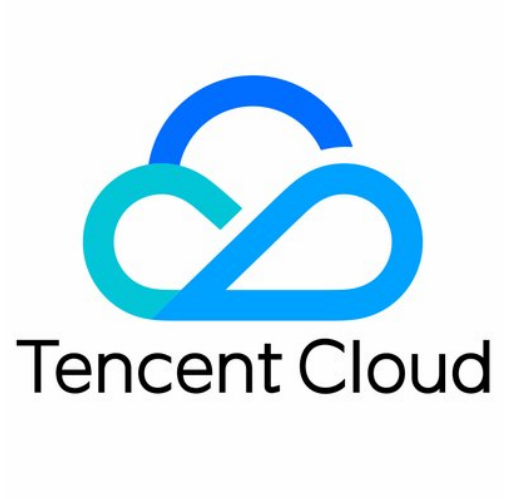
Cloud Load Balancer

Best Practise

Product Documentation



Contents

Best Practise
Enabling Gzip Configuration & Testing

Last updated : 2020-03-09 14:59:36

For instance of **public network CLB or public network CLB with fixed IP**, Gzip compression is enabled for **HTTP and HTTPS** protocols by default. Gzip feature compresses websites, which effectively reduces data volume in network transmission and speeds up access of client browser. Pay attention to the following:

Notes

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

```
gzip on;
```

- **Currently, CLB supports the following file types. You can specify the file type for compression in the gzip_types configuration item**

```
application/atom+xml application/javascript application/json application/rss+xml application/v
nd.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-compon
ent;
```

> You must enable Gzip in sync for the above file types in the business software of CVM instances of CLB.

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

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

Example of enabling Gzip 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: whether to enable or disable 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 Content-Length in the HTTP header. Default value is 1k.
  Syntax: `gzip_min_length length`
  Scopes: http, server, location

- gzip_buffers: specifies the unit of buffer for storing the data stream of the Gzip compression result. 16k means 16k is used as the unit, and memory 4 times the original data size (in 16k) will be applied for.
  Syntax: `gzip_buffers number size`
  Scopes: http, server, location

- gzip_http_version: specifies the lowest HTTP version that can use Gzip. Configure HTTP/1.0 means the lowest HTTP version that needs Gzip is 1.0, so Gzip can be compatible with HTTP/1.1 or higher. You do not need to change this since Tencent Cloud supports HTTP/1.1 across the entire network.
  Syntax: `gzip_http_version 1.0 | 1.1;`
  Scopes: http, server, location

