an HTTPS Listener. When the protocol between CLB and the backend service is set to HTTP, developers may not be able to distinguish whether the frontend request is HTTP or HTTPS.

A CLB instance adds the `X-Client-Proto` header in a request before forwarding the request to a real server:

- X-Client-Proto: http (The client request is an HTTP request.)
- X-Client-Proto: https (The client request is an HTTPS request.)

3. Getting Started Configuration

Assume that you need to configure the website `https://example.com`, so that end users can visit it securely over HTTPS when they directly enter `www.example.com` in the browser.

For information about CLB operations, see the following documents:

- Create a Cloud Load Balancer Instance
- Configuring an HTTP Listener
- Managing Real Servers

The request for accessing `www.example.com` entered by an end user is forwarded as follows:

1. The request is transferred over HTTP and accesses port 80 of the CLB listener through VIP. Then, it is forwarded to port 8080 of the real server.
2. With the configuration of rewrite in Nginx on the real server, the request passes through port 8080 and is rewritten to the `https://example.com` page.
3. At this point, the browser sends the `https://example.com` request to the corresponding HTTPS site again. The request accesses port 443 of the Cloud Load Balancer listener through VIP and is then forwarded to port 80 of the backend server.

This operation rewrites a browser user's HTTP request to a more secure HTTPS request and is imperceptible to the user. To implement the above request forwarding operation, you can configure the real server as follows:

```
server {

    listen 8080;
    server_name    example.qcloud.com;

    location / {

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

}
}
```

Alternatively, in the new version of Nginx, redirect the Nginx HTTP page to the HTTPS page by using the recommended 301 redirect:

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

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

Obtaining Real Client IPs
Real Servers Obtaining the Real Client IP Address Through CLB

Last updated: 2025-06-26 15:13:39

Introduction of Obtaining the Real Client IP Address Through CLB

Layer-4 listeners (TCP/UDP/TCP SSL/QUIC) of CLB support obtaining the real client IP addresses directly from the real servers without additional configuration. By default, the source IP addresses obtained from the real servers are the real client IP addresses.
However, when there are one or more NAT gateways between the CLB and the real servers, the real servers cannot receive the real source IP address of the client. For this scenario, you can enable Proxy Protocol configuration for the layer-4 listeners of CLB, initiate Proxy Protocol actively, and carry the real source IP address of the client to the real servers through the Proxy Protocol.

Note:
- Using this feature requires the real servers to enable Proxy Protocol simultaneously. As a result, the real server can obtain the real client IP address. If the real server does not have the capability to parse the Proxy Protocol, enabling the feature switch directly may lead to parsing exceptions on real servers, thereby affecting service availability.
- This feature does not support online hitless migration. Switching to ProxyProtocol requires stopping service for upgrade. Configure with caution.
- Only Proxy Protocol v2 is supported in CLB. It supports multiple transmission protocols, such as TCP and UDP. For more information, see The PROXY protocol.

Overview

- This feature is only supported for standard accounts and is not available for traditional accounts. For the method to determine the account type, see Checking Account Type.
- Only TCP/UDP/TCP SSL/QUIC listeners of IPv4 and IPv6 instances support this feature.
- The Proxy Protocol configuration feature for TCP/UDP listeners of IPv6 CLB remains in grayscale. If this feature is required, submit a ticket for application.

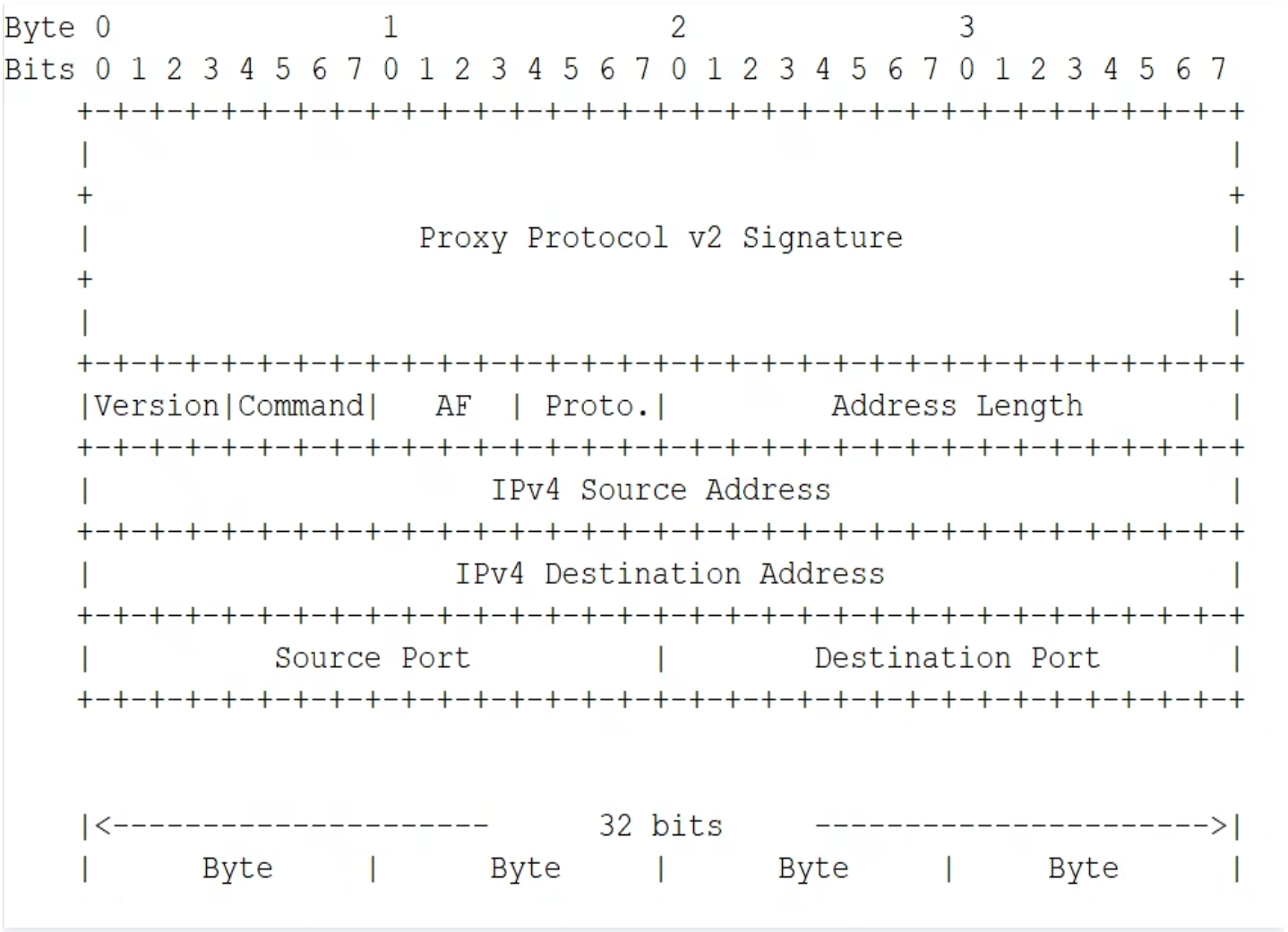
Proxy Protocol Description

The proxy server encapsulates the client's original network connection information in the request header using Proxy Protocol when forwarding requests and then sends it to the real server. The real server can obtain the real network connection information of the client, including the source IP address, source port, and transmission protocol, by parsing the Proxy Protocol header.
By using Proxy Protocol, the real server can accurately obtain the original network connection information of the client, enabling more accurate logging, access control, traffic monitoring, and other operations.

