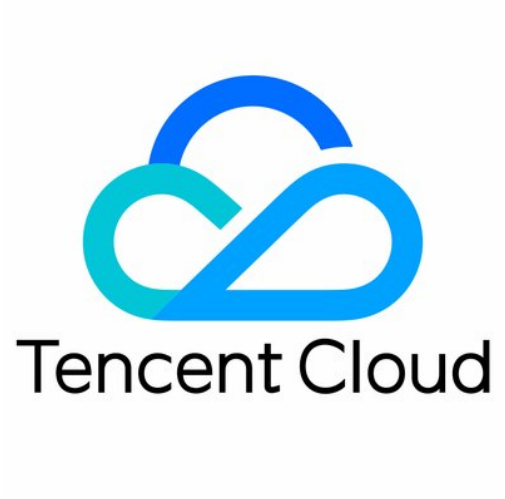
Cloud Load Balance

CLB Listeners

Product Introduction



Contents

CLB Listeners
- Introduction
- Protocols and Ports
- Polling Method
- Session Persistence
- Health Check
- Certificate Configurations
- Public Network Application-based CLB
- Redirection Configuration (Public Application CLB ONLY)

CLB Listeners Introduction

Last updated：2018-01-26 19:13:23

Cloud Load Balance service is mainly provided by cloud load balancer listener. The listener is responsible for monitoring requests on the cloud load balancer instance, delivering policies to backend CVMs and other services.

Cloud load balancer listener is configured with protocols and ports for frontend connection (client to cloud load balancer) and the ports for backend connection (cloud load balancer to backend CVM instance). In addition, cloud load balancer listener can also be configured with Session Persistence and Health Check policies.

Cloud load balancer listener can listen on the Layer-4 and Layer-7 requests on the cloud load balancer instances and deliver the requests to the backend CVM for processing. The difference between Layer-4 and Layer-7 cloud load balancers lies in whether Layer-4 information or Layer-7 information is used as the basis for determining the way of forwarding traffic when cloud load balance is performed on backend CVMs. Layer-4 cloud load balancer uses transport layer protocol and receives requests and delivers them to the backend CVMs via VIP and ports, while Layer-7 cloud load balancer uses application layer protocol and delivers traffic based on the application layer information such as URL and HTTP header.

Supported Protocol Types

Typical communication between Web applications is achieved via various layers of network, each of which provides specific communication features. According to the OSI network model, each layer has a standard communication format. Tencent Cloud's Cloud Load Balance involves Layer 4 (transport layer) and Layer 7 (application layer) in the network model.

Tencent Cloud's Cloud Load Balance supports request forwarding based on the following protocols:

- HTTP (application layer)
- HTTPS (application layer)
- TCP (transport layer)
- UDP (transport layer)

Layer-4 Protocol

If Layer-4 protocol is used for forwarding, the cloud load balancer instance forwards the request directly to the backend instance without modifying any packets. When the cloud load balancer receives the request, it attempts to open a TCP connection with the backend instance on the port specified in the listener configuration.

Layer-7 Protocol

If both frontend and backend connections use Layer-7 protocol for forwarding, the cloud load balancer resolves meaningful application layer content in the request and selects a backend CVM accordingly. Therefore, Layer-7 cloud load balancer must establish a connection with the client as a proxy of the backend CVM (three-way handshake) to receive the message containing real application layer content from client, and then determines the final choice of the internal CVM according to the specific fields in the message plus the server selection method set for the cloud load balancer.

Since the cloud load balancer is located between the client and CVM, the backend server access log only contains the IP address of the cloud load balancer. To check the actual IP address of the client, you need to use X-Forwarded-For request header. For more information, please see Obtain Client IP.

HTTPS Listener Security Mechanism

HTTPS is a secure HTTP connection that ensures the credibility of servers and clients by using SSL certificates. Cloud load balancer decrypts a request from a client with a certificate, and then sends the request to the backend instance. For more information, please see How Does SSL Work.

Difference between Layer-4 and Layer-7 Cloud Load Balancers

The difference between Layer-4 and Layer-7 cloud load balancers lies in whether Layer-4 information or Layer-7 information is used as the basis for determining the way of forwarding traffic when cloud load balance is performed on backend CVM.

Layer-4 cloud load balancer determines which traffic needs load balance based on Layer-3 IP address (VIP) and Layer-4 port number, performs NAT on the traffic to be processed, and then forwards it to the backend CVM.

Layer-7 cloud load balancer takes application layer's characteristics (such as HTTP header and URL) into account on the basis of Layer-4 cloud load balancer. For example, for the same Web server, in addition to determining the traffic to be processed based on VIP and Port 80, Layer-7 cloud load balancer can decide on whether to perform load balance based on URL, browser category and language at Layer 7. Layer-7 cloud load balancer is also known as "content exchange", which is designed to decide on the final choice

of internal CVM based on the really meaningful application layer content in message and CVM selection method set for the cloud load balancer. To select CVM based on the real application layer content, the cloud load balancer must establish a connection with the client as a proxy of the final CVM (three-way handshake) to receive the message containing real application layer content from client, and then determines the final choice of the internal CVM according to the specific fields in the message plus the server selection method set for the load balancer. In this case, the cloud load balancer is more like a proxy server. The cloud load balancer establishes TCP connection with frontend client and backend CVM separately.

Protocols and Ports

Last updated：2018-01-26 19:14:11

Tencent Cloud supports two listener modes, Layer-4 forwarding and Layer-7 forwarding. For different protocol types, they work on transport layer and application layer in the network model, respectively. For more information about listener modes supported by different types of CLB instances, please see Public Network CLB Instance and Private Network CLB Instance.

