Cloud Load Balancer

CLB Listeners

Product Documentation

Contents

CLB Listeners Overview

Last updated : 2020-07-28 16:58:25

After a Cloud Load Balancer (CLB) instance is created, you must configure a listener for it. The listener listens to incoming requests to the CLB instance and then routes them to Cloud Virtual Machine (CVM) instances according to the configured load balancing policy.

You must configure the following items when configuring a listener for a CLB instance:

1. Listening protocol and listening port. A listening port, also known as a frontend port, is used to receive and route requests to CVM instances.
2. Listening policies, such as load balancing and session persistence policies.
3. Health check policies.
4. CVM instances. Specify the IP address and service port for each CVM instance. A service port, also known as a backend port, is used by a CVM instance to receive and process requests.

Supported protocol types

A CLB listener listens to incoming layer-4 and layer-7 requests to a CLB instance and routes them to CVM instances for processing. You can configure a layer-4 or layer-7 listener depending on whether you route requests for load balancing over a layer-4 or layer-7 protocol.

- Layer-4 protocols: transport layer protocols that receive and route requests to CVM instances by using a virtual IP (VIP) and port.
- Layer-7 protocols: application layer protocols that route requests based on application layer information such as the URL and HTTP header.

Tencent Cloud CLB supports request forwarding over the following protocols:

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

> **Note :**

- The TCP SSL listener feature is currently in beta test. To try it out, please submit a ticket for application.
- The TCP SSL listener feature is only available to public network CLB but not private network CLB or classic CLB.

Layer-4 listener

Protocol	Description	Scenario
TCP	Connection-oriented and reliable transport layer protocol • The source and destination ends must perform a three-way handshake to establish a connection before data can be sent and received between them. • Session persistence based on the client IP (source IP) is supported. • The client IP can be read at the network layer. • The server can directly obtain the client IP.	TCP is suitable for scenarios where high transfer reliability and data accuracy are required with a slight compromise on transfer speed. Typical scenarios include file transfer, sending and receiving emails, and remote logins. For more information, please see Configuring a TCP Listener.
UDP	Connectionless transport layer protocol • The source and destination ends do not establish a connection nor maintain the connection status. • Each UDP connection is point-to-point. • One-to-one, one-to-many, many-to-one, and many-to-many communication are supported. • Session persistence based on the client IP (source IP) is supported. • The server can directly obtain the client IP.	UDP is suitable for scenarios where a high transfer speed is preferred over accuracy. Typical scenarios include instant messaging and online videos. For more information, please see Configuring a UDP Listener.

Protocol	Description	Scenario
TCP SSL	Secure TCP protocol • TCP SSL listeners support configuring certificates to prevent unauthorized access requests. • Unified certificate management is provided for CLB to decrypt certificates. • One-way and mutual authentication are supported. • The server can directly obtain the client IP.	TCP SSL is suitable for scenarios where high security is required for TCP and TCP-based custom protocols are supported. For more information, please see Configuring a TCP SSL Listener.

If you configure a layer-4 listener, the CLB instance establishes a TCP connection with each CVM instance on the listening port and routes requests to CVM instances. During this process, the CLB instance forwards data in passthrough mode in an efficient manner without modifying any data packets.

Layer-7 listener

Protocol	Description	Scenario
HTTP	Application layer protocol • Forwarding based on the requested domain name and URL is supported. • Cookie-based session persistence is supported.	HTTP is suitable for apps that need to identify request content, such as web apps and app services. For more information, please see Configuring an HTTP Listener.
HTTPS	Encrypted application layer protocol • Forwarding based on the requested domain name and URL is supported. • Cookie-based session persistence is supported. • Unified certificate management is provided for CLB to decrypt certificates. • One-way and mutual authentication are supported.	HTTPS is suitable for HTTP apps that require encrypted transmission. For more information, please see Configuring an HTTPS Listener.

Port configuration

Listening port (frontend port)	Service port (backend port)	Description
Through the listening port, a CLB instance receives and routes requests to CVM instances for processing. You can configure a listening port on ports 1–65535, such as port 21 (FTP), port 25 (SMTP), port 80 (HTTP), and port 443 (HTTPS).	Through the service port, a CVM instance receives and processes requests from a CLB instance. On a CLB instance, one listening port can route requests to multiple ports of multiple CVM instances.	Note the following points when configuring a listening port on a CLB instance: • You can configure the same listening port for TCP and UDP. For example, listeners `TCP:80` and `UDP:80` can co-exist. • You cannot configure the same listening port for protocols of the same type. For example, TCP, TCP SSL, HTTP, and HTTPS are all TCP protocols. You cannot configure listeners `TCP:80` and `HTTP:80` at the same time. On a CLB instance, you can configure the same service port for different CVM instances. You can also bind different listeners, for example, `HTTP:80` and `HTTPS:443` to the same port of a CVM instance.

Load Balancing Methods

Last updated : 2020-11-13 18:03:24

A load balancing method is an algorithm that allocates traffic to real servers. Each method produces different load balancing effects.

Weighted Round-Robin Scheduling

The weighted round-robin scheduling algorithm is to schedule requests to different servers based on polling. It can solve problems with imbalanced performance of different servers. It uses weight to represent the processing performance of a server and schedules requests to different servers by weight in a polling manner. It schedules servers based on the number of new connections, where servers with a higher weight receive connections earlier and have a higher chance to be polled. Servers with the same weight will process the same number of connections.

- **Advantage**: this algorithm features simplicity and high practicability. It does not need to record the status of all connections and is therefore a stateless scheduling algorithm.
- **Disadvantage**: this algorithm is relatively simple, so it is unsuitable for situations where the service time of a request changes significantly, or each request needs to consume different amounts of time. In these cases, it will cause imbalanced load distribution among servers.
- **Applicable scenario**: 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.
- **Recommendation**: 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 type or similar types), you are recommended to use weighted round-robin scheduling. If the time difference between each request is small, this algorithm is also recommended as it has low consumption and high efficiency with no need of traversal.

Weighted Least-Connection Scheduling

In actual situations, the time requests from the client spend staying on the server may vary greatly. As the working time gets longer, if a simple round-robin or random load balancing algorithm is used, the number of connection processes on each server may vary hugely, which cannot achieve load balancing effect.
Contrary to round-robin scheduling, least-connection scheduling is a dynamic scheduling algorithm

that estimates the load of a server by its active connection quantity. The scheduler needs to record the number of current established connections on each server. If a request is scheduled to a server, the number of connections will be increased by 1. If a connection stops or times out, the number of connections will be decreased by 1.
In the weighted least-connection scheduling algorithm that is based on least-connection scheduling, different weights are allocated to servers according to their processing capability. In this way, a server can receive a corresponding number of requests according to its weight, which is an improvement on least-connection scheduling.

> Suppose that the weight of a real server is wi, and the current number of connections is ci. The ci/wi values of each server are calculated in sequence. The real server with the smallest ci/wi value will be the next server that receives a new request. If there are real servers with the same ci/wi value, they will be scheduled based on weighted round-robin scheduling.

- **Advantage**: this algorithm is suitable for requests requiring long-time processing, such as FTP.
- **Disadvantage**: due to API restrictions, least-connection and session persistence cannot be enabled at the same time.
- **Applicable scenario**: this algorithm is suitable for scenarios where the time used by each request on the backend varies greatly. It is usually used in persistent connection services.
- **Recommendation**: if you need to process different requests and the service time needed by them on the backend varies greatly (such as 3 milliseconds and 3 seconds), you are recommended to use weighted least-connection scheduling to achieve load balancing.

Source Hashing Scheduling

The source hashing scheduling algorithm (ip_hash) 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.

- **Advantage**: ip_hash can map requests from a client to the same real server through the hash table. Therefore, in scenarios where session persistence is not supported, it can be used to achieve simple session persistence effect.
- **Recommendation**: this algorithm calculates the hash value of the source address of a request and distributes the request to the matched real server based on its weight. In this way, all requests from the same client IP can be distributed to the same server. This algorithm is suitable for the protocols that do not support cookie.

Choosing Load Balancing Algorithm and Configuring Weight

In order to allow real server clusters to undertake business in a stable manner in different scenarios, some cases regarding how to choose the load balancing algorithm and configure weight are provided below for your reference.

- Scenario 1:
  i. Suppose that there are 3 real servers with the same configuration (CPU and memory) and you set all their weights to 10 as they have the same performance.
  ii. 100 TCP connections have been established between each real server and the client, and a new real server is added.
  iii. In this scenario, you are recommended to use the least-connection scheduling algorithm, which can quickly increase the load of the 4th real server and reduce the pressure on the other 3 ones.
- Scenario 2:
  i. Suppose that you use Tencent Cloud services for the first time and your website was just built with low load. You are recommended to purchase real servers of the same configuration since they are all equivalent access-layer servers.
  ii. In this scenario, you can set the weights of all real servers to the default value of 10 and use the weighted round-robin scheduling algorithm to distribute the traffic.
- Scenario 3:
  i. Suppose that you have 5 real servers that undertake simple access requests to static pages, and the ratio of computing power (calculated by CPU and memory) of these servers is 9:3:3:3:1.
  ii. In this scenario, you can set the weight of the real servers to 90, 30, 30, 30, and 10, respectively. As most access requests to static web pages are of non-persistent connection type, you can use the weighted round-robin scheduling algorithm, so that the CLB instance can allocate requests based on the servers' performance ratio.
- Scenario 4:
  i. Suppose that you have 10 real servers to undertake massive amounts of web access requests and do not want to purchase more servers as that will increase the expenditure, and one of the servers often restarts due to overload.
  ii. In this scenario, you are recommended to set the weights of existing servers based on their performance and set a relatively small weight to servers with high load. In addition, you can use the least-connection scheduling algorithm to allocate requests to real servers with fewer active connections so as to avoid server overload.
- Scenario 5:

i. Suppose that you have 3 real servers for processing some persistent connections, the ratio of computing power (calculated by CPU and memory) of these servers is 3:1:1.
ii. The server with the best performance processes more requests, but you do not want it to be overloaded and want to allocate new requests to idle servers.
iii. In this scenario, you can use the least-connection scheduling algorithm and appropriately reduce the weight of the busy server, so that the CLB instance can allocate requests to real servers with fewer active connections, thereby achieving load balancing.

- Scenario 6:
  i. Suppose that you want subsequent requests from the client to be allocated to the same server. As weighted round-robin or weighted least-connection scheduling cannot ensure that requests from the same client are allocated to the same server,
  ii. To satisfy the requirements of your specific application server and maintain the "stickiness" (or "continuity") of the client sessions, you can use ip_hash to distribute the traffic. This algorithm can ensure 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.

Session Persistence

Last updated : 2020-05-22 14:13:11

Session persistence can forward requests from the same IP to the same real server. By default, a CLB instance will route requests to different real servers for load balancing; however, you can use session persistence to route requests from a specified user to the same real server, so that some applications that need to hold their session (such as shopping cart) can run properly.

Layer-4 Session Persistence

Layer-4 protocols (TCP/UDP) supports source IP-based session persistence. The session persistence duration can be set to any integer between 30 and 3600 seconds. If the time threshold is exceeded and the session has no new request, the connection will be disconnected. Session persistence is subject to the load balancing mode:

- In the mode of "weighted round-robin" where requests are distributed based on the weight of real servers, session persistence based on source IP is supported.
- In the mode of "weighted least-connection" where overall scheduling depends on server load and weight, session persistence is not supported.

Layer-7 Session Persistence

Layer-7 protocols (HTTP/HTTPS) supports session persistence based on cookie insertion (CLB inserts the cookie into the client). The session persistence duration can be set to any value between 30 and 3600 seconds. Session persistence is subject to the load balancing mode:

- In the mode of "weighted round-robin" where requests are distributed based on the weight of real servers, session persistence based on cookie insertion is supported.
- In the mode of "weighted least-connection" where overall scheduling depends on server load and weight, session persistence is not supported.
- The mode of "IP Hash" supports session persistence based on source IP but not on cookie insertion.

Connection Timeout Period

Currently, HTTP connection timeout period (`keepalive_timeout`) is 75s by default. If you want to adjust it, please enable custom configuration. If the threshold is exceeded and the session has no data transmission, the connection will be disconnected.
Currently, TCP connection timeout period is 900s by default and cannot be customized. If the threshold is exceeded and the session has no data transmission, the connection will be disconnected.

Configuring Session Persistence

1. Log in to the CLB Console and click the ID of the CLB instance to be configured with session persistence to enter its details page.
2. Select the **Listener Management** tab.
3. Click **Modify** after the CLB listener to be configured with session persistence.
4. Choose whether to enable the session persistence feature. Click the button to enable it, enter the persistence duration, and click **OK**.

Relationship Between Persistent Connection and Session Persistence

Scenario 1. HTTP layer-7 business

Assume a client accesses HTTP/1.1 protocol and `Connection:keep-alive` is configured in the header information. The client accesses CVM via a CLB instance without session persistence enabled. Can the client access the same CVM next time?

A: no.

First, HTTP keep-alive indicates TCP connection remains connected after a request is sent, so the browser can send requests via the same connection. Persistent connection reduces the time required for establishing a new connection for each request and lowers bandwidth consumption. The default timeout period of a CLB cluster is 75s (if there is no new request within 75s, TCP will be disconnected by default).

HTTP keep-alive is established between the client and a CLB instance. If cookie session persistence is disabled, the CLB instance will randomly select a CVM instance according to the polling policy. The previous persistent connection is no longer valid.

Therefore, we recommend you enable session persistence.

If the cookie session persistence period is configured as 1000s, the client will initiate a request again. Because the period between the two requests exceeds 75s, TCP connection needs to be established again. The application layer identifies the cookie and finds the CVM instance the client accessed last time so it will be assessed again this time.

Scenario 2. TCP layer-4 business

Assume a client initiates access, TCP is the transport-layer protocol, persistent connection is enabled, but session persistence based on source IP is disabled. Can the same client access the same server in the next access request?

A: not necessarily.

First, according to layer-4 implementation mechanism, when persistent connection is enabled for TCP and not closed, and the same connection is accessed in two requests, then the same client can access the same server. If the connection is closed for some reasons (such as network restart or connection timeout) during the second access request, the request may be scheduled to another real server. The default global timeout period for a persistent connection is 900s, that is, the persistent connection will be released if there is no new request in 900s.

Health Check

Last updated : 2020-11-13 14:47:37

A CLB instance can periodically send `Ping` commands to real servers, attempt to connect with or send requests to them to check their running status. These are called "health check". It uses health checks to determine the availability of real servers and prevent exceptional real servers from affecting frontend services, thereby improving overall service availability.

- If a real server is confirmed as exceptional, the CLB instance will automatically distribute new requests to other normal servers instead of forwarding them to the exceptional server; after the exceptional server returns to the normal state, the CLB instance will automatically restore its service and distribute new requests to it again.
- After health check is enabled, regardless of the weights of real servers (including 0), the CLB instance will always perform health check.
- An auto scaling group will periodically use a similar method to check the running status of each instance in the group. For more information, please see Auto Scaling.

Health Check Configuration for Layer-4 Forwarding

The health check mechanism for layer-4 forwarding is as follows: CLB initiates an access request to the server port specified in the configuration. If access to the port is normal, the real server is considered healthy. Otherwise, it is considered unhealthy.
For TCP business, SYN packets are used for health check. For UDP business, the ping command is used.

Health Check Configuration	Description	Default Value
Response timeout period	• Maximum response timeout period for health check. • If a real server fails to respond properly within the timeout period, it is considered as having an exception. • Value range: 2–60s.	2s
Check interval	• Interval between two health checks. • Value range: 5–300s.	5s
Unhealthy threshold	• If the health check results received are failures for n times (n is the entered number) in a row, the real server will be considered as unhealthy, and the status displayed in the console will be **unhealthy**.	3 times

	• Value range: 2–10.	
Healthy threshold	• If the health check results received are successes for n times (n is the entered number) in a row, the real server will be considered as healthy, and the status displayed in the console will be **healthy**. • Value range: 2–10.	3 times

Health check configuration for Layer-7 forwarding

The health check mechanism for layer-7 forwarding is as follows: CLB sends an HTTP request to the real server for health check. CLB determines whether the server is healthy based on the HTTP return value. For example, assume HTTP return values include `http_1xx`, `http_2xx`, `http_3xx`, `http_4xx` and `http_5xx`, users can define `http_1xx` and `http_2xx` as normal status based on their business needs, and configure `http_3xx` to `http_5xx` as unhealthy status.

Health Check Configuration	Description	Default Value
Check domain name	Health check domain name: • It can contain 1–80 characters. • It is the forwarded domain name by default. • Regex is not supported. If your forwarded domain name is a wildcard domain name, you should specify a fixed one (non-regex) as the health check domain name. • Supported characters: `a-z`, `0-9`, `.`, `-`.	Forwarded domain name
Check path	Health check path: • It can contain 1–200 characters. • It is `/` by default and must begin with `/`. • Regex is not supported. You are recommended to specify a fixed URL path (static page) for health checks. • Supported characters: `a-z`, `A-Z`, `0-9`, `.`, `-`, `_`, `/`, `=`, `?`.	`/`
Check interval	• Interval between two health checks. • Value range: 5–300s.	5s
Unhealthy threshold	• If the health check results received are failures for n times (n is the entered number) in a row, the real server will be considered as unhealthy, and the status displayed in the console will be **unhealthy**. • Value range: 2–10.	3 times
Healthy	• If the health check results received are successes for n times	3 times

threshold	(n is the entered number) in a row, the real server will be considered as healthy, and the status displayed in the console will be **healthy**. • Value range: 2–10.	
HTTP request method	HTTP request method for health checks. Valid values: GET, HEAD. • If HEAD is used, the server will only return the HTTP header, which can reduce backend overheads and improve request efficiency. The corresponding real server must support HEAD. • If GET is used, the real server must support GET.	GET
HTTP status code check	If the status code is the selected one, the real server is considered alive (healthy). Valid values: http_1xx, http_2xx, http_3xx, http_4xx, http_5xx.	http_1xx, http_2xx, http_3xx, http_4xx

Health Check Status

According to the health check detection conditions, the health check status of a real server may be one of the following four options:

Status	Description	Whether to Forward Traffic
Detecting	This is the state of the newly bound real server during the period of check interval * healthy threshold; for example, if the check interval is 2 seconds and the health threshold is 3 times, this will be state within 6 seconds.	CLB does not forward traffic to the real server in "detecting" status.
Healthy	The real server is normal.	CLB forwards traffic to the "healthy" real server.
Exceptional	The real server is exceptional.	• CLB does not forward traffic to "exceptional" real servers. • Under a layer-4 listener or layer-7 URL rule, if CLB detects that all real servers are unhealthy, it will activate the all-dead-all-alive logic, that is, requests will be forwarded to all real servers.
Disabled	Health check has been disabled.	CLB forwards traffic to the real server.

> **Note :**
>
> If you disable health check, CLB will forward traffic to all real servers (including exceptional ones). Therefore, we strongly recommend you enable health check to allow CLB to automatically check for and remove exceptional real servers for you.

How to Troubleshoot Health Check Issues

If an exception is detected during health check, it can be checked from various aspects such as real server bandwidth, layer-4 listener, and layer-7 protocol. For specific troubleshooting directions, please see Troubleshooting Health Check Issues

Configuring an HTTP Listener

Last updated : 2019-11-05 15:21:01

HTTP Listener Overview

You can create an HTTP listener to a CLB instance to forward HTTP requests from the client. HTTP is suitable for applications where request contents need to be identified, such as web applications and mobile apps.

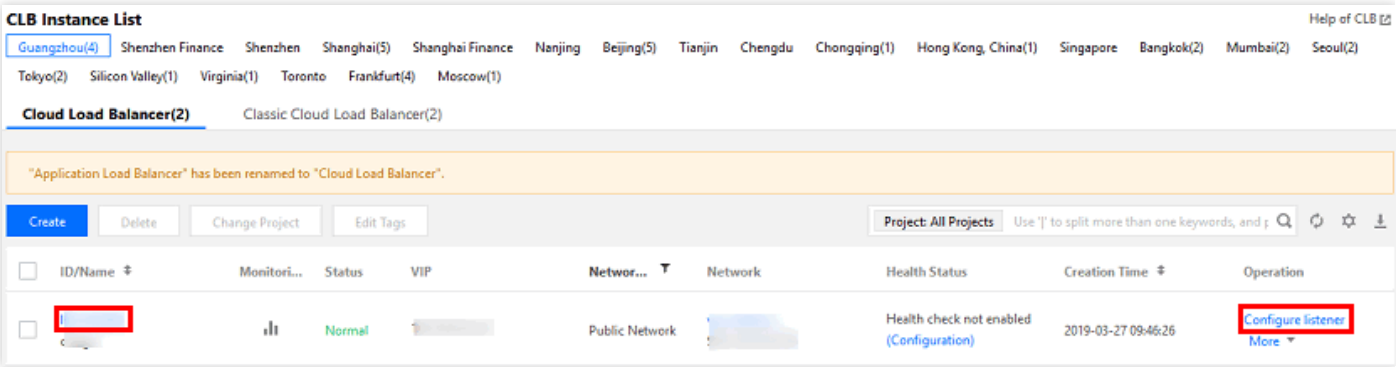
Prerequisites

You need to create a CLB instance first.

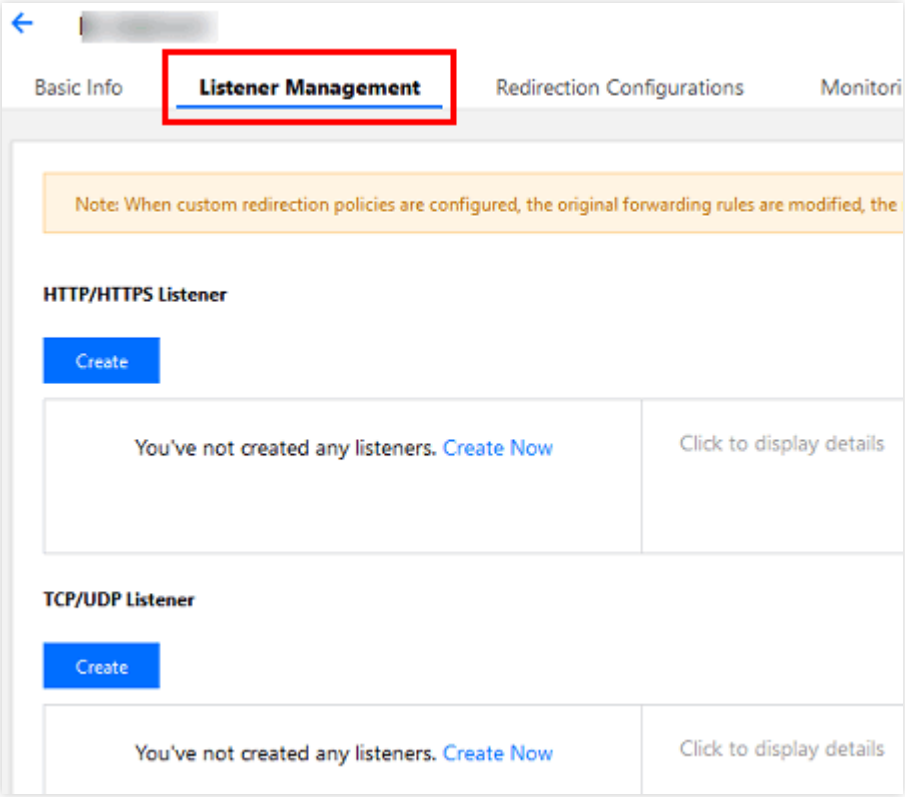
Configuring an HTTP Listener

Step 1. Open the "Listener Management" page

1. Log in to the CLB Console.
2. Select **Instance Management** on the left sidebar.
3. In the instance list, click the ID of the instance to be configured to enter the instance details page.
4. Click the **Listener Management** tab or click **Configure Listener** in the "Operation" column.



5. The "Listener Management" page is as shown below:



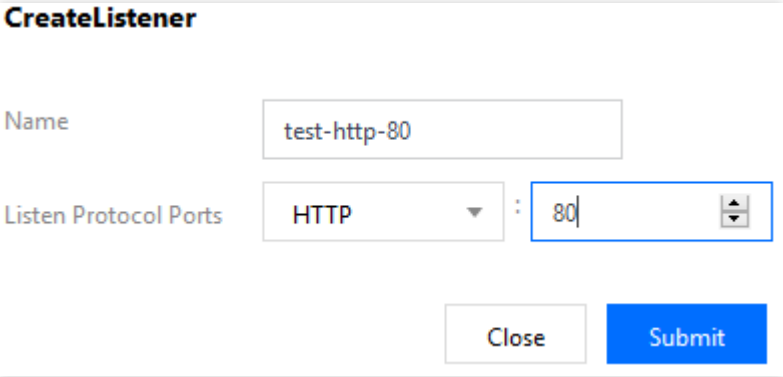
Step 2. Configure a listener

Click **Create** in **HTTP/HTTPS Listener** and configure an HTTP listener in the pop-up window.