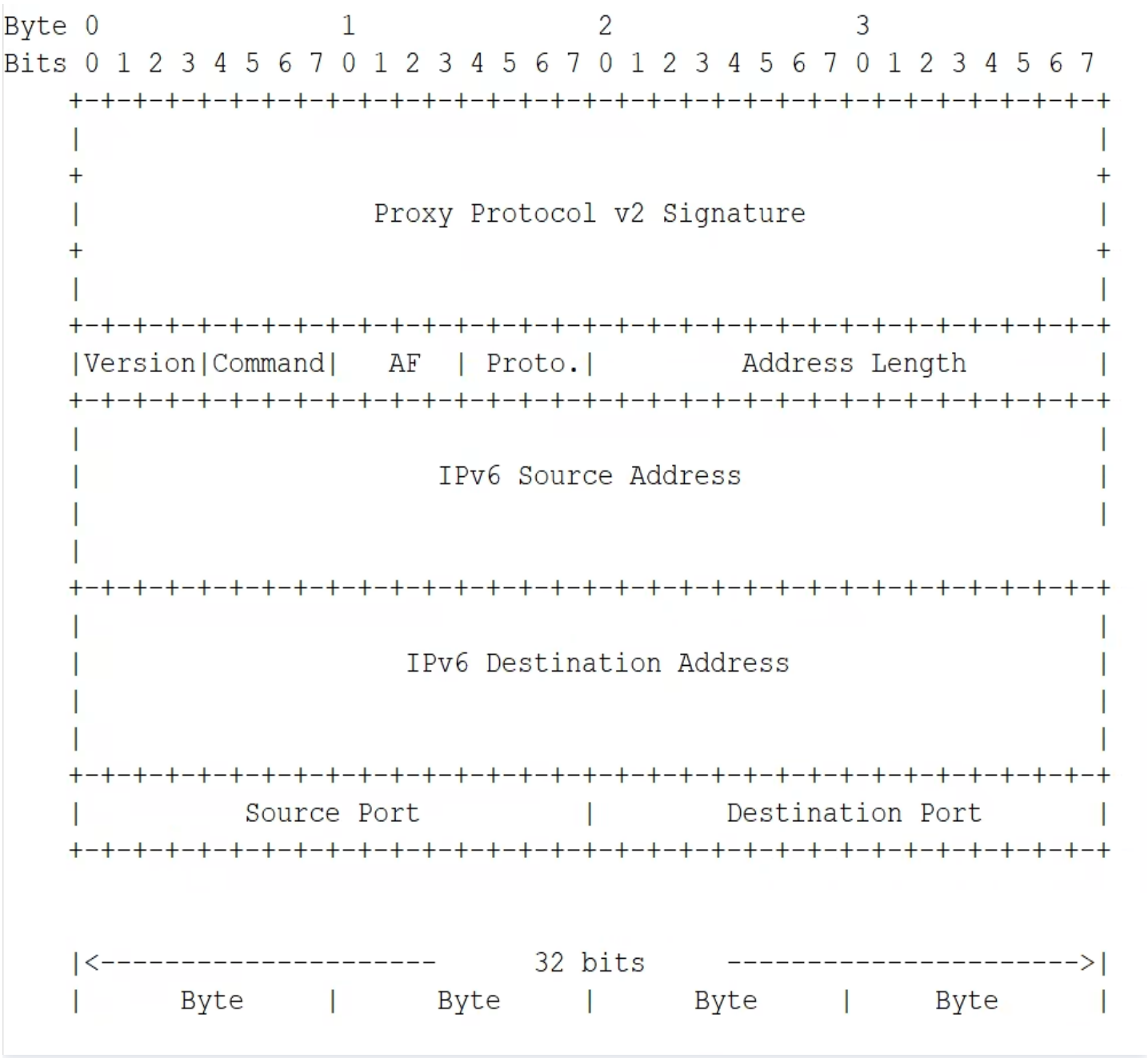
Proxy Protocol V2

Proxy Protocol V2 uses a binary format, supporting TCPv4, TCPv6, UDPv4, and UDPv6 protocols. Its format is as follows:

IPv4 Format



IPv6 Format



Prerequisites

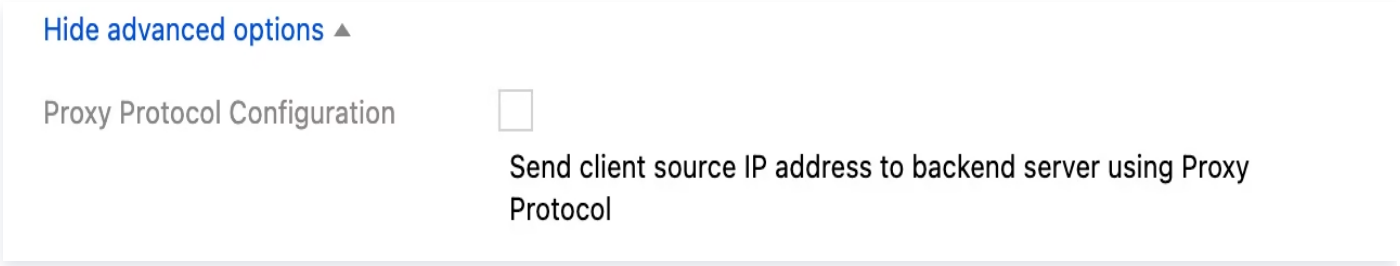
- Before Proxy Protocol is enabled, ensure your real server supports Proxy Protocol v2. Otherwise, the new connections will fail.
- This document takes the TCP listener of IPv4 CLB as an example for introduction.

Directions

Step 1: Enabling Proxy Protocol Configuration for the TCP Listener

1. Log in to the CLB console, and click **Instance Management** in the left sidebar.
2. On the CLB Instance List page, select a region in the upper left corner, and click **Configure Listener** in the operation column on the right side of the instance list.

3. Under the TCP/UDP/TCP SSL/QUIC listener, click the details of the target listener to check if ProxyProtocol is **enabled**. If it is not enabled, edit the listener, check the ProxyProtocol configuration in the advanced settings, and then submit to save the configuration.



Step 2: Enabling Proxy Protocol for the Real Server

Take CentOS 7.9 and Nginx 1.20.1 configuration as examples for introduction. The steps need to be subject to the actual usage environment.

1. Log in to the real server and execute the nginx -t command to view the path of the configuration file. The default path is /etc/nginx/nginx.conf, which is subject to the actual environment.
2. Modify the Proxy Protocol content in the configuration file and save the changes. For modification points, see the following descriptions.

```
http {
# Ensure $proxy_protocol_addr is set. This variable is used to record the real
client IP address.
log_format  main  '$proxy_protocol_addr - $remote_addr- $remote_user
[$time_local] "$request" '
                  '$status $body_bytes_sent "$http_referer" '
                  '"$http_user_agent" "$http_x_forwarded_for"';
 # Taking the listening port 80 as an example, add the proxy_protocol field.
 server {    listen 80   proxy_protocol;
 #...
   }
 }
```

3. Execute the sudo nginx -s reload command to reload the Nginx configuration file.

Step 3: Verifying That the Real Server Can Obtain the Real Client IP Address

When Nginx serves as the real server, you can determine whether the real client IP address is obtained successfully by checking Nginx logs.
The default storage path of the Nginx log file is /var/log/nginx/access.log.
In each row of logs, the IP address corresponding to the $proxy_protocol_addr variable is the real client IP address.

Obtaining Real Client IPs in IPv4 CLB Scenarios

Last updated: 2023-09-05 18:33:54

Notes on Getting Real Client IP Addresses by CLB

All layer-4 (TCP/UDP/TCP SSL) and layer-7 (HTTP/HTTPS) CLB services support getting a real client IP address directly on a backend CVM instance with no additional configuration required.

- For layer-4 CLB, the source IP address obtained on the backend CVM instance is the client IP address.
- For layer-7 CLB, when using short connections between the CLB and backend services, the source IP address obtained on the backend CVM instance is the client IP address. When using persistent connections between the CLB and backend services, the CLB no longer passes through the source IP address, and you can directly obtain the client IP address using the X-Forwarded-For or remote_addr fields. For access logs of layer-7 CLB, please refer to Configuring Access Logs to CLS.

> **Note:**
> For layer-4 CLB, the client IP address can be directly obtained with no additional configuration required on the backend CVM instance.
> For other layer-7 load balancing services with SNAT enabled, you need to configure the backend CVM instance and then use `X-Forwarded-For` to get the real client IP address.

Below are commonly used application server configuration schemes.