Layer-4 Forwarding Listener Protocol and Port Configuration

The CLB listener listens to Layer-4 requests (the 4th layer of OSI network protocol, the transport layer) on the cloud load balancer instance and distributes TCP and UDP requests to backend servers for processing. Layer-4 forwarding is achieved by the following configurations:

Frontend and Backend Protocol	Frontend Port (CLB Port)	Backend Port (Server Port)	Note
Layer-4 protocol (TCP and UDP)	Frontend ports used by CLB instances to receive requests when providing internal or external services. You can perform cloud load balance for the following TCP ports: 21 (FTP), 25 (SMTP), 80 (Http), 443 (Https), 1024-65535, etc.	Backend ports used by CVMs to provide services such as receiving traffic distributed by the CLB. In the application CLB instance, a listener port corresponds to multiple backend ports. In the classic CLB instance, a listener port corresponds to the same port on the backend.	**A frontend port must be unique**. For example, TCP port 23 and UDP port 23 cannot co-exist. **The backend ports of a classic LB instance must be unique while those of an application LB instance can be used repeatedly.** For Layer-4 forwarding, the client source IP requests are directly sent to backend servers. The feature of acquiring client IP is enabled by default (not applicable to private network CLB instances in VPCs)

Layer-7 Forwarding Listener Protocol and Port Configuration

The CLB listener listens to Layer-7 requests (the 7th layer of OSI network protocol, the application layer) on the cloud load balancer instance and distributes HTTP(s) requests to backend servers for processing. Layer-7 forwarding is achieved by the following configurations:

Frontend and Backend Protocol	Frontend Port (CLB Port)	Backend Port (Server Port)	Note
Layer-7 protocol (HTTP or HTTPS protocol)	Frontend ports used by CLB instances to receive requests when providing internal or external services. Ports from 1 to 65535 are supported.	Backend ports used by CVMs to provide services such as receiving traffic distributed by the CLB. In the application CLB instance, a listener port corresponds to multiple backend ports. In the classic CLB instance, a listener port corresponds to the same port on the backend.	**A frontend port must be unique**. For example, HTTP 80 and TCP 80 cannot co-exist. **The backend ports of a classic LB instance must be unique while those of an application LB instance can be used repeatedly.** For Layer-7 forwarding, the client source IP requests are directly sent to backend servers through configuring proxy_set_header X-Forwarded. The feature of acquiring client IP is enabled by default (not applicable to private network CLB instances in VPCs)

Polling Method

Last updated：2018-06-01 17:28:53

Polling refers to the scheduling algorithm with which load balancer distributes traffic to RS. Different polling methods combined with different weight configurations on the RS lead to different outcomes.

Weighted Round-Robin Scheduling

Weighted Round-Robin Scheduling means dispatching requests to different servers in sequence by use of polling. Weighted Round-Robin Scheduling is used to solve problems cause by performance inconsistency between servers. It uses appropriate weight values to represent each server's performance level and assign requests to servers according to these weight values and polling method. Servers with high weight values receive connections first. They process more connections than those with low weight values. Servers with the same weight value process the same number of connections.

- **Advantage**: This scheduling algorithm is simple and practical. It's a stateless scheduling method because there is no need to record the statuses of current connections.
- **Disadvantage**: Weighted Round-Robin Scheduling is relatively simple but is not suitable for situations where the service time of a request changes a lot, or where each request needs a different time length. In these cases, this round-robin scheduling will cause unbalanced load distribution among servers.
- **Application Scenario**: Backend service time needed by each request is relatively identical, in which case load distribution is optimal. This is commonly used for short connection services, such as HTTP, etc.
- **Recommendation**: If users have known that each request takes basically the same amount of backend service time, and that the RS processes requests of the same type or of similar types, it is recommended to choose Weighted Round-Robin Scheduling. Weighted Round-Robin Scheduling is also recommended when the time needed by requests varies little, because this highly efficient scheduling method consumes little resources with no need for traversing.

Weighted Least-Connection Scheduling

In practice, each request service from the client may stay at the server for a different time length. If a simple polling or random load balancing algorithm is used, the numbers of connection processes on each server may vary greatly over time, which means load balance is not actually achieved.

Weighted Least-Connection Scheduling is a dynamic scheduling algorithm which estimates a server's load condition according to its current number of active connections. Opposite to Round-Robin Scheduling, Weighted Least-Connection Scheduling is a dynamic scheduling algorithm which estimates a server's load condition according to its current number of active connections. The scheduler needs to record the number of established connections for every server. This number increases by 1 when a request is dispatched to a server, and decreases by 1 when a connection is terminated or has timed out.

Based on Least-Connection Scheduling, Weighted Least-Connection Scheduling assigns different weight values to servers based on their processing performance, which allows the servers to receive a number of service requests according to their weight values. This method is an improved version of the original Least-Connection Scheduling.

1. Suppose the weight values for RS instances are w1, w2...wi, and their numbers of current connections are c1, c2...ci. The values of ci/wi are calculated in succession, and the RS instance with the smallest value will be the next one to which the request is allocated.

2. If RS instances with the same ci/wi value exist, the scheduling is done using Weighted Round-Robin Scheduling.

3. **Advantage**: This load balancing algorithm is suitable for request services that take a long time to process, such as FTP, etc.

4. **Disadvantage**: Due to API restrictions, you cannot enable Least-Connection and session persistence at the same time.

5. **Application Scenario**: Backend service time needed by each request varies greatly. This is commonly used for persistent connection services.

6. **Recommendation**: When users need to process different requests, and the backend service time needed by these requests varies greatly (such as 3 ms and 3 sec), it is recommended to use Weighted Least-Connection Scheduling to achieve load balance.

Source Hashing Scheduling (ip_hash)

Use the source IP address of the request as Hash Key and find the corresponding server from the statically assigned hash table. The request is sent to this server if the server is available and not overloaded, otherwise the response is empty.

- **Advantage**: Requests from a certain client can be constantly mapped to the same RS through the hash table. Therefore, a simple session persistence can be achieved by use of ip_hash in scenarios where session persistence is not supported.

- **Recommendation**: Hash the source IP of a request and dispatch the request to a matching server according to RS weight. This allows all requests from the same client IP to be constantly dispatched to a certain server. This scheduling method is suitable for TCP protocols without cookie feature.

Choosing Load Balancing Algorithm and Configuring Weight

To allow RS clusters to undertake business in a stable manner in different scenarios, we provide the following reference cases regarding how to choose load balancing algorithm and configure weight.

- Scenario 1:
  If there are three RS instances with the same configuration (CPU/RAM), you may configure their weight as 10 because they have the same performance level. Each RS has established 100 TCP connections with clients, in which case you need to add another RS. In this scenario, it is recommended that you use Least-Connection Scheduling to quickly assign load to the fourth RS and lower the pressure on the other three servers.

- Scenario 2:
  Suppose you have just begun to use cloud services, your website is relatively new with low website load. In this case, it is recommended that you purchase a RS with the same configuration, because RS instances are identically the same access layer servers. In this scenario, you can configure RS weight as 10 and use Weighted Round-Robin Scheduling to distribute traffic.

