Cloud Load Balancer

Product Documentation

Contents

Cloud Load Balancer 4
- Product Introduction 4
  - CLB Overview 4
  - Use Cases 7
  - Strengths 9
  - Principles 10
  - Use limits 12
- Quick Start 14
  - Quick Start for CLB 14
  - Deploying Nginx on CentOS 25
  - Deploy Java Web in CentOS 28
- Operation Guide 35
  - CLB Instance 35
    - Creating a CLB Instance 35
    - Creating the IPv6 NAT64 CLB Instance 37
    - Deleting a CLB Instance 39
    - Configuring a CLB Security Group 40
    - Adjusting the Bandwidth Configuration of the Instance 47
  - CLB Listener 49
    - CLB Listener Overview 49
    - Configuring TCP Listener 52
    - Configuring UDP Listener 57
    - Configuring HTTP Listeners 62
    - Configuring an HTTPS Listener 70
    - Round-Robin Method 79
    - Health Examination 83
    - Session persistence 88
    - Certificate configuration 91
    - Configuring Layer-7 Redirection 95
    - Layer-7 personalized configuration 100
    - Layer-7 forwarding domain name and URL rule description 106
    - CLB supports SNI multi-domain certificates 112
  - Backend CVM 113
    - Backend CVM Overview 113
    - Add, modify, and unbind backend servers 114
    - Binding Elastic Network Interface 118
    - Backend CVM security group configuration instructions 120
  - Monitoring and Alarms 123
    - Obtain monitoring data 123
    - Monitoring metric description 125
    - Configure Alarm Policy 129
    - Alarm Metric Description 132
  - Certificate Management 134
    - Management Certificates 134
    - Certificate Requirements and Certificate Format Conversion 137
  - Log Management 141
    - View operation logs 141
    - Configuring Access Logs 143
  - Cloud Access Management 146
    - Overview 146
    - Authorization Definition 148
    - Policy Examples 154
  - Quota Management 156
- Best Practice 158
  - Method Description of CLB Enabling Gzip Configuration and Detection 158
  - Getting Started Guide for HTTPS Forwarding Configuration 161
  - How To Obtain the Client's Real IP? 164
  - Instructions on Multi Availability Zone High Availability Config 170
  - SSL certificate format requirements and format conversion instructions 171
  - Balancing Algorithm Selection and Weight Configuration Instance 175
- Purchase Guide 179
  - Billing Overview 179
  - Billing 182
    - Instance Pricing 182
    - Network Pricing 184
  - Purchase Methods 186
  - Payment Overdue 189
- Troubleshooting 191
  - UDP Health Check Exception 191
- API Documentation 195
  - Load balancer(clb) 195
    - version(2018-03-17) 195
      - API overview 195
      - CallingMethod 198
        - signaturev1 198
        - signaturev3 206
        - request 215
        - result 216
        - PublicParameter 219
      - CLB APIs 221
        - AutoRewrite 221
        - BatchDeregisterTargets 223
        - BatchModifyTargetWeight 226
        - BatchRegisterTargets 228
        - CreateListener 230
        - CreateLoadBalancer 232
        - CreateRule 233
        - DeleteCert 235
        - DeleteListener 237

- DeleteLoadBalancer ... 239
- DeleteRewrite ... 241
- DeleteRule ... 243
- DeregisterTargets ... 244
- DescribeAppIdLabel ... 246
- DescribeCerts ... 248
- DescribeCustomizedConfigAssociateList ... 250
- DescribeCustomizedConfigList ... 252
- DescribeIspInfo ... 254
- DescribeListenerListByCertId ... 256
- DescribeListeners ... 258
- DescribeLoadBalancers ... 261
- DescribeRewrite ... 262
- DescribeSubUinQuotas ... 264
- DescribeTargetHealth ... 266
- DescribeTargets ... 268
- DescribeTaskStatus ... 270
- InquiryPriceModifyLoadBalancer ... 272
- ManualRewrite ... 274
- ModifyCertAlias ... 276
- ModifyDomainAttributes ... 278
- ModifyListener ... 279
- ModifyLoadBalancerAttributes ... 280
- ModifyRule ... 281
- ModifyTargetPort ... 282
- ModifyTargetWeight ... 283
- RegisterTargets ... 284
- ReplaceCert ... 286
- SetCustomizedConfigForLoadBalancer ... 287
- SetSubUinQuotas ... 289
- UpLoadCert ... 291

• Other APIs ... 293
- DescribeLoadBalancerListByCertId ... 293
- SetLoadBalancerSecurityGroups ... 295
- SetSecurityGroupForLoadbalancers ... 296

• Data structure ... 298

• ErrorCode ... 305

Product Introduction
CLB Overview

Cloud Load Balancer (CLB) is a service that evenly distributes network traffic across multiple real servers to boost application performance and eliminate single points of failure, ensuring greater availability.

By assigning a virtual IP address (VIP), CLB consolidates multiple real servers in a given region into a unified, high-performance service pool. It then routes client requests to these servers based on the specified traffic distribution rules.

CLB continuously monitors the health of the servers in the pool and automatically isolates any that become unhealthy, addressing potential failures and improving overall service reliability.

With features such as self-service management, automated fault recovery, and protection against network attacks, CLB is well-suited for applications in sectors like enterprise operations, community platforms, e-commerce, and gaming.

COMPONENTS

A typical Cloud Load Balancer (CLB) setup consists of the following elements:

- Load Balancer Instance: The component responsible for routing incoming traffic to the backend servers.
- Backend/Real Servers: A collection of server instances that process the incoming requests.
- Virtual Private Cloud (VPC) or Classic Network: The overarching network infrastructure that connects and supports the entire system.

External requests are routed by the load balancer instance to the backend servers for processing, based on configured rules and distribution policies.

GLOSSARY

Term	Full name	Descripton
CLB	Cloud Load Balancer	Converge Cloud offers a network load balancing service that works with real servers to provide load balancing for TCP/UDP and HTTP traffic.
CLB Listener	Cloud Load Balance Listener	CLB listeners let you configure ports, load balancing methods, and health check settings, with each listener corresponding to a specific application service in the backend.

Term	Full name	Descripton
RS	Real Server	CLB directs requests to a set of real server instances for processing, following the rules defined by the user.

HOW IT WORKS

Basic Operating Principle

A CLB instance receives client traffic and forwards requests to real servers located across one or more availability zones for processing. The core functionality of the CLB service is driven by the load balancing listener, which monitors incoming requests on the CLB instance and distributes them to the backend servers based on predefined policies.

The CLB can direct traffic to the real servers by configuring forwarding protocols and ports for both the Client - CLB and CLB - Real Server connections.

It's recommended to deploy real server instances across multiple availability zones. This ensures that if one availability zone experiences failure, CLB can route traffic to healthy servers in other zones, preventing service interruptions and maintaining high availability.

Request Routing:

When a client attempts to access a service, it first resolves the CLB domain name via the DNS server, which returns the appropriate CLB IP addresses. Once the CLB listener receives the request, it uses various load balancing algorithms to forward the request to the backend servers. Converge Cloud supports multiple load balancing methods, including weighted round robin, IP hash, and weighted least connections.

Real Server Monitoring:

A CLB instance continuously monitors the health of real server instances to ensure traffic is only directed to servers that are functioning properly. If a server becomes unhealthy, the CLB will stop routing traffic to it and will resume once the server is back online and functioning normally.

Additional Services

CLB integrates with several services to enhance application availability and scalability:

- Cloud Virtual Machine Instance is a virtual server that hosts the application in the cloud.
- Auto Scaling dynamically adjusts the number of instances. When enabled with CLB, new instances are automatically added to the CLB group, and terminated instances are removed from the group.
- Converge Cloud Observability Platform provides monitoring for CLB instances and backend server health, allowing for necessary actions to be taken.
- Domains and DNSPod translates custom domain names *(e.g., www.example.com)* into the corresponding IP address *(e.g., 192.0.2.1)*, enabling the routing of requests to CLB instances.

Use Cases

Use Cases

CLB is ideal for the following use cases:

- Traffic Distribution: CLB efficiently distributes high volumes of incoming traffic across multiple backend servers to balance the load.
- Avoiding Single Points of Failure: If certain CVM instances become unavailable, CLB automatically isolates them, ensuring the application system continues to run smoothly.
- Scalable Growth: CLB allows for horizontal scaling, enabling you to expand the capacity of web and app servers based on demand.
- Global Traffic Management: With DNSPod, CLB offers global and multi-region traffic routing, facilitating remote disaster recovery and ensuring seamless service continuity.

Traffic Distribution and Elimination of Single Point of Failure

CLB can efficiently distribute business traffic across a set of real servers for optimal performance and reliability.

- When a client requests access to your service via CLB, the traffic is directed to a pool of real servers that ensure high performance and availability.
- If any real server becomes unavailable, CLB automatically isolates it and reroutes traffic to healthy CVM instances, maintaining seamless application operation.
- For setups spanning multiple availability zones, it's recommended to link a CLB instance to CVM instances across these zones, ensuring multi-zone disaster recovery at the server level.
- The session persistence feature ensures that a client’s requests are consistently directed to the same real server, enhancing access efficiency and providing a smooth user experience.

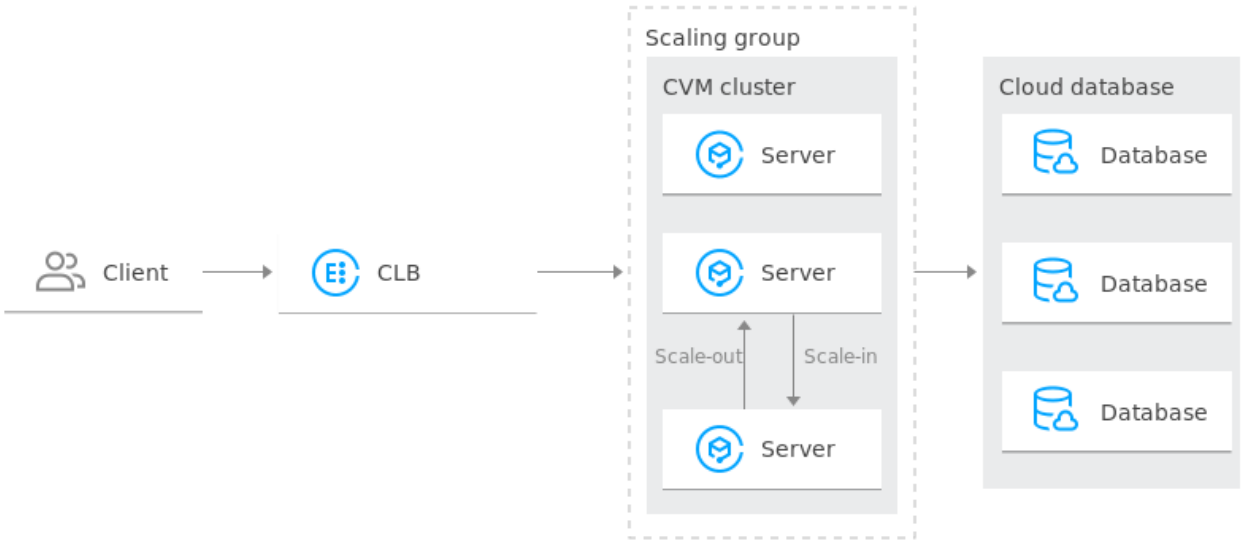
Traffic Distribution and Elimination of Single Point of Failure

Horizontal Scalability

With Auto Scaling (AS), CLB automatically adjusts the number of CVM instances based on your business requirements.