- gzip_comp_level: specifies the Gzip compression ratio with value range: 1–9. 1 is the smallest compression ratio with the fastest processing speed, while 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: indicates the Multipurpose Internet Mail Extensions (MIME) types for compression, and "text/html" type will be compressed by default. In addition, Gzip for Nginx does not compress static resource files such as JavaScript and images by default. You can configure gzip_types to specify MIME types to be compressed. Types that are not specified will not be compressed. **For example, to compress data in JSON format, you need to add application/json to this sentence**
  The supported types are below:

  ```
  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 curl command to test whether Gzip has been successfully enabled:

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

HTTPS Forwarding Configurations

Last updated : 2020-05-21 18:19:37

1. CLB Capability Description

By deeply optimizing the protocol stack and server, Tencent Cloud CLB achieves great improvement in HTTPS performance. Meanwhile, Tencent Cloud substantially reduces certificate costs through collaboration with international certificate authorities. CLB can bring significant 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 greatly simplify the certificate application process and reduce application costs.
5. Anti-CC and WAF features are provided to effectively eliminate application-layer attacks, such as slow HTTP attacks, high-frequency targeted attacks, SQL injection, and website trojans.

2. HTTP and HTTPS Header Identifiers

CLB acts as a proxy for HTTPS. Both HTTP and HTTPS requests become HTTP requests when forwarded to a backend CVM instance by CLB. In this case, you cannot distinguish whether a frontend request is in HTTP or HTTPS.

CLB implants `X-Client-Proto` into the header when it forwards the request to the real server:

X-Client-Proto: http (HTTP request on the frontend)
X-Client-Proto: https (HTTPS request on the frontend)

3. Getting Started

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.

In this case, the request for accessing `www.example.com` entered by an end user will be forwarded as below:

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. Then, the browser sends the `https://example.com` request to the corresponding HTTPS site again. The request accesses port 443 of the CLB listener through VIP and then is forwarded to port 80 of the real server.

At this point, the request forwarding process is completed.

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 redirection method:

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

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

Obtain Acutual IP for Layer 7 Load Balancing