1. Create a listener

Configuration Item	Description	Example
Name	Listener name	test-http-80
Listener protocol and listening port	• Listener protocol: CLB supports various protocols, including TCP, UDP, TCP SSL, HTTP, and HTTPS. HTTP is used in this example. • Listening port: A port used to receive requests and forward them to the real server. Port range: 1-65535. • The listening port must be unique in the same CLB instance.	HTTP:80

The specific configuration of the created HTTP listener is as shown below:



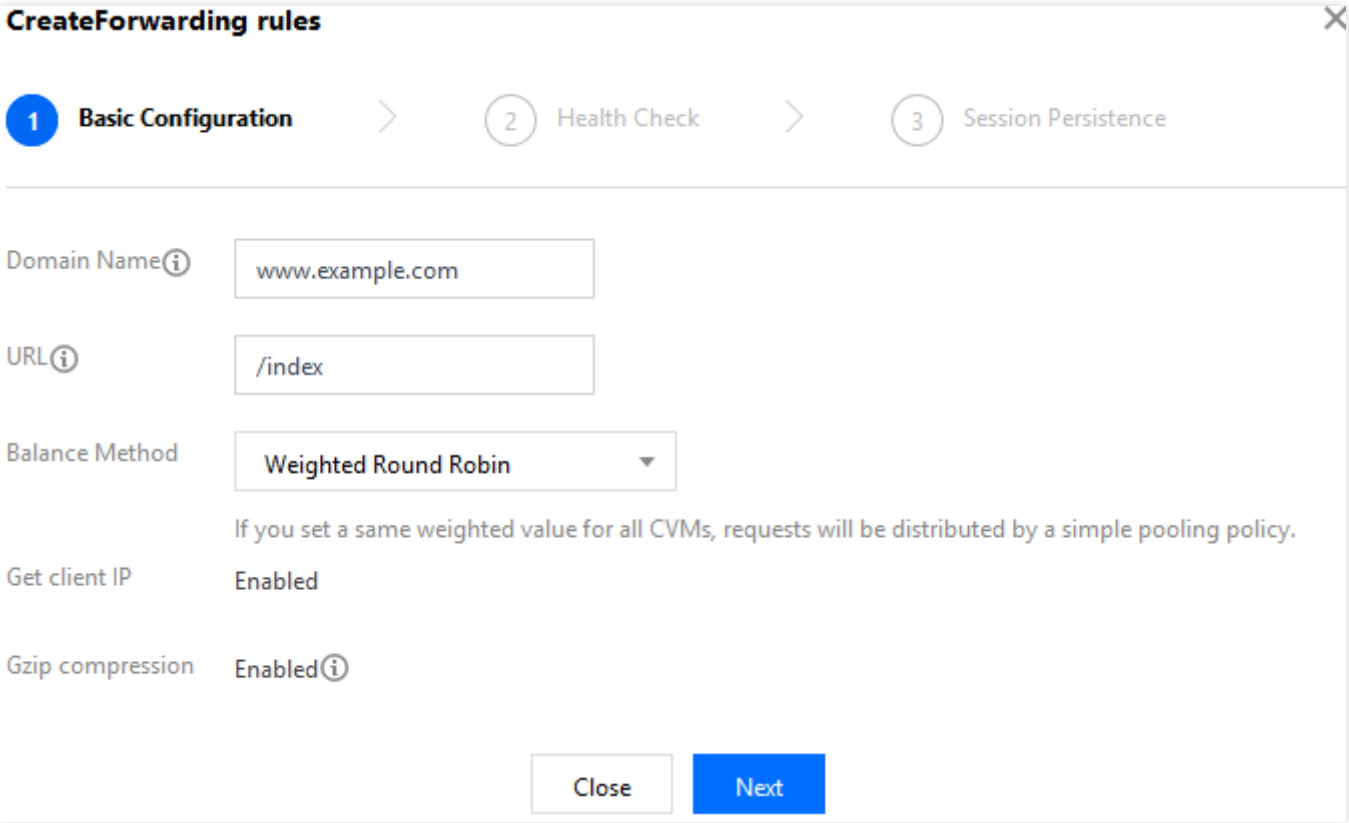
2. Create a forwarding rule

Configuration Item	Description	Example
Domain name	Request domain name. • Exact domain names are supported, such as `www.example.com` . Wildcard domain names are also supported, such as `*.example.com` and `www.example.*` , where `*` can appear only once in a single domain name. • A non-regular domain name can contain the following characters: `a-z` , `0-9` , `.` , and `-` . • Regex is supported, but they cannot contain the following characters: `"` , `{` , `}` , `;` , `¥` , ```, `~` , `'` , and `space` . • A domain name can contain 1-120 characters.	www.example.com

Configuration Item	Description	Example
URL path	Request path. • The path is in the format of `/` by default, must begin with `/`, and can contain 1-120 characters. • A non-regular URL path must begin with `/` and can contain the following characters: `a-z`, `A-Z`, `0-9`, `.`, `-`, `/`, `=`, and `?`. • Regex is supported. 1. Beginning with `=` indicates exact match. 2. Beginning with `^~` indicates that the URI begins with a general string and is not regex match. 3. Beginning with `~` indicates case-sensitive regex match. 4. Beginning with `~*` indicates non-case-sensitive regex match. 5. `/` indicates generic match, where any requests will be matched if there are no other matches. 6. A regular URL cannot contain the following characters: `”`, `{`, `}`, `;`, `¥`, ```, `~`, `’`, and `space`.	/index
Balancing method	For HTTP listeners, CLB supports three scheduling algorithms: weighted round robin (WRR), weighted least connections (WLC), and IP hash. • WRR: Requests are sequentially delivered to different real servers according to their weights. Scheduling is done based on the **number of new connections**, where servers with higher weights will undergo more polls (i.e., a higher probability), while servers with the same weight process the same number of connections. • WLC: Loads of servers are estimated according to the number of active connections to the servers. Scheduling is done based on server loads and weights. If their weights are the same, servers with fewer active connections will undergo more polls (i.e., a higher probability). • IP hash: Hash keys are used to locate the corresponding servers in the static hash table based on the source IPs of requests. If a server is available and not overloaded, requests will be delivered to it; otherwise, a null value will be returned.	WRR

Configuration Item	Description	Example
Getting client IP	Enabled by default	Enabled
Gzip compression	Enabled by default	Enabled

Select the HTTP listener for which to create a forwarding rule and click **+** on the right. The specific configuration is as shown below:



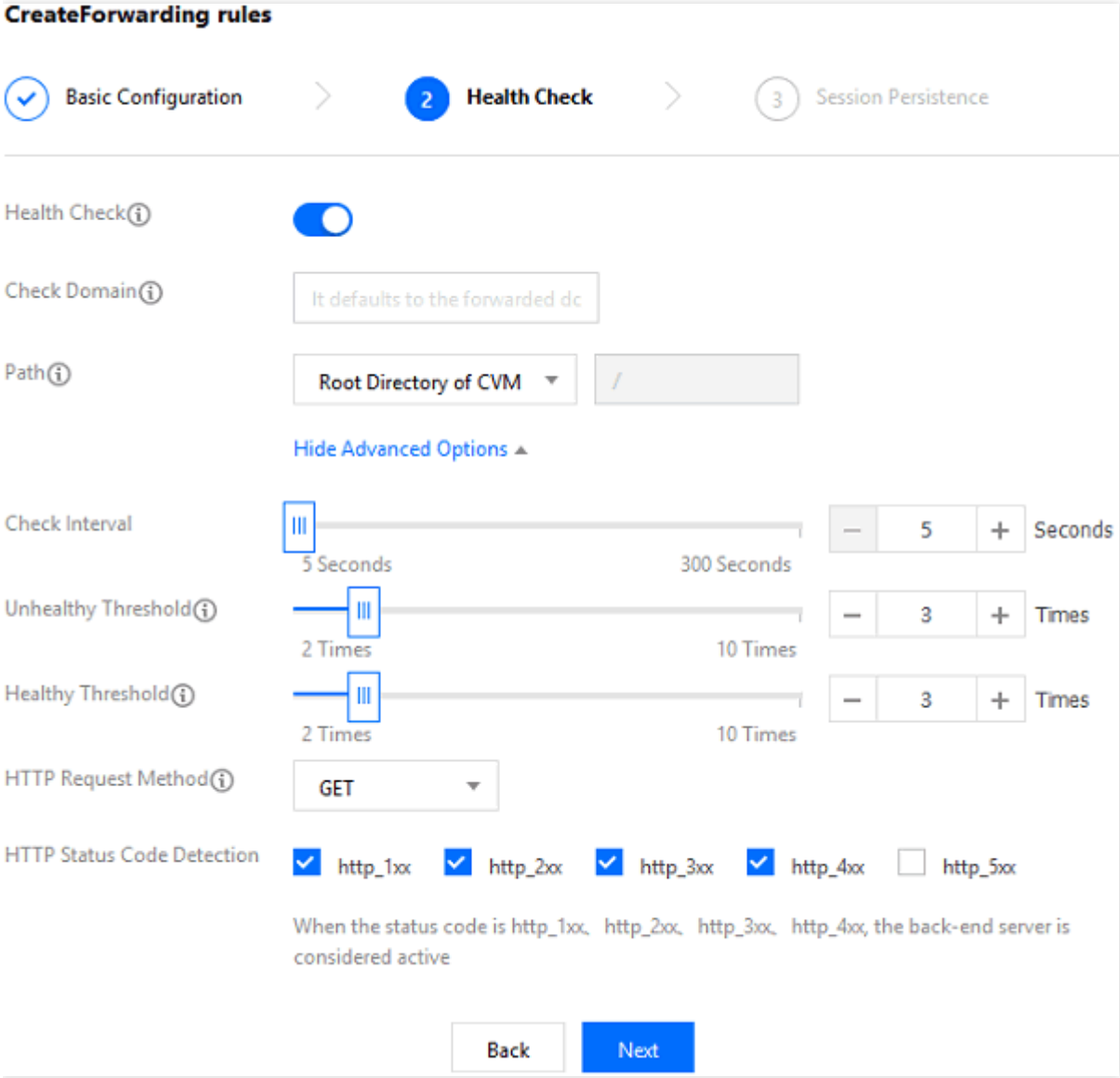
3. Health check

Configuration Item	Description	Example
Health check status	Health check can be enabled or disabled. In HTTP listeners, CLB instances send HTTP requests to the specified server port to perform health checks.	Enabled

Configuration Item	Description	Example
Check domain name	Request domain name. • The forwarding domain name is used for this parameter by default. • It can contain 1-120 supported characters: `a-z` , `0-9` , `.` , and `-` . • Regex is not supported currently. • If a wildcard domain name is entered, a fixed domain name (non-regular) should be specified as the health check domain name.	Default value (i.e., `www.example.com`)
Check path	Request path. • The path is in the format of `/` by default and must begin with `/` . • It can contain 1-120 supported characters: `a-z` , `0-9` , `.` , and `-` . • Regex is not supported currently. • It is recommended to specify a fixed URL path (static page) for health checks.	Default value (i.e., `/`)
Check interval	• Interval between two health checks. • Value range: 5-300s. Default value: 5s.	5s
Unhealthy threshold	• If the health check results received n times (n is the entered number) in a row are failures, the instance will be considered unhealthy, and the status displayed in the console will be **Abnormal**. • Value range: 2-10. Default value: 3.	3 times
Healthy threshold	• If the health check results received n times (n is the entered number) in a row are successes, the instance will be considered healthy, and the status displayed in the console will be **Healthy**. • Value range: 2-10. Default value: 3.	3 times
HTTP request method	HTTP request method for health checks. Value range: GET, HEAD. Default value: GET. • If HEAD is used, the server will only return the HTTP header, which can reduce backend overheads and improve request efficiency; the corresponding real server needs to support HEAD. • If GET is used, the real server needs to support GET.	GET

Configuration Item	Description	Example
HTTP status code check	If the status code is the selected one, the real server is considered alive (healthy). Value range: http_1xx, http_2xx, http_3xx, http_4xx, http_5xx.	Multiple selections: http_1xx, http_2xx, http_3xx, http_4xx

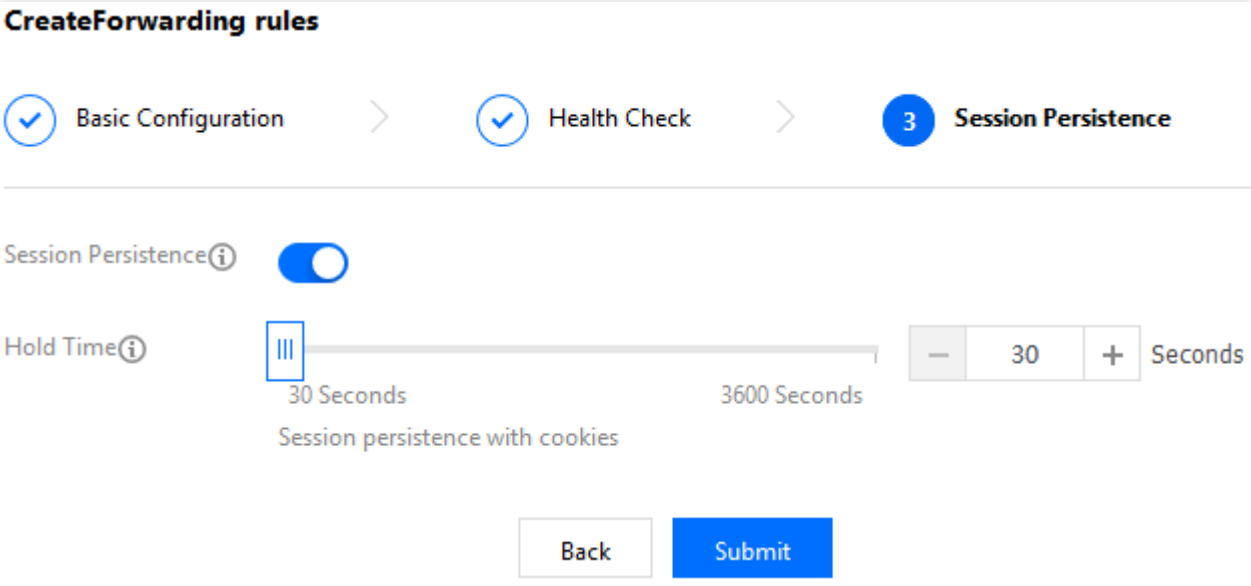
The specific configuration of health check is as shown below:



4. Session persistence

Configuration Item	Description	Example
Session persistence status	Session persistence can be enabled or disabled. • If session persistence is enabled, the CLB listener will deliver access requests from the same client to the same real server. • HTTP session persistence is implemented based on cookies, which are implanted into the client by the CLB instance. • Session persistence can be enabled for WRR scheduling but not WLC or IP hash scheduling.	Enabled
Session persistence time	Session persistence time. • If there is no new request in the connection within the session persistence time, session persistence will be interrupted automatically. • Value range: 30-3,600s.	30s

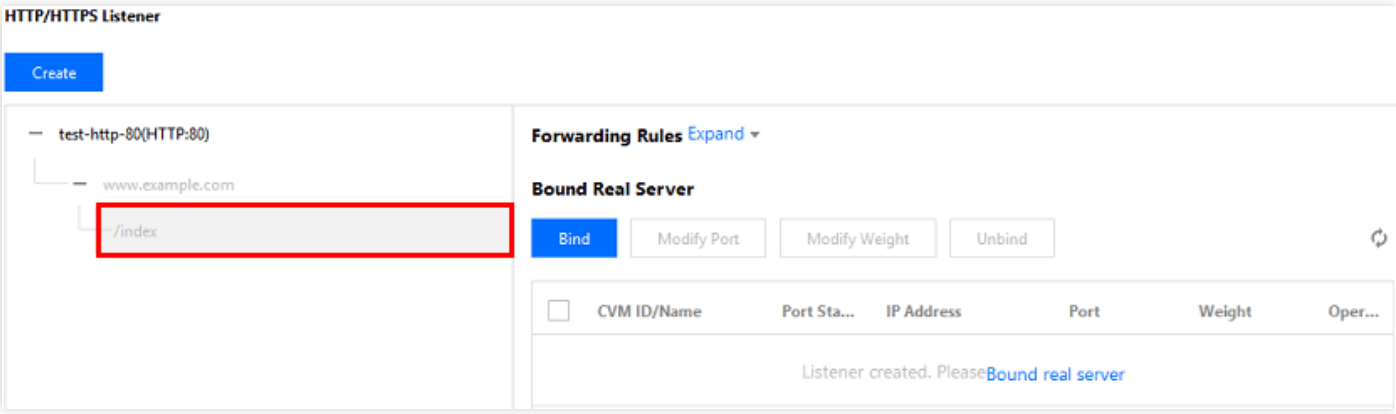
The specific configuration of session persistence is as shown below:



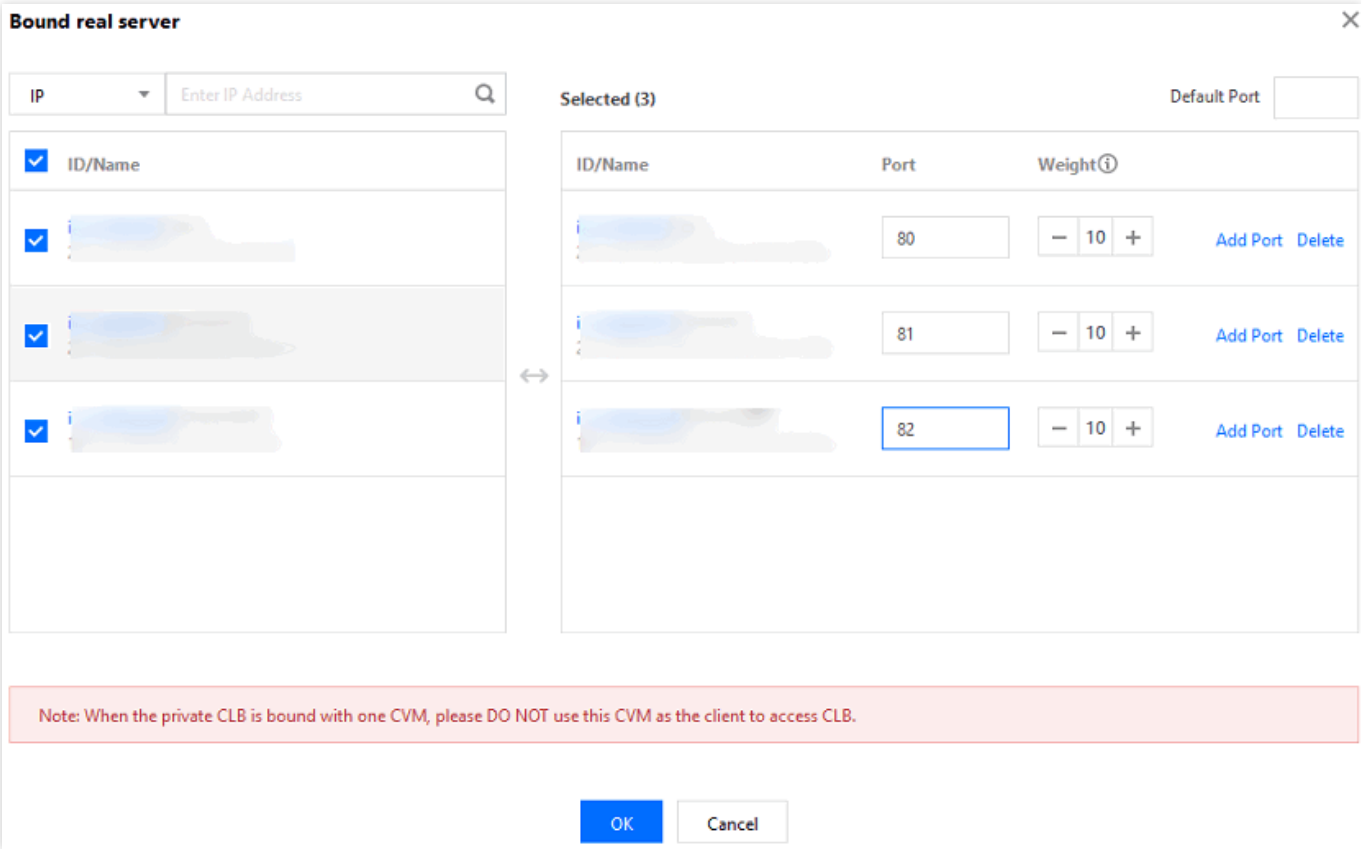
Step 3. Bind a real server

1. On the "Listener Management" page, select the created listener `HTTP:80` . Click + on the left to expand the domain names and URL paths, select the desired URL path, and view the real servers

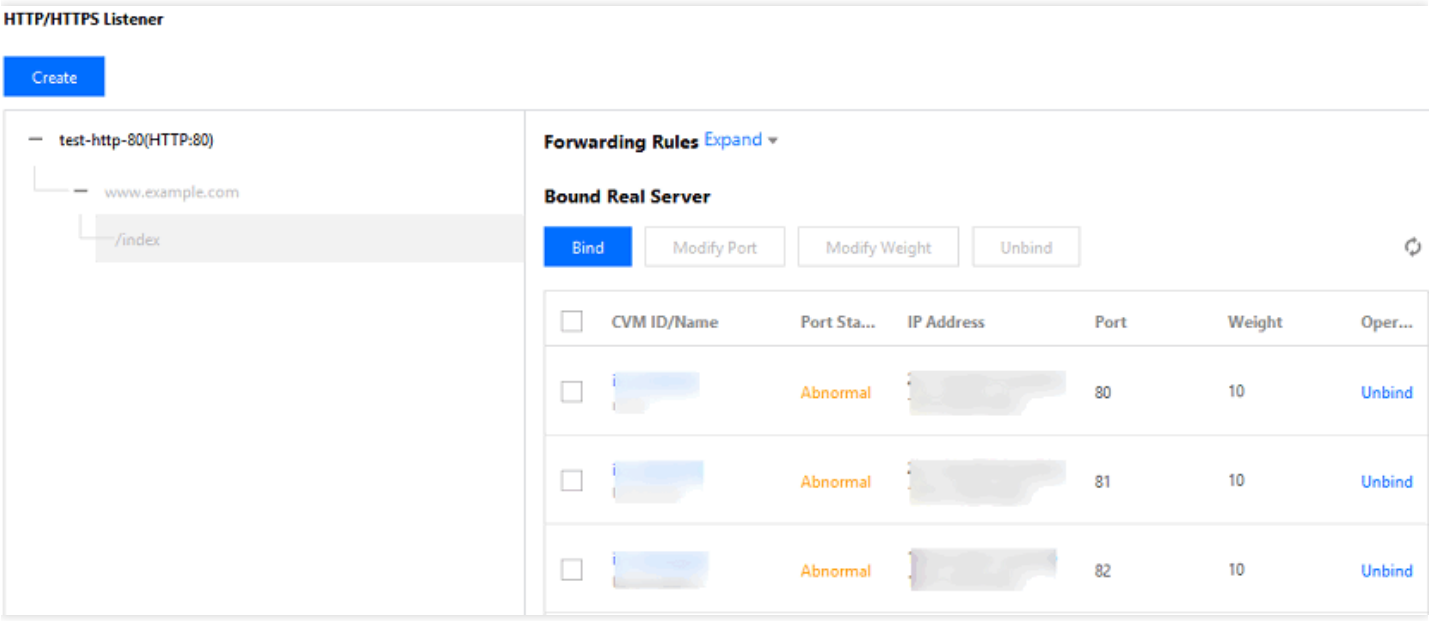
bound to the path on the right of the listener.



2. Click **Bind** and select the real server to be bound and configure the server port and weight in the pop-up window.
   i. Add Port: In the "Selected" box on the right, click **Add Port** to add multiple ports for the same CVM instance, such as ports 80, 81, and 82.
   ii. Default Port: Enter the "Default Port" first and then select the CVM instance. The port of every CVM instance is the default port.



After these three steps are completed, the HTTP listener rule has been configured as shown below:

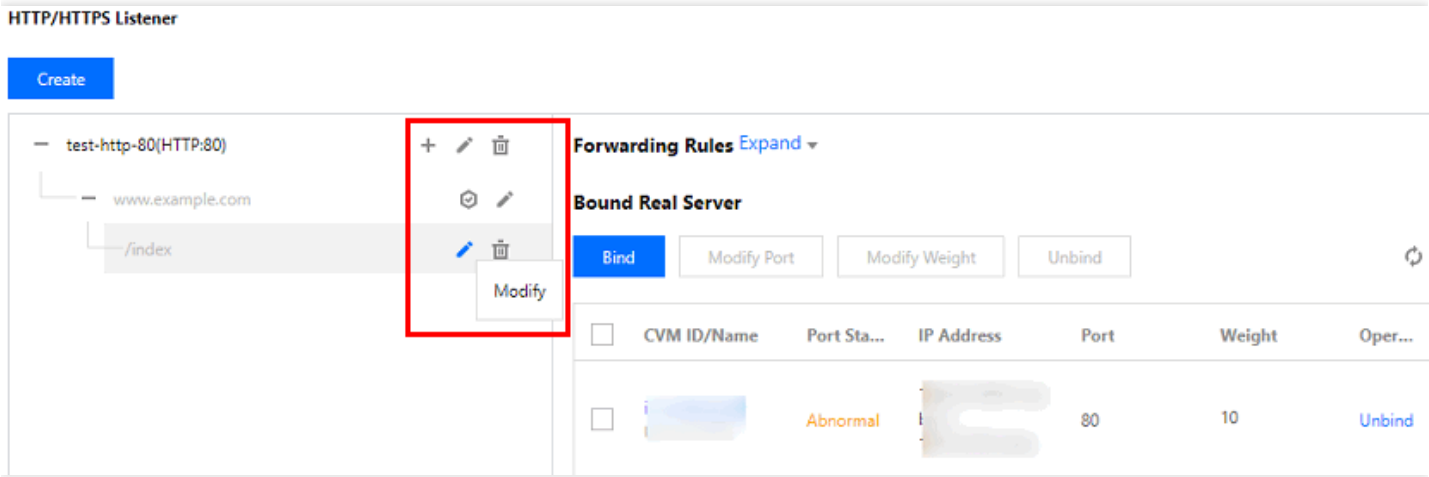


Step 4. Security group (optional)

You can configure a CLB security group to isolate public network traffic. For more information, see Configuring a CLB Security Group.

Step 5. Modify/delete a listener (optional)

If you need to modify or delete a created listener, click the listener/domain name/URL path on the "Listener Management" page and select **Modify** or **Delete**.



Configuring an HTTPS Listener

Last updated : 2019-11-05 15:21:59

HTTPS Listener Overview

You can create an HTTPS listener to a CLB instance to forward HTTPS requests from the client. HTTPS is suitable for HTTP applications where data transfer needs to be encrypted.

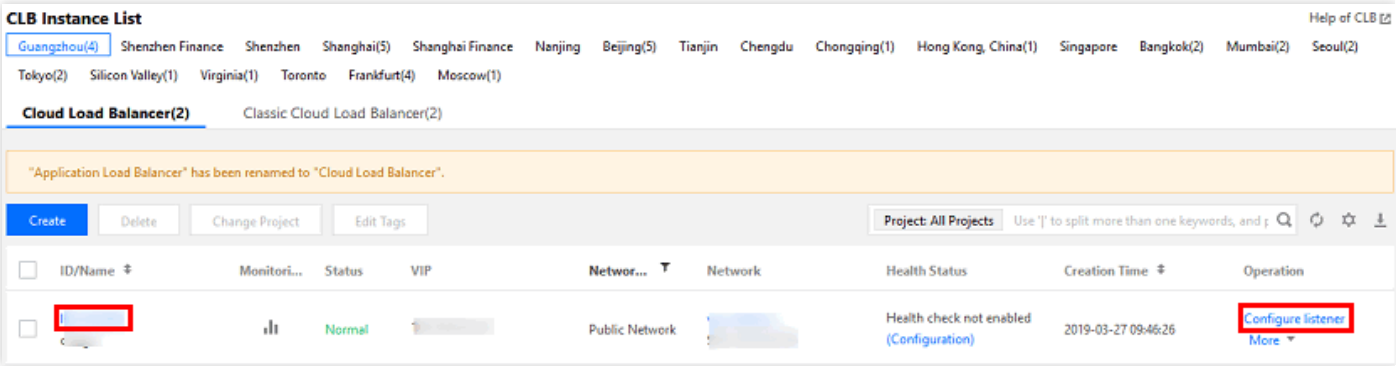
Prerequisites

You need to create a CLB instance first.

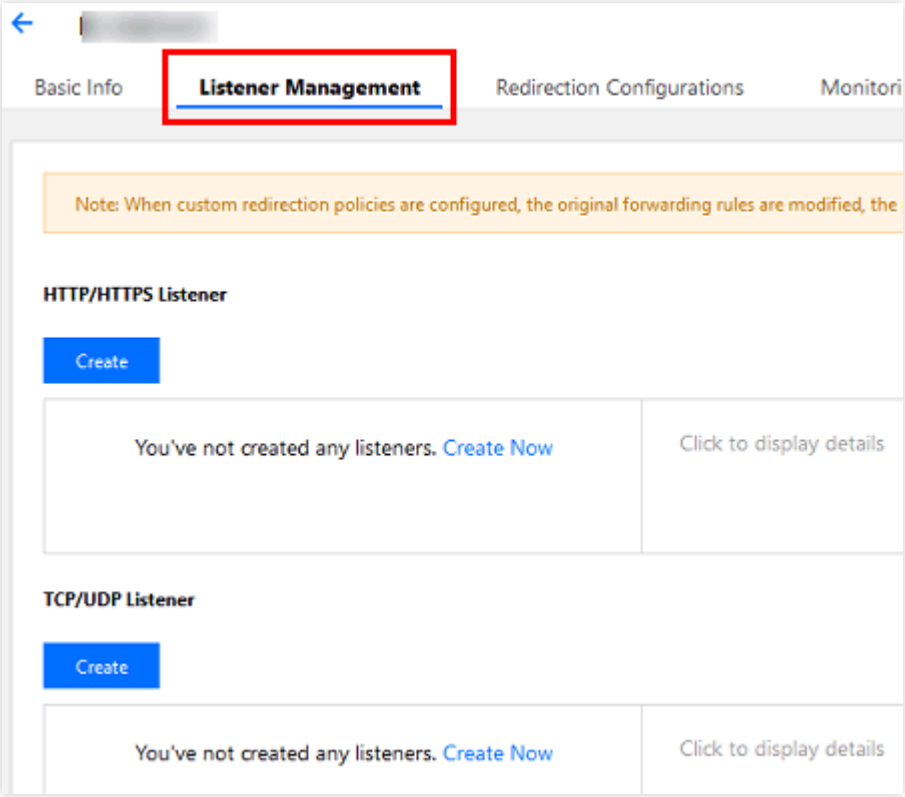
Configuring an HTTPS Listener

Step 1. Open the "Listener Management" page

1. Log in to the CLB Console.
2. Select **Instance Management** on the left sidebar.
3. In the instance list, click the ID of the instance to be configured to enter the instance details page.
4. Click the **Listener Management** tab or click **Configure Listener** in the "Operation" column.



5. The "Listener Management" page is as shown below:



Step 2. Configure a listener

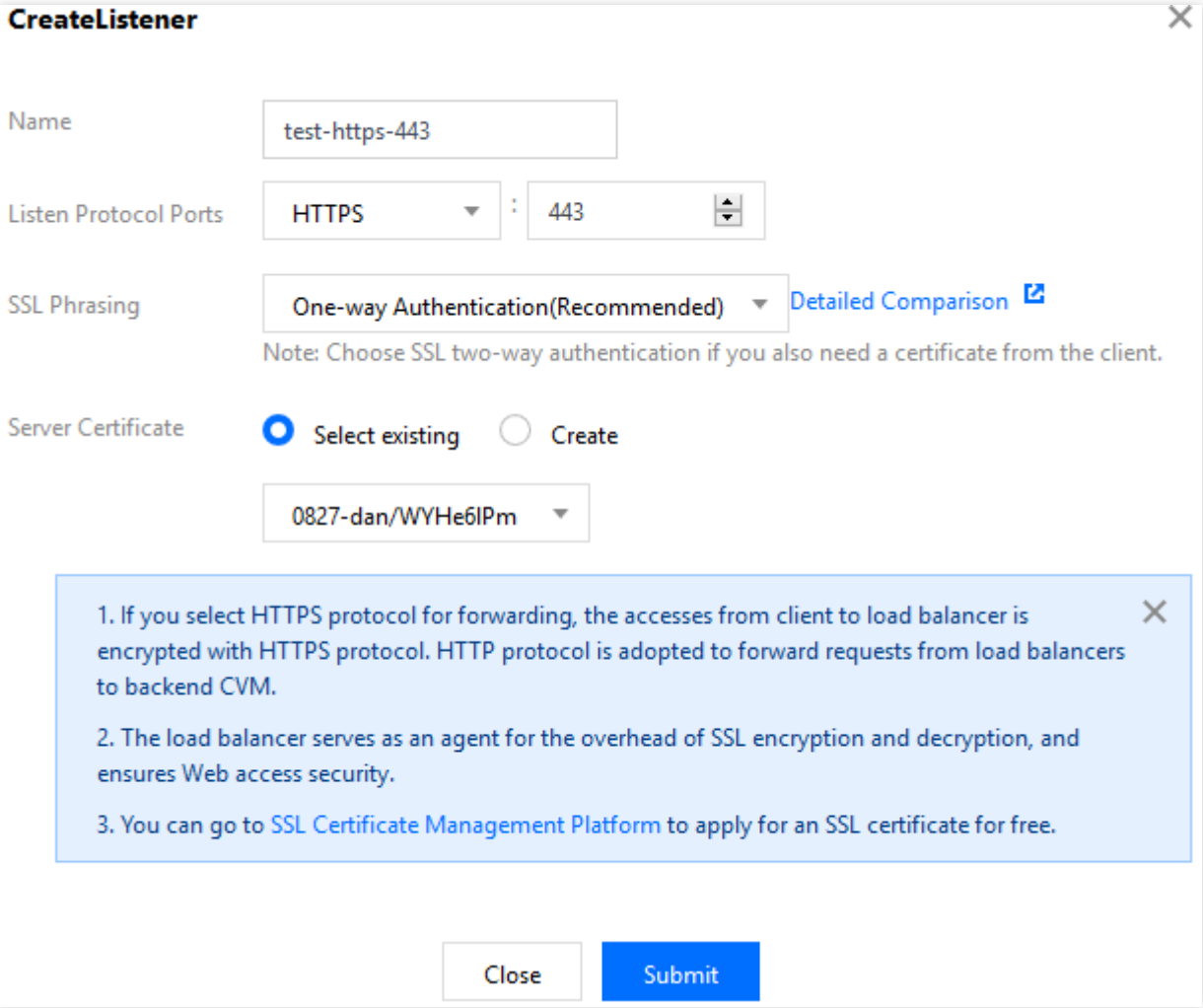
Click **Create** in **HTTP/HTTPS Listener** and configure an HTTPS listener in the pop-up window.