- You can define scaling policies to control the number of CVM instances, set up the instance environment, and ensure continuous business operation. CLB dynamically adds instances during traffic spikes to maintain high performance and removes them when demand decreases to optimize costs.

- For instance, during major events like Black Friday sales, web traffic might surge by 10 times for just a few hours. In this scenario, CLB and AS can work together to efficiently scale your infrastructure, maximizing cost savings while handling the increased load.



Strengths

Strengths

High Performance

A single CLB cluster (not just one instance) can handle hundreds of millions of concurrent connections and process millions of data packets per second. This capability makes it ideal for supporting high-traffic applications such as e-commerce websites, social media platforms, and gaming services with over 10 million daily page views.

High Availability

CLB utilizes a cluster-based deployment to ensure an uptime of up to 99.95%. Even in extreme scenarios where only one CLB physical server is operational, it can still support tens of millions of concurrent connections. The cluster system automatically removes faulty instances and retains healthy ones to ensure continuous operation of the backend servers.

High Elasticity

The CLB cluster dynamically adjusts the application's service capacity based on the business load. It works seamlessly with Auto Scaling to automatically add or remove CVM instances as needed. With real-time monitoring and precise billing, CLB eliminates the need for manual intervention, making it easier to allocate resources efficiently and avoid over-provisioning.

High Security and Stability

With its integration of the BGP Anti-DDoS system, CLB can defend against a wide range of network threats, including DDoS and web intrusion attacks. It swiftly filters out malicious traffic, preventing service interruptions. Additionally, the built-in synproxy anti-attack mechanism protects backend CVM instances from attacks before the Anti-DDoS system is activated, further strengthening data security and system stability.

CLB enforces strict traffic isolation for each tenant and provides proactive DDoS attack protection. Public network CLB automatically includes Anti-DDoS Basic. For enhanced security, CLB also supports Anti-DDoS Pro, Anti-DDoS Advanced, WAF, and other security services to safeguard your applications.

Cost Efficiency

CLB removes the need for extra investments in load balancing hardware and reduces the time spent on operational tasks, helping you save up to 99% on both hardware and labor costs. It also offers flexible billing options, allowing you to choose the best plan based on your requirements.

Principles

Principles

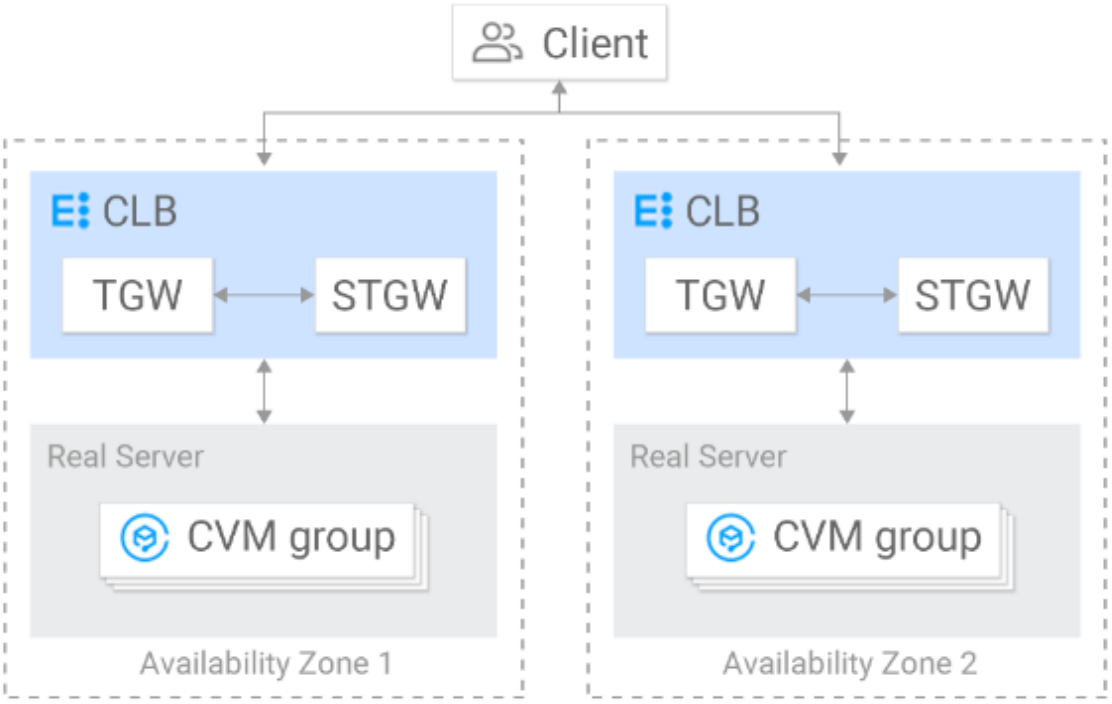
Cloud Load Balancer (CLB) offers both Layer-4 (TCP, UDP, and TCP SSL) and Layer-7 (HTTP and HTTPS) load balancing to efficiently distribute traffic across multiple backend servers. This design eliminates single points of failure and ensures high availability for your services.

Built on a clustered architecture, CLB supports session synchronization to enhance system redundancy and reliability. It can also be deployed across multiple data centers within the same region, enabling intra-city disaster recovery and ensuring continuous business operations.

Infrastructure

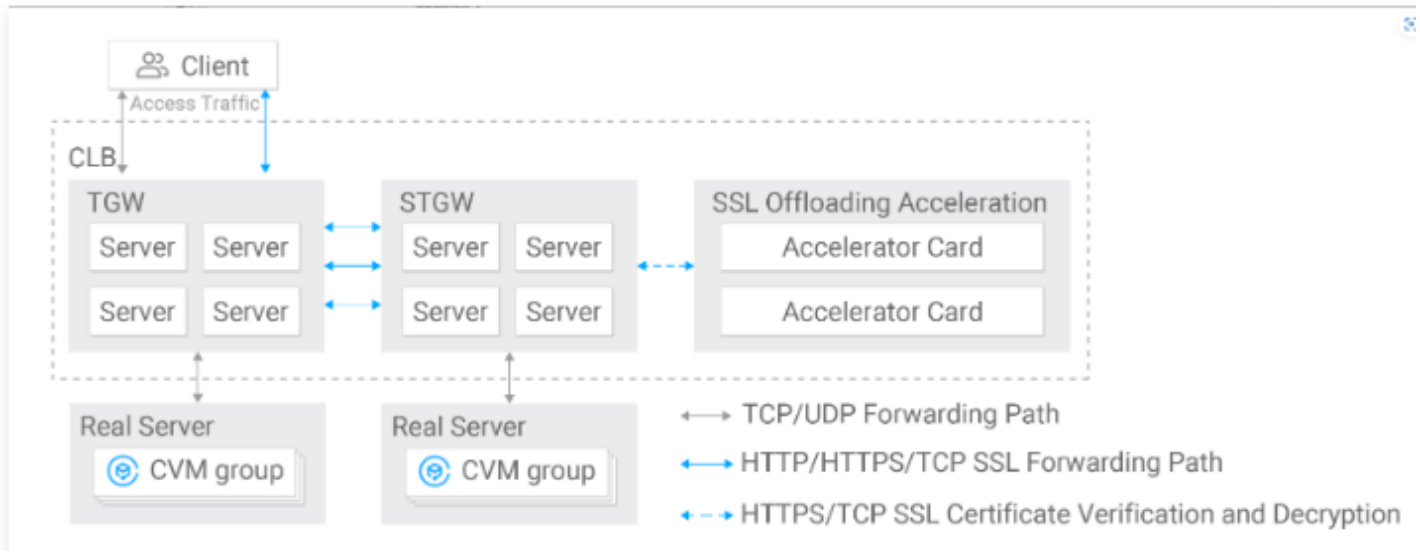
Converge Cloud CLB provides comprehensive support for Layer-4 and Layer-7 load balancing.

- Layer-4 Load Balancing is powered by the unified Converge Gateway (CGW), which offers high availability, scalability, and performance, along with strong protection against attacks. Leveraging Data Plane Development Kit (DPDK) for high-speed packet processing, CGW enables a single cluster to handle hundreds of millions of concurrent connections and tens of millions of packets per second (PPS). It serves as the access foundation for major Converge services such as Converge Games and Converge Video.
- Layer-7 Load Balancing is handled by the Secure Converge Gateway (SCGW), a custom solution built on Nginx to support high concurrency and scalability. SCGW manages a significant portion of Converge's Layer-7 traffic, supporting high-demand services like Converge News and Converge Games.



Forwarding Path

CLB handles traffic distribution, while backend servers are responsible for processing business requests. Communication between CLB and real servers occurs over Converge Cloud's private network, ensuring secure and efficient data transmission. Both CGW and SCGW are deployed in a clustered, multi-server architecture to deliver high-performance load balancing services with built-in redundancy. The CLB traffic forwarding architecture is illustrated below:



Use limits

Use Limits

General Restrictions

Converge Cloud CLB usage is subject to certain limitations, which may vary depending on the type of CLB instance. For detailed information on the available instance types and their respective limits, refer to the Instance Types documentation.