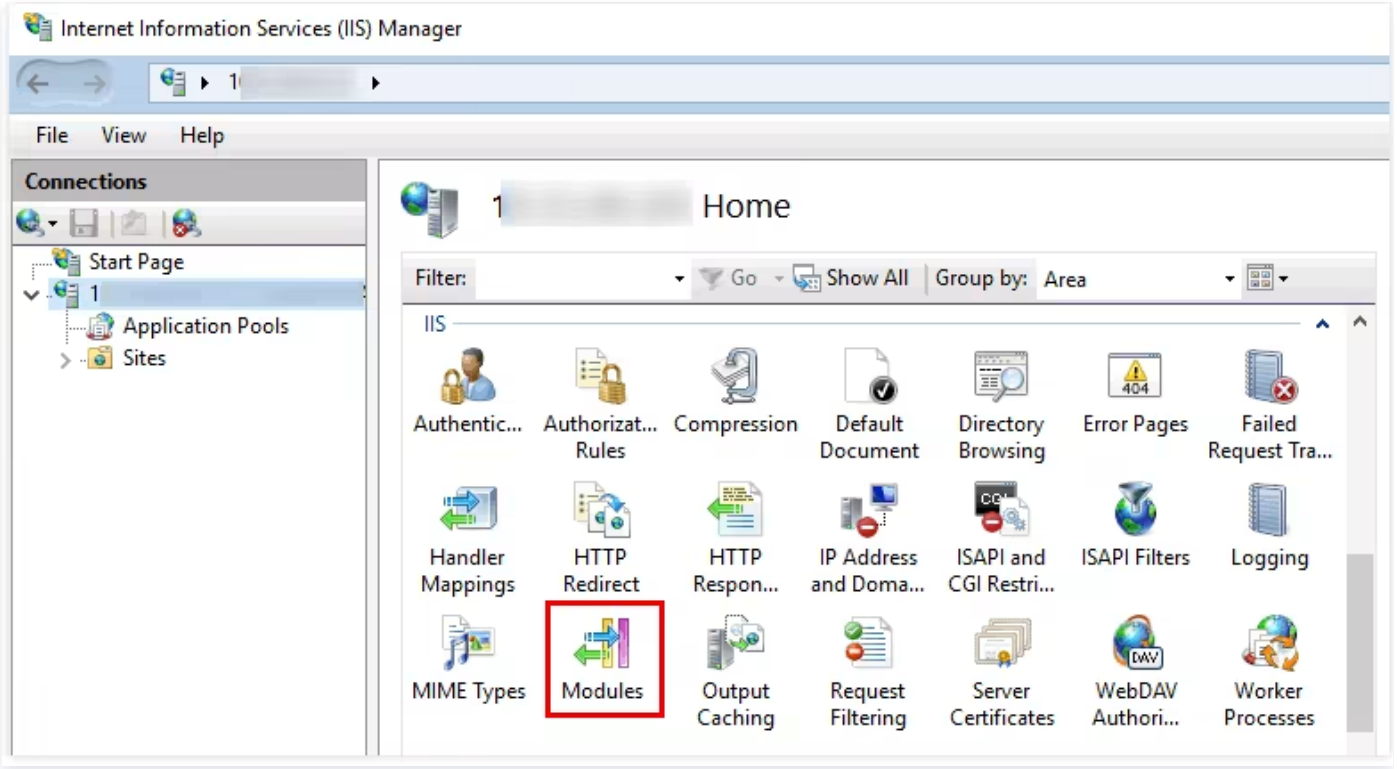
IIS 6 Configuration Scheme

1. Download and install the F5XForwardedFor plugin module. Depending on your server's operating system version, copy the `F5XForwardedFor.dll` file from either the `x86\Release` or `x64\Release` directory to a specific directory, such as `C:\ISAPIFilters`. Ensure that the IIS process has read access to this directory.
2. Open the Control Panel, select **Programs** > **Programs and Features** > **Turn Windows features on or off** > check **Internet Information Services**, and ensure that the related items under **World Wide Web Services** > **Application Development Features** > **ISAPI** are checked, then click OK.
3. Open **IIS Manager** > **ISAPI Filters**, right-click and select **Add** to display the Add/Edit Filter Properties window.
4. In the "Filter Name" field of the add window, enter "F5XForwardedFor", and in the "Executable File" field, enter the full path of `F5XForwardedFor.dll`, then click **Confirm**.
5. Restart the IIS server for the configuration to take effect.

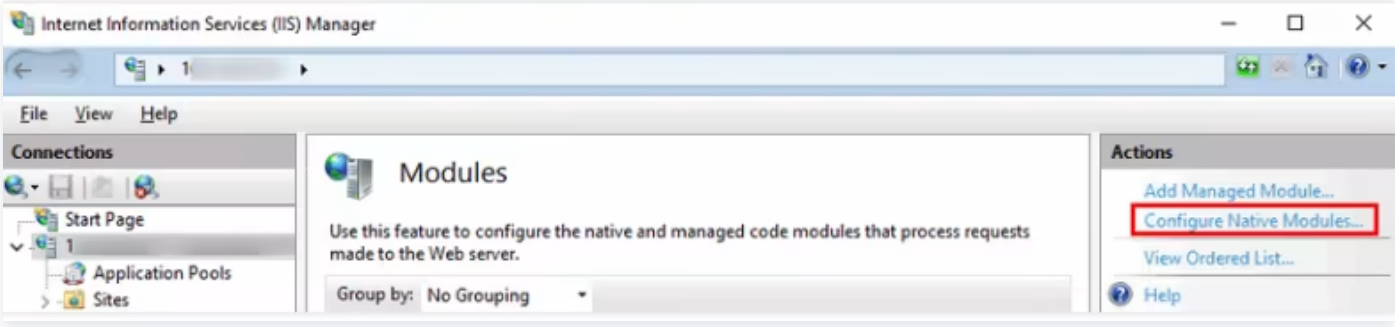
IIS 7 Configuration Scheme

1. Download and install the F5XForwardedFor plugin module. Depending on your server's operating system version, copy the `F5XFFHttpModule.dll` and `F5XFFHttpModule.ini` files from either the `x86\Release` or `x64\Release` directory to a specific folder, such as `C:\x_forwarded_for`. Ensure that the IIS process has read access to this directory.

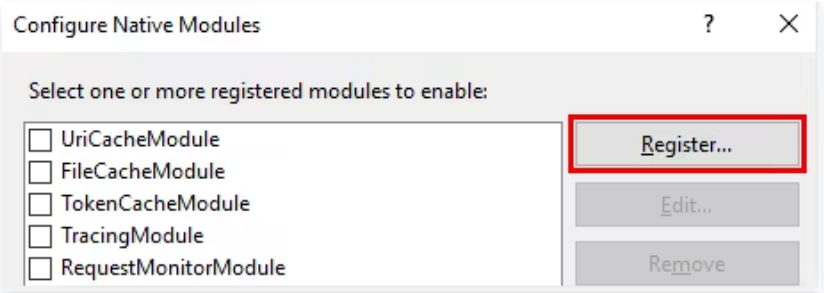
2. Open **IIS Manager**, select your IIS server, and double-click **Modules**.



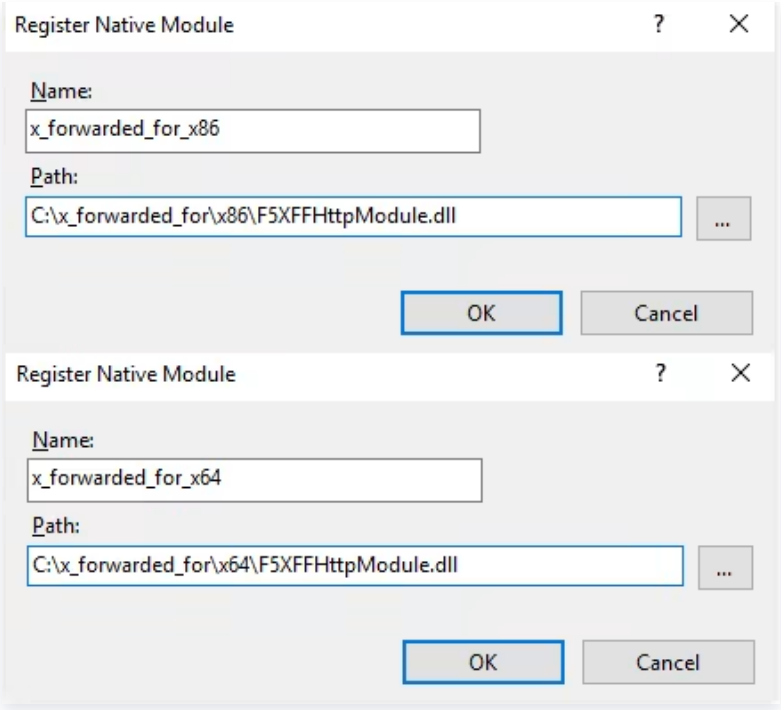
3. Click **Configure Native Modules**.