1. Create a listener

Configuration Item	Description	Example
Name	Listener name	test-https-443
Listener protocol and listening port	• Listener protocol: CLB supports various protocols, including TCP, UDP, TCP SSL, HTTP, and HTTPS. HTTPS is used in this example. • Listening port: A port used to receive requests and forward them to the real server. Port range: 1-65535. • The listening port must be unique in the same CLB instance.	HTTPS:443
SSL parsing method	One-way authentication and mutual authentication are supported	One-way authentication

Configuration Item	Description	Example
Server certificate	You can select an existing certificate in the SSL certificate service or upload a certificate	Select the existing certificate cc/UzxFoXsE

The specific configuration of the created HTTPS listener is as shown below:



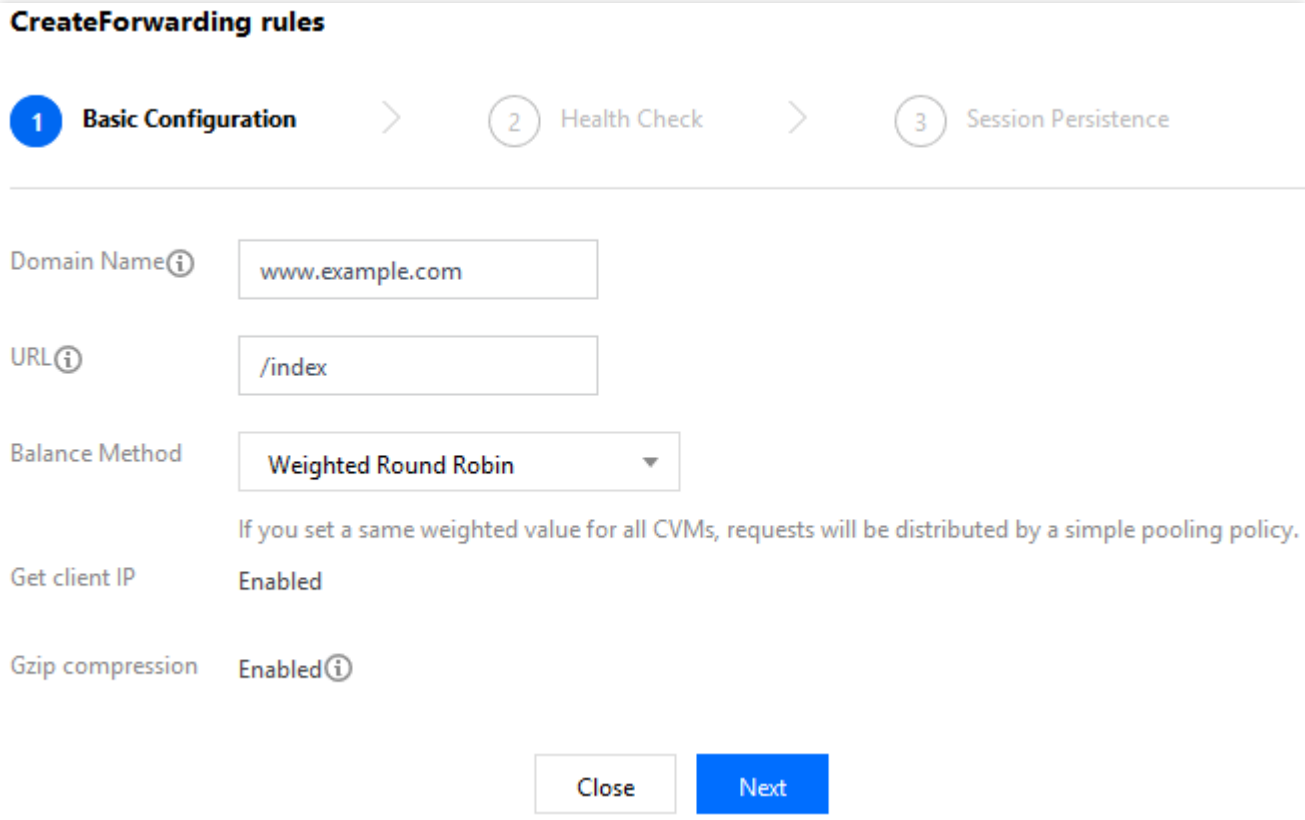
2. Create a forwarding rule

Configuration Item	Description	Example

Configuration Item	Description	Example
Domain name	Request domain name. • Exact domain names are supported, such as `www.example.com` . Wildcard domain names are also supported, such as `*.example.com` and `www.example.*` , where `*` can appear only once in a single domain name. • A non-regular domain name can contain the following characters: `a-z` , `0-9` , `.` , and `-` . • Regex is supported, but they cannot contain the following characters: `”` , `{` , `}` , `;` , `¥` , ```, `~` , `’` , and `space` . • A domain name can contain 1-120 characters.	www.example.com
URL path	Request path. • The path is in the format of `/` by default, must begin with `/` , and can contain 1-120 characters. • A non-regular URL path must begin with `/` and can contain the following characters: `a-z` , `A-Z` , `0-9` , `.` , `-` , `/` , `=` , and `?` . • Regex is supported. 1. Beginning with `=` indicates exact match. 2. Beginning with `^~` indicates that the URI begins with a general string and is not regex match. 3. Beginning with `~` indicates case-sensitive regex match. 4. Beginning with `~*` indicates non-case-insensitive regex match. 5. `/` indicates generic match, where any requests will be matched if there are no other matches. 6. A regular URL cannot contain the following characters: `”` , `{` , `}` , `;` , `¥` , ```, `~` , `’` , and `space` .	/index

Configuration Item	Description	Example
Balancing method	For HTTPS listeners, CLB supports three scheduling algorithms: weighted round robin (WRR), weighted least connections (WLC), and IP hash. • WRR: Requests are sequentially delivered to different real servers according to their weights. Scheduling is done based on the **number of new connections**, where servers with higher weights will undergo more polls (i.e., a higher probability), while servers with the same weight process the same number of connections. • WLC: Loads of servers are estimated according to the number of active connections to the servers. Scheduling is done based on server loads and weights. If their weights are the same, servers with fewer active connections will undergo more polls (i.e., a higher probability). • IP hash: Hash keys are used to locate the corresponding servers in the static hash table based on the source IPs of requests. If a server is available and not overloaded, requests will be delivered to it; otherwise, a null value will be returned.	WRR
Getting client IP	Enabled by default	Enabled
Gzip compression	Enabled by default	Enabled

Select the HTTPS listener for which to create a forwarding rule and click **+** on the right. The specific configuration is as shown below:

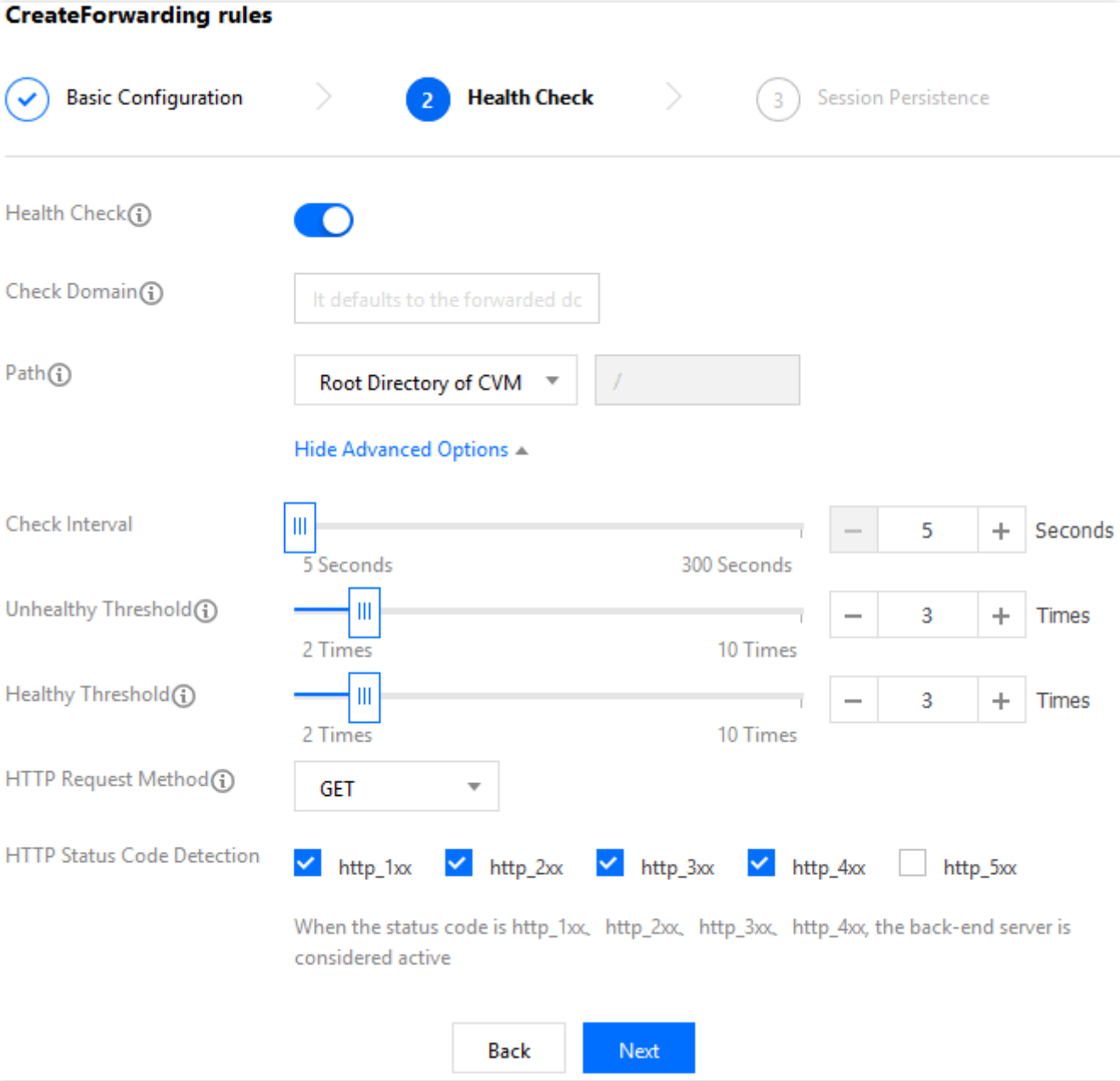


3. Health check

Configuration Item	Description	Example
Health check status	Health check can be enabled or disabled. In HTTPS listeners, CLB instances send HTTPS requests to the specified server port to perform health checks.	Enabled
Check domain name	Request domain name. • The access domain name is used for this parameter by default. • It can contain 1-120 supported characters: `a-z` , `0-9` , `.` , and `-` . • Regex is not supported currently. • If a wildcard domain name is entered, a fixed domain name (non-regular) should be specified as the health check domain name.	Default value (i.e., `www.example.com`)

Configuration Item	Description	Example
Check path	Request path. • The path is in the format of `/` by default and must begin with `/`. • It can contain 1-120 supported characters: `a-z`, `0-9`, `.`, and `-`. • Regex is not supported currently. • It is recommended to specify a fixed URL path (static page) for health checks.	Default value (i.e., `/`)
Check interval	• Interval between two health checks. • Value range: 5-300s. Default value: 5s.	5s
Unhealthy threshold	• If the health check results received n times (n is the entered number) in a row are failures, the instance will be considered unhealthy, and the status displayed in the console will be **Abnormal**. • Value range: 2-10. Default value: 3.	3 times
Healthy threshold	• If the health check results received n times (n is the entered number) in a row are successes, the instance will be considered healthy, and the status displayed in the console will be **Healthy**. • Value range: 2-10. Default value: 3.	3 times
HTTP request method	HTTP request method for health checks. Value range: GET, HEAD. Default value: GET. • If HEAD is used, the server will only return the HTTP header, which can reduce backend overheads and improve request efficiency; the corresponding real server needs to support HEAD. • If GET is used, the real server needs to support GET.	GET
HTTP status code check	If the status code is the selected one, the real server is considered alive (healthy). Value range: http_1xx, http_2xx, http_3xx, http_4xx, http_5xx.	Multiple selections: http_1xx, http_2xx, http_3xx, http_4xx

The specific configuration of health check is as shown below:

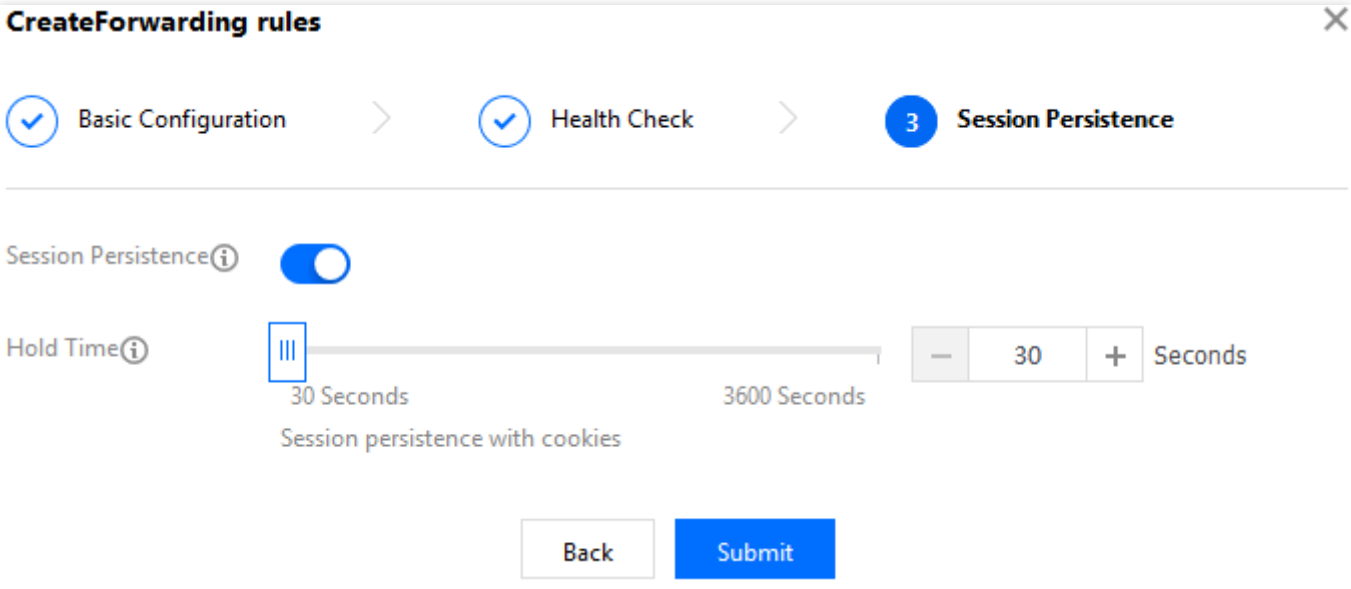


4. Session persistence

Configuration Item	Description	Example

Configuration Item	Description	Example
Session persistence status	Session persistence can be enabled or disabled. • If session persistence is enabled, the CLB listener will deliver access requests from the same client to the same real server. • HTTPS session persistence is implemented based on cookies, which are implanted into the client by the CLB instance. • Session persistence can be enabled for WRR scheduling but not WLC or IP hash scheduling.	Enabled
Session persistence time	Session persistence time • If there is no new request in the connection within the session persistence time, session persistence will be interrupted automatically. • Value range: 30-3,600s.	30s

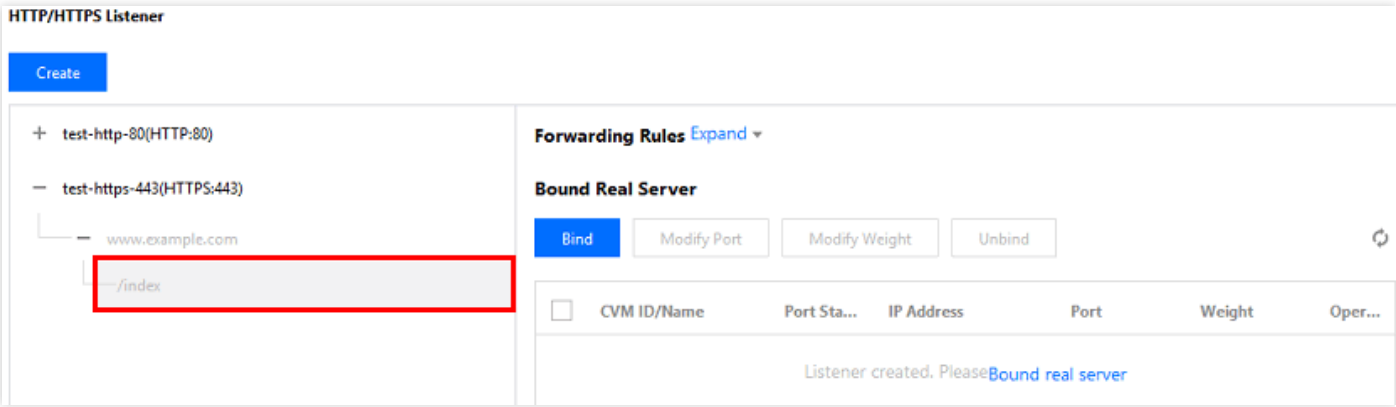
The specific configuration of session persistence is as shown below:



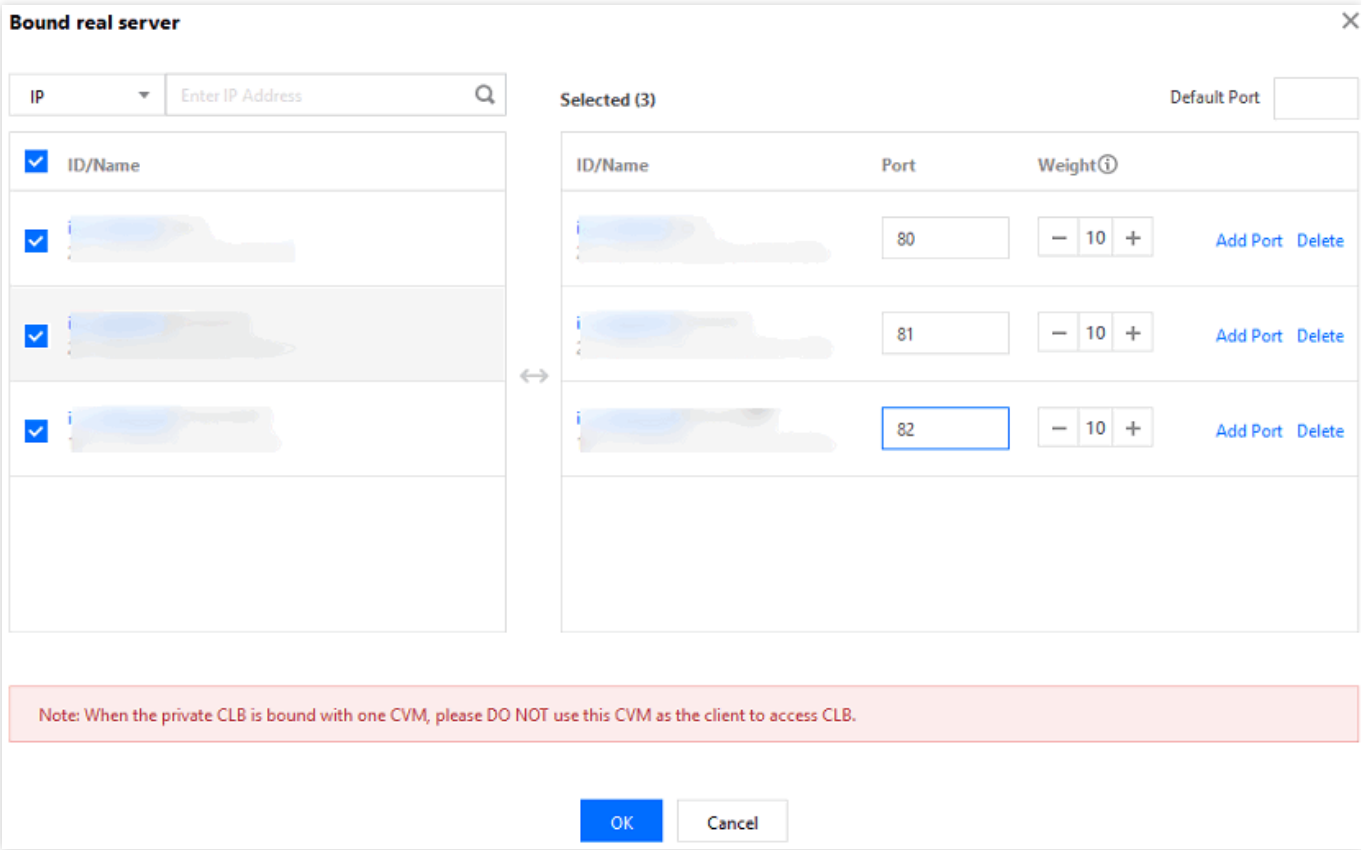
Step 3. Bind a real server

1. On the "Listener Management" page, select the created listener `HTTPS:443` . Click + on the left to expand the domain names and URL paths, select the desired URL path, and view the real servers

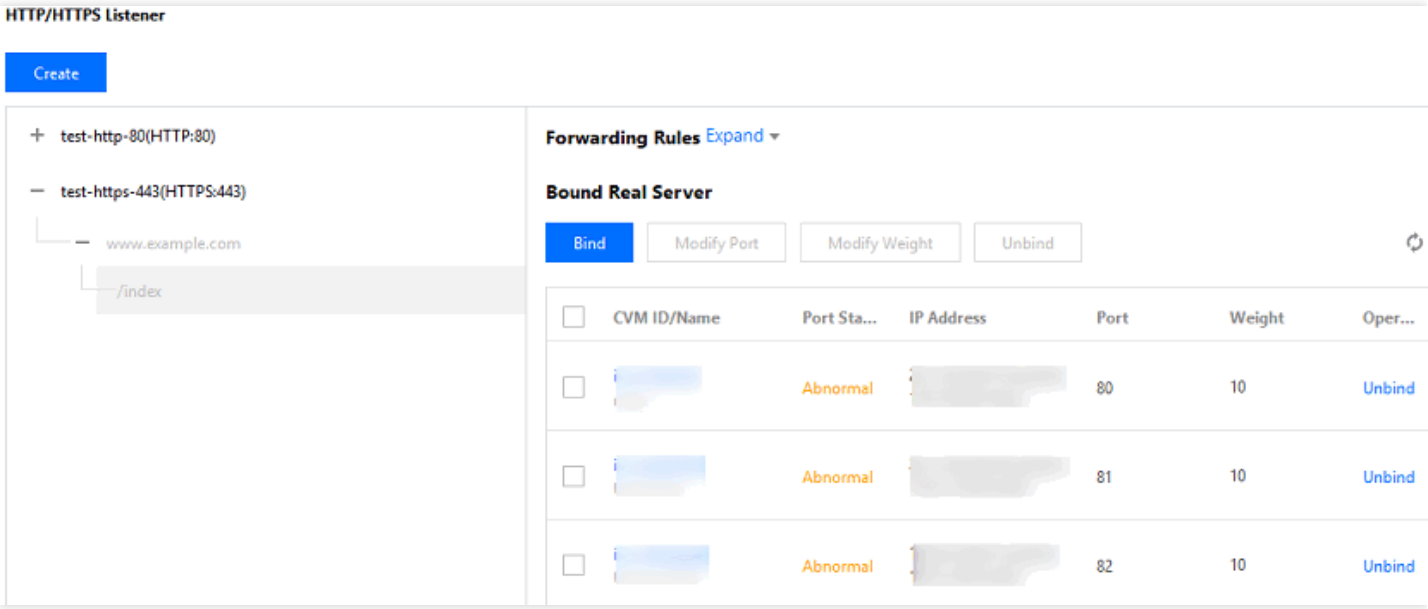
bound to the path on the right of the listener.



2. Click **Bind** and select the real server to be bound and configure the server port and weight in the pop-up window.
   i. Add Port: In the "Selected" box on the right, click **Add Port** to add multiple ports for the same CVM instance, such as ports 80, 81, and 82.
   ii. Default Port: Enter the "Default Port" first and then select the CVM instance. The port of every CVM instance is the default port.



After these three steps are completed, the HTTPS listener rule has been configured as shown below:

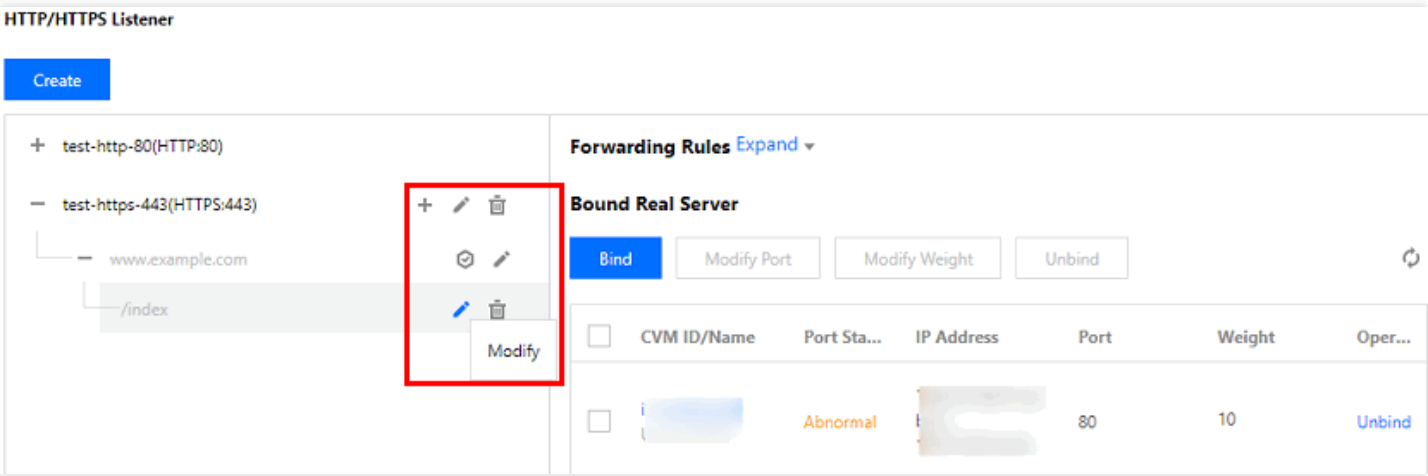


Step 4. Security group (optional)

You can configure a CLB security group to isolate public network traffic. For more information, see Configuring a CLB Security Group.

Step 5. Modify/delete a listener (optional)

If you need to modify or delete a created listener, click the listener/domain name/URL path on the "Listener Management" page and select **Modify** or **Delete**.



Configuring a UDP Listener

Last updated : 2019-11-05 15:23:23

UDP Listener Overview

You can create a UDP listener to a CLB instance to forward UDP requests from the client. UDP is suitable for scenarios that have high requirements for transfer speed but relatively low requirements for accuracy, such as instant messaging and online videos. For UDP listeners, the real server can directly get the real client IP.