Instance Type	Resource	Default Restriction
General restrictions for all instances	Maximum number of public CLB instances allowed per region	100 general IP-based instances 3 static IP-based instances under one individual account or 15 under one organizational account.
General restrictions for all instances	Limit on private network CLB instances per account within one region	100
General restrictions for all instances	Number of listeners allowed per instance	50
General restrictions for all instances	Available ports for listener configuration within an instance	An integer between 1 and 65535
CLB(formerly "application CLB")	Number of domain and URL forwarding rules configurable for an HTTP/HTTPS listener on a CLB instance	50
CLB(formerly "application CLB")	Limit on servers that can be associated with a forwarding rule in a CLB instance	100
CLB(formerly "application CLB")	Number of backend ports that can be linked to a frontend port in a CLB instance	Multiple ports
Classic CLB	Number of backend servers supported by a listener in a Classic CLB instance	100

Instance Type	Resource	Default Restriction
Classic CLB	Number of backend ports that can correspond to a frontend port in a Classic CLB instance	1 port

A CLB instance does not automatically unbind from a CVM instance. Similarly, if a pay-as-you-go CVM instance enters an isolated state (due to over 2 hours of overdue payments), it will remain bound to the CLB instance and will not be unbound automatically.

Peak Bandwidth

The concept of peak bandwidth varies according to the network billing mode, with details provided below:

Billing Mode	Difference	Description
Bill-by-traffic	Peak bandwidth represents the maximum achievable bandwidth but does not guarantee a committed bandwidth. During periods of network contention, the actual bandwidth may be restricted below the peak value.	The combined peak bandwidth of all active pay-by-traffic instances in a single region must not exceed 5 Gbps. If your application requires guaranteed or higher bandwidth, consider using the pay-by-bandwidth billing mode.
Bandwidth packages	Peak bandwidth represents the maximum achievable bandwidth but does not guarantee a committed bandwidth. During periods of network contention, the actual bandwidth may be restricted below the peak value.	The total peak bandwidth of all active instances billed under a shared bandwidth package must not exceed 50 Gbps within a single region. For bandwidth requirements beyond this limit, please contact your sales representative for further assistance.

Quick Start
Quick Start for CLB

CLB supports TCP/UDP/HTTP/HTTPS protocols and provides flexible forwarding capabilities based on domain names and URL paths. This document describes how to quickly use IPv4 CLB.

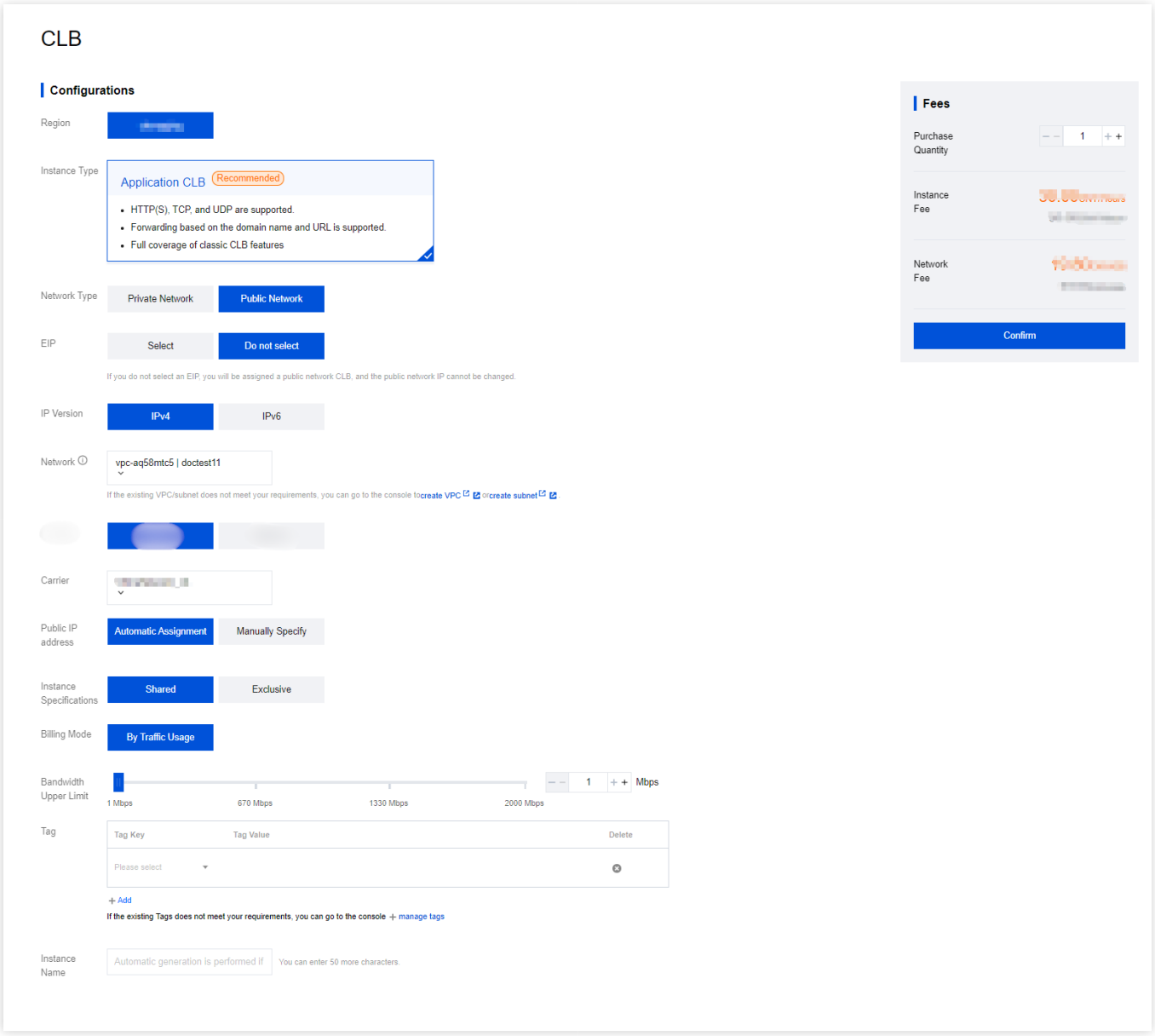
Prerequisites

1. CLB is only responsible for forwarding traffic and does not have the ability to process requests. Therefore, you need to have CVM instances that handle user requests.
   In this example, only two CVM instances are required. You can also plan the number of CVMs yourself. In this example, CVM instances rs-1 and rs-2 has been created in the NCR region.
2. In this document, HTTP forwarding is used for example to illustrate that the corresponding Web server, such as Apache, Nginx, IIS, etc, must be deployed on the CVM.
   In the example, Apache is deployed on rs-1 to return an HTML page with Hello Tomcat! This is rs-1! and deployed on rs-2 to return an HTML page with Hello Tomcat! This is rs-2! in order to verify the results. For more information about CVM deployment, see Deployment of Java Web on Linux (CentOS) and Installment and Configuration of PHP on Windows.
3. Access the public network IP + path of the CVM. If the displayed result is the page you deployed, it means that the service has been deployed successfully.

   > Note:
   >
   > In this example, the service return values deployed by the backend server are different and are only used for feature verification. In practice, to ensure that all users have a consistent experience, exactly the same services are usually deployed on the backend servers.

Create a CLB instance

1. log in to Converge Cloud CLB .
2. Select a region, click Create, and then enter the Cloud Load Balancer LB window.
3. Configure the CLB instance parameters. The configuration example in this document is shown in the figure. Configure according to the actual situation.



4. Click Confirm Activation. After the instance is successfully created, you can see the newly created instance in the list.

Create a CLB listener

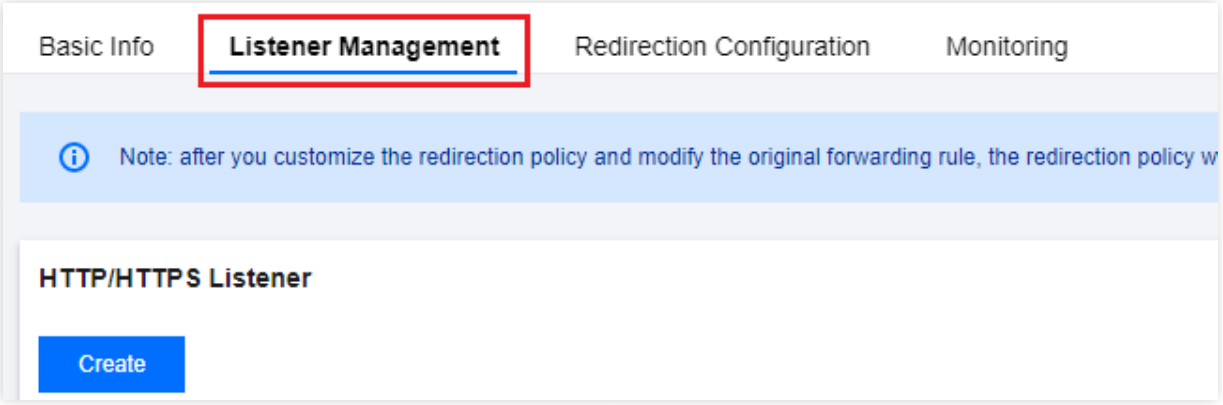
The Load Balance Listener is responsible for the actual forwarding by specifying the protocol and port. In this document, the configuration of CLB forwarding the client's HTTP request is used as an example.