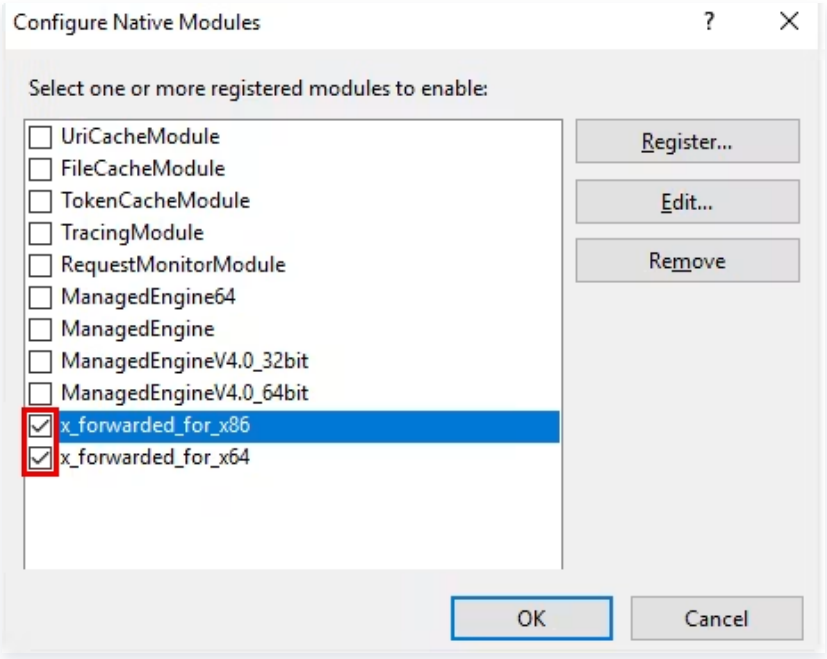
4. In the pop-up window, click **Register**.



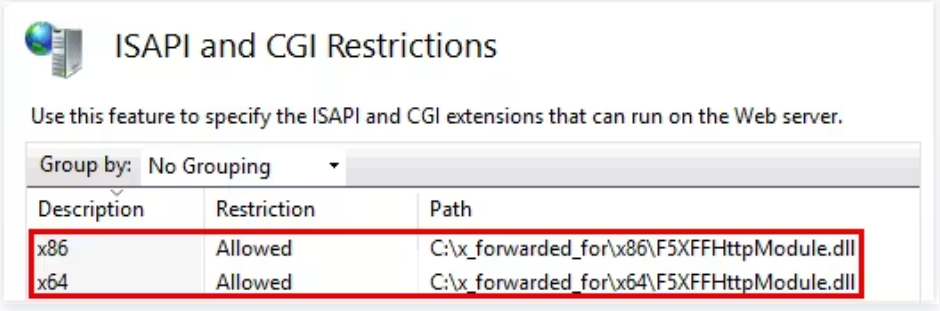
5. Add the downloaded DLL files as shown below:



6. After adding the files, check them and click **OK**.



7. Add the above two DLL files in "ISAPI and CGI Restrictions" and set the restrictions to "Allow".



8. Restart the IIS server for the configuration to take effect.

Apache Configuration Scheme

1. Install the third-party Apache module "mod_rpaf".

```
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. Modify the Apache configuration file `/etc/httpd/conf/httpd.conf` by adding the following to the end of the file:

```
LoadModule rpaf_module modules/mod_rpaf-2.0.so
RPAFenable On
RPAFsethostname On
RPAFproxy_ips IP address (The IP address is not the public IP address provided
by CLB. For the specific IP address, query the Apache logs. Generally, there are
two IP addresses and you need to enter both of them.)
RPAFheader X-Forwarded-For
```

3. After adding the above content, restart Apache.

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

Nginx Configuration Scheme

1. When Nginx is used as the server, the real client IP address can be obtained using the http_realip_module. By default, this module is not installed in Nginx, and you need to recompile Nginx to add `--with-http_realip_module`.

```
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. Modify the `nginx.conf` file.

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

Modify the configuration fields and information as follows:

Note
Here, you need to change `xx.xx.xx.xx` to the IP address or IP range of the upstream proxy server.

```
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;

# Modify the configuration fields and information as follows:
set_real_ip_from xx.xx.xx.xx;
real_ip_header X-Forwarded-For;
real_ip_recursive on;
```

3. Restart Nginx.

```
service nginx restart
```

4. View Nginx access logs to get the real client IP.

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

Obtaining Real Client IPs via TOA in Hybrid Cloud Deployment

Last updated: 2023-09-05 18:38:18

This document describes how the layer-4 (TCP) CLB service obtains the real client IP address via TOA in hybrid cloud deployment and NAT64 CLB scenarios.

- Enabling TOA in the Console
- Loading the TOA Module
- Adapting Backend Services
- (Optional) Monitoring TOA Module Status

Note:
- Only NAT64 CLB instances in Beijing, Shanghai, and Guangzhou regions can obtain the real client IP address via TOA.
- Layer-4 (TCP) CLB can obtain the real client source IP via TOA, while Layer-4 (UDP) and layer-7 (HTTP/HTTPS) CLB cannot.
- This feature is currently in beta testing. To try it out, please submit a ticket.

Scenarios

Hybrid cloud deployment

In hybrid cloud deployment scenarios, IP addresses of the IDC and VPC may overlap, so an SNAT IP address is required. For the server, the real client IP address is invisible and needs to be obtained via TOA.

NAT64 CLB

In NAT64 CLB scenarios, the real IPv6 client IP is translated to an IPv4 public IP, which is invisible to the real server.
In this case, the real client IP address can be obtained via TOA, that is, the TCP packets transmit the real client IP address to the server after you insert the real client IP address into the field `TCP option`, and the client can obtain the real client IP address by calling the API of the TOA kernel module.