Last updated : 2020-05-26 13:54:58

Notes on Getting Real Client IPs by CLB

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

- For layer-4 CLB, the source IP obtained on the backend CVM instance is the client IP.
- For layer-7 CLB, you can use the `X-Forwarded-For` or `remote_addr` field to directly get the client IP. For the access logs of layer-7 CLB, please see Storing Access Logs in CLS.

- For CLB, the client IP 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.

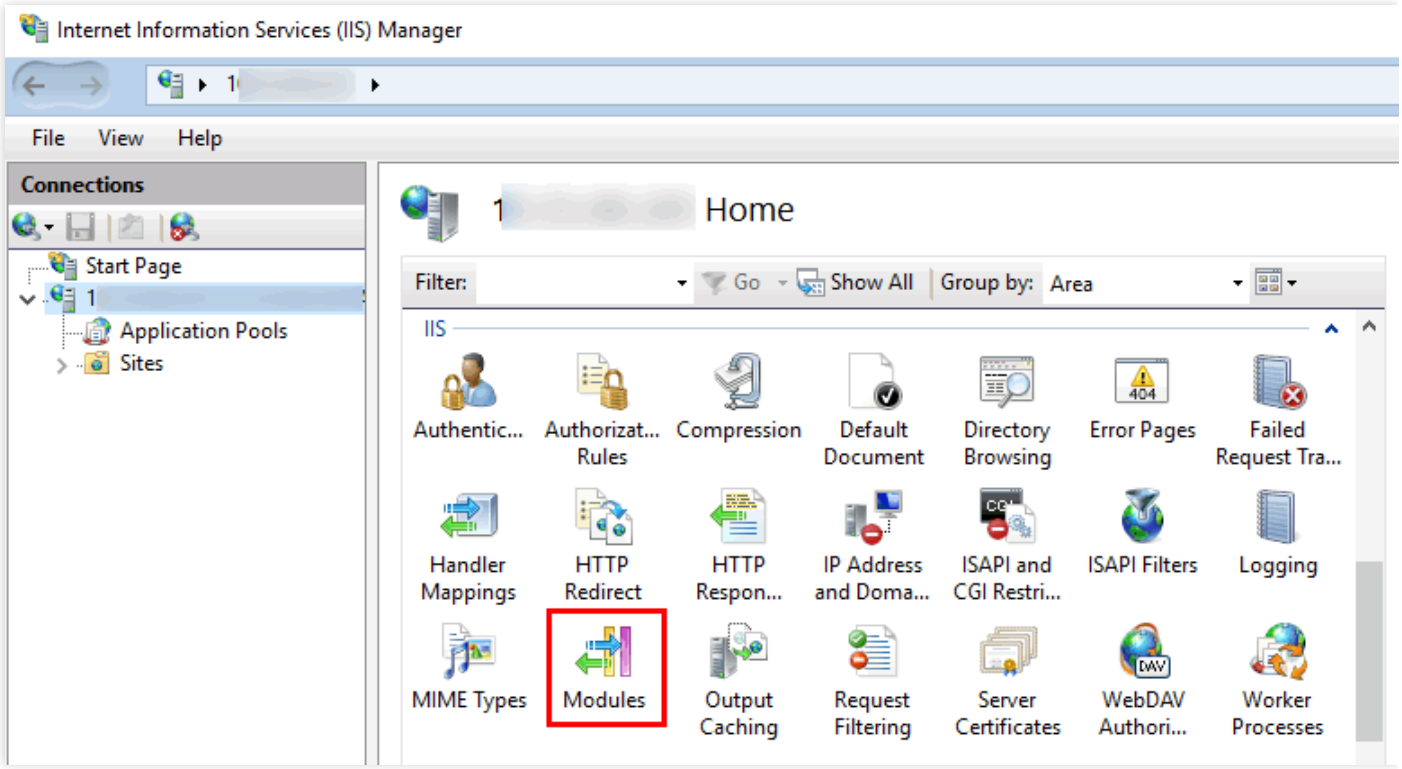
Below are commonly used application server configuration schemes.

IIS 6 Configuration Scheme

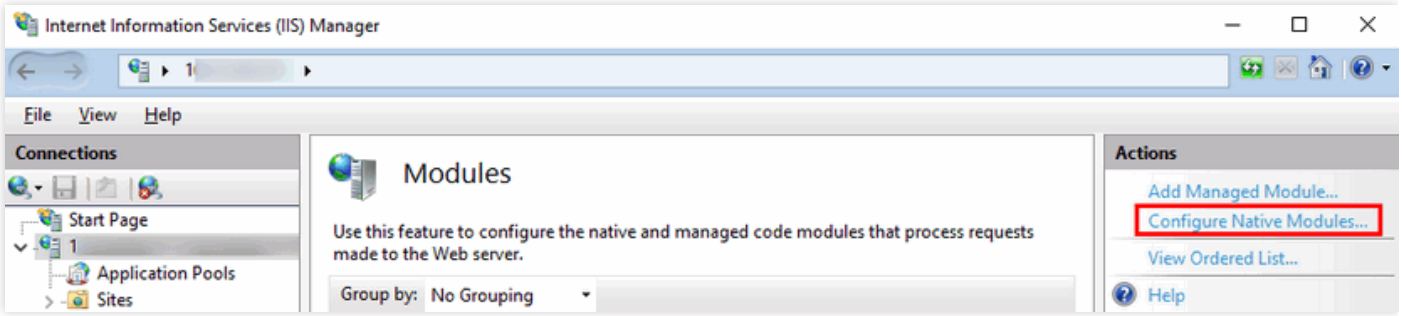
1. Download and install the F5XForwardedFor plugin module, copy `F5XForwardedFor.dll` in the `x86¥Release` or `x64¥Release` directory based on your server operating system version to a certain directory (such as `C:¥ISAPIFilters` in this document), and make sure that the IIS process has read permission to this directory.
2. Open the IIS Manager, find the currently opened website, right-click the website, and select **Properties** to open the properties page.
3. On the properties page, switch to **ISAPI Filters** and click **Add** to pop up the "Add" window.
4. In the "Add" window, enter "F5XForwardedFor" for "Filter Name" and the full path to `F5XForwardedFor.dll` for "Executable" and then click **OK**.