Configure HTTP listening protocol and port

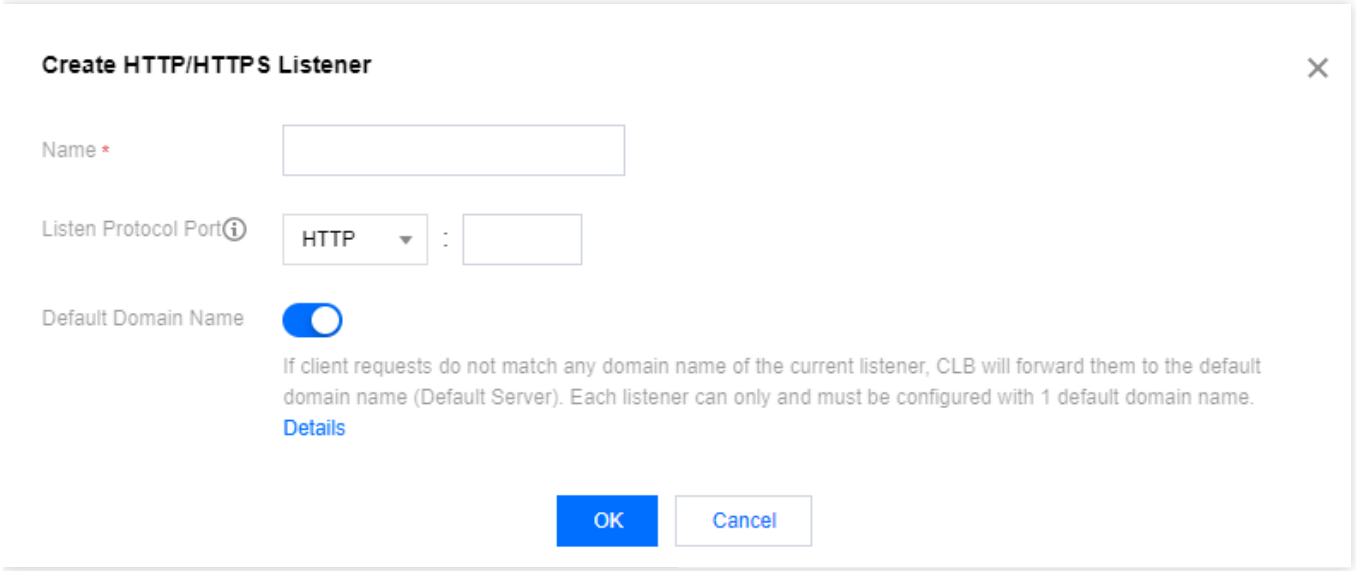
1. Log in to the cloud console, click Cloud Products > Cloud Computing & Networking > Cloud Load Balancer to open the CLB console.
2. In the LB Instance List, find the created CLB instance clb-test and click the instance ID to open the CLB

details page.

3. In the Basic Info module, you can click the edit icon after the name to modify the instance name.
4. In Listener Management, click Create under HTTP/HTTPS Listener to create a new Load Balance Listener.



5. In the dialog box that pops up, configure the following parameters, and click [Confirm] when done.

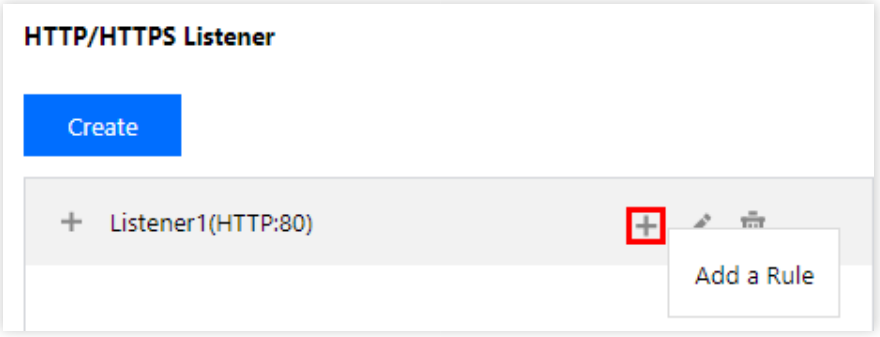


Parameter	Description
Name	The name of the listener. In this example, it can be customized as Listener1.
Listen Protocol Port	The listener's protocol and listening port. - The listening protocols include HTTP and HTTPS. In this example, HTTP is selected. - The listening port is used to receive requests and forward them to the backend server. The port range is 1 - 65535. Among them, ports 843, 1,020, 1,433, 1,434, 3,306, 3,389, 6,006, 20,000, 36,000, 42,222, 48,369, 56,000, and 65,010 are reserved for the system and are not open to the public yet. - The listening ports in the same CLB instance cannot be repeated.
Default Domain	You can choose to enable or disable it. - Enable: When the client request does not match any domain name of this listener, CLB will

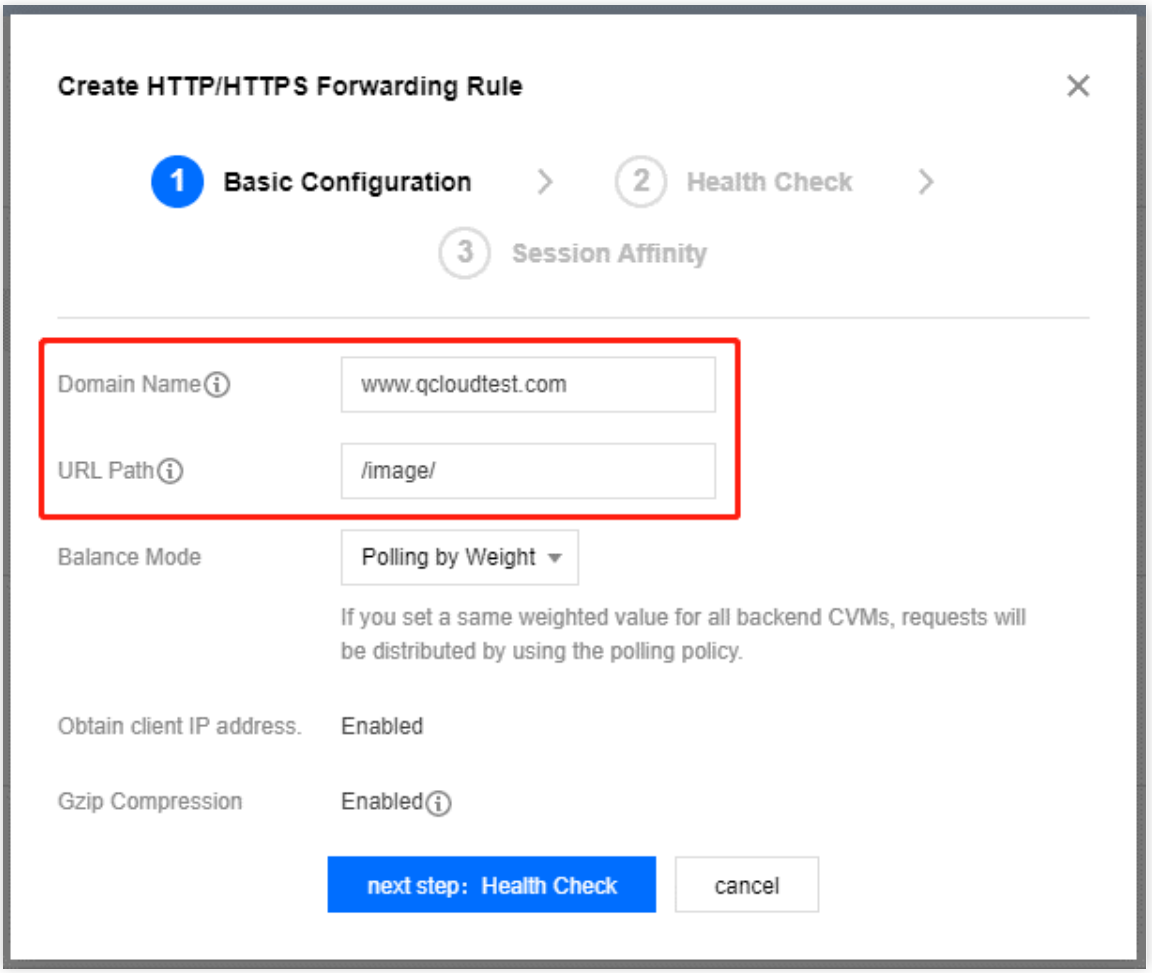
Parameter	Description
Name	forward the request to the default domain name (Default Server). Each listener can only be configured with one default domain name. - If your Layer-7 listener has been configured with a default domain name, client requests that do not match other rules will be forwarded to the default domain name. - If your Layer-7 listener is not configured with a default domain name, client requests that do not match other rules will be forwarded to the first domain name loaded by the CLB. Since the loading order may be inconsistent with the console configuration order, it is not necessarily the first one configured by the console. - Disable: When the client request does not match any domain name of this listener, the request will not be forwarded.

Configuring the forwarding rules for the listener

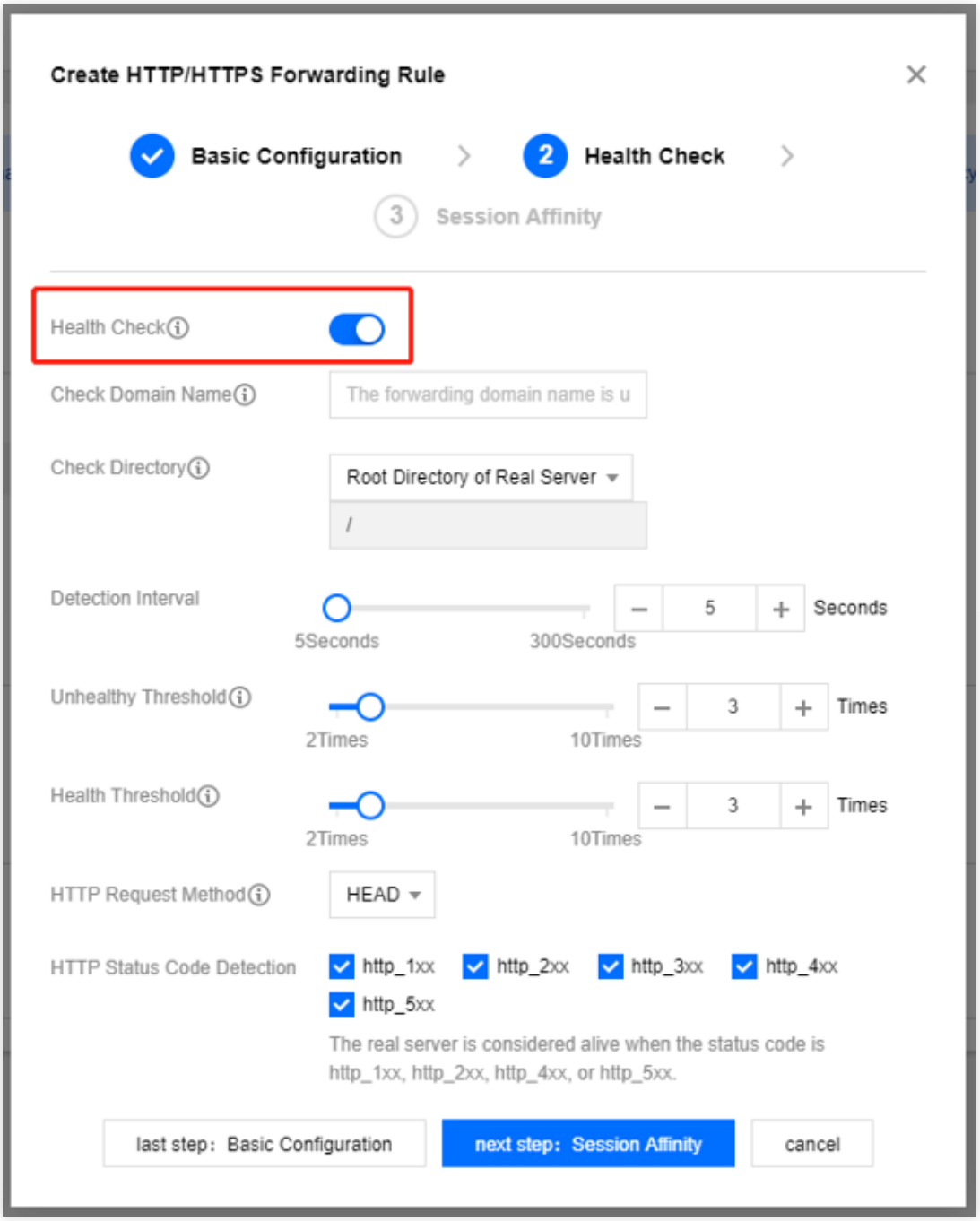
1. In "Listener Management", select the newly created listener Listener1, click +, and start adding rules.