- Scenario 3:
  You use 5 servers to satisfy the access need for a simple static website, the ratio of their computing capability is 9:3:3:3:1 (calculated based on their CPUs and RAMs). In this scenario, you can set the weights of these RS to 90, 30, 30, 30, 10, respectively. Since most access requests towards static websites are short connection requests, you can use Weighted Round-Robin Scheduling to allow the load balancer instance to assign requests according to the performance ratio of each RS.

- Scenario 4:
  You use 10 RS to support huge amount of Web access requests, and are not planning to purchase additional RS to save cost. One of the RS often restarts due to overload. In this scenario, it is recommended that you set the weight based on the performance level of each RS and set a relatively

low weight value for the RS whose load is too high. In addition, you can use Least-Connection load balancing algorithm to assign requests to RS with fewer active connections to reduce the pressure on overloading RS.

- Scenario 5:
  You use 3 RS to process several persistent connection requests, the ratio of these servers' computing capability is 3:1:1 (calculated based on their CPUs and RAMs). In this case, the server with the best performance processes more requests. To avoid overloading, you wish to assign new requests to idle servers. In this scenario, you can use Least-Connection load balancing algorithm and lower the weight of busy servers as appropriate. This allows load balancer to assign requests to RS with fewer active connections to achieve load balance.

- Scenario 6:
  Suppose you want the requests from subsequent clients to be assigned to the same server. However, neither Weighted Round-Robin nor Weighted Least-Connection scheduling method can guarantee that requests from the same client are assigned to the same server. In this scenario, in order to meet your requirements for specific application servers and ensure "stickiness" or "persistence" of client sessions, we can use ip-hash load balancing method to distribute traffic. This ensures that requests from the same client are always distributed to the same RS. (Not including the situations where the number of servers changes, or the server becomes unavailable)

Session Persistence

Last updated：2018-09-10 15:18:49

Session persistence allows the requests from the same IP to be forwarded to the same backend CVM. By default, load balancer routes each request to a different backend CVM instance. However, you can use session persistence feature to route requests from a specific user to the same backend CVM instance, so that some applications (such as shopping cart) that need to maintain the session state can work properly.

Layer-4 Session Persistence

Layer-4 forwarding scenario supports simple session persistence. The session persistence duration can be set to any integer value within the range of `30-3600` seconds. If the time threshold is exceeded and there is no new request in the session, the session will be disconnected.

Layer-7 Session Persistence

Layer-7 forwarding scenario supports session persistence based on cookie insertion (the cookie is stuffed into the client by load balancer). Session duration range is 30-3600s. For more information on session persistence based on cookie insertion, please see Session Persistence Principles.

Connection Timeout

HTTP connection timeout (keepalive_timeout) is not adjustable and the default is `75` seconds. If there is no data transfer in the session for a period exceeding the time threshold, the session will be disconnected. TCP connection timeout is not adjustable and the default is `900` seconds. If there is no data transfer in the session for a period exceeding the time threshold, the session will be disconnected.

Configuring Session Persistence

1. Log in to the CLB Console, then click the ID of the load balancer instance that needs session persistence configurations to go to its details page.

2. Click the **Modify** button next to the load balancer listener that needs session persistence configurations.

3. Set whether to enable the session persistence feature. Click the button to enable it, then enter the persistence duration, and click **OK**.

Relationship Between Persistent Connection and Session Persistence

Scenario 1: HTTP Layer-7 Business

Assume that the client access protocol is HTTP/1.1, and "Connection: keep-alive" is set in the header information. If the backend CVM is accessed through the CLB with session persistence disabled, is it possible to access the same CVM next time?

A: No.

First of all, HTTP keep-alive means that the TCP connection remains connected after requests are sent. Therefore, the browser can continue to send requests over the same connection. Keeping connected can save the time and bandwidth taken to set up a new connection for each request. The default timeout for a CLB cluster is `75` seconds(if there is no new request within `75` seconds, the TCP connection is disconnected by default).

HTTP keep-alive is established by the client with the CLB. If the cookie session persistence is disabled, the CLB will randomly select a backend CVM according to the polling policy. The previous persistent connection is in vain.

Therefore, it is recommended to enable session persistence.

If the cookie session persistence is set to `1,000` seconds and the client initiates another request, TCP connection needs to be re-established, because it has been more than 75s since the last request is sent. The application layer finds the same CVM by cookie, and the CVM accessed by the client is still the one used for the last access.

Scenario 2: TCP Layer-4 Business

Assume that the client initiates the access using TCP as the transport layer protocol, and persistent connection is enabled, but source IP-based session persistence is disabled, can the same client access the same CVM next time?

A: Uncertain.

First of all, according to the layer-4 implementation mechanism, if the persistent connection is enabled for TCP and remains connected, the same CVM can be accessed, because the two accesses use the same connection. However, if the first connection is released due to some reason (network restart or connection timeout), the second access may be dispatched to other backend CVMs. Besides, the global timeout for

persistent connection is `900` seconds by default, which means that the persistent connection will be released if there is no new request.

Health Check

Last updated：2018-09-10 14:57:07

Tencent Cloud's cloud load balancer instances can periodically send Ping to backend CVMs, make attempts to connect with them or send requests to them to test their running status. This is called "health check".

If it is concluded that a backend CVM instance is unhealthy, the cloud load balancer instance will not forward requests to the instance. But health check will be performed on all backend CVMs, whether healthy or unhealthy, and when the unhealthy instance returns to normal state, the cloud load balancer instance will forward new requests to it.

Auto scaling group regularly checks the running status of instances within each group in a similar way. For more information, please see Auto Scaling product documentation.

Definitions of Health Check Configuration Fields

Response Timeout: The maximum time-out for the response to health check. If a backend CVM does not respond properly within the time limit, it is considered that the health check fails.

Health Check Interval: Time interval between health checks.

Unhealthy Threshold: If a failure result is returned for the health check for n consecutive times (n is the input value), the backend CVM is identified as unhealthy and marked "Unhealthy" on the console.

Healthy Threshold: If a success result is returned for the health check for n consecutive times (n is the input value), the backend CVM is identified as healthy and marked "Healthy" on the console.

Normal Status Code: This is only applicable to HTTP check method. Specify the HTTP status code used to verify that the health check is normal. Option values include `http_1xx`, `http_2xx`, `http_3xx`, `http_4xx` and `http_5xx`. Multiple choices are allowed. By default or if no choice is made, it is set to `http_2xx`.