Description

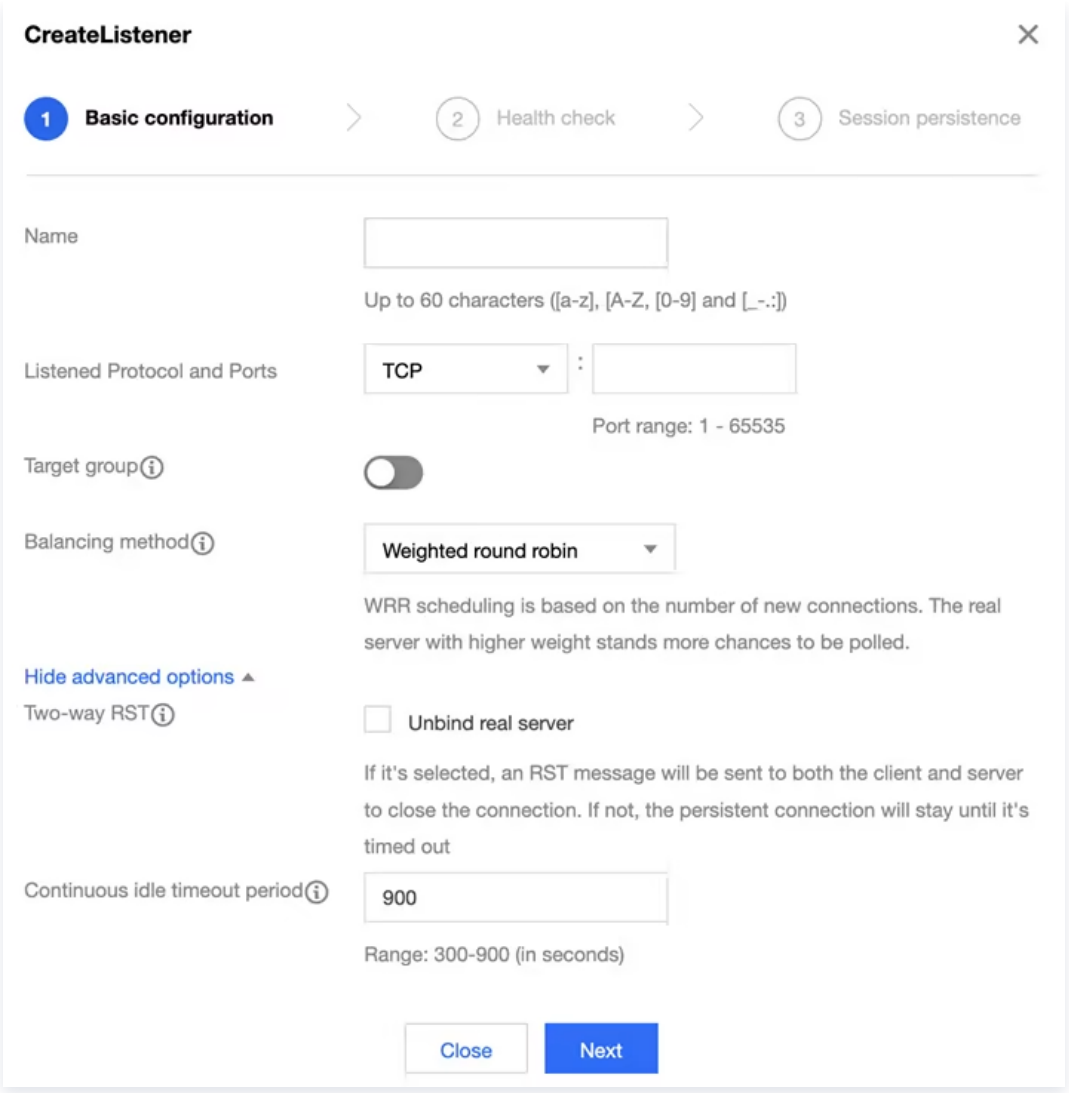
Resource Limits
- The kernel version of the TOA compilation environment must be consistent with that of the service environment.
- In a container environment, the TOA kernel module needs to be loaded in the host.
- To load the TOA kernel module, the root permission is required.

Compatibility Limits

- UDP listeners cannot obtain the real client IP address via TOA.
- If TOA-related operations are already performed on devices between the client and the real server, the real server may fail to obtain the real client IP address.
- After TOA is inserted, it takes effect only on new connections.
- TOA may degrade the server performance as it needs to perform additional processing such as extracting the address from the field `TCP option`.
- Compatibility issues may come up when Tencent Cloud TOA is used with other user-defined TOA modules.
- The TOA module embedded in Tencent's proprietary TencentOS supports obtaining the real client IP address in hybrid cloud deployment scenarios. Therefore, if the server system is TencentOS and deployed in a hybrid cloud, you can try executing the `modprobe toa` command to load and use it. It is important to note that TencentOS and other Linux distributions have separate TOA modules, and they are not compatible with each other.

Enabling TOA in the Console

1. A NAT64 CLB instance has been created. For more information, see Creating IPv6 NAT64 Cloud Load Balance Instances.
2. Log in to the Cloud Load Balance Console and create a TCP listener. For more information, see Configuring TCP Listener.
3. Enable TOA in the "Create listener " window.



Loading TOA

1. Download and decompress the TOA package corresponding to the version of Linux OS on Tencent Cloud.

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. After the decompression is completed, execute the cd command to enter the recently decompressed folder, and run the following command to load the module:

```
insmod toa.ko
```

3. Run the following command to verify if the TOA module has been successfully loaded. If the message "toa load success" is displayed, it indicates a successful load.

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

4. After successfully loading, include the `toa.ko` file in the startup script (the ko file needs to be reloaded upon server restart).
5. (Optional) If TOA is no longer needed, run the following command to uninstall it.

```
rmmod toa
```

6. (Optional) Run the following command to check whether the TOA module has been successfully uninstalled. If the message "TOA unloaded" appears, the uninstallation is successful.

```
dmesg -T
```

If you cannot find an installation package above for your OS version, you can download the general source package for Linux OS and compile it to obtain the `toa.ko` file. This general version supports most Linux distributions (e.g., CentOS 7, CentOS 8, Ubuntu 16.04, and Ubuntu 18.04).

Note
Due to the numerous Linux kernel versions and the vast market of Linux distribution operating systems with various editions, it is advisable to compile the TOA source package on your system to address potential compatibility issues with kernel modules.

1. Download the source package.

Note
If your OS is Linux, download the Linux TOA source package; if it is TencentOS, download the TLinux TOA source package.

- Linux

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

- TLinux

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

2. To compile the Linux environment for TOA, you need to install the GCC compiler, Make tool and kernel development package first.

Installation Steps for CentOS Environment

```
yum install gcc
yum install make
//Install the kernel development package. The version of the package
header file and library must be consistent with the kernel version.
yum install kernel-devel-uname -r
yum install devtoolset-8
```

Installation Steps for Ubuntu and Debian Environments

```
apt-get install gcc
apt-get install make
//Install the kernel development package. The version of the package
header file and library must be consistent with the kernel version.
apt-get install linux-headers-uname -r
apt-get install devtoolset-8
```

Installation Procedure for SUSE Environment

```
zypper install gcc
zypper install make
//Install the kernel development package. The version of the package
header file and library must be consistent with the kernel version.
zypper install kernel-default-devel
zypper install devtoolset-8
```

3. Modify the PATH environment variable to `PATH=/opt/rh/devtoolset-8/root/bin:$PATH`. Before compiling, please ensure that the kernel gcc compilation version is consistent with the gcc version. You can use the `dmesg | grep 'Linux version'` command to view the kernel gcc compilation version information.
4. Compile the source code to generate the toa.ko file. If there are no `warnings` or `errors` during the compilation process, it indicates a successful compilation. Using the corresponding source package for the Linux system as an example:

```
tar zxvf tgw_toa_linux_ver.tar.gz
cd tgw_toa_linux_ver//Enter the decompressed directory tgw_toa
make
```

5. Upon successful compilation of toa.ko, execute the operation of loading the TOA module as described in Step 2 above.

Adapting the Real Sever

Hybrid Cloud Deployment Scenarios

When adapting backend services in a hybrid cloud deployment scenario, there is no need for code modification. Simply call the standard interface in Linux network programming to obtain the real source IP of the visiting user. For example, the following C code sample.

```
struct sockaddr v4addr;
len = sizeof(struct sockaddr);
//get_peer_name is the standard Linux network programming API.
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 Scenarios

In the NAT64 CLB scenario, the TOA source address pass-through function is used. After the backend server inserts the toa.ko kernel module, the source code of the application program must be modified to adapt to the function of obtaining the real source IP.

1. Define a data structure to store the IP address.

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

2. Define TOA variables and make calls to get the real source IPv6 address.

```
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. Obtain the source IP address.

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

A complete configuration sample is as follows:

```
//Define TOA variables. Set TOA_BASE_CTL to 4096.
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,
};
//Define a data structure to store the IP address.
struct toa_nat64_peer {
    struct in6_addr saddr;
    uint16_t sport;
};
//Declare the variable that is used to store the address.
struct toa_nat64_peer client_addr;
.……
//Get the file descriptor of the client, where listenfd is the listening
file descriptor of the server.
client_fd = accept(listenfd, (struct sockaddr*)&caddr, &length);
//Make calls to get the real client IP address of the user in the NAT64
scenario.
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));
    //Obtain the source IP and source port information
    printf("real client [%s]:%d\n", from, ntohs(client_addr.sport));
}
```