5. Restart the IIS server for the configuration to take effect.

IIS 7 Configuration Scheme

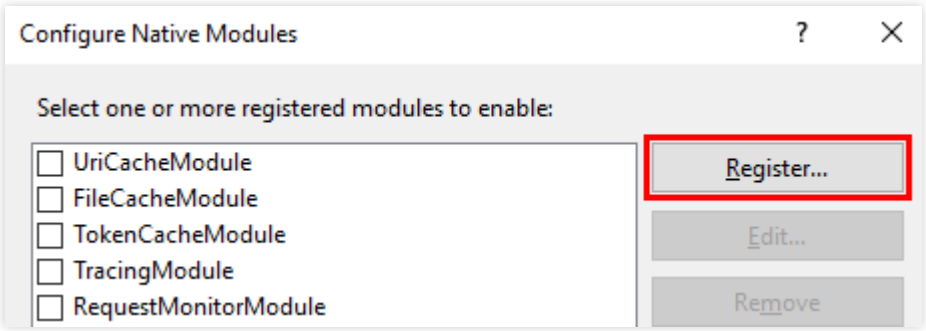
1. Download and install the F5XForwardedFor plugin module, copy `F5XFFHttpModule.dll` and `F5XFFHttpModule.ini` in the `x86¥Release` or `x64¥Release` directory based on your server operating system version to a certain directory (such as `C:¥x_forwarded_for` in this document), and make sure that the IIS process has read permission to this directory.
2. Select **IIS Server** and double-click **Modules**.



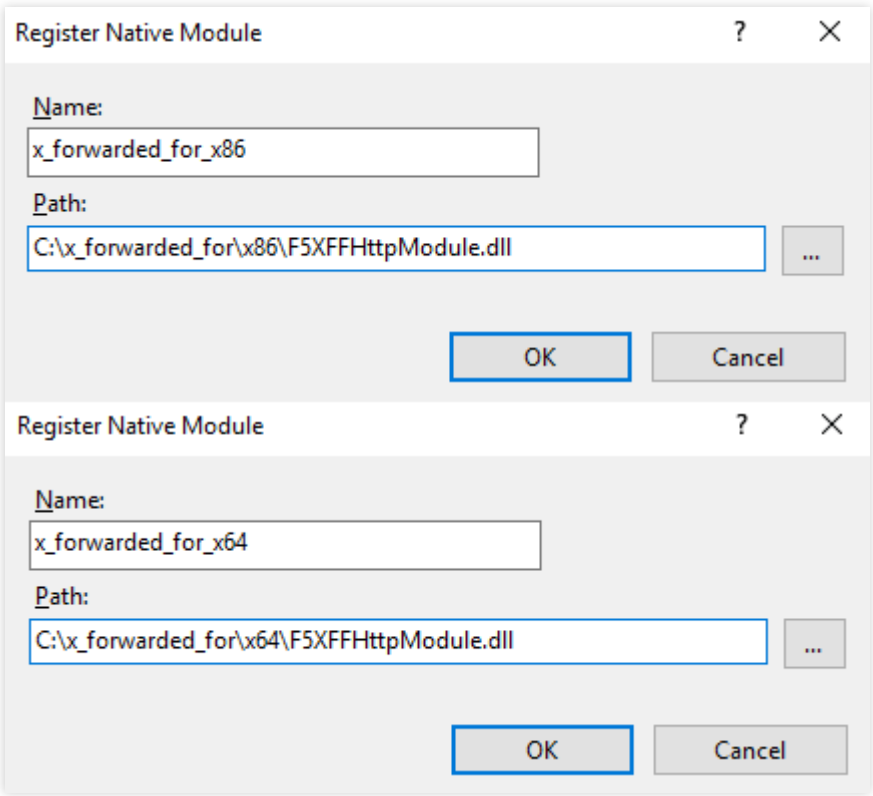
3. Click **Configure Native Modules**.



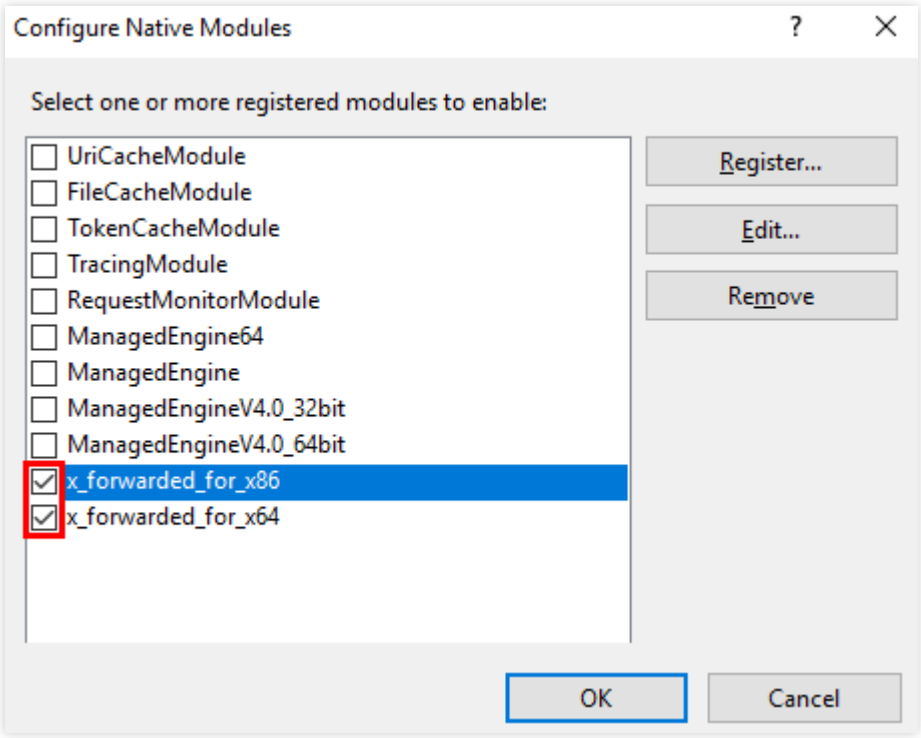
4. In the pop-up box, click **Register**.



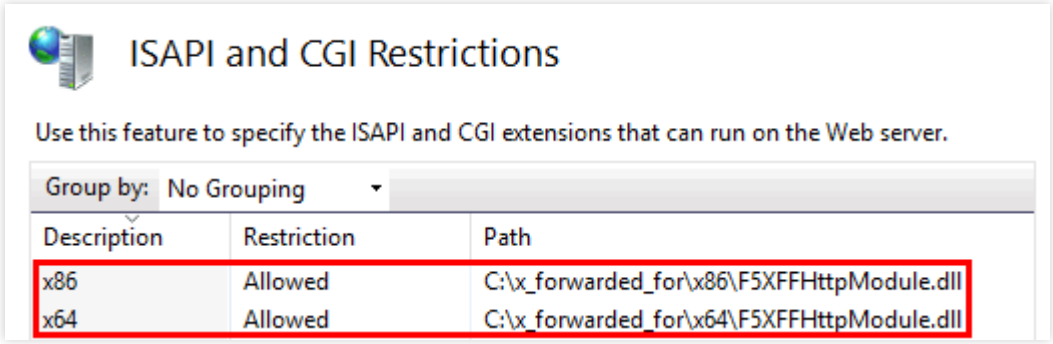
5 Add the downloaded DLL files as shown below:



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



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



7. 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 `/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 (this is not the public IP provided by CLB. For the specific IP, plea
se 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. You can use `http_realip_module` to get the real client IP when Nginx is used as the server. However, this module is not installed in Nginx by default, and you need to recompile Nginx to add `--with-http_realip_module` .