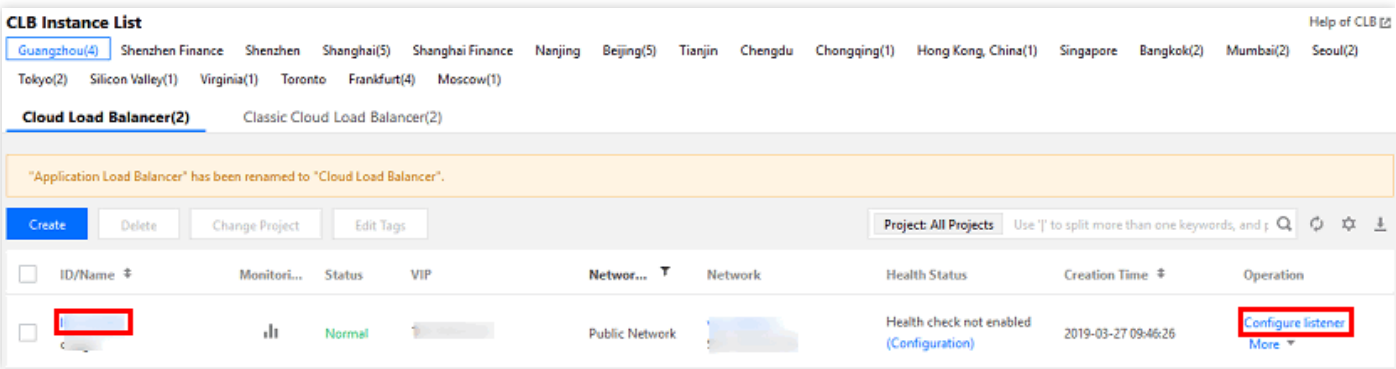
Prerequisites

You need to create a CLB instance first.

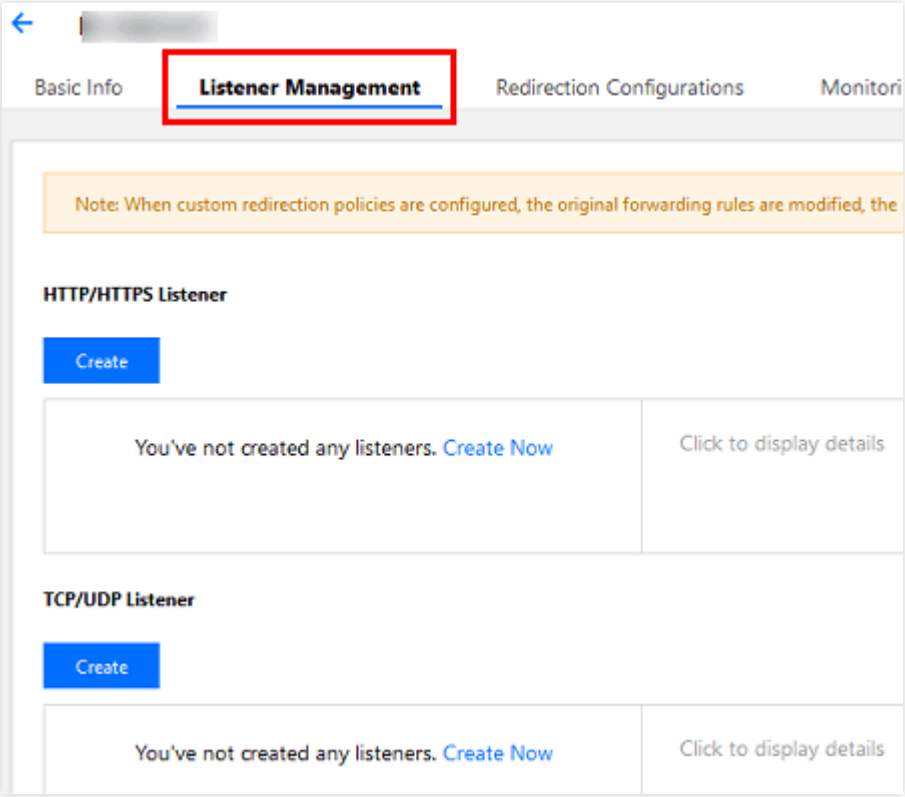
Configuring a UDP Listener

Step 1. Open the "Listener Management" page

1. Log in to the CLB Console.
2. Select **Instance Management** on the left sidebar.
3. In the instance list, click the ID of the instance to be configured to enter the instance details page.
4. Click the **Listener Management** tab or click **Configure Listener** in the "Operation" column.



5. The "Listener Management" page is as shown below:



Step 2. Configure a listener

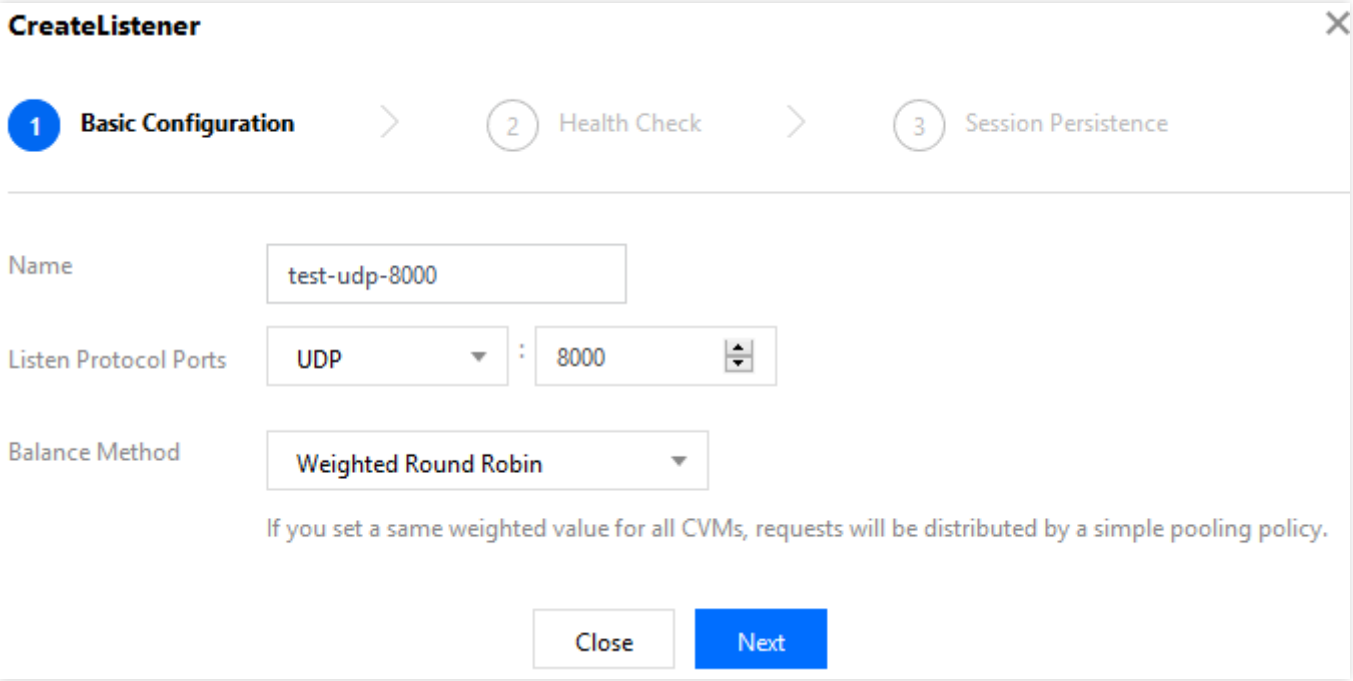
Click **Create** in **TCP/UDP Listener** and configure a UDP listener in the pop-up window.

1. Basic configuration

Configuration Item	Description	Example
Name	Listener name	test-udp-8000
Listener protocol and listening port	Listener protocol and listening port. • Listener protocol: CLB supports various protocols, including TCP, UDP, HTTP, and HTTPS. UDP is used in this example. • Listening port: A port used to receive requests and forward them to the real server. Port range: 1-65535. • The listener port must be unique in the same CLB instance.	UDP:8000

Configuration Item	Description	Example
Balancing method	For UDP listeners, CLB supports two scheduling algorithms: weighted round robin (WRR) and weighted least connections (WLC). • WRR: Requests are sequentially delivered to different real servers according to their weights. Scheduling is done based on the **number of new connections**, where servers with higher weights will undergo more polls (i.e., a higher probability), while servers with the same weight process the same number of connections. • WLC: Loads of servers are estimated according to the number of active connections to the servers. Scheduling is done based on server loads and weights. If their weights are the same, servers with fewer active connections will undergo more polls (i.e., a higher probability).	WRR

The specific configuration of the created UDP listener is as shown below:

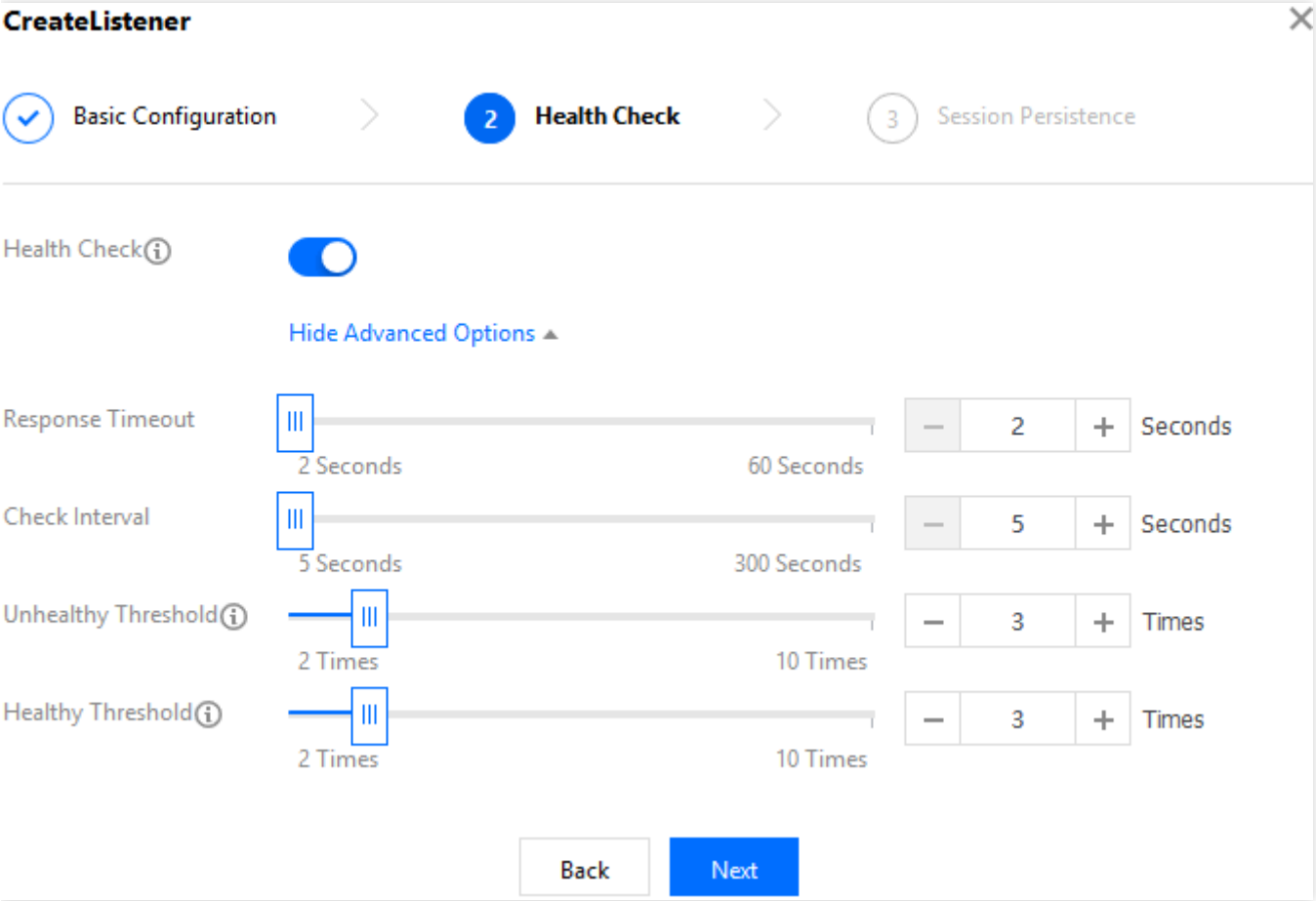


2. Health check

Configuration Item	Description	Example

Configuration Item	Description	Example
Health check status	Health check can be enabled or disabled. In UDP listeners, CLB instances send ping commands to the server to perform health checks.	Enabled
Response timeout period	• Maximum response timeout period for health checks. • If a real server fails to respond correctly within the timeout period, it is considered abnormal. • Value range: 2-60s. Default value: 2s.	2s
Check interval	• Interval between two health checks. • Value range: 5-300s. Default value: 5s.	5s
Unhealthy threshold	• If the health check results received n times (n is the entered number) in a row are failures, the instance will be considered unhealthy, and the status displayed in the console will be **Abnormal**. • Value range: 2-10. Default value: 3.	3 times
Healthy threshold	• If the health check results received n times (n is the entered number) in a row are successes, the instance will be considered healthy, and the status displayed in the console will be **Healthy**. • Value range: 2-10. Default value: 3.	3 times

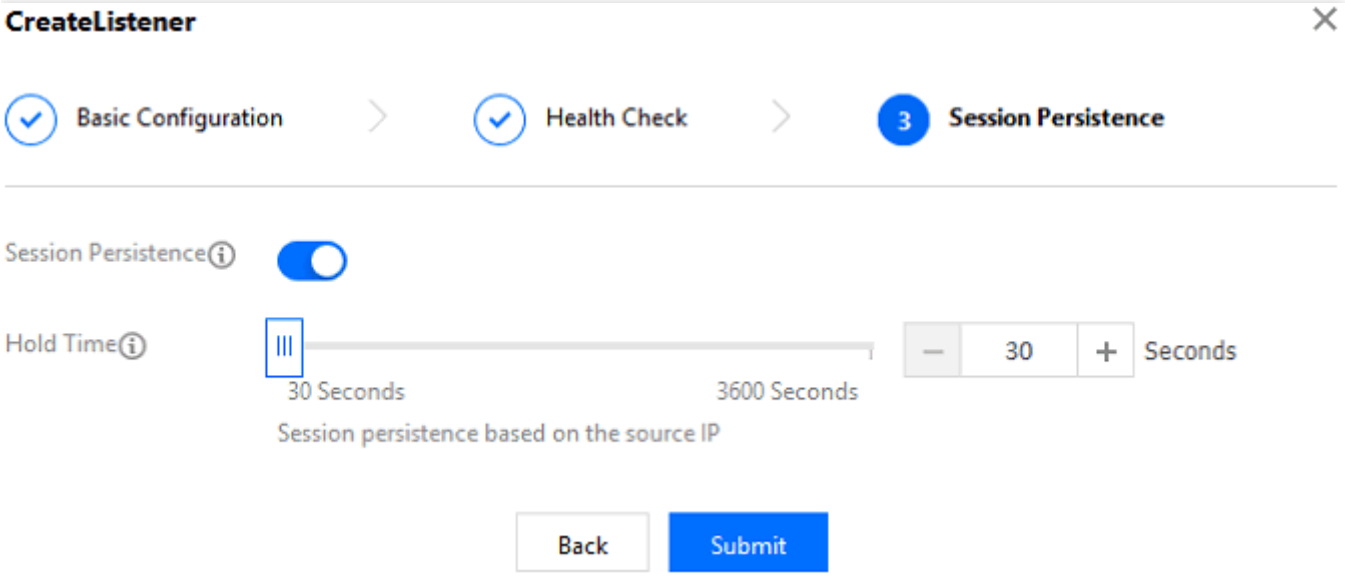
The specific configuration of health check is as shown below:



3. Session persistence

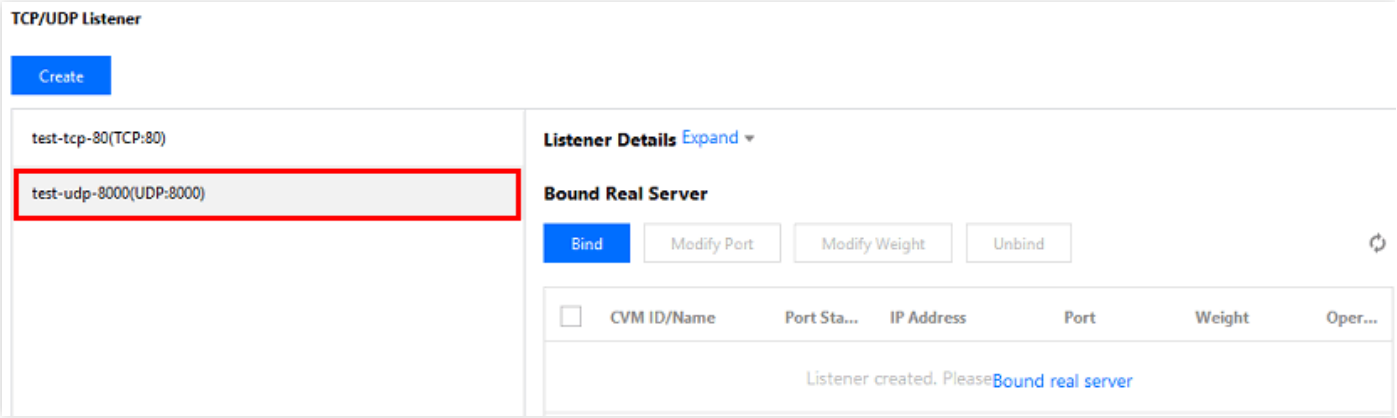
Configuration Item	Description	Example
Session persistence status	Session persistence can be enabled or disabled. • If session persistence is enabled, the CLB listener will deliver access requests from the same client to the same real server. • UDP session persistence is implemented based on client IP addresses, i.e., access requests from the same IP address are forwarded to the same real server. • Session persistence can be enabled for WRR scheduling but not WLC scheduling.	Enabled
Session persistence time	Session persistence time. • If there is no new request in the connection within the session persistence time, session persistence will be interrupted automatically. • Value range: 30-3,600s.	30s

The specific configuration of session persistence is as shown below:

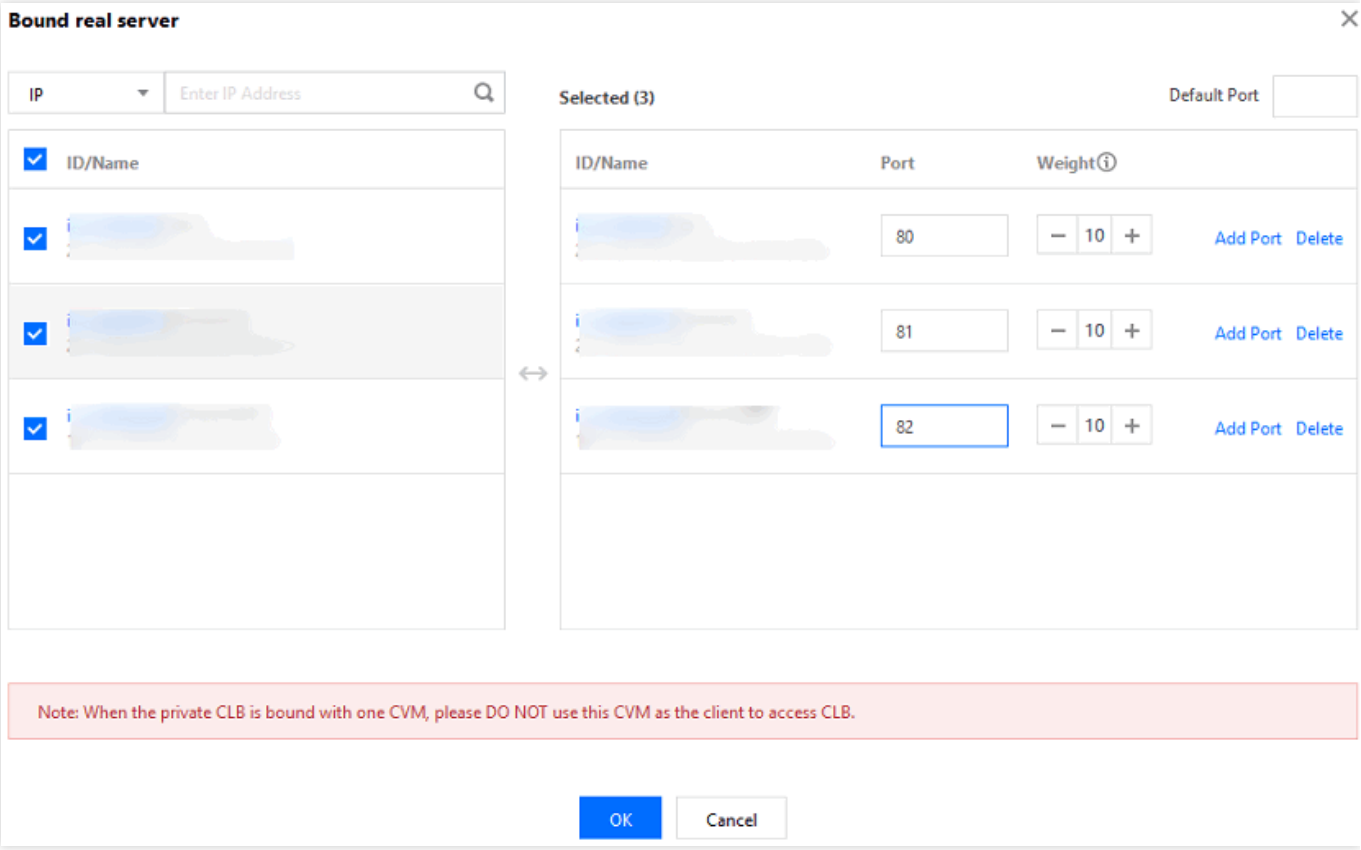


Step 3. Bind a real server

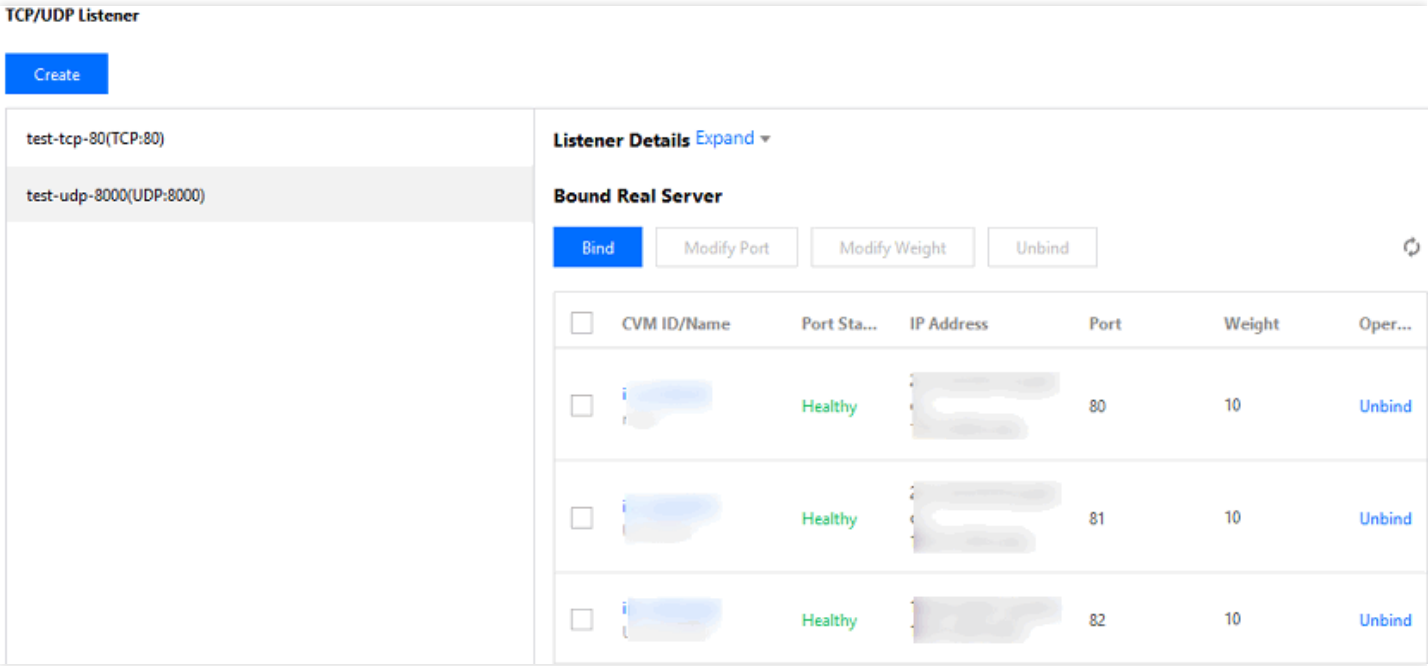
1. On the "Listener Management" page, click the created listener `UDP:8000` to view the bound real servers on the right of the listener.



2. Click **Bind** and select the real server to be bound and configure the server port and weight in the pop-up window.
   i. Add Port: In the "Selected" box on the right, click **Add Port** to add multiple ports for the same CVM instance, such as ports 80, 81, and 82.
   ii. Default Port: Enter the "Default Port" first and then select the CVM instance. The port of every CVM instance is the default port.



After these three steps are completed, the UDP listener rule has been configured as shown below:

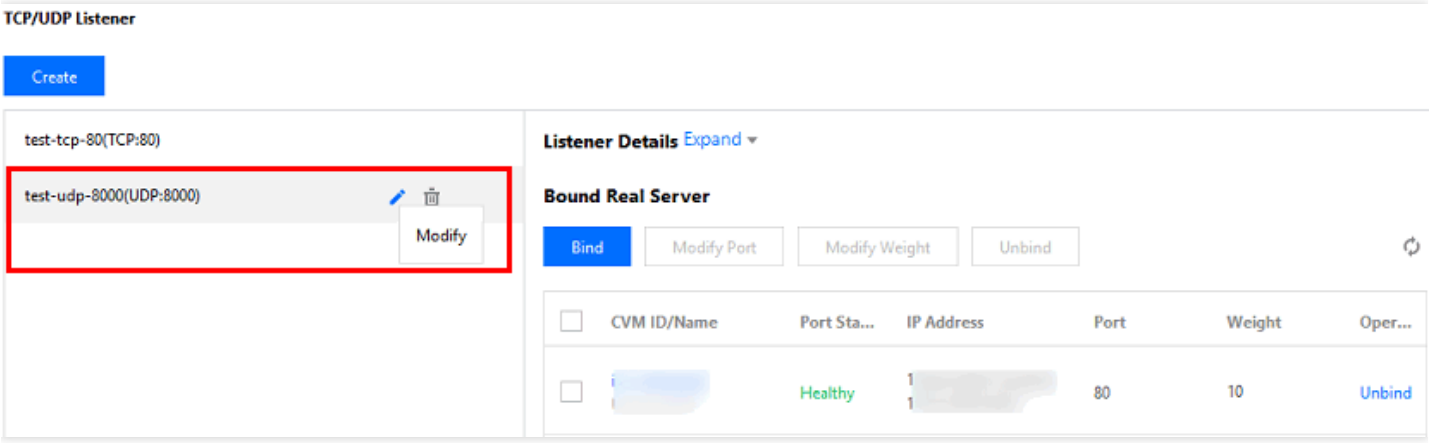


Step 4. Security group (optional)

You can configure a CLB security group to isolate public network traffic. For more information, see Configuring a CLB Security Group.

Step 5. Modify/delete a listener (optional)

If you need to modify or delete a created listener, click the listener on the "Listener Management" page and select **Modify** or **Delete**.



Configuring a TCP Listener

Last updated : 2019-11-05 15:24:03

TCP Listener Overview

You can create a TCP listener to a CLB instance to forward TCP requests from the client. TCP is suitable for scenarios that have high requirements for reliability and data accuracy but relatively low requirements for transfer speed, such as file transfer, email messaging, and remote login. For TCP listeners, the real server can directly get the real client IP.

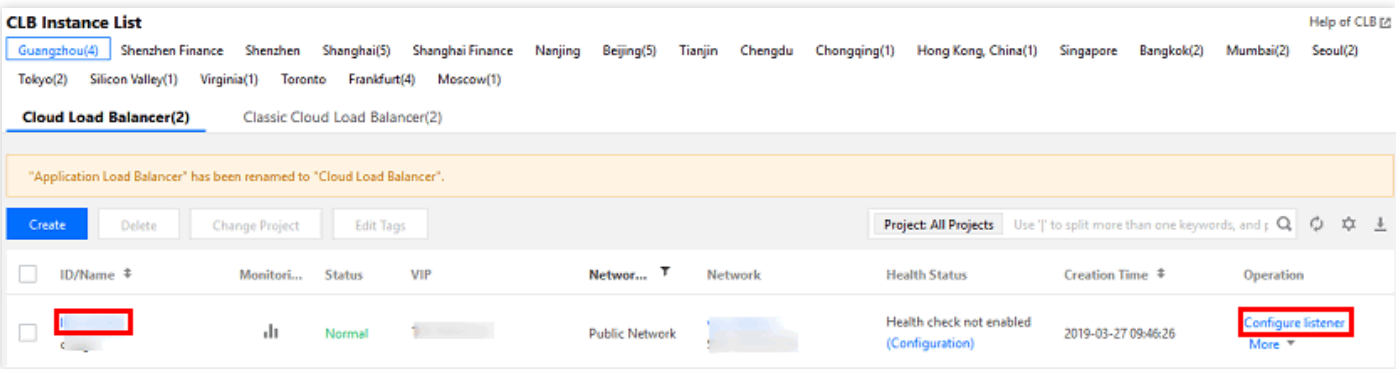
Prerequisites

You need to create a CLB instance first.