Hybrid Cloud Deployment and NAT64 CLB Scenarios

In hybrid cloud deployment and NAT64 CLB mixed-use scenarios, the TOA source address pass-through function is utilized. After the backend server inserts the toa.ko kernel module, it is necessary to modify the source code of the application to adapt to the function of obtaining the real source IP.
A complete configuration sample is as follows:

```
//Define TOA variables. Set TOA_BASE_CTL to 4096.
```

```
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,
};

//Define a data structure to store the IP address.
struct toa_nat64_peer {
    struct in6_addr saddr;
    uint16_t sport;
};

//Declare the variable that is used to store the address.
struct toa_nat64_peer client_addr_nat64;
.......
//Get the file descriptor of the client, where listenfd is the listening
file descriptor of the server.
//Make calls to get the real client IP in the NAT64 scenario.
char from[40];
int len = sizeof(struct toa_nat64_peer);
int ret;
ret = getsockopt(client_fd, IPPROTO_IP, TOA_SO_GET_LOOKUP,
&client_addr_nat64, &len);
if (ret == 0) {
    inet_ntop(AF_INET6, &(client_addr_nat64.saddr), from, sizeof(from));
    //Obtain the source IP and source port information.
    printf("real client v6 [%s]:%d\n", from,
ntohs(client_addr_nat64.sport));
} else if (ret != 0) {
    struct sockaddr v4addr;
    len = sizeof(struct sockaddr);
    //Retrieve the source IP and source port information, note that the
obtained source address is relevant to:
    //In the hybrid cloud deployment scenario, the connection's true source
address is the SNAT IP;
    //In non-hybrid cloud deployment scenarios, the client address without
SNAT and NAT64 is also the genuine source IP address.
    //Hence, the purpose of this function is to obtain the genuine client
address and port.
    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));
    }
}
```

(Optional) Monitoring TOA Status

To ensure the stability of the TOA kernel module, it also provides monitoring capabilities. After inserting the toa.ko kernel module, you can monitor the working status of the TOA module on the host of the container using either of the following methods.

Method 1: Checking the IPv6 address stored in TOA

Run the following command to check the IPv6 address stored in TOA.

> ⚠ **Note:**
> Note that executing this command may lower the performance. Avoid frequent usage of this command.

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

Method 2: Viewing TOA-related Counting Status

Run the following command to check the TOA metrics.

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

The monitoring metrics are described as follows:

Description	Note
syn_recv_sock_toa	Receives connections with TOA information.
syn_recv_sock_no_toa	Receives connections without TOA information.
getname_toa_ok	This count increases when you call `getsockopt` and get the source IP successfully or when you call `accept` to receive client requests.
getname_toa_mismatch	When invoking getsockopt to obtain the source IP, this count increases when the types do not match. For instance, if an IPv4 source IP is stored in a client connection instead of an IPv6 address, this count will increment.
getname_toa_empty	This count increases when the `getsockopt` function is called in a client file descriptor that does not contain TOA.
ip6_address_alloc	Allocates space to store the information when TOA gets the source IP and source port saved in the TCP data packet.
ip6_address_free	When the connection is released, TOA will release the memory previously used to save the source IP and source port. If all connections

	are closed, the total count of `ip6_address_alloc` for each CPU should be equal to the count of this metric.

FAQ

Why is there a need to modify the server program even after the TOA module has been inserted in the NAT64 CLB scenario?

This is due to the change in IP type. In hybrid cloud deployment scenarios, IPv4 Fullnat conversion is performed. In this case, the real client source IP is still converted from one IPv4 IP to another IPv4 IP, so the IP type remains unchanged. However, in NAT64 CLB scenarios, the real client source IP is converted from IPv6 to IPv4, causing a change in IP type. Therefore, to understand this IPv6 IP, the server-side program must be modified to comprehend the meaning of this IPv6 address.

How can I determine if my system is based on a Linux distribution or Tencent's TLinux kernel?

- Execute the following command to ascertain the kernel version. If the version result includes `tlinux`, it signifies a TLinux system. Conversely, it indicates a Linux distribution.

```
uname -a
```

- You can also check the version by running the following command. If `tlinux` or `tl2` is returned, you are using TLinux OS.

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

How do I perform preliminary checks when I failed to obtain the source IP address?

1. Run the following command to check whether TOA has been loaded.

```
lsmod | grep toa
```

2. Check whether the server program has made correct calls to obtain the source IP address. You can refer to Adapting Backend Services above.
3. Capture TCP packets on the server and check whether the packets contain the source IP information.
   - If `unknown-200` is displayed in the tcp option output, it indicates that the actual source IP has been inserted into the tcp option after passing through SNAT.
   - If `unknown-253` is displayed, the real client IPv6 address is inserted in the NAT64 scenario.

```
[root@VM-0-133-centos ]# tcpdump -i any "ip[40:1]==200" -c 100
dropped privs to tcpdump
tcpdump: verbose output suppressed, use -v or -vv for full protocol decode
listening on any, link-type LINUX_SLL (Linux cooked v1), capture size 262144 bytes
18:04:24.864649 IP 192.168.0.177.23638 > VM-0-133-centos.webcache: Flags [.], ack 3309146461, win 229, options [unknown-200 0xa
2662ac13bca,nop,nop,TS val 2243901958 ecr 3395797654], length 0
18:04:24.864679 IP 192.168.0.177.23638 > VM-0-133-centos.webcache: Flags [P.], seq 0:154, ack 1, win 229, options [unknown-200
0xa2662ac13bca,nop,nop,TS val 2243901958 ecr 3395797654], length 154: HTTP: GET /data/1K HTTP/1.1
```

4. In the previous step, if it is confirmed that the packet carrying the TOA address has entered the server, compile the DEBUG version of toa.ko. This allows for further positioning through the kernel log. In the downloaded TOA source code directory, add the DEBUG compile option in the Makefile.

```
endif
PWD       := $(shell pwd)

ccflags-y += -DTOA_NAT64_ENABLE -DTOA_DEBUG_ENABLE

ifeq ($(DEBUG), 1)
ccflags-y += -g -O0
endif
```

5. Run the following command to compile again.

```
make clean
make
```

6. Run the following commands to uninstall the original `toa.ko` and install the latest one.

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

7. Run the following command to observe the kernel log.

```
dmesg -Tw
```

If you see the following message, TOA is working normally. You can further check whether the server program has made calls to obtain the real client IP address, or whether the API is used incorrectly.

```
[Wed Dec 29 18:07:11 2021] [DEBUG] TOA: inet_getname_toa called, sk->sk_user_data is 000000003088927f
[Wed Dec 29 18:07:11 2021] [DEBUG] TOA: inet_getname_toa: set new sockaddr, ip 192.168.0.177 -> 42.193.59.202, port 49682 -> 41
618
```

8. If you failed to find out the problem with the preceding steps, please submit a ticket.

Implementing HA Across Multiple AZs

Last updated: 2025-11-06 18:21:47

Tencent Cloud Load Balancer (CLB) ensures high availability through a multi-dimensional approach, including system architecture and product configuration. Based on your business requirements and scenarios, you can choose from various disaster recovery options, such as cross-region disaster recovery and cross-availability zone disaster recovery within the same region.

CLB Cluster High Availability

Tencent Cloud Load Balancer (CLB) instances employ cluster deployment, supporting session synchronization, eliminating single points of failure, enhancing system redundancy, and ensuring service stability. All CLB instances possess cluster high availability.

- Layer 4 load balancing in Tencent Cloud Load Balancer is primarily implemented through Tencent's proprietary Unified Access Gateway (Tencent Gateway, TGW). TGW boasts high reliability, strong scalability, high performance, and robust attack resistance. It supports Data Plane Development Kit (DPDK) for high-performance forwarding, with a single cluster capable of handling billions of concurrent connections and tens of millions of PPS. Numerous Tencent internal services, including Tencent Games, Tencent Video, WeChat, and QQ, utilize TGW for access.
- Layer 7 load balancing in Tencent Cloud Load Balancer is primarily implemented through Secure Tencent Gateway (STGW). STGW is a large-scale, concurrent layer 7 load balancing service developed by Tencent based on Nginx, which carries a significant amount of layer 7 traffic for Tencent's internal services.

Single CLB Instance High Availability

Non-domain Public Network CLB

Non-domain-based public network CLB instances provide services in the form of VIPs, with an SLA of 99.95%. There are two deployment schemes for the VIP's associated cluster:

Deployment mode	Cluster Disaster Recovery	Cross-Availability Zone Disaster Recovery
Single-AZ	This feature is supported.	Unavailable
Multi-AZ Deployment	This feature is supported.	Yes, the primary and secondary availability zone mode is supported. When the primary availability zone fails, Cloud Load Balancer can automatically switch to the secondary availability zone and restore services within a very short time.