```
wget http://nginx.org/download/nginx-1.14.0.tar.gz
tar zxvf nginx-1.14.0.tar.gz
cd nginx-1.14.0
./configure --user=www --group=www --with-http_stub_status_module --without-http-cache --with-
http_ssl_module --with-http_realip_module
make
make install
```

2. Modify `nginx.conf` .

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

Modify the content in red as follows:

```


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 IP address; (this is not the public IP provided by CLB. For the specific IP, ple
real_ip_header X-Forwarded-For;
```

3. Restart Nginx.

```
service nginx restart
```

Notes on SNAT and Non-SNAT Types for Private Network CLB

Last updated : 2020-04-01 17:15:12

SNAT is not performed for any VPC-based private network CLB instances (i.e., instances configured with VPC when creating) purchased after December 15, 2016, which means the access IP obtained from the server is the real client IP. To ensure normal operations of your business, please note:

1. For private network CLB instances purchased after December 15, 2016, after a security group policy is enabled, you must allow the inbound rules of all client IPs to ensure normal access.
2. For the private network CLB instances purchased before December 15, 2016, if you want to convert them to non-SNAT type, please submit a ticket for assistance. After the conversion is made, the access IP obtained from the server will be the client IP, and the business will not be interrupted during the conversion.

Configure Multiple Availability Zones

Last updated : 2020-03-09 14:59:36

CLB instance with multiple availability zones and high availability

CLB instance supports disaster recovery across availability zones. For example, multiple clusters can be deployed in two availability zones in the same region: Hong Kong (China) Zone 1 and Hong Kong (China) Zone 2. This achieves disaster recovery across availability zones in the same region. With this feature, a CLB instance can forward frontend access traffic to other availability zones in the same region within 10 seconds if the entire availability zone fails, restoring service capability.

FAQs and use cases

Question 1: If CLB instance `test1` is configured for Hong Kong (China) Zones 1 and Zone 2, what is the policy for inbound public network traffic of the client?

Hong Kong (China) Zones 1 and Zone 2 have a pair of IP resource pools, which can be seen as IP resources with equivalent load balancing capability. Developers do not need to figure out between master cluster and slave cluster. When they purchase a CLB instance and bind it to CVM, two sets of rules will be generated and written into the two clusters, achieving high availability.

Question 2: Suppose that CLB instance test1 is configured for Hong Kong (China) Zones 1 and 2, and bound to 100 real servers in each availability zone. During business operation, 1 million HTTP persistent connections (with TCP connection kept alive) are established in each zone. If the entire CLB cluster in Zone 1 fails and becomes unavailable, what will happen to the business?

When CLB instance in Hong Kong (China) Zone 1 fails, all current persistent connections will be closed, while non-persistent connections will not be affected. The disaster recovery architecture will automatically bind the 100 servers in each zone to CLB instance in Hong Kong (China) Zone 2 within 10s, immediately restoring business capability with no manual intervention required.

Question 3: Which type of CLB is compatible with multi-AZ disaster recovery? Does it cost extra fees?

Multi-AZ disaster recovery is free of charge and currently supported by public network CLB, not private network CLB.

SSL Certificate Format

Last updated : 2020-03-09 14:59:37

1. Certificate format description

1.1. Certificate format requirements

- The certificate needs to be applied for should be in PEM format on Linux. CLB does not support certificates in other formats. For more information, please see Converting Certificates to PEM Format.
- If your certificate is issued by a root CA, the certificate is unique, and the configured website will be considered trustworthy by accessing devices such as browsers with no additional certificates required.
- If your certificate is issued by an intermediate CA, your certificate file will consist of multiple certificates. In this case, you need to combine the server certificates and intermediate certificates manually for uploading.
- If your certificate has a certificate chain, please convert it to PEM format and combine with the certificate content for uploading.
- The splicing rule follows that the server certificate content should be before the intermediate certificate content, with no blank lines in between.

> You can check for applicable rules or instructions provided by the CA (if any) alongside the certificate.

1.2. Certificate format and certificate chain format

Below are examples of certificate and certificate chain formats. Please confirm that the formats are correct before the uploading:

1. Certificate issued by a root CA: PEM format on Linux, as shown below:

   Certificate rules:

   - Your certificate should start with "——-BEGIN CERTIFICATE——-" and end with "——-END CERTIFICATE——-", which should be uploaded together.
   - All lines must contain 64 characters, except for the the last line which can contain less than 64 characters.

2. Certificate chain from an intermediate CA:

   ——-BEGIN CERTIFICATE——-
   ——-END CERTIFICATE——-
   ——-BEGIN CERTIFICATE——-
   ——-END CERTIFICATE——-
   ——-BEGIN CERTIFICATE——-
   ——-END CERTIFICATE——-

   Certificate chain rules:

   - No blank lines between certificates.
   - All certificates should meet requirements in 1.1. Certificate format requirements.

2. RSA private key format requirements

Below is an example:

An RSA private key can include all private keys (RSA and DSA), public keys (RSA and DSA), and (x509) certificates. It stores data in Base64-encoded DER format and is wrapped by ASCII headers, making it suitable for transmission in text mode between systems.

RSA private key rules:

- Your certificate should start with "——-BEGIN RSA PRIVATE KEY——-" and end with "——-END RSA PRIVATE KEY——-", which should be uploaded together.
- All lines must contain 64 characters, except for the last line which can contain less than 64 characters.

If your private key does not start with "——-BEGIN PRIVATE KEY——-" or end with "——-END PRIVATE KEY——-", you can convert it in the following way:

```
openssl rsa -in old_server_key.pem -out new_server_key.pem
```

You can then upload the `new_server_key.pem` content together with the certificate.

3. Converting certificates to PEM format

Currently, CLB only supports certificates in PEM format. Certificates in other formats need to be converted to PEM format before they can be uploaded to CLB. We recommend you use OpenSSL to convert the format. The following shows how to convert several popular certificte formats to PEM.

3.1. DER to PEM

DER format is generally used on Java platforms.

Certificate conversion: `openssl x509 -inform der -in certificate.cer -out certificate.pem`

Private key conversion: `openssl rsa -inform DER -outform PEM -in privatekey.der -out privatekey.pem`

3.2. P7B to PEM

P7B format is generally used on Windows Server and Tomcat.

Certificate conversion: `openssl pkcs7 -print_certs -in incertificat.p7b -out outcertificate.cer`

You need to get the content of "——-BEGIN CERTIFICATE——-" and "——-END CERTIFICATE——-" in `outcertificat.cer` and upload it as certificate.

Private key conversion: Private keys can generally be exported on IIS servers.

3.3. PFX to PEM

PFX format is generally used on Windows Server.

Certificate conversion: `openssl pkcs12 -in certname.pfx -nokeys -out cert.pem`

Private key conversion: `openssl pkcs12 -in certname.pfx -nocerts -out key.pem -nodes`

3.4. CER/CRT to PEM

You can convert certificates in CER/CRT formats to PEM by directly modifying their file extensions. For example, you can directly rename the "servertest.crt" certificate file to "servertest.pem".

Algorithms and Weight Configuration

Last updated : 2020-10-20 10:19:47

Comparative Analysis of CLB Algorithms

Weighted round-robin scheduling

- **How it works**
  Round-robin scheduling is to schedule requests to different servers based on polling, i.e., the system performs `i = (i + 1) mod n` to select server i in each scheduling. Weighted round-robin scheduling can solve performance imbalance of different servers. It uses weight to represent the processing performance of a server and schedules requests to different servers by weight through polling. Servers with higher weight receive connections faster but need to process more connections than those with lower weight. Servers with the same weight process the same number of connections.
- **Advantages**
  This algorithm features simplicity and high practicability. It is a stateless scheduling algorithm that does not record the status of all connections.
- **Disadvantages**
  Weighted round-robin scheduling is relatively simple, but is not suitable for scenarios where the service time of a request changes significantly, or each request consumes different amounts of time. In these scenarios, round-robin scheduling may cause load distribution imbalance among servers.
- **Use Cases**
  This algorithm is suitable for scenarios where each request consumes basically the same amount of time on the backend with the best loading performance. It is usually used in non-persistent connection services such as HTTP service.
- **Recommendations**
  If you know that each request consumes basically the same amount of time on the backend (for example, requests processed by a real server are of the same or similar types), we recommend you use weighted round-robin scheduling. If the time difference between each request is small, we recommend you use this algorithm because it has low consumption, high efficiency and no need for traversal.

Weighted least-connection scheduling

- **How it works**
  In actual situations, the time each client request spends on the server may vary greatly. If a simple

round-robin or random load balancing algorithm is used as the working time gets longer, the number of connection processes on each server may vary greatly and load balancing may not be achieved. Contrary to round-robin scheduling, least-connection scheduling is a dynamic scheduling algorithm that estimates the load on a server based on its number of active connections. The scheduler records the number of connections currently established on each server. If a request is scheduled to a server, its number of connections increases by 1. If a connection stops or times out, its number of connections decreases by 1. The weighted least-connection scheduling algorithm is based on and improves least-connection scheduling. Different weights are allocated to servers according to their processing performance. In this way, server receives a number of requests based on its weight.

i. Suppose that the weight of a real server is wi and its current number of connections is ci. The ci/wi value of each server is calculated in sequence. The real server with the least ci/wi value will be the next server to receive a new request.

ii. For real servers with the same ci/wi value, they will be scheduled based on weighted round-robin scheduling.

- **Advantages**