Health Check Configuration for Layer-4 Forwarding

Under the health check mechanism for Layer-4 forwarding, cloud load balancer initiates an access request to the CVM port specified in the configuration. If the access to the port is normal, the backend CVM is considered normal, otherwise it is considered abnormal. For TCP services, SYN packet is used for the check. For UDP services, Ping command is used for the check.

- Response timeout: 2-60 seconds

- Check interval: 5-300 seconds
- Unhealthy threshold: 2-10 times (When the response timeout has happened to a healthy backend CVM for the specified number of times, the backend CVM is considered unhealthy)
- Healthy threshold: 2-10 times (When the response timeout has happened to an unhealthy backend CVM for the specified number of times, the backend CVM is considered healthy)

Health Check Configuration for Layer-7 Forwarding

Under the health check mechanism for Layer-7 forwarding, the cloud load balancer sends an HTTP request to the backend CVM to check the backend services. The cloud load balancer determines the running status of the service based on whether the returned value of HTTP is `http_2xx` or `http_4xx`. In the future, users will be allowed to customize the descriptions of the statuses represented by response codes. For example, in a certain scenario, HTTP returned values include `http_1xx`, `http_2xx`, `http_3xx`, `http_4xx` and `http_5xx`. Users can, based on business needs, define `http_1xx` and `http_2xx` as normal status and the values from `http_3xx` to `http_5xx` as abnormal status.

- Response timeout cannot be configured. Default is 5 seconds.
- Check interval is 5-300 seconds, default is 6 seconds.
- Unhealthy threshold: 2-10 times, default is 3 times. When the response timeout has happened to a healthy backend CVM for the specified number of times, the backend CVM is considered unhealthy.
- Healthy threshold: 2-10 times, default is 3 times. When the response timeout has happened to an unhealthy backend CVM for the specified number of times, the backend CVM is considered healthy.
- HTTP Request Method: by default, the HEAD method is used. The server only returns the HTTP header information. The corresponding backend service needs to support HEAD. Selecting HEAD method can reduce back-end overhead and improve the request efficiency. If the GET method is used, the backend service needs to support GET.

How to Troubleshoot in Health Check

Layer-4 Troubleshooting

Under TCP protocol, cloud load balancer uses SYN packet for the check while under UDP protocol, it uses Ping command for the check.

When a backend CVM port is marked "unhealthy" in the page, you should conduct troubleshooting using the following procedures:

- Check whether the service of the backend CVM is affected by a configuration or the security group. For information on how to ensure the normal operation of service by controlling the access to the backend

CVM, please see Access Control for the Backend CVM.

- Use the `netstat` command to check if there is a process listening on the backend CVM's port. If no such process is found, restart the service.

Layer-7 Troubleshooting

For Layer-7 services (HTTP/HTTPS protocol), when an exception is detected in the health check by a listener process, the troubleshooting can be performed in the following ways:

1) The Layer-7 health check service of cloud load balancer communicates with the backend CVM via private network, so you need to log in to the server to check whether the application server port is being listened on normally at the private network address; if not, you should move the listening of application server port to the private network address to ensure the normal communication between cloud load balancer system and backend CVM.

Assume that the both the frontend and the backend ports of the cloud load balancer are 80, the CVM's private IP is: 1.1.1.10

The server on Windows system uses the following command:

```
netstat -ano | findstr :80
```

The server on Linux system uses the following command:

```
netstat -anp | grep :80
```

If you can see the listening status at 1.1.1.10:80 or 0.0.0.0:80, the configuration is considered normal.

2) Make sure that the backend port configured in the cloud load balancer listener has been enabled on the backend CVM.

For Layer-4 cloud load balancer, it is considered normal as long as the backend port telnet gives a response. You can use `telnet 1.1.1.10 80` to test. For Layer-7 cloud load balancer, such HTTP status codes as 200 indicate a normal state. The test is conducted as follows:

- On Windows system, directly input private IP in the CVM browser to check whether it is normal. In this example, `http://1.1.1.10` is input;
- On Linux system, use the `curl-I` command to check if the status is `HTTP/1.1 200 OK`. In this example, `curl -I 1.1.1.10` command is used

3) Check whether there is a firewall or other security software inside the backend CVM. This type of software can easily block the local IP address of the cloud load balancer system, causing the failure of cloud load balancer system to communicate with the backend CVM.

Check whether the firewall of private network on server allows port 80. You can temporarily disable the firewall for the test.

- For Windows system, run the `firewall.cpl' entry to disable the firewall
- For Linux system, input `/etc/init.d/iptables stop` to disable the firewall

4) Check if the parameter settings of cloud load balancer health check are correct. It is recommended to complete the settings by referring to the health check default parameter values provided in this document.

5) The recommended test file specified for health check is a simple page in HTML form and is only used for check returned results. Dynamic scripting languages such as php are not recommended.

6) Check whether there is a high load on the backend CVM that leads to slow response of CVM to provide service.

7) Check the HTTP request method. If you use the HEAD method, the backend service must support HEAD. If it is a GET method, the backend service must support GET.

Notes about too frequent health check

Health check packets are sent too frequently. Each health check packet is sent every 5 seconds as configured in the console. But the backend RS finds that one or more health check requests are received in one second. Why is that?

Too frequent health check is caused by the implementation mechanism of CLB backend health check. Assume that 1 million requests from client are distributed on four LB backend physical machines before being sent to the CVM, and each LB backend physical machine conducts health check separately. If the LB instance is set to send a health check request every 5 seconds, each physical machine on the LB backend sends a health check request every 5 seconds. That's why the backend CVM receives multiple health check requests. For example, if the cluster to which an LB instance belongs has eight physical machines, and each machine sends a request every 5 seconds, the backend CVM may receive 8 health check requests in 5 seconds.

The advantages of this implementation solution are high efficiency, accurate check, and avoidance of mis-removal. For example, if one of eight physical machines in the LB instance cluster fails, the other seven machines can still forward traffic normally.

Therefore, if your backend CVM is checked too frequently, you can set the check interval to be much longer, such as, 15 seconds.

Certificate Configurations

Last updated：2018-08-21 15:55:25

Common Procedure for Applying for Certificates

- Generate private key locally: openssl genrsa -out privateKey.pem 2048, where privateKey.pem will be your private key file. Make sure to place the file in a secure place for safekeeping
- Generate certificate request file: openssl req -new -key privateKey.pem -out server.csr, where server.csr will be your certificate request file which is used to apply for certificate
- Acquire the content of the request file and go to authorities (such as CA) to apply for certificate

Certificate Format Requirement

- The format of the certificate to be applied by the user is a .pem file under linux environment. Cloud load balancer does not support certificates of other types. If your certificate is of another type, please refer to the "Certificate Types Supported by Cloud Load Balancers and Conversion Method" section of this document.