Currently, the public network multi-AZ disaster recovery is in beta testing and is free of charge. If you are interested, please submit a beta test application.

Domain-based Public Network CLB

"Domain name-based public network CLB" builds upon the foundation of "non-domain name-based public network CLB" by adding a layer of DNS service. The SLA increases from 99.95% to 99.99%, and it can automatically replace faulty VIPs, enhancing availability. For more information, please refer to the Announcement of Domain Name-Based Public Network Cloud Load Balancer Launch .

Private CLB

Private network CLB instances adopt a proximity-based access architecture, with a single CLB instance deployed across one or multiple availability zones. When a client accesses the CLB, the traffic is automatically directed to the cluster with the lowest latency in the available zones before being forwarded to the real servers.
Private network CLB does not currently support cross-availability zone disaster recovery switching. If a CLB cluster in a specific availability zone becomes unavailable, it will affect access from sources within that availability zone. If the access source is from another availability zone and has nearby access in the corresponding zone, the traffic will not be affected. However, if the access source is from another availability zone, the corresponding zone does not have nearby access, and the default points to the failed CLB, the traffic will be impacted.

High Availability with Multiple CLB Instances

If you have extremely high availability requirements, the inherent availability mechanisms of CLB instances may not be sufficient to meet your needs in scenarios such as network attacks, cross-region switching, or configuration errors. You can create multiple CLB instances and use Tencent Cloud DNS to manage and distribute traffic.
Comparison between Multi-CLB Instance High Availability and Domain-based Public CLB:

Comparison Item	Domain-based Public Network CLB	High Availability with Multiple CLB Instances
SLA	99.99%	99.95%
Input Failover	The service offers link detection and disaster recovery switching capabilities, eliminating concerns about single IP entry point interruptions. In case of a single IP failure, the system can automatically switch the faulty IP, minimizing the impact on your business operations.	Depending on your configured DNS resolution and switching policies, it is essential for your business to promptly detect and switch as needed.
Ops	Only a single instance needs to be configured.	Multiple CLB instances and corresponding DNS resolution policies need to be configured.
Cost effectiveness	The cost is relatively low, as only CLB-related fees are charged.	Deploying multiple CLB instances and components such as DNS resolution results in higher costs.

Regions	CLB instances are deployed within a single region as part of a cluster.	You can choose CLB instances across multiple regions.

Best Practices:

- If your business is deployed in a single region, we recommend prioritizing the domain name-based public CLB solution, which automatically switches faulty IPs.
- If your business is deployed across multiple regions and has stringent disaster recovery requirements, it is recommended to opt for a high availability solution with multiple CLB instances.

Backend Service High Availability

Tencent Cloud Load Balancer (CLB) determines the availability of real servers through health checks, preventing frontend businesses from being affected by real server exceptions and improving the overall availability of businesses.
Once health checks are enabled, regardless of the weight of the real servers (including a weight of 0), Tencent Cloud Load Balancer instances will perform health checks. You can view the health check status in the **Health Status** column on the instance list page or on the listener's bound real server details page.
For more information on the health check mechanism, please refer to Health Check Overview.

Load Balancing Algorithm Selection and Weight Configuration Examples

Last updated: 2023-09-05 18:44:36

Comparative analysis of CLB algorithms

Weighted round-robin scheduling

Weighted round-robin scheduling is a method that sequentially distributes requests to different servers in a round-robin manner. This algorithm addresses the issue of varying server performance by assigning appropriate weights to represent each server's processing capabilities. Requests are allocated to servers based on their weights and the round-robin method. Scheduling is performed based on the number of new connections, with higher-weighted servers receiving connections first. The higher the weight, the more likely a server is to be polled, and servers with the same weight process the same number of connections.

- **Advantages**: Simple and practical, no need to track the status of all current connections, making it a stateless scheduling method.
- **Disadvantages**: Relatively simple, and in cases where the request service time varies greatly or each request consumes different amounts of time, it can easily lead to imbalanced server workloads.
- **Applicable scenarios**: Performs best when the backend time consumed by each request is roughly equal. Commonly used for short-lived connections, such as HTTP.
- **User Recommendation**: Weighted round-robin scheduling is recommended when it is known that each request occupies a similar amount of backend time, and the backend servers process the same or similar types of requests. It is also recommended when the difference in request times is small, as this implementation consumes less resources, does not require traversal, and offers higher efficiency.

Weighted least-connection scheduling

- **Principle**
  In real-world scenarios, the time each client request spends on a server may vary significantly. As the workload increases, using simple round-robin or random balancing algorithms may result in a significant difference in the number of connection processes on each server, which does not achieve true cloud load balancing. The least-connection scheduling is a dynamic algorithm that evaluates server workloads based on the number of active connections. In contrast to round-robin scheduling, the scheduler needs to track the number of established connections for each server. When a request is scheduled to a server, its connection count increases by one; when the connection is terminated or times out, the count decreases by one. The weighted least-connection scheduling algorithm builds upon the least-connection scheduling by assigning different weights to servers based on their processing capabilities, allowing them to accept service requests proportional to their assigned weights. This is an improvement over the basic least-connection scheduling algorithm.

  1.1 Suppose the weight of a real server is Wi (i = 1... n) and its current number of connections is Ci (i = 1... n). The Ci/Wi value of each server is calculated in sequence. The real server with the smallest Ci/Wi value will be the next server to receive a new request.

1.2 For real servers with the same Ci/Wi value, they will be scheduled based on weighted round robin scheduling.

- **Advantages**
  This balancing algorithm is suitable for long-duration request services, such as FTP and other applications.
- **Disadvantages**
  Due to interface limitations, the minimum number of connections and session persistence features cannot be enabled simultaneously.
- **Applicable Scenarios**
  Situations where the backend time occupied by each request varies significantly. Commonly used for long-lived connections.
- **User Recommendation**
  If users need to handle different requests with varying backend processing times, such as a significant difference in the order of magnitude between 3ms and 3s, it is recommended to use the weighted least-connection scheduling algorithm for implementing Cloud Load Balancer.

Source hashing scheduling (ip_hash)

- **Principle**
  Using the source IP address of the request as the hash key, the corresponding server is found from the statically assigned hash table. If the server is available and not overloaded, the request is sent to that server; otherwise, null is returned.
- **Advantages**
  The ip_hash method achieves partial session persistence by remembering the source IP, ensuring that a specific client's requests are consistently mapped to the same real server through the hash table. Therefore, ip_hash can be used for scheduling in scenarios where session persistence is not supported.
- **User Recommendation**
  Hash the source address of the request and combine it with the backend server's weight to distribute the request to a matching server. This ensures that requests from the same client IP are always dispatched to a specific server. This method is suitable for Cloud Load Balance with TCP protocol and no cookie functionality.

Load Balancing Algorithm Selection and Weight Configuration Examples

In the upcoming new features of Cloud Load Balancer, **Layer-7 forwarding will support the least connections balancing method**. To enable users to maintain stable business operations with RS clusters in different scenarios, we provide several examples of CLB selection and weight configuration for user reference.

- Scenario 1
  Suppose there are 3 real servers (RS) with identical configurations (CPU / memory). Due to their consistent performance, users can set the weight of each RS to 10. Assume that each RS has established 100 TCP connections with the client side, and a new RS is added. In this scenario, it is recommended to use the least-connection balancing method, as this configuration can quickly increase the load on the fourth RS and reduce the pressure on the other 3 servers.