  This algorithm is suitable for requests requiring long-time processing, such as FTP.
- **Disadvantages**
  Due to API restrictions, least-connection and session persistence cannot be enabled at the same time.
- **Use Cases**
  This algorithm is suitable for scenarios where the time used by each request on the backend varies greatly. It is usually used in persistence connection services.
- **Recommendations**
  If you need to process different requests and their service time on the backend varies greatly (such as 3 milliseconds and 3 seconds), we recommend you use weighted least-connection scheduling to achieve load balancing.

Source hashing scheduling (ip_hash)

- **How it works**
  Source hashing scheduling uses the source IP address of the request as the hash key and finds the corresponding server from the statically assigned hash table. The request will be sent to this server if it is available and not overloaded; otherwise, null will be returned.
- **Advantages**
  ip_hash can achieve certain session persistence by remembering the source IP and mapping requests from a client to the same real server through the hash table. In scenarios where session persistence is not supported, ip_hash can be used for scheduling.

- **Recommendations**
  This algorithm calculates the hash value of the source address of a request and distributes the request to the corresponding real server based on its weight. This allows requests from the same client IP to be distributed to the same server. This algorithm is suitable for load balancing over TCP protocol that does not support cookie.

Choosing Load Balancing Algorithm and Configuring Weight

In the upcoming features of CLB, **layer-7 forwarding will support the least-connection balancing method.** We provide examples for your reference on how to choose load balancing algorithm and configure weight, so you can ensure that real server clusters undertake business in different scenarios with stability.

- Scenario 1
  Suppose there are 3 real servers with the same configuration (CPU and memory) and you configure their weights all to 10. Suppose 100 TCP connections have been established between each real server and the client. If a new real server is added, we recommend you use the least-connection scheduling algorithm, which can quickly increase the load of the 4th server and reduce the pressure on the other 3 servers.
- Scenario 2
  Suppose you use Tencent Cloud services for the first time. Your website has just been built and has low load. We recommend you purchase real servers of the same configuration because they are all access-layer servers. In this scenario, you can configure the weights of all real servers to 10 and use weighted round-robin scheduling algorithm to distribute traffic.
- Scenario 3
  Suppose you have 5 real servers to undertake simple access requests to static websites, and the ratio of their computing power (calculated by CPU and memory) is 9:3:3:3:1. In this scenario, you can configure the weights of servers to 90, 30, 30, 30, and 10 respectively. As access requests to static websites are mostly of non-persistence connection type, you can use weighted round-robin scheduling algorithm, so CLB can allocate requests based on the performance ratio of real servers.
- Scenario 4
  Suppose you have 10 real servers to undertake massive amounts of web access requests, and you do not want to purchase more servers. One of the servers often restarts due to overload. In this scenario, we recommend you configure the weights of existing servers based on their performance. Server with higher load should have a smaller weight. In addition, you can use least-

connection scheduling algorithm to allocate requests to real servers with fewer active connections to avoid server overload.

- Scenario 5
  Suppose you have 3 real servers to process persistent connections, and the ratio of their computing power (calculated by CPU and memory) is 3:1:1. The server with the best performance processes more requests, but you do not want it to be overloaded. Instead, you want to allocate new requests to idle servers. In this scenario, you can use least-connection scheduling algorithm and appropriately reduce the weights of busy servers, so CLB can allocate requests to real servers with fewer active connections, thereby achieving load balancing.
- Scenario 6
  Suppose you want subsequent requests from the client to be allocated to the same server. The weighted round-robin or weighted least-connection scheduling cannot ensure that requests from the same client are allocated to the same server. To meet the requirements of your specified application server and maintain the "stickiness" (or "continuity") of client sessions, we recommend you use ip_hash to distribute the traffic. This algorithm ensures that all requests from the same client will be distributed to the same real server, unless the number of servers changes or the server becomes unavailable.