- A certificate issued by root CA is unique, in which case you will not need additional certificates and your configured site will be considered as trusted by accessing devices such as browsers.

- A certificate file issued by intermediate CA includes multiple certificates, you need to concatenate server certificate and intermediate certificate manually before uploading.

- If your certificate has a certificate chain, please convert its content into PEM format, and upload it with the certificate content.

- Concatenation rule: The server certificate comes first, followed by the intermediate certificate. No blank lines in between. Note: CAs will usually provide corresponding instructions when issuing certificates, make sure to read the rules.

Here are examples of certificate format and certificate chain format. Please confirm the format before uploading:

1.Certificate issued by root CA: Certificate is pem format under linux environment. Example:

Rules for certificate:

- [--- BEGIN CERTIFICATE ---, --- END CERTIFICATE ---] are the beginning and end, which should be uploaded with the content;
- Each line contains 64 characters, but the last line can contain less than 64 characters;

2.Certificate chain issued by intermediate CA:
---BEGIN CERTIFICATE---
---END CERTIFICATE---
---BEGIN CERTIFICATE---
---END CERTIFICATE---
---BEGIN CERTIFICATE---
---END CERTIFICATE---

Rules for certificate chain:

- No blank line is allowed between certificates;
- Each certificate shall comply with the certificate format rules described above;

RSA Private Key Format Requirement

Example:

RSA private key can include all private keys (RSA and DSA), public keys (RSA and DSA), and (x509) certificates. It stores DER data encoded with Base64 and is enclosed by ascii header, being suitable for textual transfer between systems.

Rules for RSA private key:

- [---BEGIN RSA PRIVATE KEY---, ---END RSA PRIVATE KEY---] are the beginning and end, which should be uploaded with the content;
- Each line contains 64 characters, but the last line can contain less than 64 characters;

If the private key is generated using other methods than the one described above and has a format of [--- BEGIN PRIVATE KEY ---, --- END PRIVATE KEY ---], you can convert the format as follows:

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

Then upload the content of new_server_key.pem with the certificate.

How to Convert Certificate into PEM Format

Currently, cloud load balance only supports certificates of PEM format. Any non-PEM certificates are required to be converted to PEM format before being uploaded to cloud load balance. It is recommended to use openssl tool for the conversion. Here are some common methods for converting the certificate format to PEM format.

Converting DER to PEM

DER format generally occurs in Java platform.

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`

Converting P7B to PEM

P7B format generally occurs in Windows Server and Tomcat.

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

Obtain [--- BEGIN CERTIFICATE ---, --- END CERTIFICATE ---] content in outcertificat.cer as a certificate for upload.

Private key conversion: no private key

Converting PFX to PEM

PFX format generally occurs in Windows Server.

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

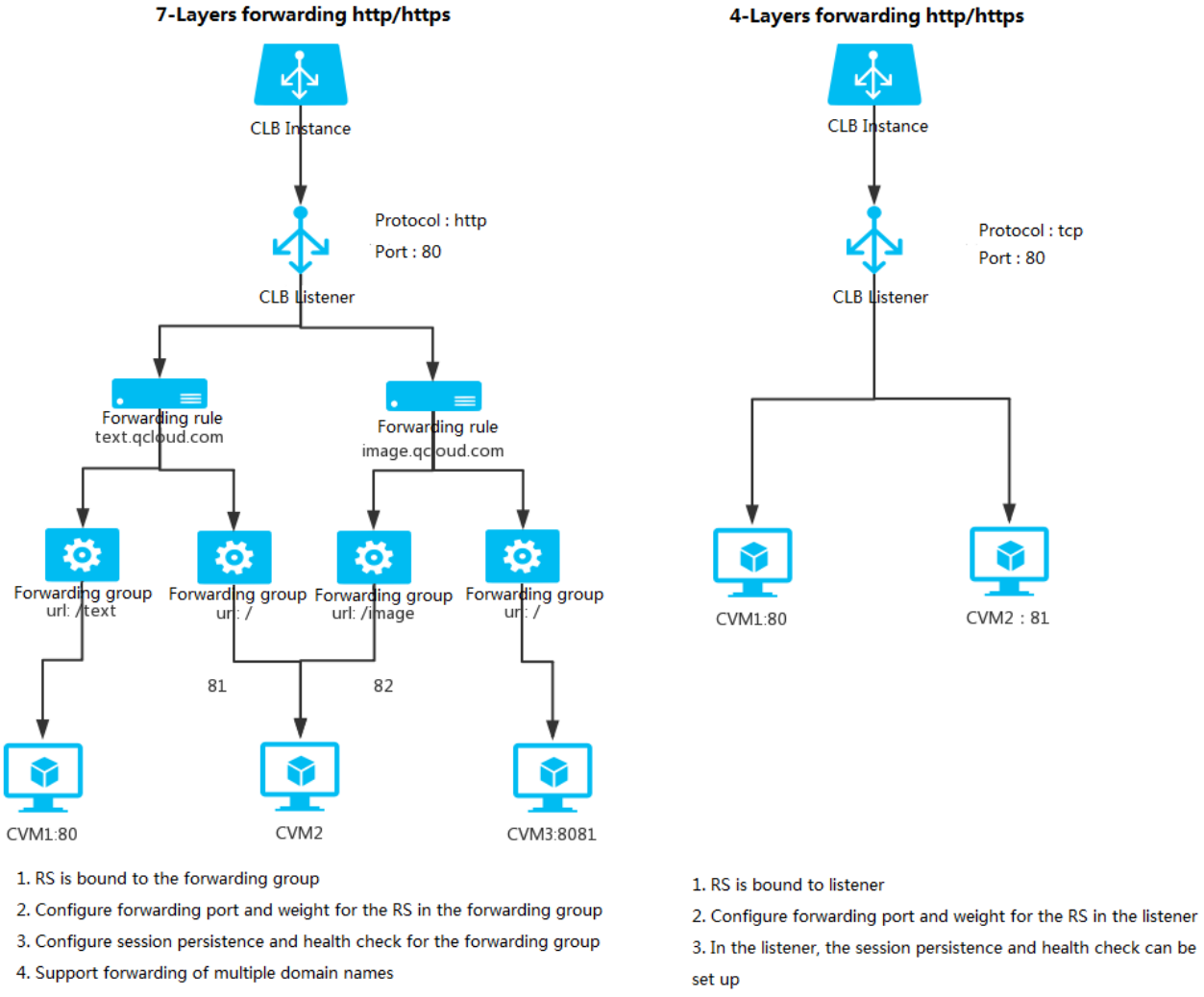
Private key conversion: `openssl pkcs12 -in certname.pfx -nocerts -out key.pem -nodes`

Public Network Application-based CLB

Last updated：2018-10-08 17:39:32

Business Process Chart

The Layer-7 business process and Layer-4 business process of the public network application-based CLB are displayed as follows:



- In the Layer-7 forwarding http/https of the public network application CLB, you can add a domain name for the new forwarding rule in the listener of a LB instance.