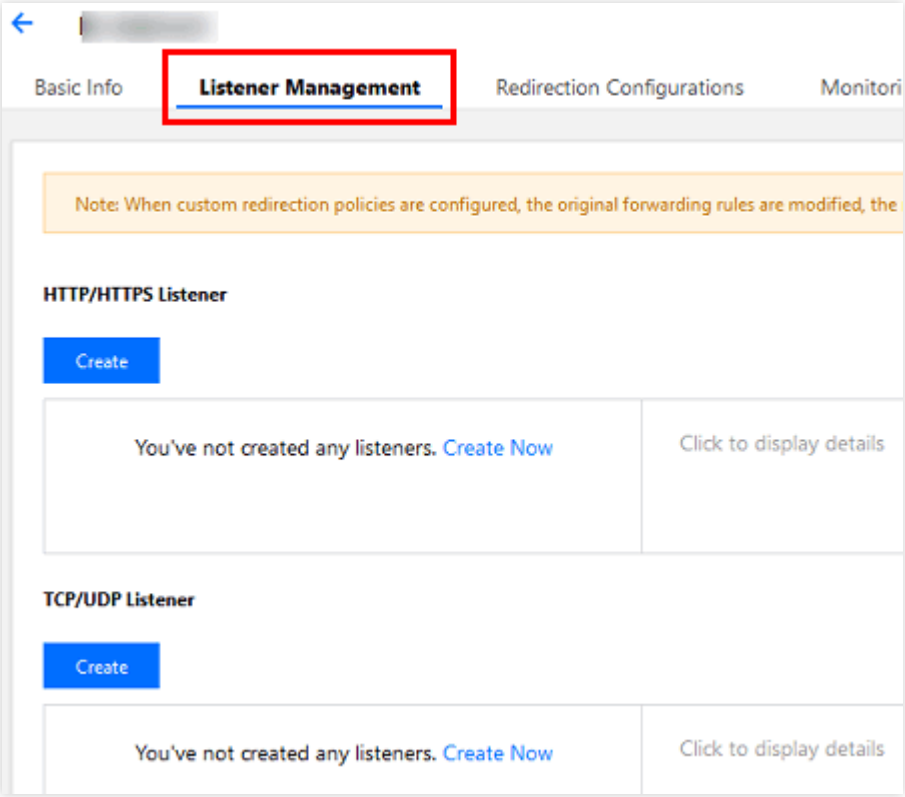
Configuring a TCP Listener

Step 1. Open the "Listener Management" page

1. Log in to the CLB Console.
2. Select **Instance Management** on the left sidebar.
3. In the instance list, click the ID of the instance to be configured to enter the instance details page.
4. Click the **Listener Management** tab or click **Configure Listener** in the "Operation" column.



5. The "Listener Management" page is as shown below:



Step 2. Configure a listener

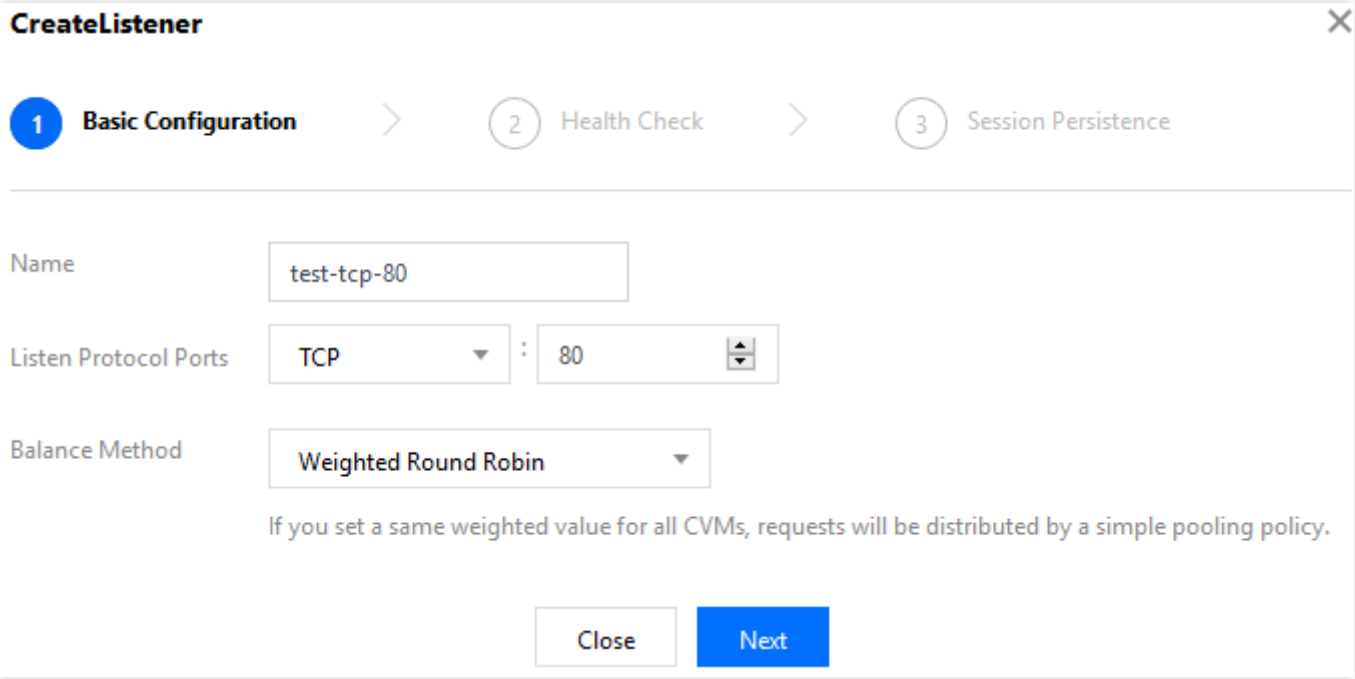
Click **Create** in **TCP/UDP Listener** and configure a TCP listener in the pop-up window.

1. Basic configuration

Configuration Item	Description	Example
Name	Listener name	test-tcp-80
Listener protocol and listening port	Listener protocol and listening port. • Listener protocol: CLB supports various protocols, including TCP, UDP, HTTP, and HTTPS. TCP is used in this example. • Listening port: A port used to receive requests and forward them to the real server. Port range: 1-65535. • The listener port must be unique in the same CLB instance.	TCP:80

Configuration Item	Description	Example
Balancing method	For TCP listeners, CLB supports two scheduling algorithms: weighted round robin (WRR) and weighted least connections (WLC). • WRR: Requests are sequentially delivered to different real servers according to their weights. Scheduling is done based on the **number of new connections**, where servers with higher weights will undergo more polls (i.e., a higher probability), while servers with the same weight process the same number of connections. • WLC: Loads of servers are estimated according to the number of active connections to the servers. Scheduling is done based on server loads and weights. If their weights are the same, servers with fewer active connections will undergo more polls (i.e., a higher probability).	WRR

The specific configuration of the created TCP listener is as shown below:

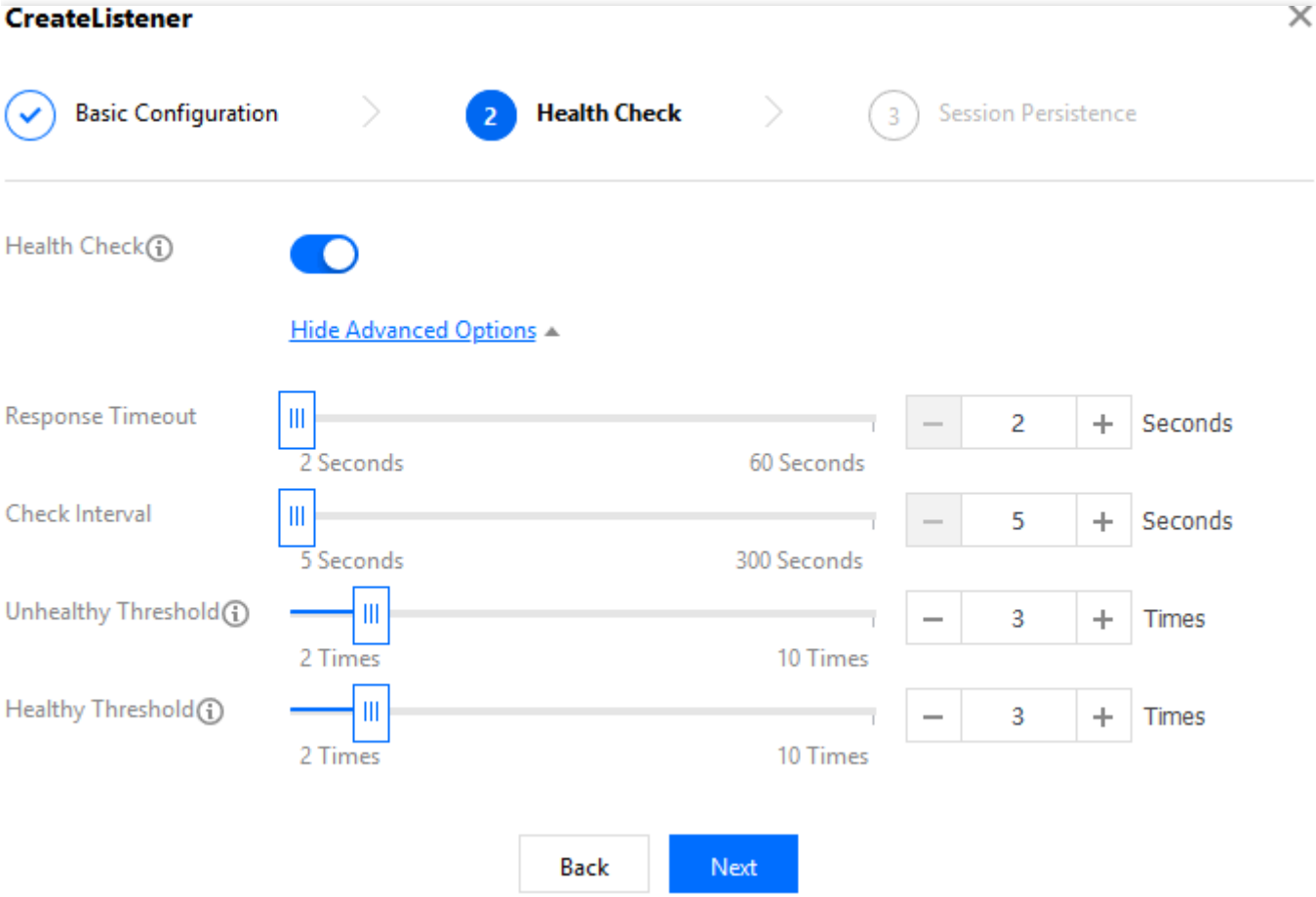


2. Health check

Configuration Item	Description	Example

Configuration Item	Description	Example
Health check status	Health check can be enabled or disabled. In TCP listeners, CLB instances send SYN packets to the specified server port to perform health checks.	Enabled
Response timeout period	• Maximum response timeout period for health checks. • If a real server fails to respond correctly within the timeout period, it is considered abnormal. • Value range: 2-60s. Default value: 2s.	2s
Check interval	• Interval between two health checks. • Value range: 5-300s. Default value: 5s.	5s
Unhealthy threshold	• If the health check results received n times (n is the entered number) in a row are failures, the instance will be considered unhealthy, and the status displayed in the console will be **Abnormal**. • Value range: 2-10. Default value: 3.	3 times
Healthy threshold	• If the health check results received n times (n is the entered number) in a row are successes, the instance will be considered healthy, and the status displayed in the console will be **Healthy**. • Value range: 2-10. Default value: 3.	3 times

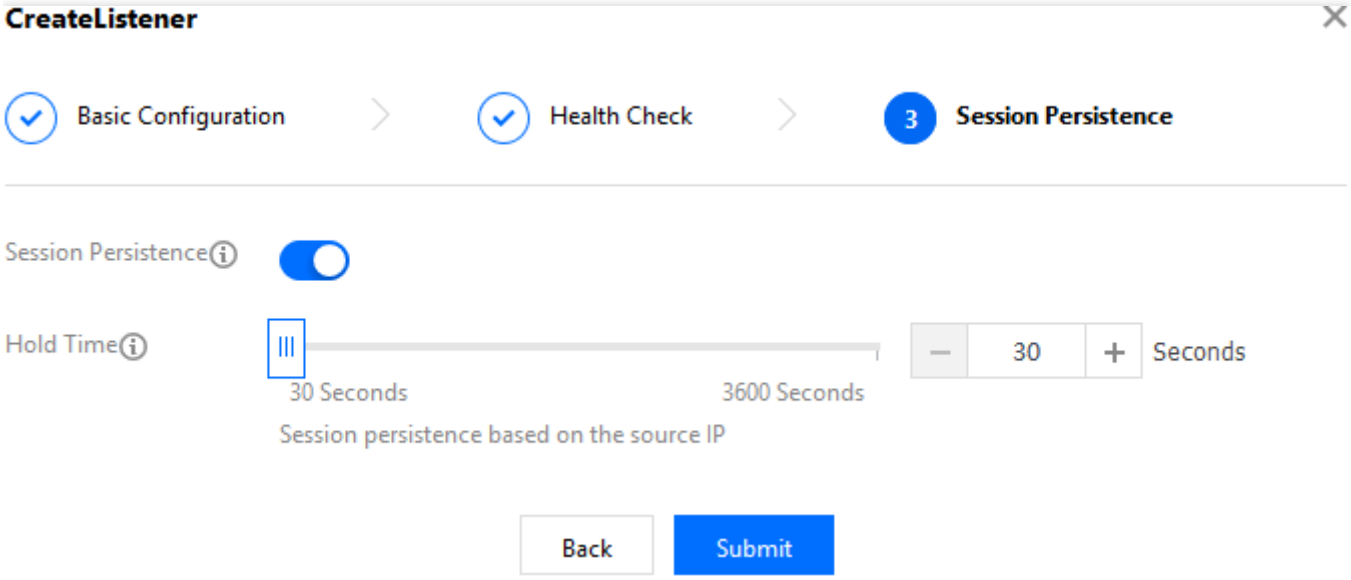
The specific configuration of health check is as shown below:



3. Session persistence

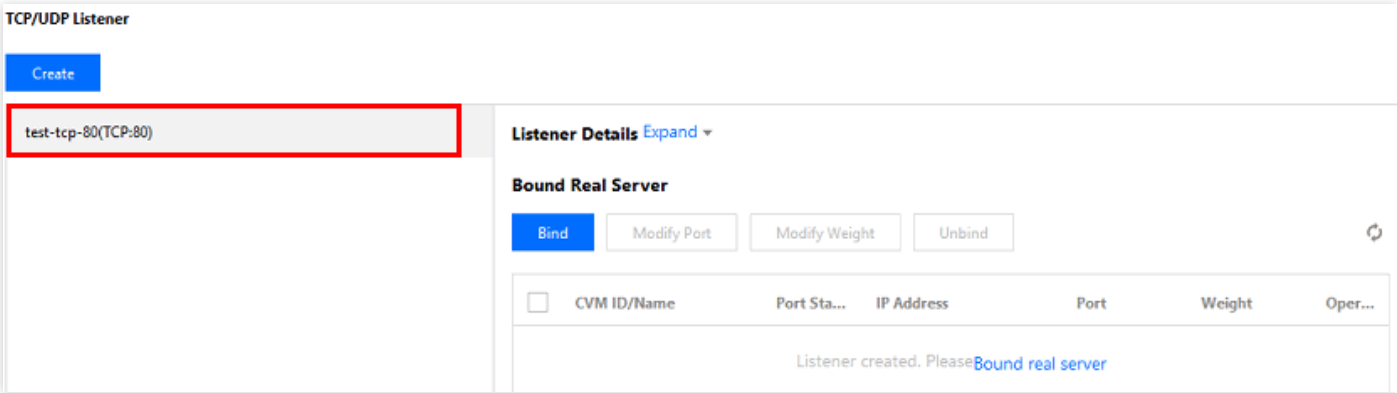
Configuration Item	Description	Example
Session persistence status	Session persistence can be enabled or disabled. • If session persistence is enabled, the CLB listener will deliver access requests from the same client to the same real server. • TCP session persistence is implemented based on client IP addresses, i.e., access requests from the same IP address are forwarded to the same real server. • Session persistence can be enabled for WRR scheduling but not WLC scheduling.	Enabled
Session persistence time	Session persistence time. • If there is no new request in the connection within the session persistence time, session persistence will be interrupted automatically. • Value range: 30-3,600s.	30s

The specific configuration of session persistence is as shown below:

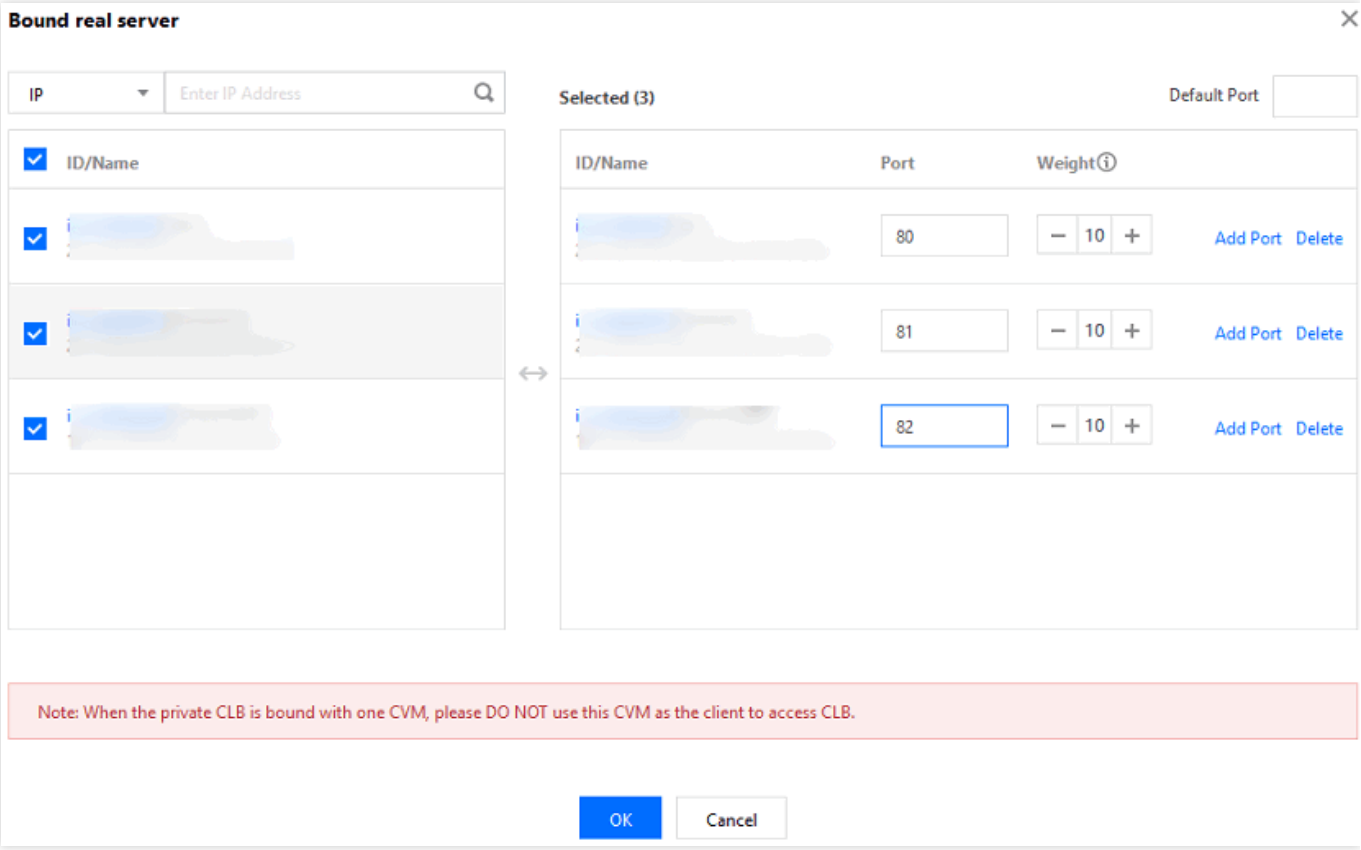


Step 3. Bind a real server

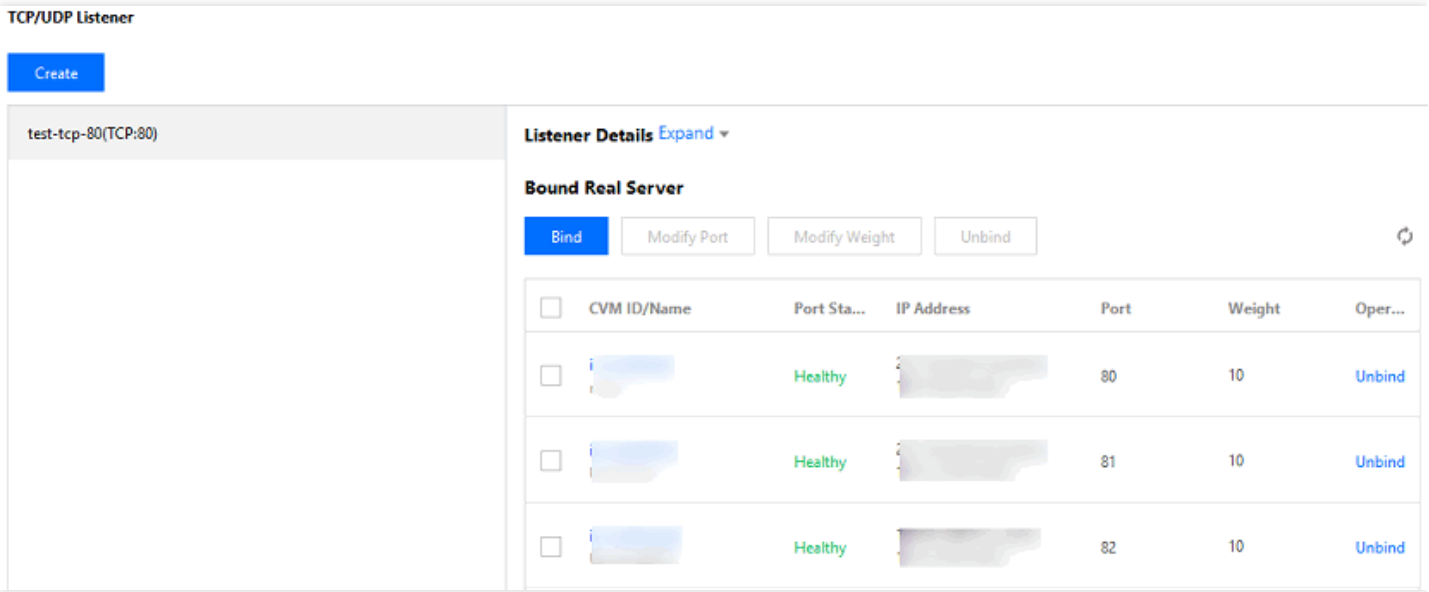
1. On the "Listener Management" page, click the created listener `TCP:80` to view the bound real servers on the right of the listener.



2. Click **Bind** and select the real server to be bound and configure the server port and weight in the pop-up window.
   i. Add Port: In the "Selected" box on the right, click **Add Port** to add multiple ports for the same CVM instance, such as ports 80, 81, and 82.
   ii. Default Port: Enter the "Default Port" first and then select the CVM instance. The port of every CVM instance is the default port.



After these three steps are completed, the TCP listener rule has been configured as shown below:

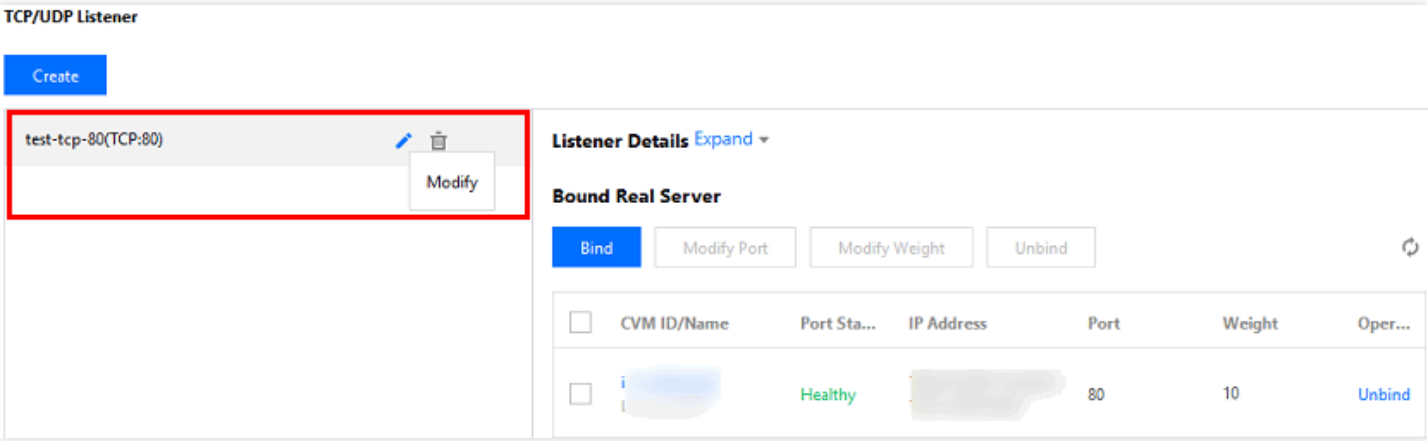


Step 4. Security group (optional)

You can configure a CLB security group to isolate public network traffic. For more information, see Configuring a CLB Security Group.

Step 5. Modify/delete a listener (optional)

If you need to modify or delete a created listener, click the listener on the "Listener Management" page and select **Modify** or **Delete**.



Configuring a TCP SSL Listener

Last updated : 2019-11-05 15:25:05

TCP SSL Listener Overview

You can create a TCP SSL listener to a CLB instance to forward encrypted TCP requests from the client. TCP SSL is applicable to scenarios where ultra-high performance and large-scale TLS offloading are required. For TCP SSL listeners, the real server can directly get the real client IP.

> The TCP SSL listener feature is currently in beta test and only available to public network CLB but not private network CLB or classic CLB. If you want to use it, please submit a ticket for application.

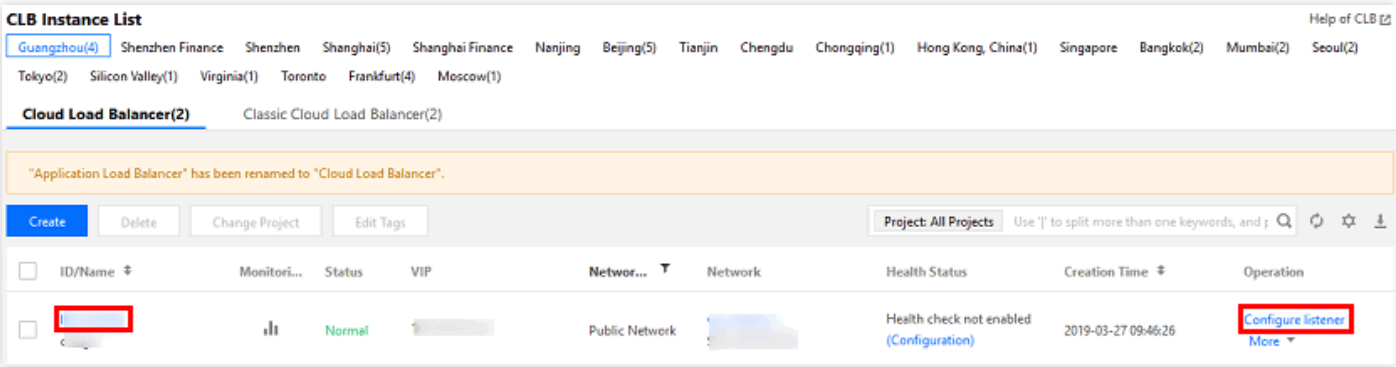
Prerequisites

You need to create a CLB instance first.