2. In the forwarding rule dialog box that pops up, set basic configuration information such as domain name, URL path, and balancing method, and click next step: Health Check.
   - Domain name: the domain name used by your backend service. In this example, www.qcloudtest.com is used. Domain name supports wildcards. For details, see Configuration Instructions.
   - URL Path: the access path of your backend service. In this example, /image/ is used.
   - Select Polling by Weight for the balance mode.



3. Configuring health check: Enable health check, check whether the domain name uses default forwarding domain name and forwarding path, and click next step: Session Affinity.



Parameter	Description
Health Check	You can either enable or disable the health check feature. We recommend you to enable health check. It can help you automatically check and remove abnormal backend CVM ports.
Check Domain Name	Health check domain name: - Length limit: 1 to 80 characters. - The request domain is set to the forwarding domain by default. - Regular expressions are not supported. When your forwarding domain name is a wildcard name, you need to specify a fixed domain (non-regular) name as the health check domain name.

Parameter	Description
	- The supported character set is a to z, 0 to 9, ., and -.
Check Directory.	The health check path can be set to the backend server root directory or a specified URL: - Length limit: 1 to 200 characters. - The default is / and must start with /. - Regular expressions are not supported. We recommend you to specify a fixed URL path (static page) for health check. - The supported character set is: a to z, A to Z, 0 to 9, ., -, _, /, =, and ?. .
Detection Interval	- The interval for the CLB health check. - Configurable range: 5 - 300 seconds.
Unhealthy Threshold	- If the health check results received for n (a value filled in) consecutive times fail, it is identified as unhealthy and the console displays it as an exception. - Configurable range: 2 - 10 times.
Health Threshold	- If the health check results received for n (a value filled in) consecutive times are successful, it is identified as healthy and the console displays it as healthy. - Configurable range: 2 - 10 times.
HTTP Request Method	HTTP request method for the health checks, optional: GET or HEAD, the default is GET. - If the HEAD method is used, the server only returns the HTTP header information, which can reduce backend overhead and improve request efficiency. The corresponding backend service must support HEAD. - If the GET method is used, the backend service only needs to support GET.
Normal Status Code	When the status code is the selected status code, the backend server is considered alive, that is, the health check is normal. Optional codes: http_1xx, http_2xx, http_3xx, http_4xx and http_5xx. You can select multiple status codes.

4. Session Affinity: Enable session affinity, set the persistence time, and click Submit to complete the configuration of the listener forwarding rule.

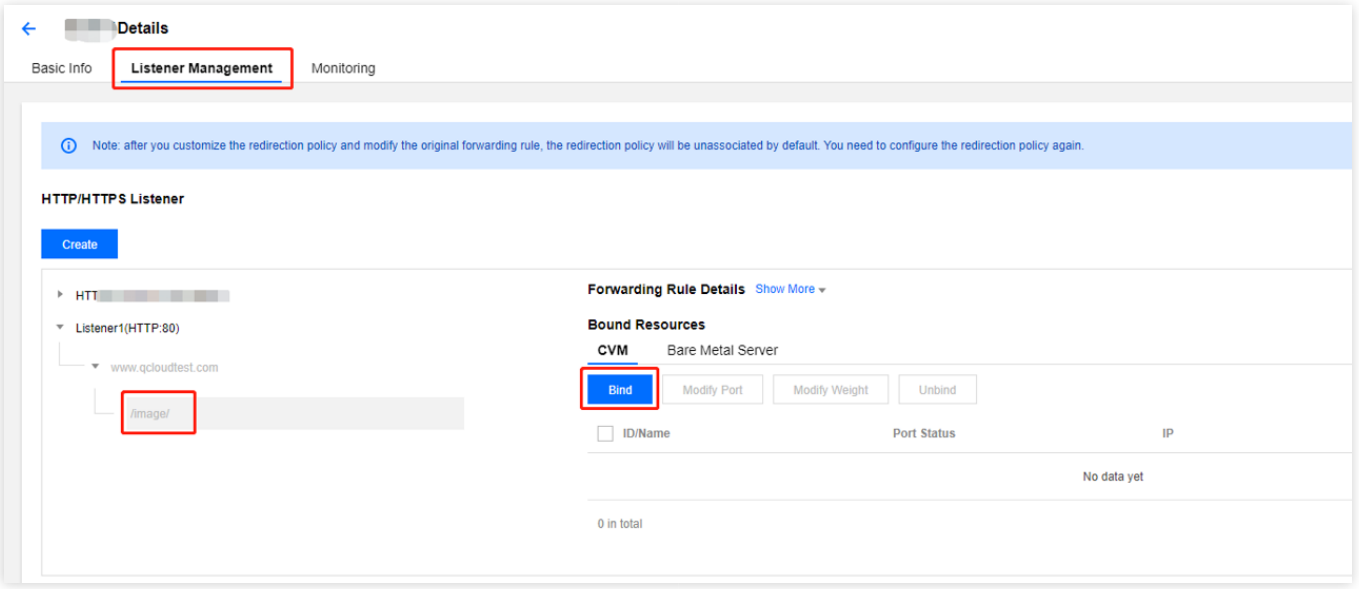
   For more information about Load Balance Listener, see Load Balance Listener Overview.

   > Note:
   >
   > - One listener (i.e. listening protocol: port) can be configured with multiple domain names, and one domain name can be configured with multiple URL paths. Select the listener or domain names and click the + sign to create a new rule.

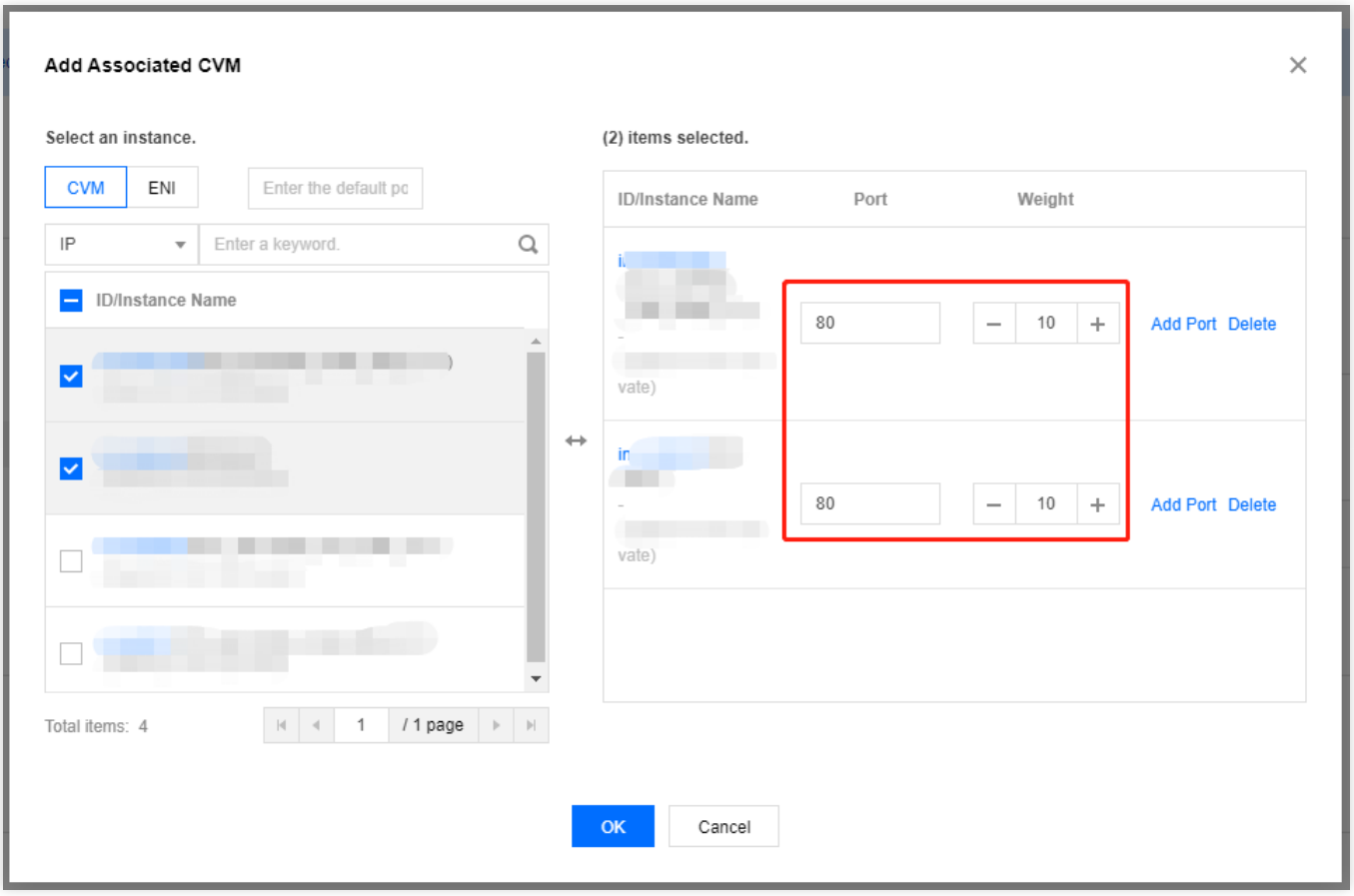
- Session persistence: If the user disables session persistence and select round-robin scheduling, requests will be allocated to different backend servers in turn. If the user enables session persistence, or disables session persistence and selects IP Hash scheduling, requests will be continuously allocated to the same backend server.
  >

Binding Cloud Virtual Machine (CVM)

1. On the Listener Management page, select the triangle icon next to the newly created listener to expand Listener1. Select and expand the domain name and URL path in turn. The information of CVM bound to the URL path is displayed on the right side of the screen. Click Bind.