- When only one forwarding rule is created, you can access the service by accessing VIP + URL, for it can correspond to the appropriate forwarding rule.
- When multiple forwarding rules are created, you will be prompted that a random domain name + URL may be accessed by simply accessing VIP+URL. You should access domain name + URL directly to enable the specified forwarding rules to take effect. In other words, when more than one forwarding rules are set, it is not recommended to access service by means of VIP + URL, because a VIP may correspond to multiple domain names. Instead, you should use specific domain name + URL to access the service.

Notes about Configuration of Forwarding Rules

Domain Configuration Rules

In the public network application CLB, when a domain is configured using the forwarding rules of Layer-7 listener, a regular expression should be adopted with a length of 1-80 characters.

- Character sets supported in non-regular domain names are as follows:
  `a-z` `0-9` `.` `-`

- A wild card domain name only supports
  `*.example.com` or `www.example.*`. Only one * can occur in a single domain name.

- Character sets not supported in the regular expression of domain names are as follows:
  `"` `{` `}` `;` `~` `'` `` ` `` blank space

- The followings is an example of regular domain names supported by application-based cloud load balancers:
  `~^www\d+\.example\.com$`

Health Check Configuration Rules

If the domain name you entered is a wildcard or regular one, you need to specify a fixed domain name for the health check. The character sets supported in a domain name for health check are as followings:
`a-z` `0-9` `_` `.` `-`

In the public network application CLB, when a health check path is configured using the forwarding rules of Layer-7 listener, Default is "/" and must start with "/" with a maximum of 80 characters. Regular expression is not supported. It is recommended to specify a fixed URL path (static page) for health check.

The character sets supported in path configuration for health check are as followings:
`a-z` `A-Z` `0-9` `_` `.` `-` `/` `=` `?`

Example of Domain Name Matching Rules:

1. Enter IP instead of domain name in the forwarding rule, and configure multiple URLs in the forwarding group. The service is accessed via VIP + URL.

2. Configure the full domain name in the forwarding rule and configure multiple URLs in the forwarding group. The service is accessed via Domain + URL.

3. Configure wildcard domain name in the forwarding rule and configure multiple URLs in the forwarding group. The service is accessed via matching Domain + URL. When customers want to point different domain names to the same URL, this method applies. Take example.qcould.com as an example, the format is as follows:
   `example.qcloud.com` exactly matches the example.qcloud.com domain name
   `\*.qcloud.com` matches all domain names ending in qcloud.com
   `example.qcloud.\*` matches all domain names beginning with example.qcloud

   > Note: If the requested domain name does not match any of the forwarding rules, a 403 error is returned.

4. Configure the domain name in the forwarding rule and configure the fuzzy matching URL in the forwarding group. Use the prefix match to add the wildcard $ at the end for complete match.
   For example, a customer wants to match any file that ends with gif, jpg, or bmp by configuring the forwarding group URL ~* .(gif|jpg|bmp)$.

Notes about Forwarding Group URL Matching Rules

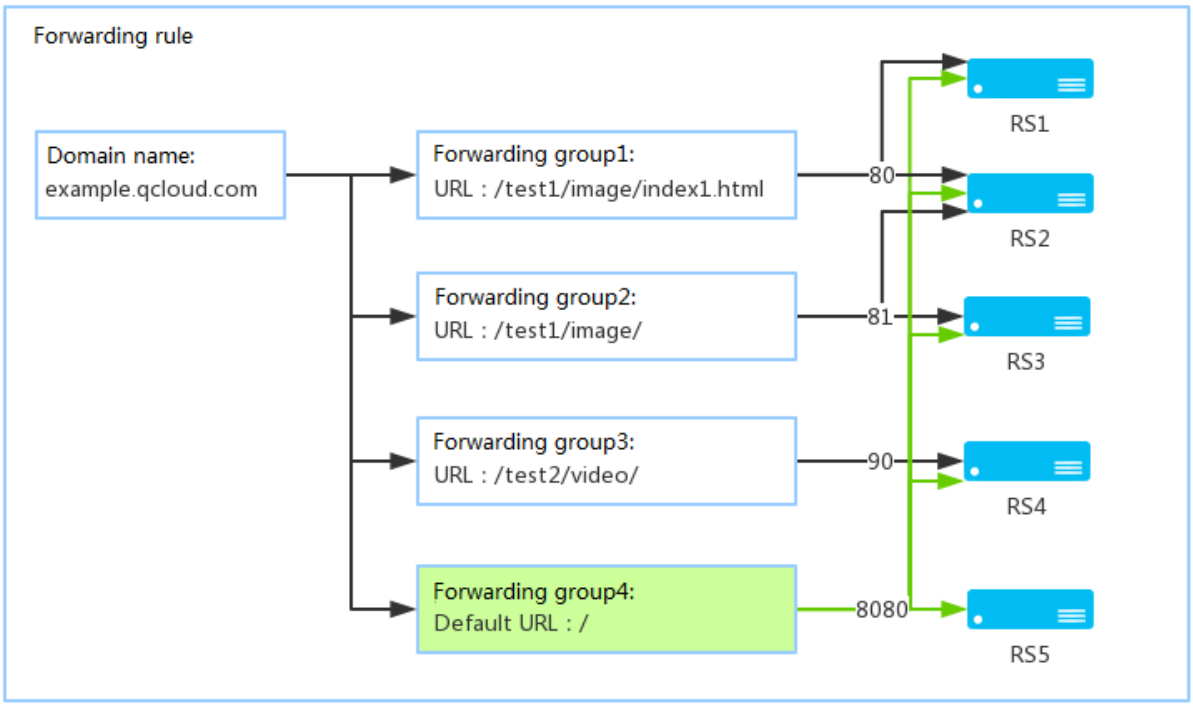
URL Configuration Rules

In the public network application CLB, the forwarding path URL of Layer-7 listener is "/" by default and must start with "/" with a maximum of 80 characters. Regular expressions are supported.