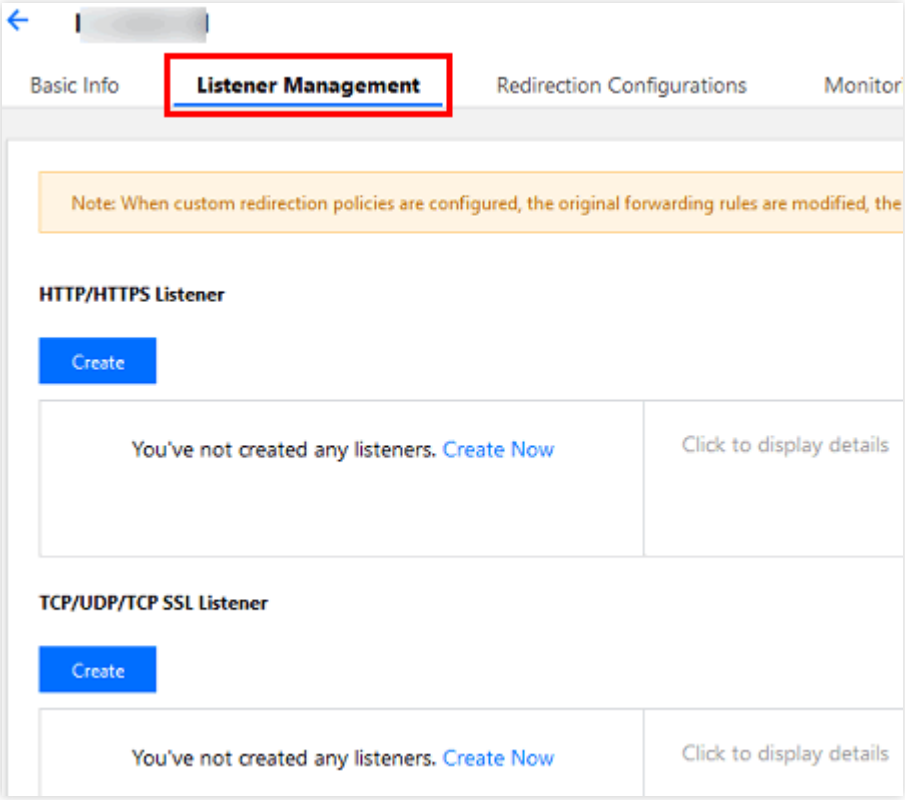
Configuring a TCP SSL Listener

Step 1. Open the "Listener Management" page

1. Log in to the CLB Console.
2. Select **Instance Management** on the left sidebar.
3. In the instance list, click the ID of the instance to be configured to enter the instance details page.
4. Click the **Listener Management** tab or click **Configure Listener** in the "Operation" column.



5. The "Listener Management" page is as shown below:



Step 2. Configure a listener

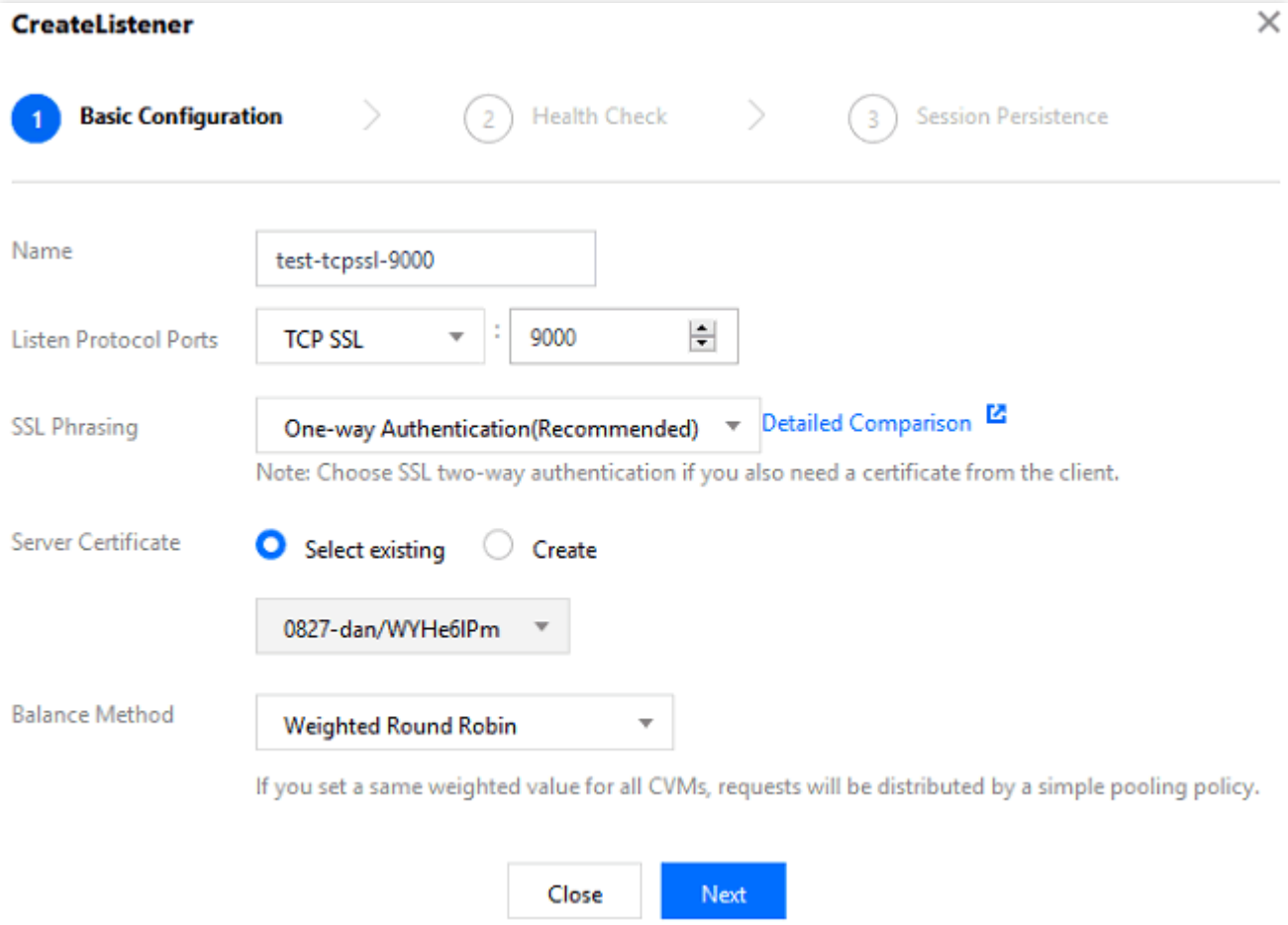
Click **Create** in **TCP/UDP/TCP SSL Listener** and configure a TCP SSL listener in the pop-up window.

1. Basic configuration

Configuration Item	Description	Example
Name	Listener name	test-tcpssl-9000
Listener protocol and listening port	Listener protocol and listening port. • Listener protocol: CLB supports various protocols, including TCP, UDP, TCP SSL, HTTP, and HTTPS. TCP SSL is used in this example. • Listening port: A port used to receive requests and forward them to the real server. Port range: 1-65535. • The listener port must be unique in the same CLB instance.	TCP SSL:9000
SSL parsing method	One-way authentication and mutual authentication are supported	One-way authentication

Configuration Item	Description	Example
Server certificate	You can select an existing certificate in the SSL certificate service or upload a certificate	Select the existing certificate cc/UzxFoXsE
Balancing method	For TCP SSL listeners, CLB supports two scheduling algorithms: weighted round robin (WRR) and weighted least connections (WLC). • WRR: Requests are sequentially delivered to different real servers according to their weights. Scheduling is done based on the **number of new connections**, where servers with higher weights will undergo more polls (i.e., a higher probability), while servers with the same weight process the same number of connections. • WLC: Loads of servers are estimated according to the number of active connections to the servers. Scheduling is done based on server loads and weights. If their weights are the same, servers with fewer active connections will undergo more polls (i.e., a higher probability).	WRR

The specific configuration of the created TCP SSL listener is as shown below:

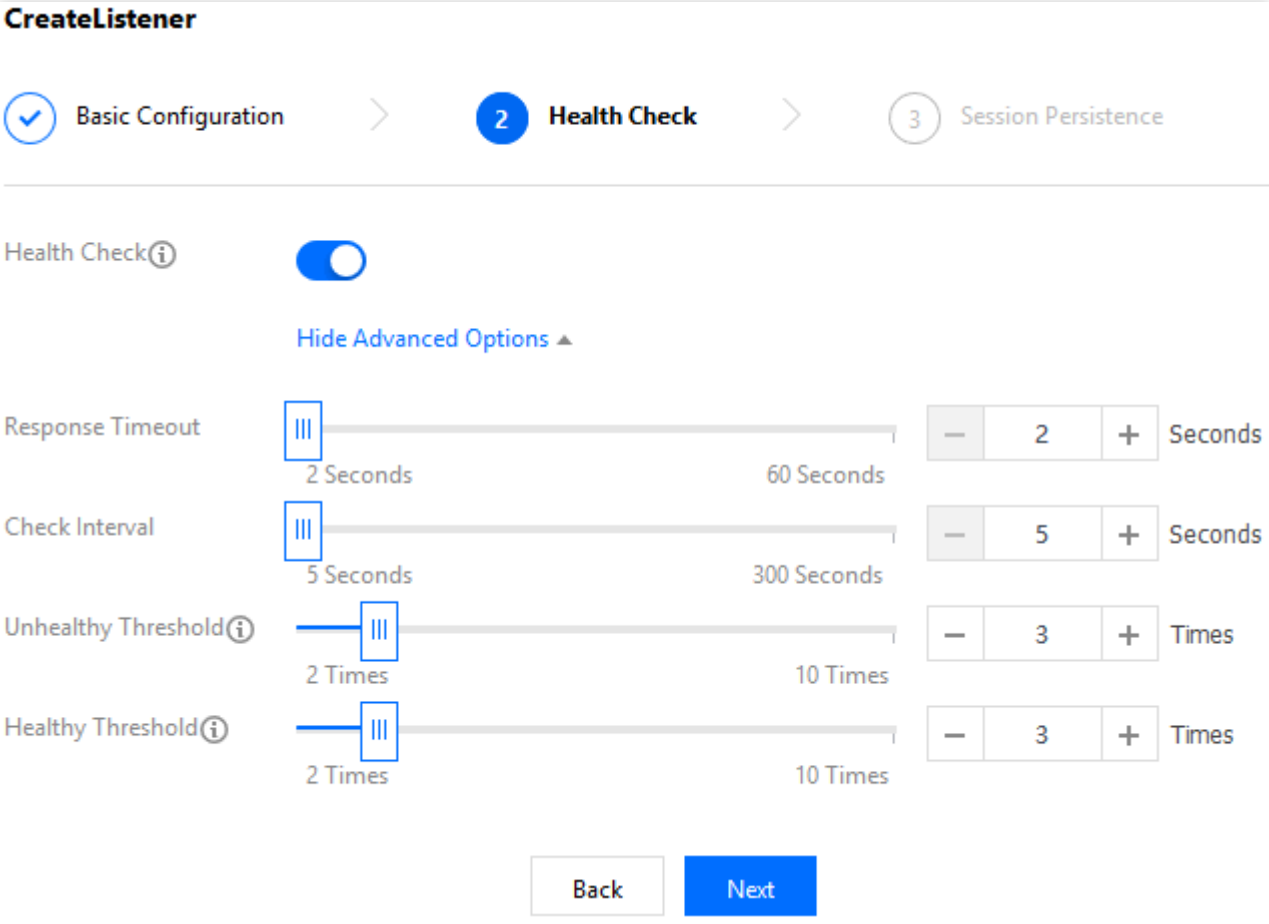


2. Health check

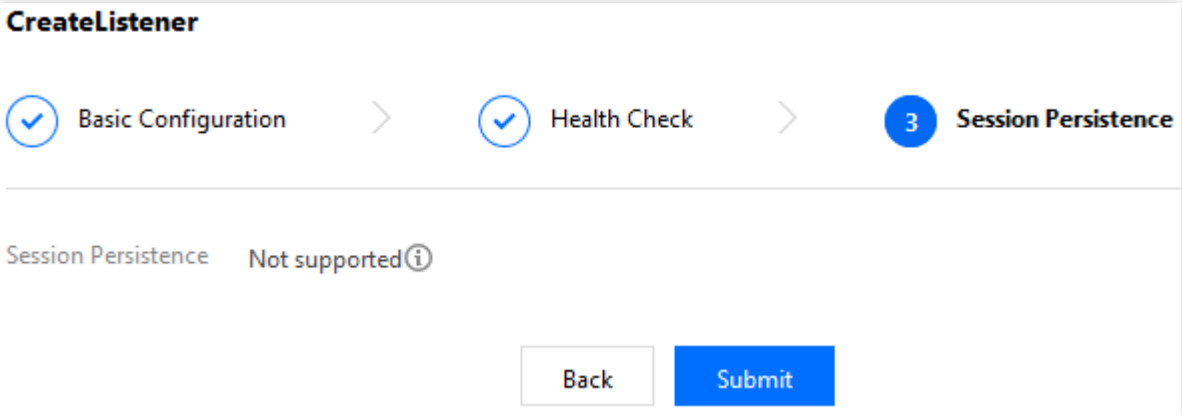
Configuration Item	Description	Example
Health check status	Health check can be enabled or disabled. In TCP SSL listeners, CLB instances send SYN packets to the specified server port to perform health checks.	Enabled
Response timeout period	• Maximum response timeout period for health checks. • If a real server fails to respond correctly within the timeout period, it is considered abnormal. • Value range: 2-60s. Default value: 2s.	2s
Check interval	• Interval between two health checks. • Value range: 5-300s. Default value: 5s.	5s

Configuration Item	Description	Example
Unhealthy threshold	• If the health check results received n times (n is the entered number) in a row are failures, the instance will be considered unhealthy, and the status displayed in the console will be **Abnormal**. • Value range: 2-10. Default value: 3.	3 times
Healthy threshold	• If the health check results received n times (n is the entered number) in a row are successes, the instance will be considered healthy, and the status displayed in the console will be **Healthy**. • Value range: 2-10. Default value: 3.	3 times

The specific configuration of health check is as shown below:

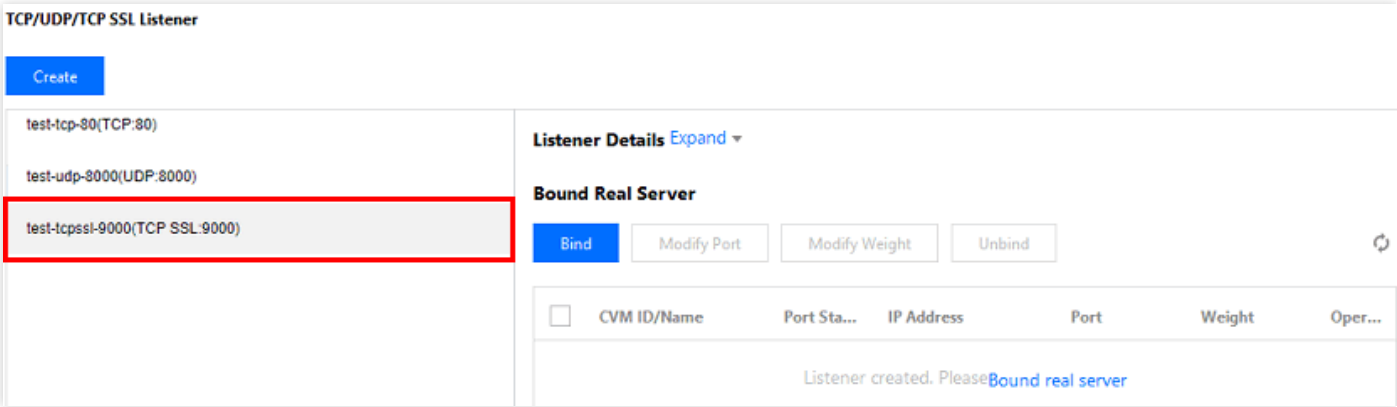


3. Session persistence (not supported currently)

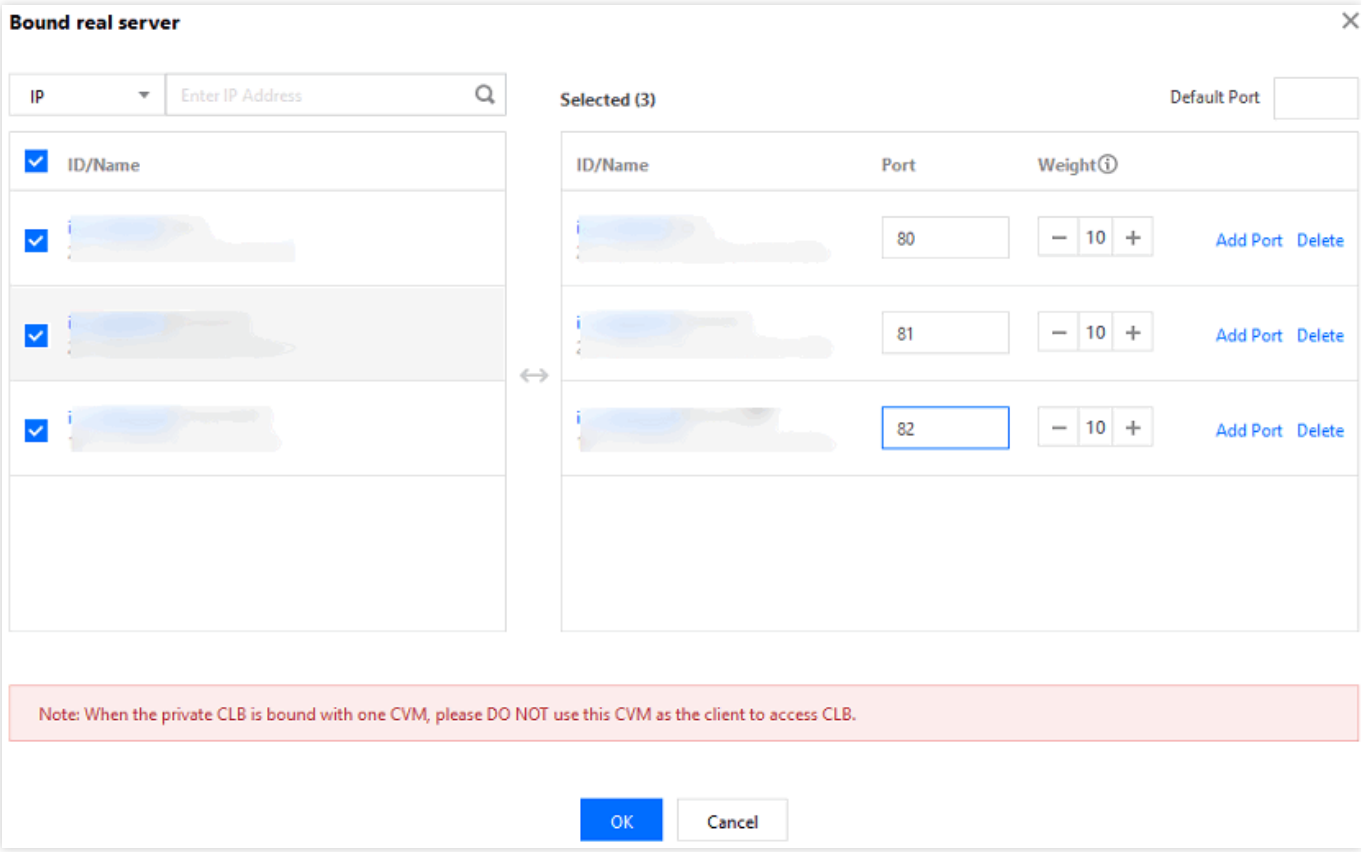


Step 3. Bind a real server

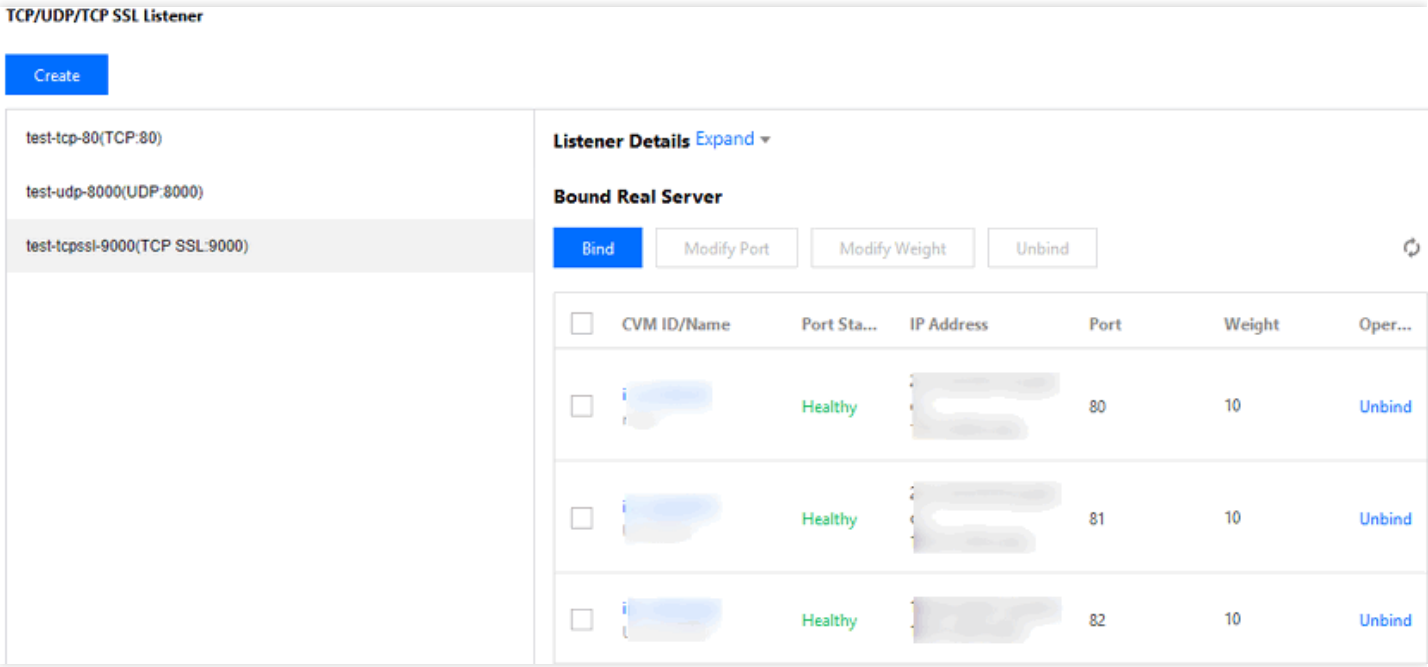
1. On the "Listener Management" page, click the created listener `TCP SSL:9000` to view the bound real servers on the right of the listener.



2. Click **Bind** and select the real server to be bound and configure the server port and weight in the pop-up window.
   i. Add Port: In the "Selected" box on the right, click **Add Port** to add multiple ports for the same CVM instance, such as ports 80, 81, and 82.
   ii. Default Port: Enter the "Default Port" first and then select the CVM instance. The port of every CVM instance is the default port.



After these three steps are completed, the TCP SSL listener rule has been configured as shown below:

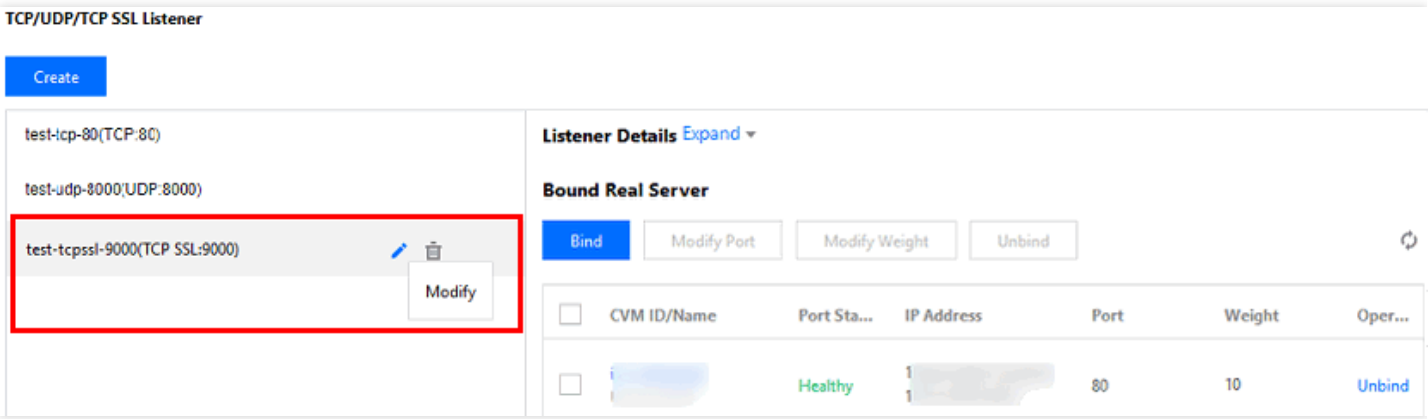


Step 4. Security group (optional)

You can configure a CLB security group to isolate public network traffic. For more information, see Configuring a CLB Security Group.

Step 5. Modify/delete a listener (optional)

If you need to modify or delete a created listener, click the listener on the "Listener Management" page and select **Modify** or **Delete**.

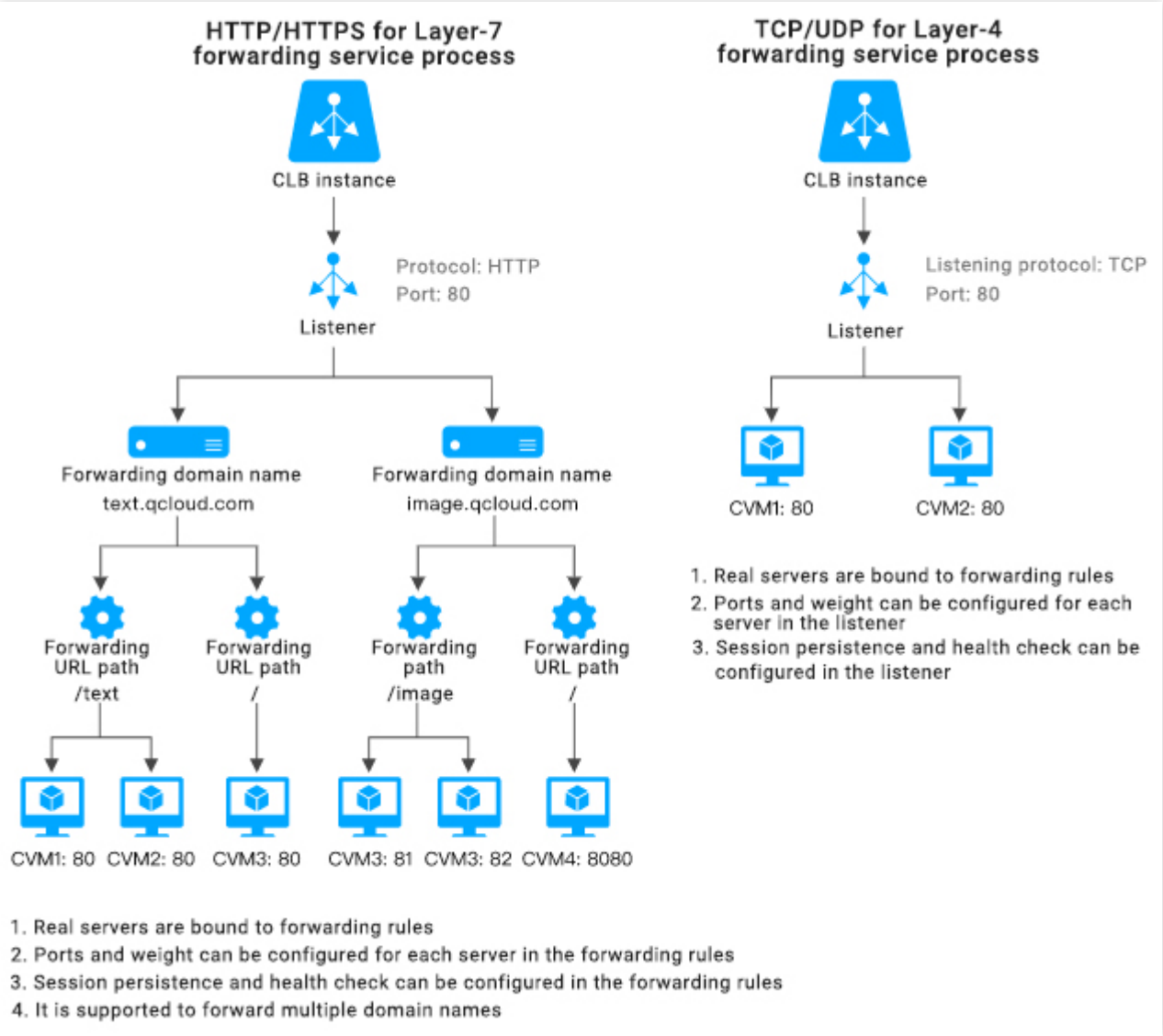


Layer-7 Domain Name Forwarding and URL Rules

Last updated : 2020-08-10 15:34:41

Business Flow Chart

The business flows of layer-7 and layer-4 CLB (formerly application CLB) are as shown below:



If layer-7 CLB is used to forward HTTP/HTTPS protocols, you can add a corresponding domain name when creating the forwarding rule in a CLB listener.

- If only one forwarding rule is created, you can access the corresponding forwarding rule and the service through VIP+URL.

- If multiple forwarding rules are created, the use of VIP+URL does not guarantee access to a specified domain name+URL. You should access a domain name+URL directly to make sure a forwarding rule has taken effect. In other words, when you configure multiple forwarding rules, a VIP may correspond to multiple domain names. In this case, we recommend you access the service via specified domain name+URL instead of VIP+URL.

Layer-7 Forwarding Configuration Description

Forwarded domain name configuration rules

Layer-7 CLB can forward requests from different domain names and URLs to different servers for processing. A layer-7 listener can be configured with multiple domain names, each of which can be configured with multiple forwarding paths.