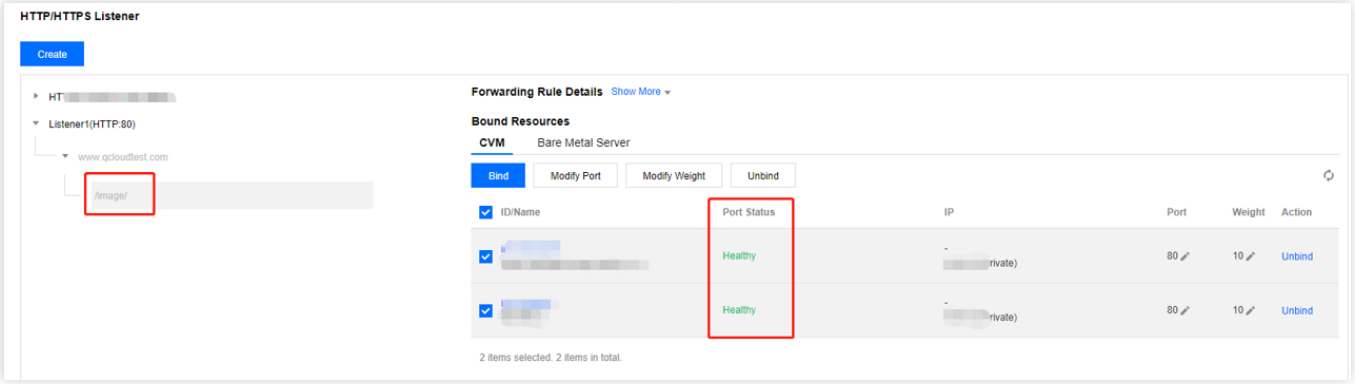


2. In the pop-up box, select the CVM instances rs-1 and rs-2 in the same region as the CLB. Set the CVM ports to 80 and the CVM weights to the default value 10.



3. Click OK to complete binding.

4. Expand the listener to the URL path dimensionality to view the bound CVM and its health check status. When the status is healthy, it means that the CVM can normally process the requests forwarded by the CLB.



> Note:
>
> One forwarding rule (listening protocol + port + domain name + URL path) can bind multiple ports of the same CVM. If the user deploys the same service in ports 80 and 81 of rs-1 , the CLB supports the forwarding rules in the example binds ports 80 and 81 of rs-1 at the same time, and both ports will receive the request forwarded by CLB.

Verifying the CLB Service

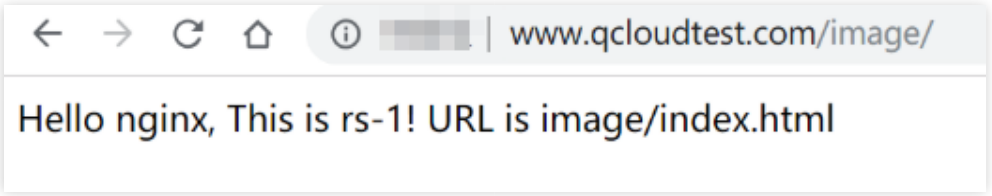
After you configure CLB, you can verify whether the architecture is effective, that is, verify different backend CVMs are accessed through different domain names + URLs under a CLB instance, or, verify content-based routing feature.

Method 1: Configuring hosts to point the domain name to CLB

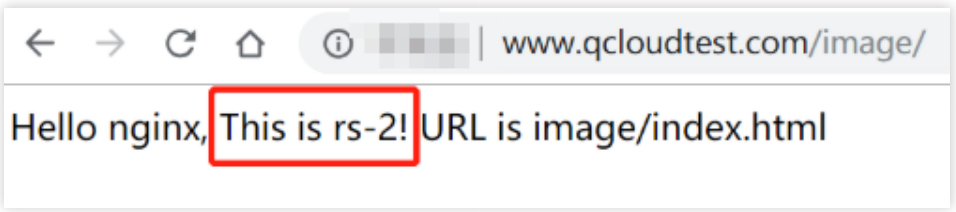
1. On a Windows system, go to the directory C:\Windows\System32\drivers\etc , modify the hosts file, and map the domain name to the VIP of the CLB instance.



2. To verify whether the hosts are configured successfully, you can run cmd and use the ping command to check whether the domain name is successfully bound to the VIP. If there is a data packet, the binding is successful.
3. Enter the access path http://www.qcloudtest.com/image/ in the browser to test the CLB service. As shown in the following figure, the request has been forwarded by CLB to the CVM rs-1, and the CVM processed and returned the request normally.



4. The round-robin scheduling algorithm of this listener is Weighted Round Robin, and the weights of the two CVMs are both 10. Refresh the browser and send the request again. You can see that this request has been forwarded by CLB to the CVM rs-2.



> Note:
>
> The / in image/ must be retained. It indicates that image is the default directory, not a file name.

Method 2: Configuring DNS to point the domain name to CLB

1. Log in to the cloud console and click Cloud Products > Domain Name Services > VPCDNS.
2. Click the Domain Name you purchased, click the Add Record button on the Domain DNS Management page, add an A record for the domain, and enter the following content:
   - Record type: A record .
   - Host record: This is the domain name prefix. In this example, to resolve all prefixes, set it to *.qcloudtest.com .
   - Record value: Enter the CLB IP address here.
   - Weight: Set this to its default value 100.
3. After addition, click OK.
   It takes some time for DNS to propagate the record across the Internet. To test whether the domain name is resolved normally, you can directly access the bound CNAME domain name (such as www.qcloudtest.com in this example) after the DNS record for a period of time to verify CLB.

Deploying Nginx on CentOS

Software Versions

The software versions in the example steps in this document are as follows. In actual operations, the actual software versions should prevail.

- Operating system: CentOS 7.5
- Nginx version: Nginx 1.16.1

Installing Nginx

1. After the purchase is completed, click Log In on the CVM Details page to log in to CVM directly. After you enter your username and password, start building the Nginx environment. For more information on how to create a CVM instance, see Creating a CVM Instance.

   ```
   # Install Nginx:
   yum -y install nginx
   # View the Nginx version:
   nginx -v
   # View the Nginx installation directory:
   rpm -ql nginx
   # Start Nginx:
   service nginx start
   ```

2. Access the public IP address of the CVM. If the following page appears, it indicates that Nginx has been deployed.



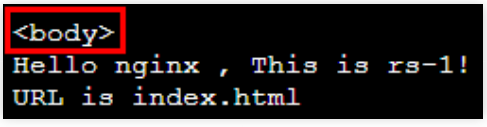
3. The default root directory of Nginx is /usr/share/nginx/html . Directly modify the index.html static page under html to identify the particularity of this page. The relevant operations are as follows:

   i. Execute the following command to go to the index.html static page under html:

```
bash
vim /usr/share/nginx/html/index.html
```

   ii. Press i to enter the edit mode, and enter the following content in the <body></body> tag:

```
# It is recommended to directly enter the following content below <body>:
Hello nginx , This is rs-1!
URL is index.html
```



   3. Press Esc and enter :wq to save the change.

4. CLB (formerly Application CLB) can forward requests based on the path of the backend server. Deploy static pages under the /image path. The relevant operations are as follows:

   i. Execute the following commands in sequence to create a directory image and enter it:

```
mkdir /usr/share/nginx/html/image
```

cd /usr/share/nginx/html/image

ii. Execute the following command to create an index.html static page in the image directory:

vim index.html

iii. Press i to enter the edit mode and enter the following content on the page:

Hello nginx , This is rs-1!
URL is image/index.html

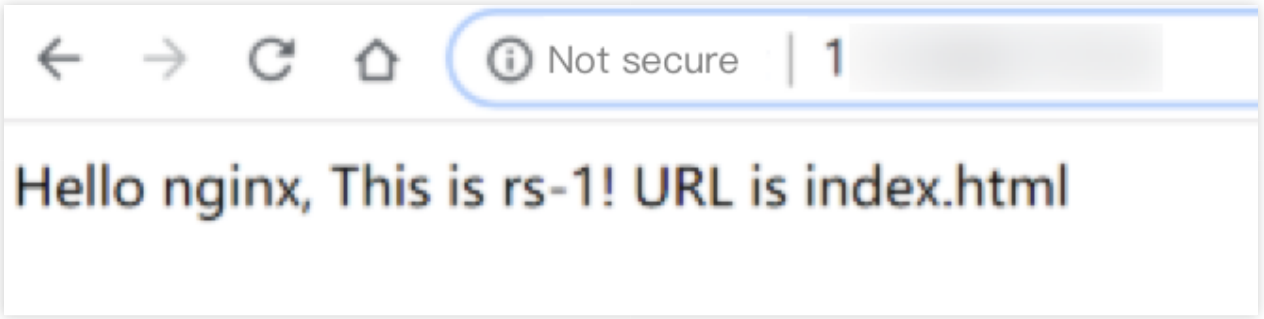
iv. Press Esc and enter :wq to save the edits.

> Note:
>
> The default port for Nginx is port 80 . To modify the port, modify the configuration file and restart Nginx.

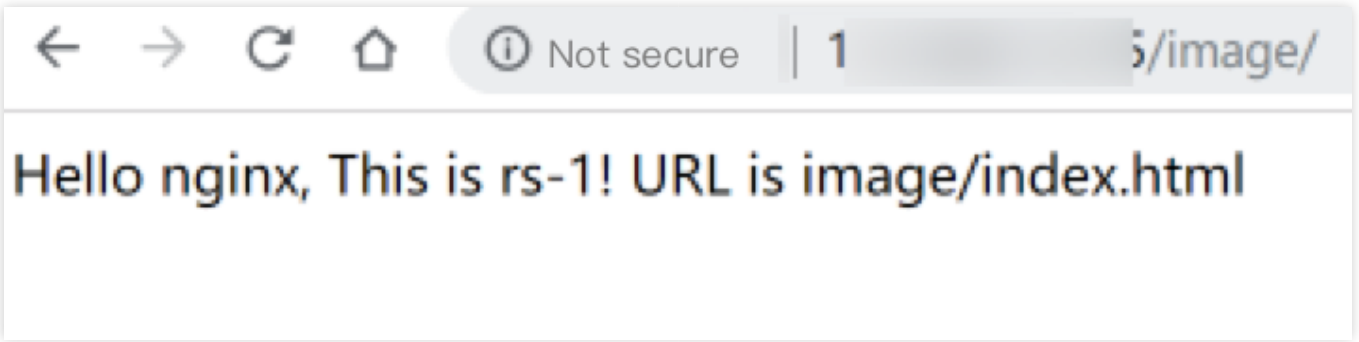
Verifying the Nginx Service

Access the public IP address and path of CVM. If a static page displays that deployment has been completed, it proves that Nginx has been deployed successfully.