A non-regular URL needs to start with "/ "and supports the following character sets:
`a-z` `A-Z` `0-9` `_` `.` `-` `/` `=` `?`

A regular URL does not support the following character sets:
" { } ; \ ` ~ ' blank space

Example of URL Matching Rules



1. Match rules: Exact match should be prior to the fuzzy match

   Example: After configuring forwarding rules and forwarding groups according to the following figure, the following requests will be matched to different forwarding groups in sequence:

   a. example.qloud.com/test1/image/index1.html The request is forwarded to the backend CVM associated with forwarding group 1 due to the exact match of the URL rule set by forwarding group 1. Port 80 for RS1 and RS2 is illustrated in the figure.

   b. example.qloud.com/test1/image/hello.html Since this request does not exactly match the first rule, it continues to match the rules in forwarding group 2 and finds that the fuzzy match succeeds. Therefore, the request will be forwarded to the backend CVM associated with forwarding group 2. Port 81 for RS2 and RS3 is illustrated in the figure.

c. example.qloud.com/test2/video/mp4/ Since this request does not match exactly the first two rules, it continues to match downwards and finds that it can do a fuzzy match with the rules in forwarding group 3. Therefore, the request will be forwarded to the backend CVM associated with forwarding group 3. Port 90 for RS4 is illustrated in the figure.

d. example.qloud.com/test3/hello/index.html Since this request does not match the rules in the first three forwarding groups, a match with the common rule "default URL" set by the user should apply. nginx will serve as a reverse proxy server to forward the request to the backend application server such as FastCGI (php) and tomcat (jsp).

e. example.qloud.com/test2/ Since this request does not exactly match the rules in the first three forwarding groups, a match with the common rule "default URL" set by the user should apply.

2. If the service does not work properly in the user's URL rules, it will not be redirected to other pages after the successful match.
   For example: If the client requests example.qloud.com/test1/image/index1.html and matches the URL rules of forwarding group 1, but the backend CVM of forwarding group 1 is running abnormally and shows 404 error page. Then, the user may only see 404 error page rather than other pages when accessing.

3. ***It is recommended that you point the default URL to a stable page (such as a static page and homepage) and bind it to all backend CVMs.*** If none of the rules match, the system points the request to the default URL page. Otherwise, a 404 error page may appear.

4. If you do not configure the default URL, and none of the forwarding rules match, a 404 error is returned when you access the service.

Redirection Configuration (Public Application CLB ONLY)

Last updated：2018-08-21 16:12:18

LoadBalance team has launched the exclusive capability of **public network application-based LB** in April: custom redirect. This feature can solve two major problems:

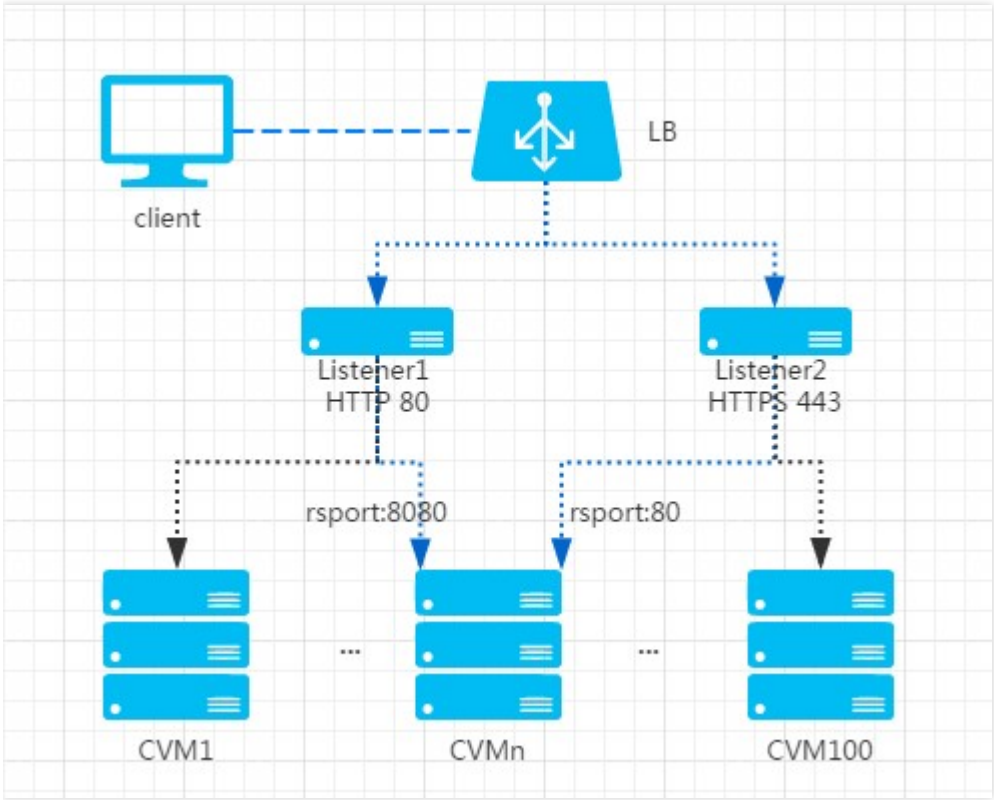
1. Forced HTTPS: With LoadBalance proxy, when Web service is accessed by browsers on PC and mobile via HTTP requests, the HTTPS respond is returned. By default, HTTPS is used forcibly by browsers to access web pages.
2. Custom redirect: When Web services need to be temporarily deactivated (in case of e-commerce products sold out, page maintenance, update and upgrade), redirect capability is required. Without redirect, visitors can only get a `404/503` error message caused by old address in users' favorites and search engine database, which affects user experience and results in access traffic loss. In addition, the search engine scores accumulated on this page are also wasted.