- Scenario 2
  Suppose a user is new to cloud services and has a recently established website with a low workload. In this case, it is recommended to purchase RS instances with the same configuration, making all RS instances indistinguishable access layer servers. In this scenario, the user can set the weight of all RS instances to 10 and use the weighted round-robin balancing method for traffic distribution.
- Scenario 3
  The user has 5 servers for hosting simple static website access, and the ratio of their compute capabilities is 9:3:3:3:1 (calculated by CPU and memory). In this scenario, the user can set the RS weight ratios to 90, 30, 30, 30, and 10, respectively. Since most static website access involves short connection requests, the weighted round-robin balancing method can be used, allowing the CLB to distribute requests according to the performance ratios of the RS.
- Scenario 4
  A user has 10 real servers (RS) to handle a massive amount of web access requests and does not want to purchase additional RS to increase expenditure. One of the RS often restarts due to high workload. In this scenario, it is recommended that the user sets appropriate weights for each RS based on their performance, assigning a smaller weight to the overloaded RS. Additionally, adopting the least-connection Cloud Load Balance method can help distribute requests to RS with fewer active connections, thus addressing the issue of excessive workload on a particular RS.
- Scenario 5
  A user has 3 real servers (RS) for processing several long-lasting connection requests, and the compute capability ratio of these servers is 3:1:1 (calculated by CPU and memory). In this situation, the highest-performing server handles more requests, and the user does not want to overload it. Instead, they prefer to allocate new requests to idle servers. In this scenario, the least-connection scheduling method can be adopted, and the weight of the busy server can be reduced appropriately. This allows the CLB to allocate requests to RS with fewer active connections, achieving cloud load balancing.
- Scenario 6
  A user wishes to have subsequent client requests allocated to the same server. Using weighted round-robin or weighted least-connections methods cannot guarantee that requests from the same client are assigned to a specific server. To accommodate the needs of a customer's specific application server and ensure that client sessions have "stickiness" or "persistence," we can adopt the ip_hash load balancing method for traffic distribution in this scenario. This method ensures that requests from the same client are always directed to the same real server (except when the number of servers changes or the server becomes unavailable).

Difference Between Resetting Weight to 0 and Unbinding Real Server

- Resetting the weight to 0: TCP listeners keep forwarding client requests in existing connections to real servers. UDP listeners keep forwarding client requests in connections to real servers with the same quintuple information as the listeners. HTTP/HTTPS listeners keep forwarding client requests in existing connections to real servers.
- Unbinding the real server: TCP/UDP listeners stop forwarding client requests in existing connections. HTTP/HTTPS listeners keep forwarding client requests in existing connections.

Documentation

Modifying real server weight

Configuring WAF protection for CLB listening domain names

Last updated: 2023-09-05 18:50:28

Cloud Load Balancer-based Web Application Firewall (WAF) binds domain names with Cloud Load Balancer listeners to inspect and intercept HTTP or HTTPS traffic passing through the listeners. This document explains how to provide web security protection for domain names added to Cloud Load Balancer using the CLB-based WAF.

Preparations

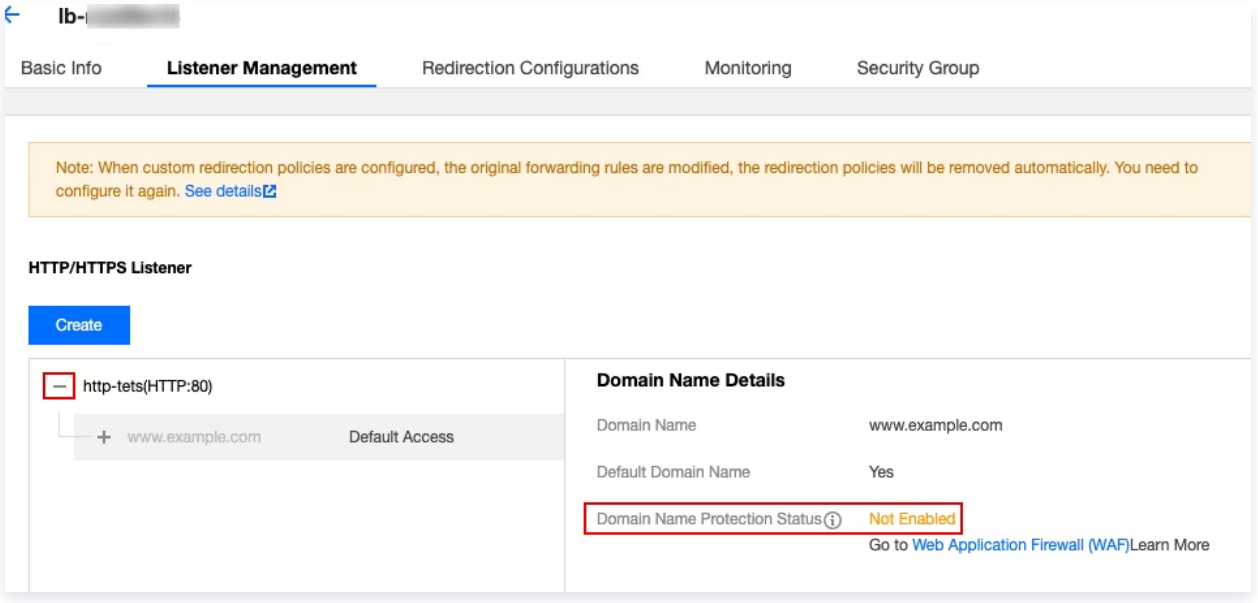
- You have successfully created an HTTP or HTTPS listener, and the domain name can be accessed normally. For more information, please refer to Getting Started with Cloud Load Balancer.
- You have successfully purchased the Cloud Load Balancer-based WAF. For more information on purchasing, please refer to Purchase Guide. If you haven't purchased it yet, you can opt for a 7-day free trial of the Cloud Load Balancer-based WAF.

Instructions

Step 1: Confirm the CLB domain name configuration

This document takes the domain name `www.example.com` as an example.

1. Log in to the Cloud Load Balancer console and click **Instance management** in the left sidebar.
2. On the **Instance Management** page, select the instance region and then click **Configure Listener** on the right of the target instance in the "Operation" column.
3. On the **Listener Management** tab, click + on the left of the target listener in the **HTTP/HTTPS Listener** section to view domain details.



4. Confirm the Cloud Load Balancer domain configuration information as follows: The Cloud Load Balancer instance ID is "lb-f8lm**", the listener name is "http-test", the listener forwarding rule listens

to the domain name `www.example.com`, and the domain protection status is "Not enabled" (all IDs, names, and domain names are subject to actual conditions).

Step 2: Add a domain name in the WAF console and bind it to a CLB instance

To apply protection to a domain name with the CLB WAF service, you need to add a CLB-listening domain name in WAF and bind it with a CLB listener.

1. Log in to the Web Application Firewall console, and select **Asset Center** > **Access Management** > **Domain Access** in the left sidebar.
2. On the Domain Access page, click **Add Domain**, configure the relevant parameters, and click **OK** to confirm.

   Field Descriptions

   - **Associated Instance:** Select the Cloud Load Balancer-based instance and its name.
   - **Domain:** In the domain input box, add the domain `www.example.com` that requires protection.
   - **Proxy status:** Choose whether you have used proxies such as DDoS protection, CDN, or Cloud Acceleration based on your actual situation.

     > **Note**
     > If "Yes" is selected, WAF will retrieve the client's actual IP address from the XFF field as the source address, which may pose a risk of source IP spoofing.

   - **Domestic Region:** Select based on your actual needs.
   - **Select the corresponding Cloud Load Balancer listener for the domain name:** Choose and configure the listener information for the access domain according to actual needs.

     > **Note**
     > The current WAF supports traffic protection for both public and private network types of Cloud Load Balancer instances. You can view and filter by the network type field.

3. Click **Confirm** to return to the domain name access. In the domain name access, you can view the protected domain name `www.example.com`, along with the Cloud Load Balancer ID, name, VIP, and listener information.

Step 3: Verify the result

1. The WAF binds with the domain name and corresponding CLB listener to protect the domain traffic passing through the CLB listener. To verify whether the CLB-based WAF is effective, please ensure that the domain names added under different CLB instances can be accessed normally from your local computer.

   > **Note**
   > Verify whether the domain name added to the Cloud Load Balancer is accessible. For IPv4 domain name requests, please refer to Verifying Cloud Load Balancer Service in the Getting Started with Cloud Load Balancer guide. For IPv6 domain name requests, please refer to Step

> 4: Testing IPv6 Cloud Load Balancer in the Getting Started with IPv6 Cloud Load Balancer guide.

2. Enter the URL `http://www.example.com/?test=alert(123)` in the browser. If a block page is returned, it indicates that the Web Application Firewall is functioning properly.

> **Note**
> In this case, `www.example.com` is the domain name. Here, you need to replace the domain name with the actual domain name you have added.