- The index.html page of rs-1:



- The /image/index.html page of rs-1:



Deploy Java Web in CentOS

Software Versions

The software versions in the example steps in this document are as follows. In actual operations,see the actual software versions.

- Operating system: CentOS 7.5
- Tomcat version: apache-tomcat-8.5.39
- JDK version: JDK 1.8.0_201

Install JDK

After you purchase the lCLB service, click Log In on the CVM details page to log in to the CVM directly. Enter your username and password to start building the Java Web environment. For information on how to create a CVM instance, see CVM - Create an Instance.

Download JDK

You can enter the following command:

```
mkdir /usr/java  # Create a Java folder
cd /usr/java     # Enter the Java folder
```

```
# Upload the JDK installation package (recommended).
It is recommended that you use  or other tools to upload the JDK installation package to the aforementione
d java folder, and then decompress the installation package.
or
# Directly use the command (It is recommended that you use the method of uploading the JDK installation p
ackage): wget download link. The compressed package downloaded cannot be decompressed. This is becaus
e the compressed package downloaded directly does not accept the Oracle BSD license by default. Everyone'
s cookies are different. Go to https://www.oracle.com/technetwork/java/javase/downloads/jdk8-download
s-2133151.html to agree to the license agreement and obtain the download link with your own cookies.
wget --no-check-certificate --no-cookies --header "Cookie: oraclelicense=accept-securebackup-cookie" http
s://download.oracle.com/otn-pub/java/jdk/8u201-b09/42970487e3af4f5aa5bca3f542482c60/jdk-8u201-linu
x-x64.tar.gz
```

```
# Unzip
chmod +x jdk-8u201-linux-x64.tar.gz
tar -xzvf jdk-8u201-linux-x64.tar.gz
```

Set the relevant environment variables

1. Open /etc/profile document.

   vi /etc/profile

2. Press the i key to enter the edit mode and add the following information to the file.

   ```
   # set java environment
   export JAVA_HOME=/usr/java/jdk1.8.0_201
   export CLASSPATH=$JAVA_HOME/lib/tools.jar:$JAVA_HOME/lib/dt.jar:$JAVA_HOME/lib
   export PATH=$JAVA_HOME/bin:$PATH
   ```

3. Press the Esc key to exit edit mode and enter :wq to save and close the file.

4. Load environment variables.

   source /etc/profile

Check whether JDK is installed successfully

Run java -version . When the JDK version information is displayed, it means that the JDK has been installed successfully.

```
[root@emma /]# java -version
java version "1.8.0_201"
Java(TM) SE Runtime Environment (build 1.8.0_201-b09)
Java HotSpot(TM) 64-Bit Server VM (build 25.201-b09, mixed mode)
```

Install Tomcat

Download Tomcat

You can enter the following command:

```
# The image address changes and the Tomcat version is continuously upgraded. If the download link is invalid, please go to the Tomcat official website (https://tomcat.apache.org/download-80.cgi) to select the appropriate installation package address.
wget http://mirrors.tuna.tsinghua.edu.cn/apache/tomcat/tomcat-8/v8.5.39/bin/apache-tomcat-8.5.39.tar.gz
tar -xzvf apache-tomcat-8.5.39.tar.gz
mv apache-tomcat-8.5.39 /usr/local/tomcat/
```

In /usr/local/tomcat/ directory contains the following files:

- bin: script file, including scripts for starting and shutting down Tomcat services.

- conf: various global configuration files, with the most important of which being server.xml and web.xml.
- webapps: Tomcat's main Web publishing directory. By default, Web application files are placed in this directory.
- logs: stores the log files when Tomcat is executed.

> Note:
>
> If the download link is invalid, please replace it with Tomcat official website latest download link.

Adding Users

```
# Create a general user www to run Tomcat
useradd www
# Create the website root directory
mkdir -p /data/wwwroot/default
# Upload the Java Web project file WAR package to be deployed to the website root directory, and then cha
nge the file permission in the website root directory to www. This example will create a new Tomcat testing p
age directly in the root directory of the website:
echo Hello Tomcat! > /data/wwwroot/default/index.jsp
chown -R www.www /data/wwwroot
```

Set JVM memory parameters

1. Create a /usr/local/tomcat/bin/setenv.sh script file.

   ```
   vi /usr/local/tomcat/bin/setenv.sh
   ```

2. Press the i key to enter edit mode and add the following content.

   ```
   JAVA_OPTS='-Djava.security.egd=file:/dev/./urandom -server -Xms256m -Xmx496m -Dfile.encoding=U
   TF-8'
   ```

3. Press Esc to exit edit mode, enter :wq! to save and exit.

Configure server.xml

1. Switch to /usr/local/tomcat/conf/ table of contents.

   ```
   cd /usr/local/tomcat/conf/
   ```

2. Back up the server.xml file.

   ```
   mv server.xml server_default.xml
   ```

3. Create a new server.xml file.

vi server.xml

4. Press the i key to enter edit mode and add the following content.

```
<?xml version="1.0" encoding="UTF-8"?>
<Server port="8006" shutdown="SHUTDOWN">
<Listener className="org.apache.catalina.core.JreMemoryLeakPreventionListener"/>
<Listener className="org.apache.catalina.mbeans.GlobalResourcesLifecycleListener"/>
<Listener className="org.apache.catalina.core.ThreadLocalLeakPreventionListener"/>
<Listener className="org.apache.catalina.core.AprLifecycleListener"/>
<GlobalNamingResources>
<Resource name="UserDatabase" auth="Container"
 type="org.apache.catalina.UserDatabase"
 description="User database that can be updated and saved"
 factory="org.apache.catalina.users.MemoryUserDatabaseFactory"
 pathname="conf/tomcat-users.xml"/>
</GlobalNamingResources>
<Service name="Catalina">
<Connector port="8080"
 protocol="HTTP/1.1"
 connectionTimeout="20000"
 redirectPort="8443"
 maxThreads="1000"
 minSpareThreads="20"
 acceptCount="1000"
 maxHttpHeaderSize="65536"
 debug="0"
 disableUploadTimeout="true"
 useBodyEncodingForURI="true"
 enableLookups="false"
 URIEncoding="UTF-8"/>
<Engine name="Catalina" defaultHost="localhost">
<Realm className="org.apache.catalina.realm.LockOutRealm">
<Realm className="org.apache.catalina.realm.UserDatabaseRealm"
  resourceName="UserDatabase"/>
</Realm>
<Host name="localhost" appBase="/data/wwwroot/default" unpackWARs="true" autoDeploy="true">
<Context path="" docBase="/data/wwwroot/default" debug="0" reloadable="false" crossContext="tru
e"/>
<Valve className="org.apache.catalina.valves.AccessLogValve" directory="logs"
prefix="localhost_access_log." suffix=".txt" pattern="%h %l %u %t &amp;quot;%r&amp;quot; %s %b" />
</Host>
</Engine>
</Service>
</Server>
```

5. Press Esc to exit edit mode, enter :wq! to save and exit.

Start Tomcat

Method A

Enter the bin directory of the Tomcat server and execute ./startup.sh command to start the Tomcat server.

```
cd /usr/local/tomcat/bin
./startup.sh
```

The results are as follows:

```
[root@emma bin]# ./startup.sh
Using CATALINA_BASE:   /usr/local/tomcat
Using CATALINA_HOME:   /usr/local/tomcat
Using CATALINA_TMPDIR: /usr/local/tomcat/temp
Using JRE_HOME:        /usr/java/jdk1.8.0_201
Using CLASSPATH:       /usr/local/tomcat/bin/bootstrap.jar:/usr/local/tomcat/bin/tomcat-juli.jar
Tomcat started.
```

Method B

1. Set up a shortcut start, and you can start Tomcat from anywhere with service Tomcat start.

   ```
   wget https://github.com/lj2007331/oneinstack/raw/master/init.d/Tomcat-init
   mv Tomcat-init /etc/init.d/tomcat
   chmod +x /etc/init.d/tomcat
   ```

2. Run the following command to set the startup script JAVA_HOME.

   ```
   sed -i 's@^export JAVA_HOME=.*@export JAVA_HOME=/usr/java/jdk1.8.0_201@' /etc/init.d/tomcat
   ```

3. Set up auto-start.

   ```
   chkconfig --add tomcat
   chkconfig tomcat on
   ```

4. Start Tomcat.

   ```
   # Start Tomcat
   service tomcat start
   # View Tomcat running status
   service tomcat status
   ```

```
# Shut down Tomcat
service tomcat stop
```

The results are as follows:

```
[root@emma bin]# service tomcat start
Tomcat is already running (pid: 27293)
[root@emma bin]# service tomcat status
Tomcat is running with pid: 27293
[root@emma bin]# service tomcat stop
Stoping Tomcat
Using CATALINA_BASE:   /usr/local/tomcat
Using CATALINA_HOME:   /usr/local/tomcat
Using CATALINA_TMPDIR: /usr/local/tomcat/temp
Using JRE_HOME:        /usr/java/jdk1.8.0_201
Using CLASSPATH:       /usr/local/tomcat/bin/bootstrap.jar:/usr/local/tomcat/bin/tomcat-juli.jar
waiting for processes to exit
```

5. If it prompts that you do not have permission, switch to the root user and modify the permission.

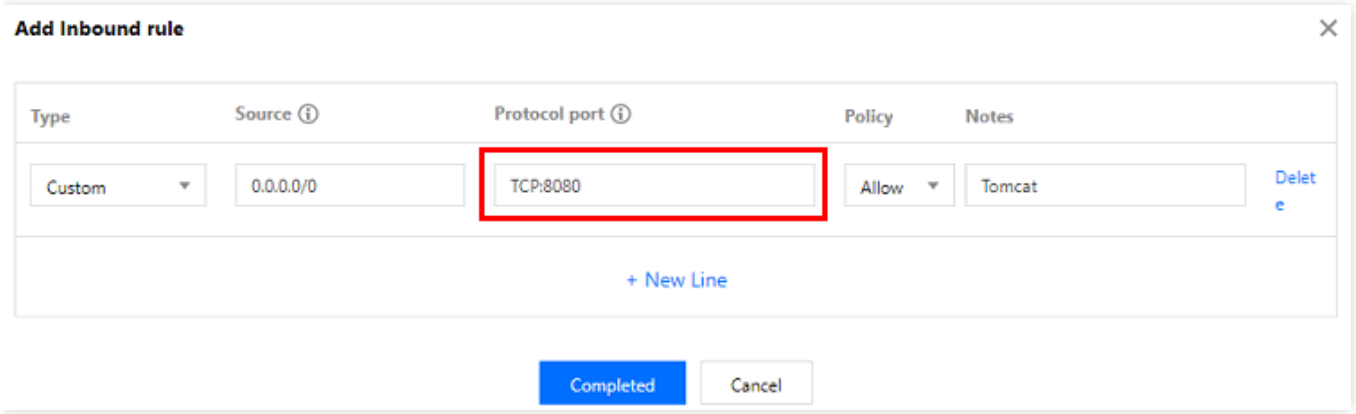
```
cd /usr/local
chmod -R 777 tomcat
```

6. Enter http://public IP:port in the browser address bar. (The port is the connector port set in server.xml) for access. When the page shown below appears, the installation is successful.