1. CLB Acts as a Proxy for HTTPS Requests

1.1 Traditional solution of nginx

(1) Solution description

Assume that a developer has purchased a load balancer bound with 100 backend CVMs to configure the website https://example.com. The developer wants users to directly access the website securely through HTTPS protocol by simply entering www.example.com in the browser. (That is, regardless of HTTP/HTTPS requests, HTTPS response is returned to force the use of encryption capability)
In this case, the request for accessing www.example.com entered by user is forwarded as follows:
The request is transmitted over HTTP protocol and accesses port 80 of the load balancer listener via VIP. Then, it is forwarded to port 8080 of the backend CVM.
By configuring rewrite operation on nginx of the Tencent Cloud backend CVM, the request passes through the port 8080 and is re-written to the https://example.com page.
The browser sends the https://example.com request to the corresponding HTTPS site again. The request accesses port 443 of the load balancer listener via VIP, and is forwarded to the port 80 of the backend CVM. Now, the request forwarding process is finished. The figure below shows the architecture:



(2) Detailed configuration

1. If the domain names of the HTTP and HTTPS services requested by a user are identical and the default HTTPS port is 443, the backend CVM can be configured as follows to achieve the above request forwarding operation:

```
server {
listen 80;
server_name example.com;

location / {
client_max_body_size 200m;
rewrite ^/.(.*) https://$host/$1 redirect; //Configure rewrite on CVM
}
}
```

2. If the domain names of the HTTP and HTTPS services requested by a user are different or the default HTTPS port is not 443, the user needs to specify URL and port and configure the backend CVM as follows to achieve the above request forwarding operation:

```
server {
listen 80;
server_name example.com;

location / {
client_max_body_size 200m;
rewrite ^/.(.*) https://xxx.xxx.xx:10011/x redirect; //Configure rewrite on CVM
}
}
```

(3) CLB acts as a proxy for HTTPS

In the above architecture, CLB mainly acts as a proxy for HTTPS. Both HTTP and HTTPS requests will become HTTP requests when forwarded to the backend CVM by CLB. Therefore, **if HTTPS protocol is used, the request is encrypted when transmitted from client to LB. However, the request is still transmitted as plaintext from LB to the backend CVM.** In this case, the developer cannot distinguish between HTTP and HTTPS requests.

To solve this problem, X-Client-Proto is placed into the header when Tencent CLB forwards the request to the backend CVM to help the developer determine the request type based on the header content:

- X-Client-Proto: http (frontend request is an HTTP request)
- X-Client-Proto: https (frontend request is an HTTPS request)

(4) Existing problems

- Complicated configuration: If you have multiple domain + uri and 100 backend CVMs, you need to repeat the configuration on 100 CVMs. And for each additional domain + uri, you need to refresh it on 100 backend CVMs.
- Computing cost: Determining whether redirect is required consumes CPU resources of backend CVMs.

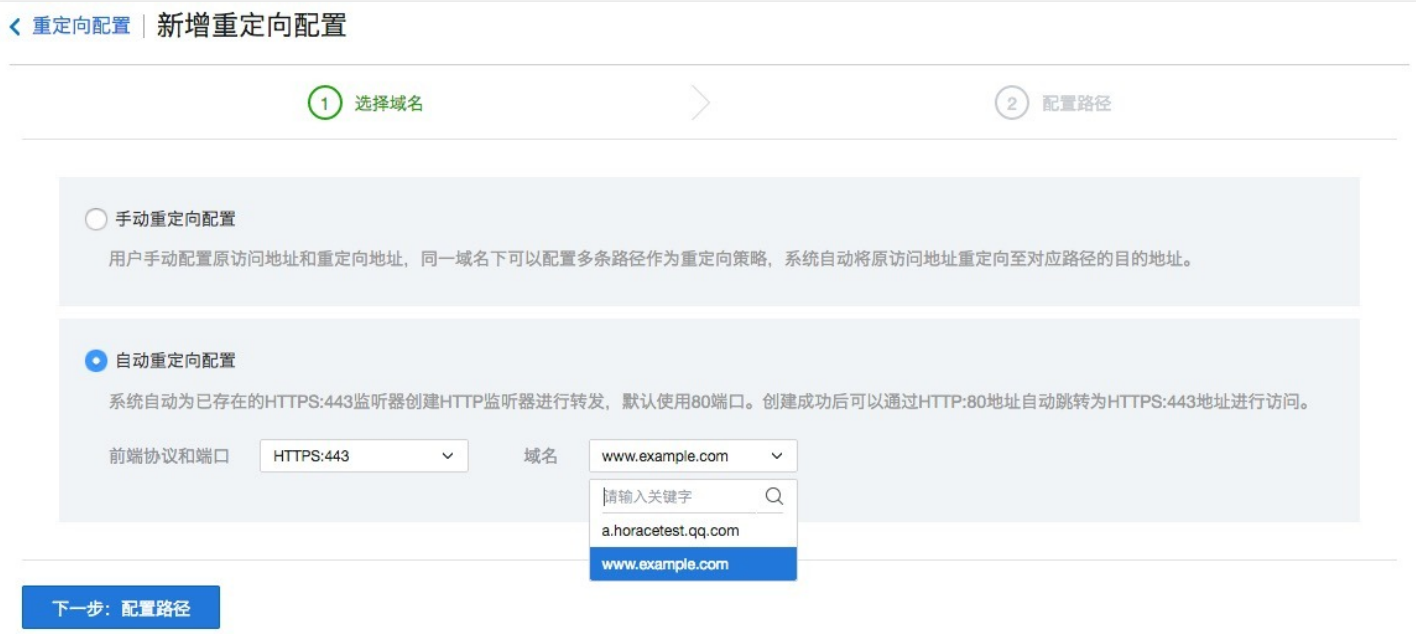
1.2 Forcibly redirect HTTP request from CLB

(1) Solution description

Assume that a developer needs to configure the website https://example.com. The developer wants users to directly access the website securely through HTTPS protocol by simply entering www.example.com in the browser. www.example.com is not merely an address, and hundreds of URLs may be associated at the backend (with regular expression matching), which leads to hundreds of real servers. Therefore, it is difficult to configure CVMs one by one. Tencent Cloud supports forcing HTTPS redirect with just one click.

First, configure the LB HTTPS listener in the Tencent CLB console to set up Web environment for https://example.com.

Second, enable redirect in the application-based CLB console. Overall redirect at the domain name level is supported.



(2) Solution advantages

- Configure once only: Forced HTTPS redirect can be implemented with one domain name and one-time configuration.
- Update: If the number of URLs of HTTPS service changes, you just need to use this feature again in the console for a refresh.

2. Notes

- Session persistence: If the client has accessed example.com/bbs/test/123.html and the session persistence is enabled for the backend CVM, when redirect is enabled and traffic flows into example.com/bbs/test/456.html, the original session persistence mechanism will become invalid.
- TCP/UDP redirect: Redirect at IP + Port level is not supported, but will be available in subsequent versions.