- Length limit for forwarded domain name: 1–80 characters.
- It cannot begin with `_`.
- It is supported to specify domain names, such as `www.example.com`.
- Wildcard domain names are supported, but currently only those in the format of `*.example.com` or `www.example.*`, that is, wildcard domain names begin or end with `*` which appears only once.
- For non-regex forwarded domain names, valid character sets include `a-z`, `0-9`, `.`, `-` and `_`.
- Forwarded domain name supports regex. Regex domain names:
  - Support character sets including `S`, `a-z`, `0-9`, `.`, `-`, `?`, `=`, `~`, `_`, `-`, `+`, `¥`, `^`, `*`, `!`, `$`, `&`, `|`, `(`, `)`, `[`, and `]`.
  - Must begin with `~` which can appear only once.
  - An example of regex domain name supported by CLB may be `~^www¥d+¥.example¥.com$`.

Forwarded domain name matching description

General matching policy for forwarded domain name

1. If you enter an IP address instead of a domain name in the forwarding rule and configure multiple URLs in the forwarding group, VIP+URLs will be used to access the service.
2. If you configure a full domain name in the forwarding rule and multiple URLs in the forwarding group, domain name+URLs will be used to access the service.
3. If you configure a wildcard domain name in the forwarding rule and multiple URLs in the forwarding group, you will access the service through the matching of requested domain name and URLs. To have different domain names point to the same URL, you can use this method for configuration. Taking `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 with `qcloud.com`.

- `example.qcloud.*` matches all domain names beginning with `example.qcloud`.

4. If you configure a domain name in the forwarding rule and a URL for fuzzy matching in the forwarding group, you can initiate full matching by using prefix matches and adding a suffixed wildcard `$`.
   For example, if you configure `URL ~*.(gif|jpg|bmp)$`. in the forwarding group, hopefully it will match any files that end with gif, jpg, or bmp.

Default-domain-name-policy-for-forwarded-domain-name">

Default domain name policy for forwarded domain name

If a client request cannot be matched with any domain name of this listener, CLB will forward the request to the default domain name (`Default Server`) to make the default rule controllable. Only one default domain name can be configured under one listener.
For example, the `HTTP:80` listener of CLB instance 1 is configured with two domain names: `www.test1.com` and `www.test2.com`, where `www.test1.com` is the default domain name. When a user visits `www.example.com`, since no domain name is matched, CLB will forward the request to the default domain name `www.test1.com`.

- Before May 18, 2020, the default domain name is optional for layer-7 listeners.
  - If your layer-7 listener has a default domain name configured, client requests that do not match other rules will be forwarded to it.
  - If your layer-7 listener has no default domain name configured, client requests that do not match other rules will be forwarded to the first domain name loaded by CLB (its loading order may be different from that configured in the console; therefore, it may not be the first one configured in the console).
- Starting from May 18, 2020:
  - All new layer-7 listeners must have a default domain name configured: the first rule of a layer-7 listener must enable the default domain name. When an API is called to create a layer-7 rule, CLB will automatically set the `DefaultServer` field to `true`.
  - For all listeners that have a default domain name configured, you need to specify a new default domain name when modifying or deleting the existing default domain name: when you perform the operation in the console, you need to specify a new default domain name; when you perform the operation by calling an API, if you do not set a new default domain name, CLB will set the earliest-created one among the remaining domain names as the new default domain name.
  - For existing rules without a default domain name, you can directly configure a default domain name based on your business needs as instructed in operation 4 below. If you don't do so, Tencent Cloud will set the first domain name loaded by CLB as the default

domain name. This operation has no impact on the business. Existing listeners will be all processed before June 19, 2020.

The above policy will be implemented gradually starting from May 18, 2020, and the effective date for each instance may vary slightly. As of June 20, 2020, all layer-7 listeners that have a forwarded domain name will have a default domain name.

The following four operations can be performed on the default domain name:

- **Operation 1**: when configuring the first forwarding rule for the layer-7 listener, the default domain name must be in "enabled" status.
- **Operation 2**: disable the current default domain name.
  - If there are multiple domain names under a listener, when disabling the current default domain name, you need to specify a new default domain name.
  - If a listener has only one domain name and the domain name is the default domain name, it cannot be disabled.
- **Operation 3**: delete the default domain name.
  - If there are multiple domain names under a listener, when you delete a rule under the default domain name:
    - If the rule is not the last rule of the default domain name, you can delete it directly.
    - If the rule is the last rule of the default domain name, you need to set a new default domain name.
  - If there is only one domain name under a listener, you can directly delete all rules without setting a new default domain name.
- **Operation 4**: you can quickly modify the default domain name in the listener list.

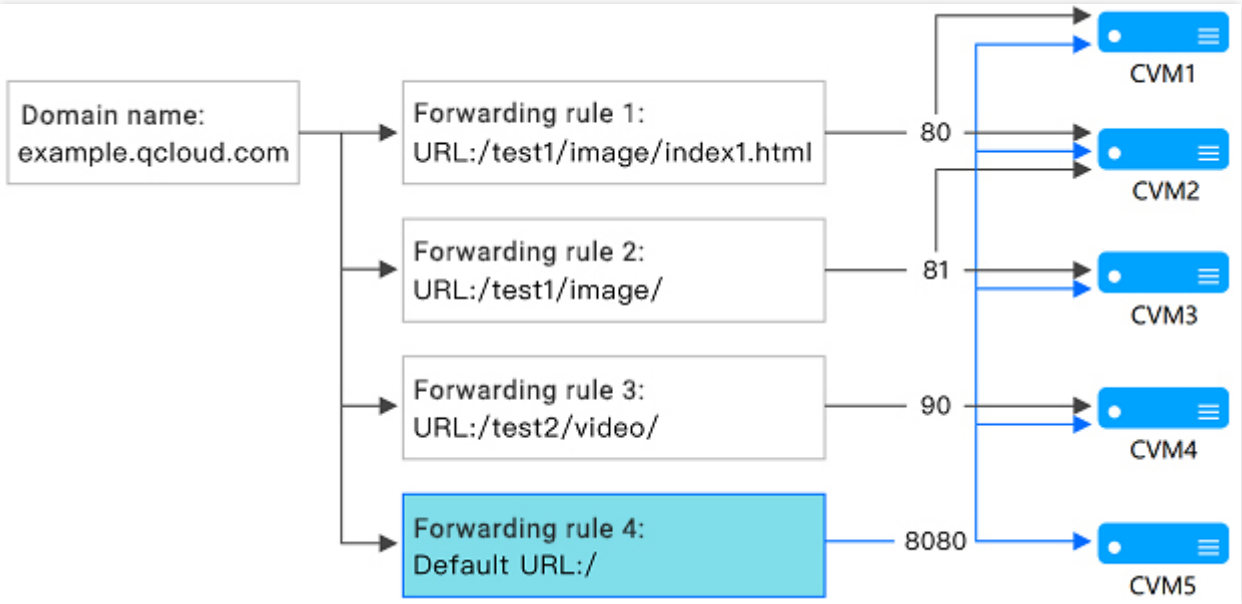
Forwarded URL path configuration rules

Layer-7 CLB can forward requests from different URLs to different servers for processing, and multiple forwarded URL paths can be configured for a single domain name.

- Length limit of forwarded URL: 1–200 characters.
- A non-regex forwarded URL must start with `/`., with valid character sets including `a-z`, `A-Z`, `0-9`, `.`, `-`, `_`, `/`, `=` and `?`.
- Forwarded URL supports regex:
  - A regex URL must begin with `~` which can appear only once.
  - For a regex URL, the valid character sets include `a-z`, `A-Z`, `0-9`, `.`, `-`, `_`, `/`, `=`, `?`, `~`, `^`, `*`, `$`, and `:`.
  - An example of regex URL may be `~*.png$`.

- The matching rules for a forwarded URL are as follows:
  - Beginning with `=` indicates exact match.
  - Beginning with `^~` indicates that the URL starts with a regular string and is not for regex match.
  - Beginning with `~` indicates case-sensitive regex match.
  - Beginning with `~*` indicates case-insensitive regex match.
  - `/` indicates generic match, where any requests will be matched if there are no other matches.

Forwarded URL path matching description



1. Matching rules: based on longest prefix match, exact match is performed first followed by fuzzy match.
   For example, after you configure the forwarding rules and forwarding groups as shown above, the following requests will be matched into different forwarding rules in sequence.
   i. Because `example.qloud.com/test1/image/index1.html` exactly matches the URL rule configured by forwarding rule 1, the request will be forwarded to the real servers associated with forwarding rule 1, i.e., port 80 of CVM1 and CVM2 in the figure.
   ii. Because `example.qloud.com/test1/image/hello.html` has no exact match, it will match forwarding rule 2 based on longest prefix match; therefore, the request will be forwarded to the real servers associated with forwarding rule 2, i.e., port 81 of CVM2 and CVM3 in the figure.
   iii. Because `example.qloud.com/test2/video/mp4/` has no exact match, it will match forwarding rule 3 based on longest prefix match; therefore, the request will be forwarded to the real server associated with forwarding rule 3, i.e., port 90 of CVM4 in the figure.

iv. Because `example.qloud.com/test3/hello/index.html` has no exact match, it will match the root directory's default URL `example.qloud.com/` by longest prefix match. In this case, Nginx will forward the request to the real server such as FastCGI (php) or Tomcat (jsp), while Nginx will exist as a reverse proxy server.
v. Because `example.qloud.com/test2/` has no exact match, it will match the root directory's default URL `example.qloud.com/` by longest prefix match.

2. If the service does not work properly in the set URL rules, it will not be redirected to other pages after successful match.
   For example, the client requests `example.qloud.com/test1/image/index1.html` and matches it with forwarding rule 1. However, the real server of forwarding rule 1 has an exception and a 404 error page appears. You will see the 404 error page, but not being redirected to other pages.
3. You are recommended to point the default URL to a stable page (such as a static page or homepage) and bind it to all real servers. If none of the rules match, the system will point the request to the default URL page; otherwise, a 404 error may occur.
4. If you do not set the default URL, and none of the forwarding rules match, a 404 error will be returned when you access the service.
5. Note on the slash at the end of the layer-7 URL path: if the URL you set ends with `/`, but the access request from the client does not contain `/`, then the request will be redirected to a rule ending with `/` (301 redirect).
   For example, under the `HTTP:80` listener, the configured domain name is `www.test.com`:
   i. If the URL set under this domain name is `/abc/`:
      - When the client accesses `www.test.com/abc`, it will be redirected to `www.test.com/abc/`.
      - When the client accesses `www.test.com/abc/`, it will match `www.test.com/abc/`.
   ii. If the URL set under this domain name is `/abc`:
      - When the client accesses `www.test.com/abc`, it will match `www.test.com/abc`.
      - When the client accesses `www.test.com/abc/`, it will also match `www.test.com/abc`.

Layer-7 Health Check Configuration Description

Health check domain name configuration rules

A health check domain name is the domain name used by layer-7 CLB to detect the health status of a real server.

- Length limit: 1–80 characters.
- Default: forwarded domain name.
- Regex is not supported. If your forwarded domain name is a wildcard domain name, you should specify a fixed one (non-regex).
- Valid character sets include `a-z`, `0-9`, `.`, `-`, and `_`.

Health check path configuration rules

A health check path is the URL path used by layer-7 CLB to detect the health status of a real server.

- Length limit: 1–200 characters.
- Default: `/`, with which the path must begin.
- Regex is not supported. You are recommended to specify a fixed URL (static page) for health check.
- Valid character sets include `a-z`, `A-Z`, `0-9`, `.`, `-`, `_`, `/`, `=`, `?`, and `:`.

SNI Support for Binding Multiple Certificates to a CLB Instance

Last updated : 2020-10-20 10:36:41

Server Name Indication (SNI) is designed to solve the problem that one server can only use one certificate so as to improve SSL/TLS extensions of the server and the client. If a server supports SNI, it means that the server can be bound to multiple certificates. To use SNI for the client, the domain name to connect to should be specified before SSL/TLS connections to the server are established, and then the server will return an appropriate certificate based on the domain name.

Use Cases

A layer-7 HTTPS CLB listener supports SNI, i.e., binding multiple certificates, which can be used by different domain names in the listening rules. For example, in the same `HTTPS:443` listener of a CLB instance, you can use certificate 1 and certificate 2 for `*.test.com` and `*.example.com` respectively to forward requests from these domain names to two different sets of servers.

Prerequisites

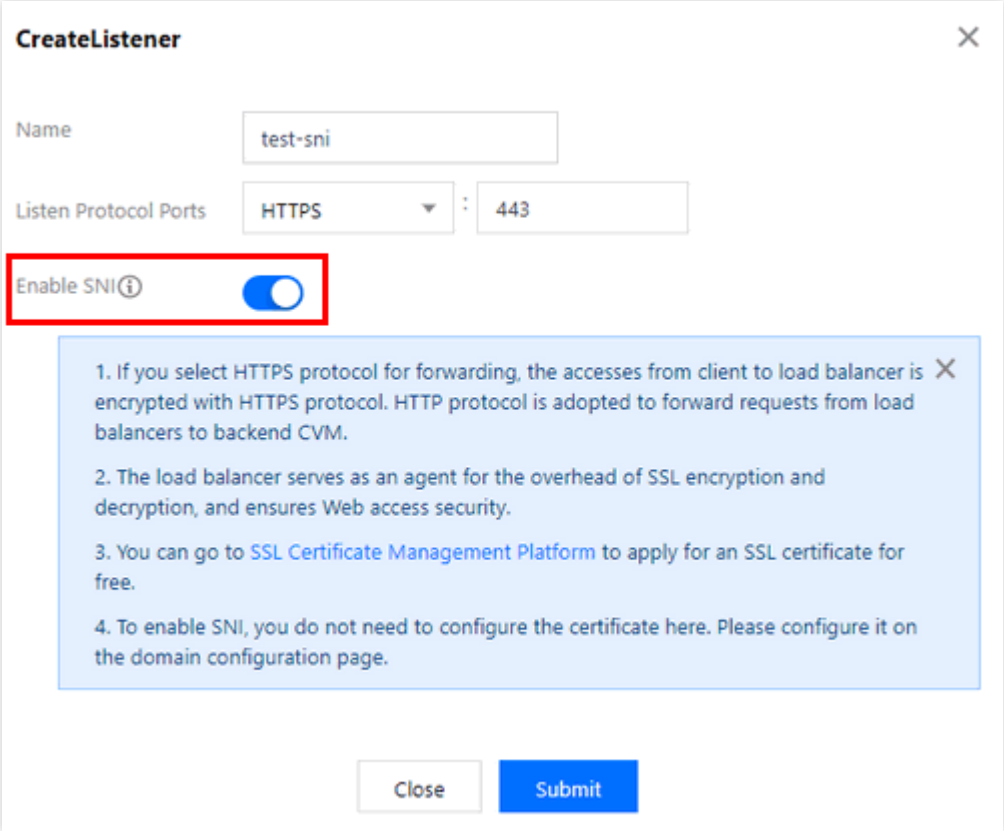
You have purchased a CLB instance.

> Classic CLB does not support forwarding based on domain name and URL; therefore, it does not support SNI.

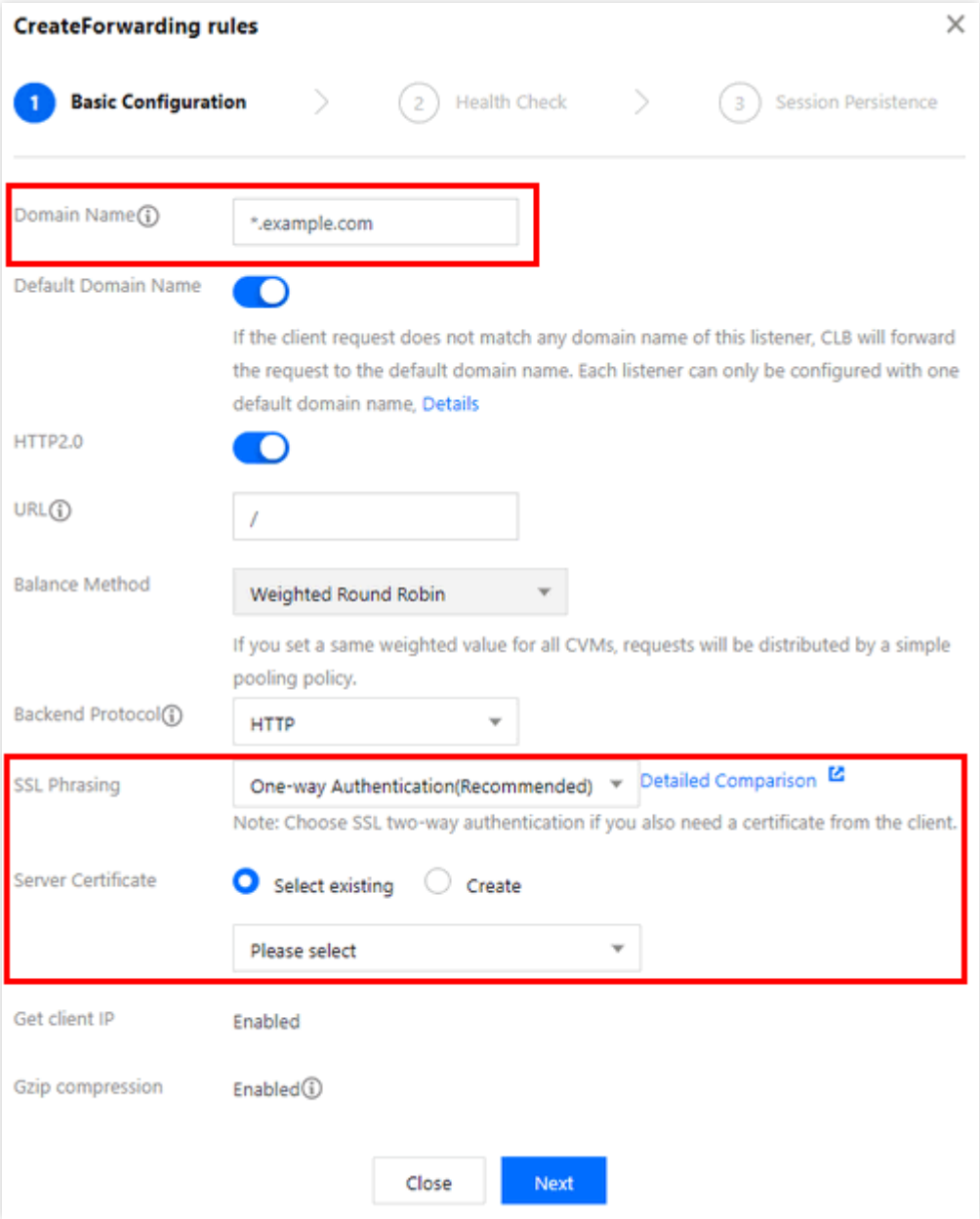
Directions

1. Log in to the CLB console.

2. Configure an HTTPS listener and enable SNI.



3. When adding a forwarding rule to the listener, configure different server certificates for different domain names. Then, click **Next** and configure health check and session persistence.



Configuring Layer-7 Redirection

Last updated : 2020-05-21 18:19:36

CLB supports layer-7 redirection, so that you can configure redirection on layer-7 HTTP/HTTPS listeners.

> - Session persistence: if the client accesses `example.com/bbs/test/123.html` and session persistence has been enabled on the backend CVM, after redirection is enabled to forward traffic to `example.com/bbs/test/456.html` , the original session persistence mechanism will not take effect.

- TCP/UDP redirection: redirection at IP + port level is not supported currently but will be available in subsequent versions.

Redirection Overview

1. Automatic redirection
   - Overview
     For an existing `HTTPS:443` listener, an HTTP listener (port 80) will be created automatically by the system for forwarding. Requests sent to `HTTP:80` will be automatically redirected to `HTTPS:443` .
   - Use case
     Forced HTTPS redirection, i.e., redirecting HTTP requests to HTTPS. When a user accesses a web service in a PC or mobile browser over HTTP, CLB will redirect all requests sent to `HTTP:80` to `HTTPS:443` for forwarding.
   - Scheme advantages
     - Set-and-forget configuration: forced HTTPS redirection can be implemented for a domain name with only one configuration operation needed.
     - Convenient update: if the number of URLs of the HTTPS service changes, you only need to use this feature again in the console for refreshing.
2. Manual redirection
   - Overview
     You can configure 1-to-1 redirection. For example, in a CLB instance, you can configure redirection of `listener 1 / domain name 1 / URL 1` to `listener 2 / domain name 2 / URL 2` .

- Use case
  Single-path redirection. For example, if you want to temporarily deactivate your web business in cases such as product sellout, page maintenance, or update and upgrade, the original page needs to be redirected to a new page. If no redirection is performed, the old address in a visitor's favorites and search engine database will return a `404/503` error message page, degrading the user experience and resulting in traffic waste.

Automatic Redirection

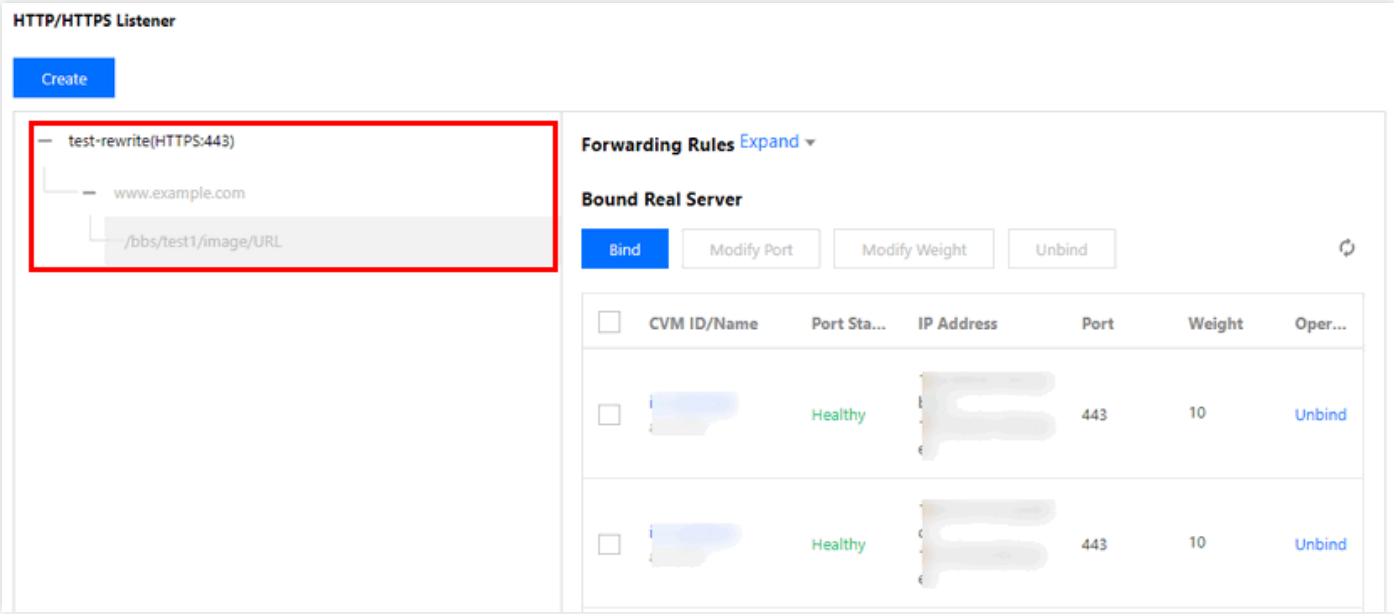
CLB supports one-click forced redirection from HTTP to HTTPS.
Assume that you need to configure the website `https://www.example.com` , so that end users can visit it securely over HTTPS no matter whether they send HTTP requests (`http://www.example.com`) or HTTPS requests (`https://www.example.com`) in the browser.

Prerequisites

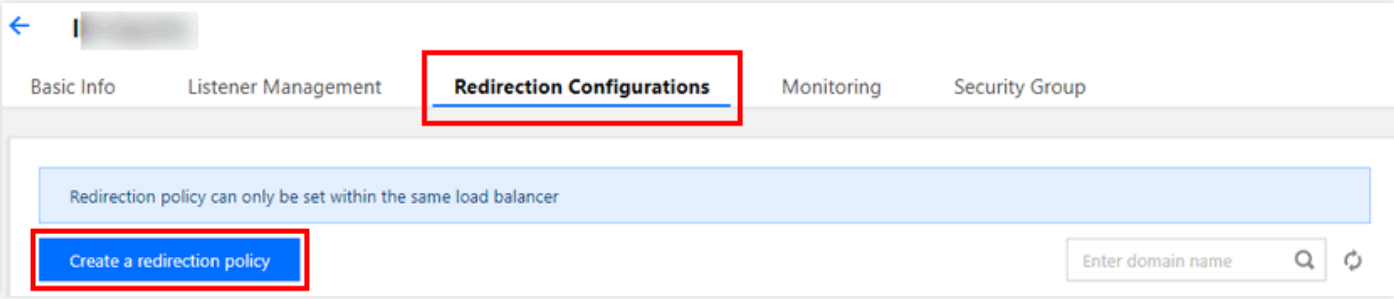
The `HTTPS:443` listener has been configured.

Directions

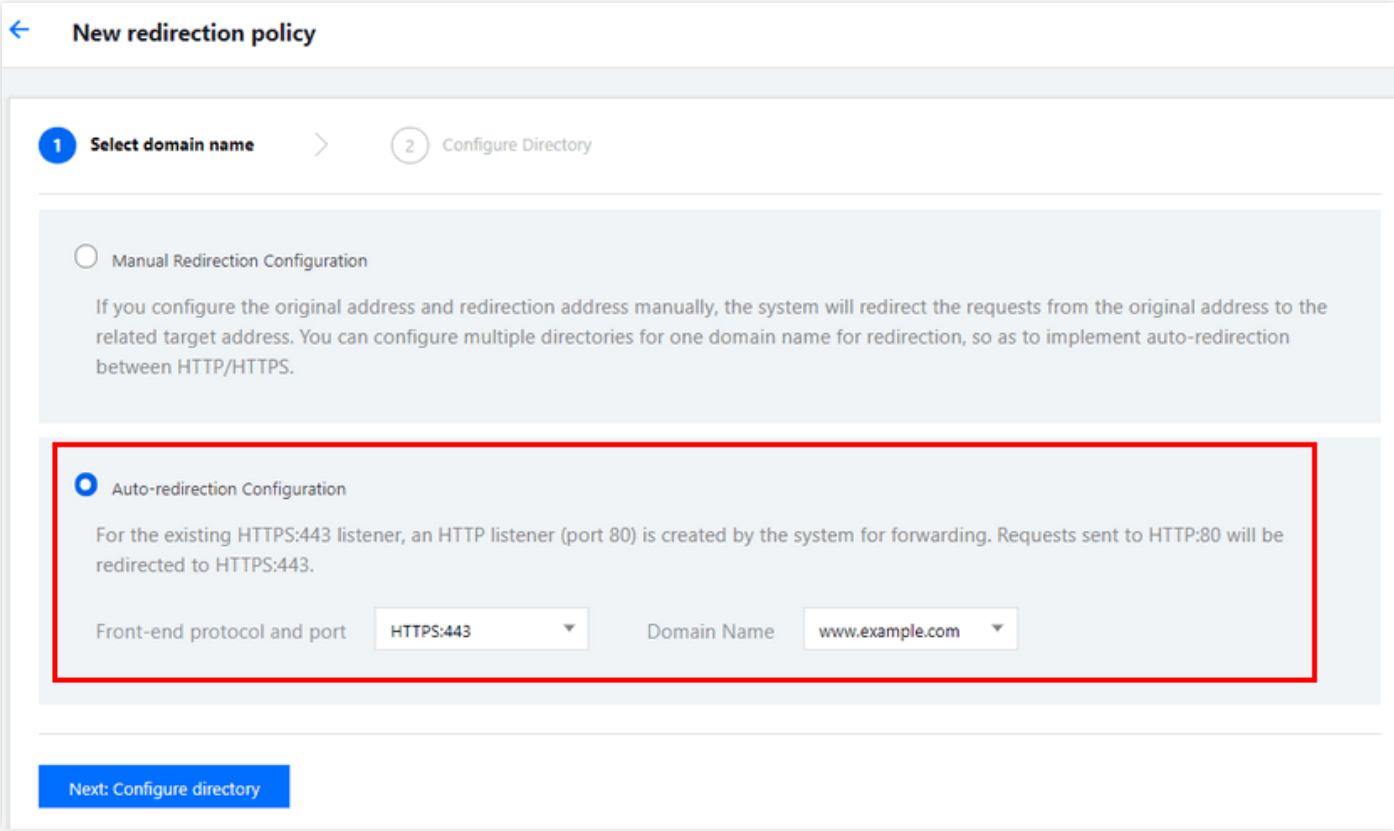
1. Configure the CLB HTTPS listener in the CLB Console and set up the web environment of `https://example.com` . For more information, please see Configuring an HTTPS Listener .
2. The result of the HTTPS listener configuration is as shown below:



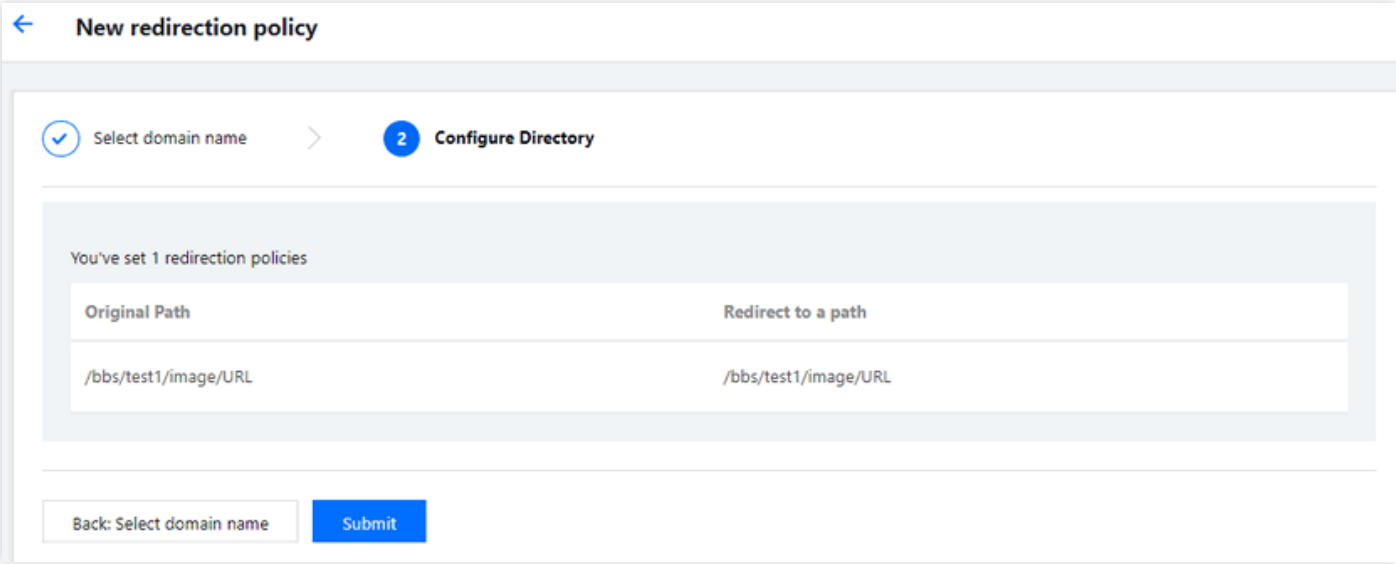
3. On the "Redirection Configuration" tab in CLB instance details, click **Create Redirection Configuration**.



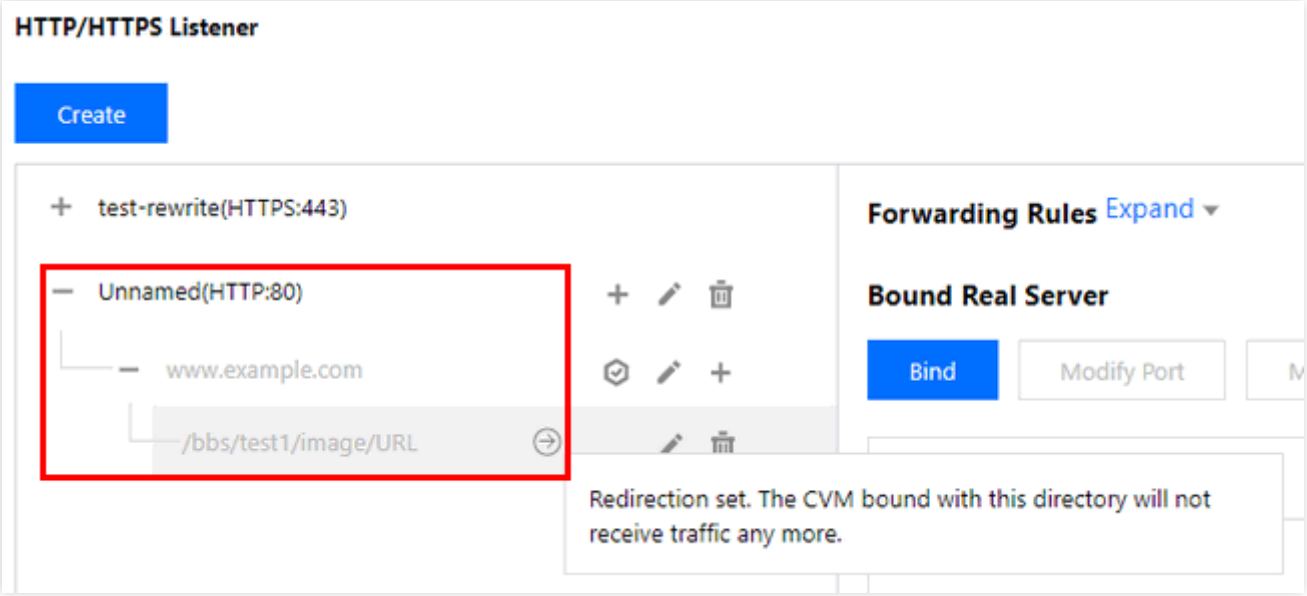
4. Select **Automatic Redirection Configuration**, select the configured HTTPS listener and domain name, and click **Next: Configure Path**.



5. Click **Submit** to complete the configuration.



6. The result after the redirection is configured is as shown below. As you can see, the `HTTP:80` listener has been automatically configured for the `HTTPS:443` listener, and all HTTP traffic will be automatically redirected to HTTPS.



Manual Redirection

CLB supports configuring 1-to-1 redirection.

For example, your business uses a `forsale` page for a promotion campaign and needs to redirect

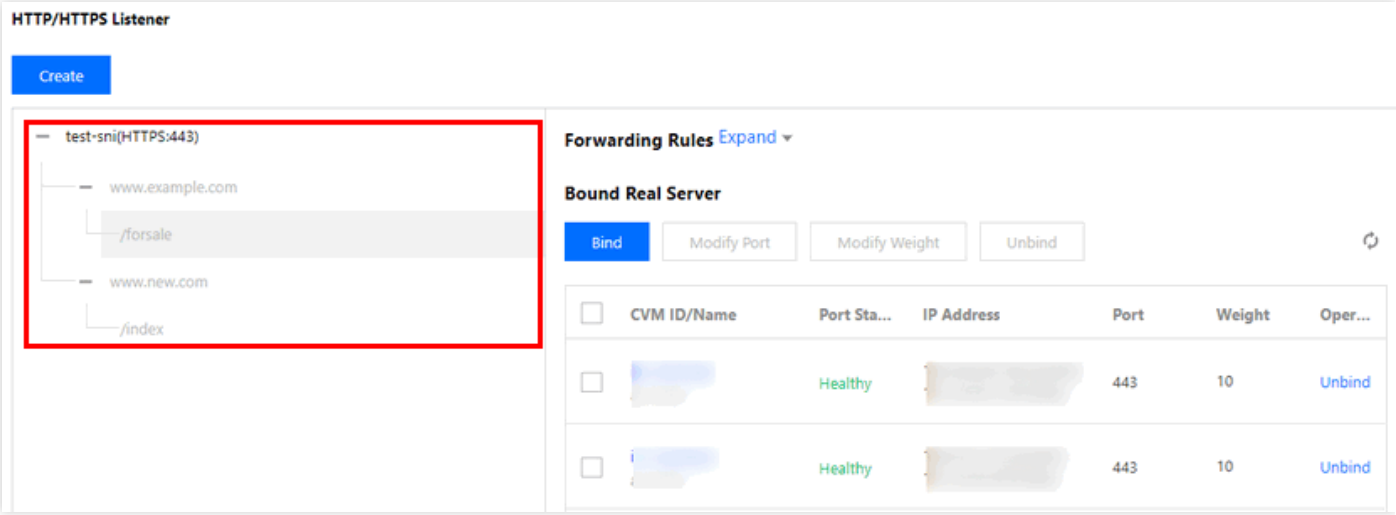
the campaign page `https://www.example.com/forsale` to the new homepage `https://www.new.com/index` after the campaign ends.

Prerequisites

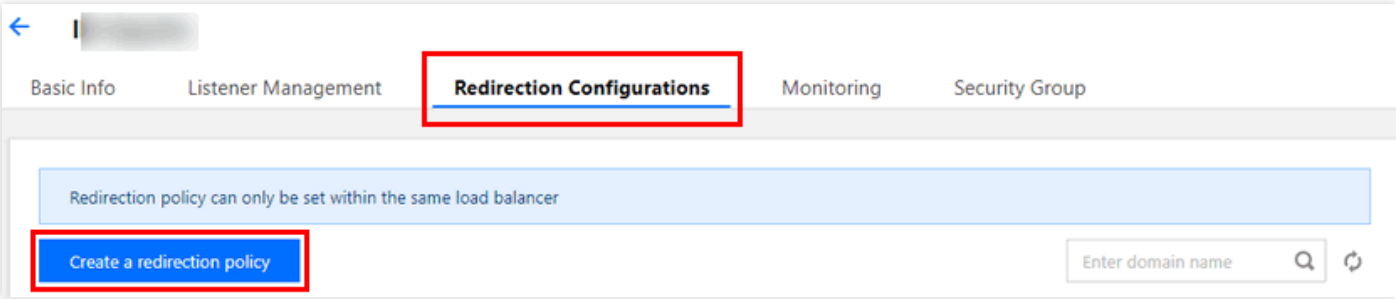
- An HTTPS listener has been configured.
- The forwarded domain name `https://www.example.com/forsale` has been configured.
- The forwarded domain name and path `https://www.new.com/index` has been configured.

Directions

1. Configure the CLB HTTPS listener in the CLB Console and set up the web environment of `https://example.com` . For more information, please see Configuring HTTPS Listener.
2. The result of the HTTPS configuration is as shown below:

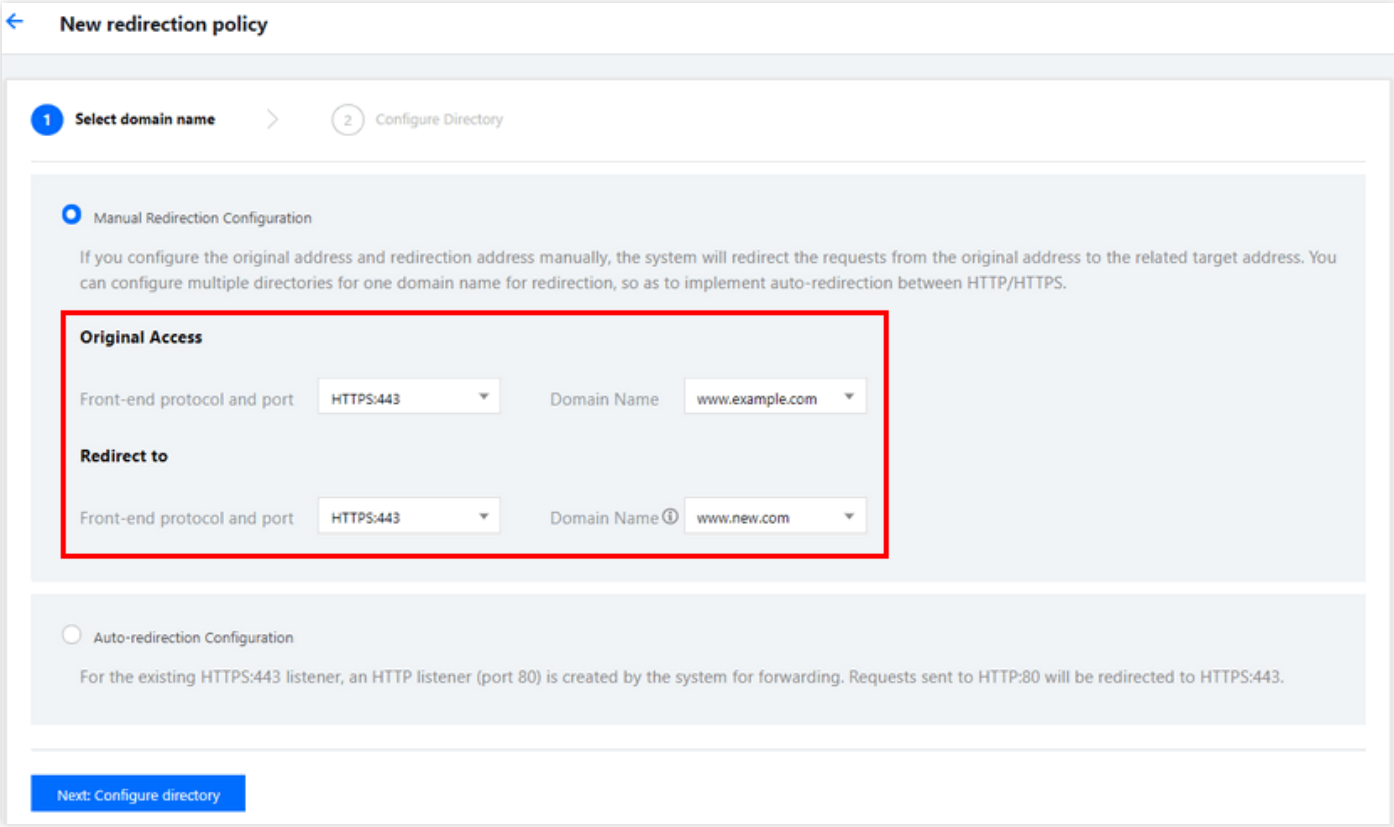


3. On the "Redirection Configuration" tab in CLB instance details, click **Create Redirection Configuration**.

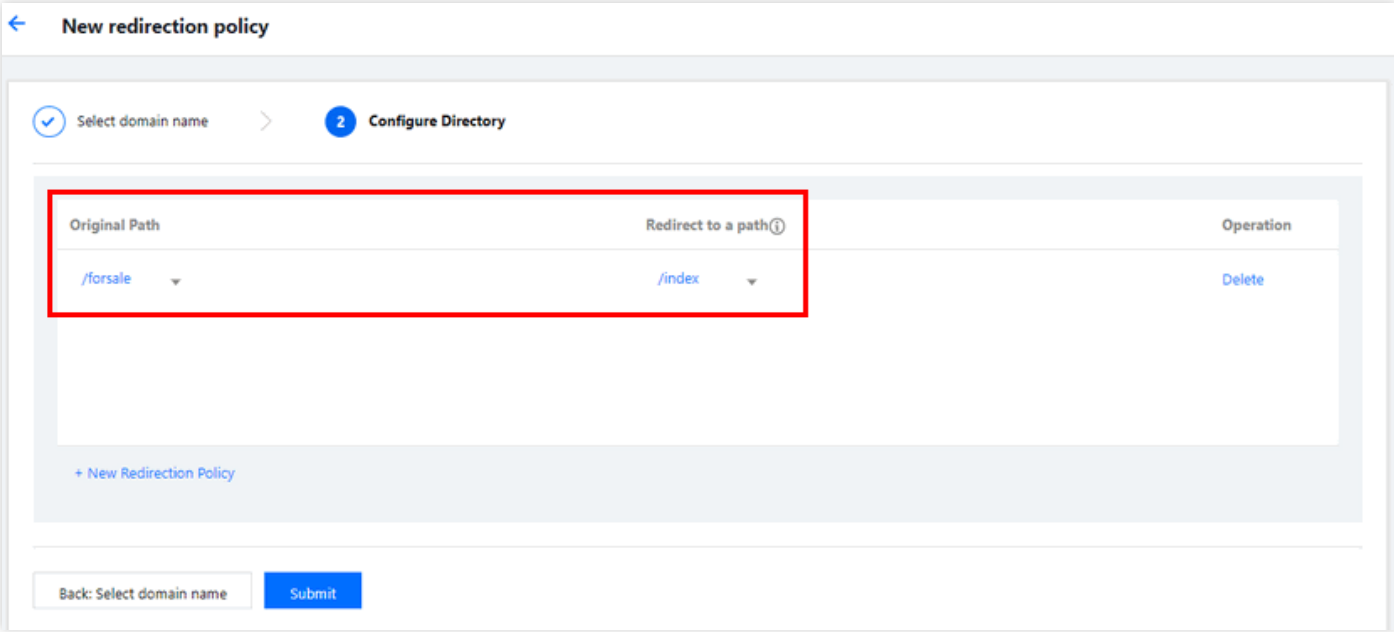


4. Select **Manual Redirection Configuration**, select the originally accessed frontend protocol port `HTTPS:443` and domain name `https://www.example.com/forsale` , select the frontend protocol port `HTTPS:443` and domain name `https://www.new.com/index` after redirection, and click **Next:**

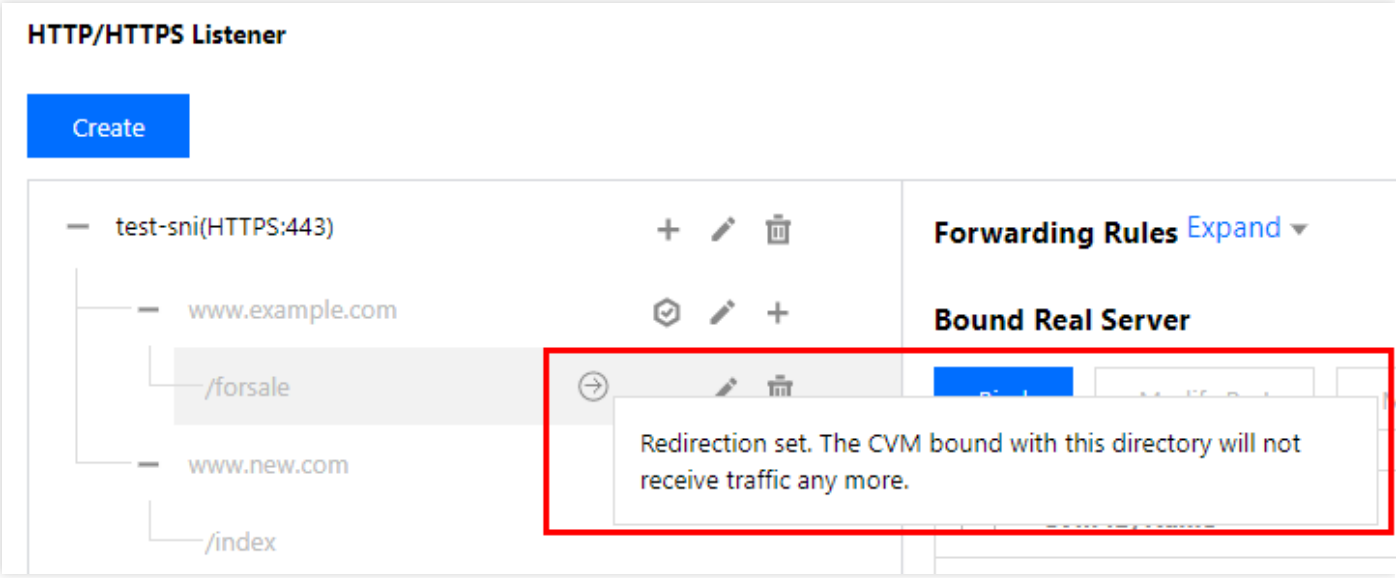
Configure Path.



5. Select `/forsale` for the original access path and `/index` for the access path after redirection, and click **Submit** to complete the configuration.



6. The result of the redirection configuration is as shown below. As you can see, in the `HTTPS:443` listener, `https://www.example.com/forsale` has been redirected to `https://www.new.com/index`.



Custom Configurations of CLB Instances

Last updated : 2019-10-18 13:17:47

CLB supports custom configuration, allowing you to set the configuration parameters for a single CLB instance, such as `client_max_body_size` and `ssl_protocols`, so as to meet your unique needs.

> The CLB custom configuration feature is currently in beta test. If you want to use it, please submit a ticket for application.

CLB Custom Configuration Parameter Descriptions

Currently, CLB custom configuration supports the following fields:

Configuration Field	Default Value/Recommended Value	Parameter Range	Description
ssl_protocols	TLSv1 TLSv1.1 TLSv1.2	TLSv1 TLSv1.1 TLSv1.2	Version of TLS protocol used; TLSv1.3 will be added later.
ssl_ciphers	See further below	See further below	Encryption suite.
client_header_timeout	60s	[30-120]s	Timeout period of obtaining a client request header; in case of timeout, a 408 error will be returned.
client_header_buffer_size	4 KB	[1-64] KB	Size of default buffer where a client request header is stored.

Configuration Field	Default Value/Recommended Value	Parameter Range	Description
client_body_timeout	60s	[30-120]s	Timeout period of obtaining a client request body, which is not the time for obtaining the entire body but refers to the idle period without data transmission; in case of timeout, a 408 error will be returned.
client_max_body_size	60 MB	[1-256] MB	Maximum size of client request body, which may need to be modified for uploads; if the maximum size is exceeded, a 413 error will be returned.
keepalive_timeout	75s	[0-3,600]s	Client-server persistent connection hold time; if it is set to 0, persistent connection is prohibited.
add_header	Custom	-	Specific header field returned to the client in the format of add_header xxx yyy.
more_set_headers	Custom	-	Specific header field returned to the client in the format of more_set_headers "A:B".
proxy_connect_timeout	4s	[4-120]s	Timeout period of upstream backend connection.
proxy_read_timeout	60s	[30-3,600]s	Timeout period of reading upstream backend response.
proxy_send_timeout	60s	[30-3,600]s	Timeout period of sending a request to the upstream backend.

Configuration Field	Default Value/Recommended Value	Parameter Range	Description
server_tokens	on	on; off	on means displaying version information, while off means hiding version information.
keepalive_requests	100	[0-10,000]	Maximum number of requests that can be sent over the client-server persistent connection.
proxy_buffer_size	4 KB	[4-16] KB	Size of server response header, which is the size of a single buffer set in `proxy_buffer` by default; to use `proxy_buffer_size`, `proxy_buffers` must be set at the same time.
proxy_buffers	1; 4 KB	[1-8] [4-8] KB	Buffer quantity and size.
proxy_set_header	X-Real-Port $remote_port	-	proxy_set_header only supports `X-Real-Port $remote_port` but not other custom fields.

ssl_ciphers Configuration Instructions

The ssl_ciphers encryption suite being configured must be in the same format as that used by OpenSSL. The algorithm list is one or more `<cipher strings>`; multiple algorithms should be separated with ":"; ALL represents all algorithms, "!" indicates not to enable an algorithm, and "+" indicates to move an algorithm to the last place.
The encryption algorithm for default forced disabling is: `!aNULL:!eNULL:!EXPORT:!DES:!RC4:!MD5:!PSK:!DHE`.

Default value:

```
ECDHE-RSA-AES128-GCM-SHA256:ECDHE-ECDSA-AES128-GCM-SHA256:ECDHE-RSA-AES256-GCM-SHA384:ECDHE-ECDSA
-AES256-GCM-SHA384:ECDHE-RSA-CHACHA20-POLY1305:kEDH+AESGCM:ECDHE-RSA-AES128-SHA256:ECDHE-ECDSA-AE
S128-SHA256:ECDHE-RSA-AES128-SHA:ECDHE-ECDSA-AES128-SHA:ECDHE-RSA-AES256-SHA384:ECDHE-ECDSA-AES25
```

```
6-SHA384:ECDHE-RSA-AES256-SHA:ECDHE-ECDSA-AES256-SHA:AES128-GCM-SHA256:AES256-GCM-SHA384:AES128:A
ES256:AES:HIGH:!aNULL:!eNULL:!EXPORT:!DES:!RC4:!MD5:!PSK:!DHE:3DES;
```

Parameter range:

```
ECDH-ECDSA-AES128-SHA256:ECDH-RSA-AES256-SHA:ECDH-ECDSA-AES256-SHA:SRP-DSS-AES-256-CBC-SHA:SRP-AE
S-128-CBC-SHA:ECDH-RSA-AES128-SHA256:DH-RSA-AES128-SHA256:DH-RSA-CAMELLIA128-SHA:DH-DSS-AES256-GC
M-SHA384:DH-RSA-AES256-SHA256:AES256-SHA256:SEED-SHA:CAMELLIA256-SHA:ECDH-RSA-AES256-SHA384:ECDH-
ECDSA-AES128-GCM-SHA256:DH-RSA-AES128-SHA:DH-RSA-AES128-GCM-SHA256:DH-DSS-AES128-SHA:ECDH-RSA-AES
128-SHA:DH-DSS-CAMELLIA256-SHA:SRP-AES-256-CBC-SHA:DH-DSS-AES128-SHA256:SRP-RSA-AES-256-CBC-SHA:E
CDH-ECDSA-AES256-GCM-SHA384:ECDH-RSA-AES256-GCM-SHA384:DH-DSS-AES256-SHA256:ECDH-ECDSA-AES256-SHA
384:AES128-SHA:DH-DSS-AES128-GCM-SHA256:AES128-SHA256:DH-RSA-SEED-SHA:ECDH-ECDSA-AES128-SHA:IDEA-
CBC-SHA:AES128-GCM-SHA256:DH-RSA-CAMELLIA256-SHA:CAMELLIA128-SHA:DH-RSA-AES256-GCM-SHA384:SRP-RSA
-AES-128-CBC-SHA:SRP-DSS-AES-128-CBC-SHA:ECDH-RSA-AES128-GCM-SHA256:DH-DSS-CAMELLIA128-SHA:DH-DSS
-SEED-SHA:AES256-SHA:DH-RSA-AES256-SHA:kEDH+AESGCM:AES256-GCM-SHA384:DH-DSS-AES256-SHA:HIGH:AES12
8:AES256:AES:!aNULL:!eNULL:!EXPORT:!DES:!RC4:!MD5:!PSK:!DHE
```

CLB Custom Configuration Examples

1. Log in to the CLB Console, click the custom configuration page on the left sidebar, and click **Create** to create a custom configuration file where configuration items should end with `;`.

2. Click **Bind to Instance** and select the CLB instance that you need to bind to in the same region.

3. You can view the corresponding custom configuration information on the instance list page.

Default configuration sample code:

```
ssl_protocols TLSv1 TLSv1.1 TLSv1.2;
client_header_timeout 60s;
client_header_buffer_size 4k;
client_body_timeout 60s;
client_max_body_size 60M;
keepalive_timeout 75s;
add_header xxx yyy;
more_set_headers "A:B";
proxy_connect_timeout 4s;
proxy_read_timeout 60s;
proxy_send_timeout 60s;
```

- Each region can have up to 200 custom configurations.
- Currently, one instance can be bound to only one custom configuration.
- Custom configurations are valid only for HTTP/HTTPS **CLB (former Application CLB)** listeners.

Using QUIC Protocol on CLB

Last updated : 2020-08-05 15:40:54

The QUIC protocol helps you access applications faster and achieves multiplexing with no reconnection required in scenarios such as weak network or frequent switch between Wi-Fi and 4G. This document introduces how to configure QUIC protocol in the CLB Console.

QUIC Overview

Quick UDP Internet Connection (QUIC is a transport layer network protocol designed by Google, multiplexing concurrent data streams using UDP. Compared with the popular TCP+TLS+HTTP2 protocol, QUIC has the following advantages:

- Reduce the time to establish a connection.
- Improve congestion control.
- Multiplex without head-of-line (HOL) blocking.
- Connection migration.

After QUIC is enabled, the client can establish a QUIC connection with a CLB instance. If the QUIC connection fails due to negotiation between the client and the CLB instance, HTTPS or HTTP/2 will be used. However, the CLB instance and the real server still use the HTTP1.x protocol.

> **Note :**
>
> Currently, CLB supports QUIC Q044 and earlier versions.

Use Limits

- The QUIC protocol in CLB is currently in beta test. To use it, please submit an application.
- The QUIC protocol is now available in Beijing, Shanghai, and Mumbai regions.
- Currently, only public network CLB with Layer-7 HTTPS listeners supports the QUIC protocol.

Directions

1. Create a CLB instance as needed. For more information, see Creating CLB Instances.

> **Note :**
>
> When creating a CLB instance, select “Beijing”, “Shanghai” or “Mumbai” for **Region**, and “Public network” for **Network type**.

2. Log in to the CLB Console, and click **CLB Instance List** on the left sidebar.

3. On the **Instance Management** page, select the **Cloud Load Balancer** tab.

4. Locate the public network CLB instance created in Beijing, Shanghai or Mumbai region, and click **Configure listener** under the **Operation** column.

5. On the **Listener Management** page, click **Create** under **HTTP/HTTPS Listener**.

6. On the **CreateListener** page, choose “HTTPS” for **Listen Protocol Ports**. Complete other configurations, and click **Submit**.

7. On the **Listener Management** page, click the **+** symbol next to the listener you just created.

8. On the **CreateForwarding rules** page, enable **QUIC** and create a Layer-7 rule. Fill in relevant fields and click **Next** to complete the basic configuration.

> **Note :**
>
> - Currently, a HTTPS listener can only enable the QUIC protocol for one domain name.
> - If you enabled the QUIC protocol when creating a HTTPS listener, you can enable or disable the QUIC protocol later as needed. If you did not enable the QUIC protocol when creating a HTTPS listener, you cannot enable it later.
> - Based on the UDP protocol, QUIC will use the UDP port of a CLB instance. If you enable QUIC for a HTTPS listener, UDP and TCP ports will be used. For example, you enable QUIC for the HTTPS:443 listener, both TCP:443 and UDP:443 ports are used, and you cannot create the TCP:443 or UDP:443 listener.

Subsequent Operations

After the basic configuration is completed, you can configure health check and session persistence.