Configure Security Group

If access is unsuccessful, check the security group. In the above example, the connector port in server.xml is 8,080, so TCP:8,080 needs to be allowed in the security group bound to the corresponding cloud server.


Add Inbound rule
Type
Source
Protocol port
Policy
Notes
Custom
0.0.0.0/0
TCP:8080
Allow
Tomcat
Delet
e
+ New Line
Completed
Cancel

Operation Guide
CLB Instance
Creating a CLB Instance

This document describes how to purchase CLB.

1. Log in to the cloud platform console, choose Cloud Products > Cloud Computing & Networking > Cloud Load Balancer to go to the CLB console. Click LB Instance List in the left sidebar.
2. On the LB Instance List page, click Create in the upper left corner.
3. On the CLB Service Purchase page, select the following configurations and click Confirm.

Parameter	Description
Region	Select the region to which the CLB instance belongs.
Instance Type	Only Application CLB is supported.
Network Type	Configure a public or private network CLB service based on your business scenarios. The system will assign a public or private network service address based on your choice. - Public network: Public network CLB charges instance fees and public network fees. - Private network: Private network CLB is free.
Elastic IP (EIP)	- If you do not select an EIP address, the cloud platform will assign a public network CLB for you, and the public network IP address cannot be changed. - If you select an EIP address, the cloud platform will assign an EIP address and a private network CLB for you. The EIP address and private network CLB can be bound/unbound as needed.
IP Version	The IPv4 version can be selected.
Network	Select a VPC.
AZ Type	AZ types are divided into the single-AZ type and the multi-AZ type. A multi-AZ type consists of a primary AZ and a secondary AZ. The primary AZ handles traffic, while the secondary AZ does not handle traffic unless the primary AZ becomes unavailable. To improve the high availability of the business, it is recommended to select the multi-AZ type.
Carrier	Local Carriers are supported.
Instance Specifications	Both private and public network IPv4 CLB instances support selecting instance specifications. Instances are classified into shared instances and dedicated

Parameter	Description
	instances. Dedicated instances reside on the physical cluster specified by you.
Cluster Tag	When the instance specification of private and public network IPv4 CLB is a dedicated cluster, you can select a cluster tag. Cluster tags are classified into layer-4 cluster tags and layer-7 cluster tags.
Cloud Dedicated Cluster (CDC)	Supported only in CDC scenarios. The final instance is created on the CDC.
Public IP Address	If you have specific requirements for the IP address of the instance, you can manually specify it. Currently supported CLB instances include: - Private and public network IPv4 instances
Public Network Bandwidth	Only the billing-by-traffic-usage mode is supported.
Bandwidth Limit	The bandwidth limit is 2 Gbps.
Tag	Select a tag key and tag value, or create a tag. For details, see Creating a Tag.
Instance Name	You can name it after creation or immediately.
Purchase Quantity	Select the number of instances to be purchased.

Creating the IPv6 NAT64 CLB Instance

> Description:
>
> IPv6 NAT64 CLB does not support obtaining the client IP.

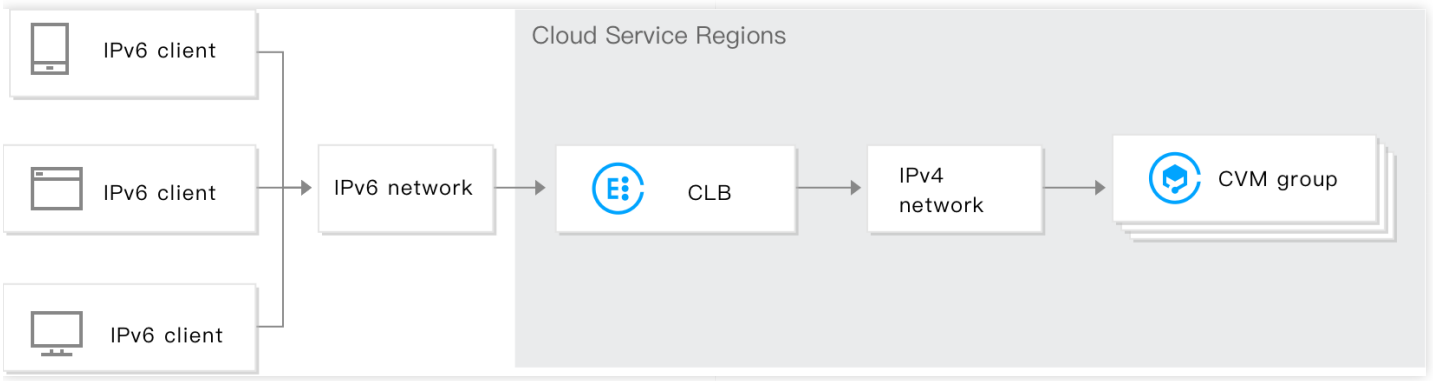
CLB supports creating the IPv6 NAT64 CLB instance. Converge Cloud will allocate an IPv6 public network address (VIP in the IPv6 Version) to the instance. The VIP will forward requests from the IPv6 client's end to the IPv4 CVM at the backend.

What is IPv6 NAT64 CLB

IPv6 NAT64 CLB is the Cloud Load Balancer implemented based on NAT64 IPv6 transition technology. With the IPv6 NAT64 CLB, the backend CVM can quickly access IPv6 users without any IPv6 transformation.

IPv6 NAT64 CLB Architecture

The architecture of the IPv6 NAT64 CLB is shown in the figure below.



When accessing the IPv6 NAT64 CLB through an IPv6 network, the CLB can seamlessly translate IPv6 addresses to IPv4 addresses, adapting to existing services and quickly enabling IPv6 transformation.

The Strengths of IPv6 NAT64 CLB

Converge Cloud IPv6 NAT64 CLB has the following strengths in helping businesses quickly access IPv6:

- Quick access: Access IPv6 in seconds and go online quickly as soon as you buy and use it.
- Smooth business transition: Businesses can smoothly access IPv6 only based on client modification, not requiring modifying any backend businesses. IPv6 NAT64 CLB supports the access from IPv6

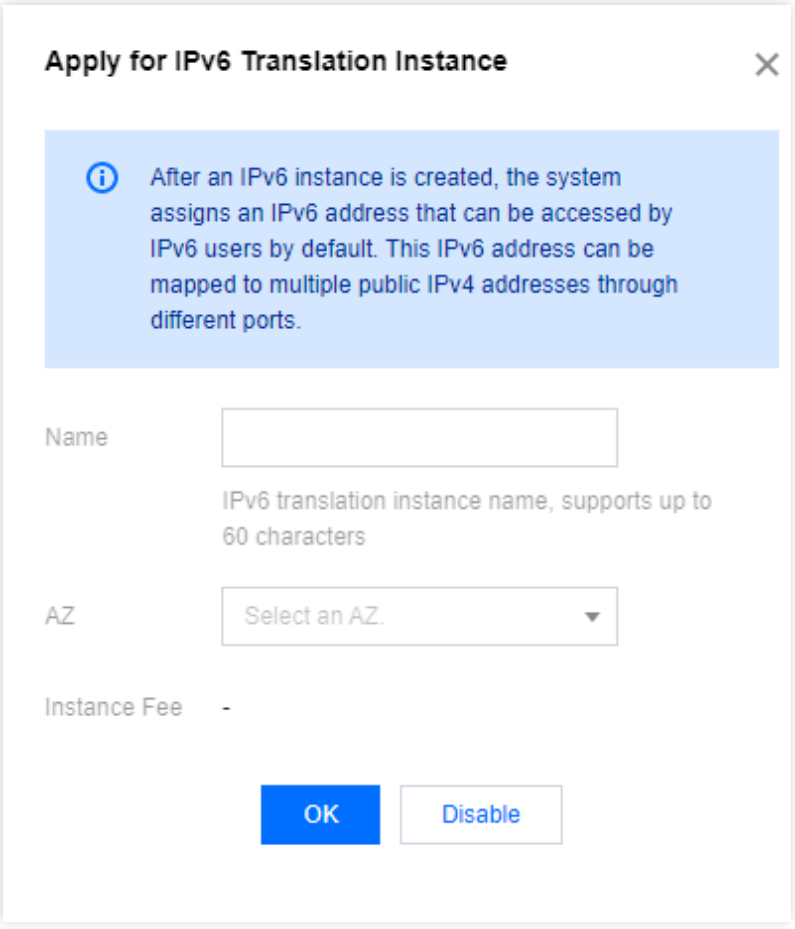
clients and converts IPv6 messages into IPv4 messages. Applications on the backend CVMs do not need to be aware of IPv6 and can still be deployed and work in their original form.

- Easy to use: IPv6 NAT64 CLB is compatible with the original IPv4 CLB operation traffic, with zero learning cost and a low threshold for use.

Operation Guide

Creating the IPv6 NAT64 CLB

1. Log in to the CLB console.
2. Click the IPv6 Translation Instance in the left sidebar to enter the IPv6 Translation Instance page.
3. Click Create and the Apply for IPv6 Translation Instance dialog box pops up.
    - Name: Enter the instance name.
    - AZ: Select the AZ where the CLB instance is located.



4. Click OK to complete the creation of the instance.

Using the IPv6 NAT64 CLB

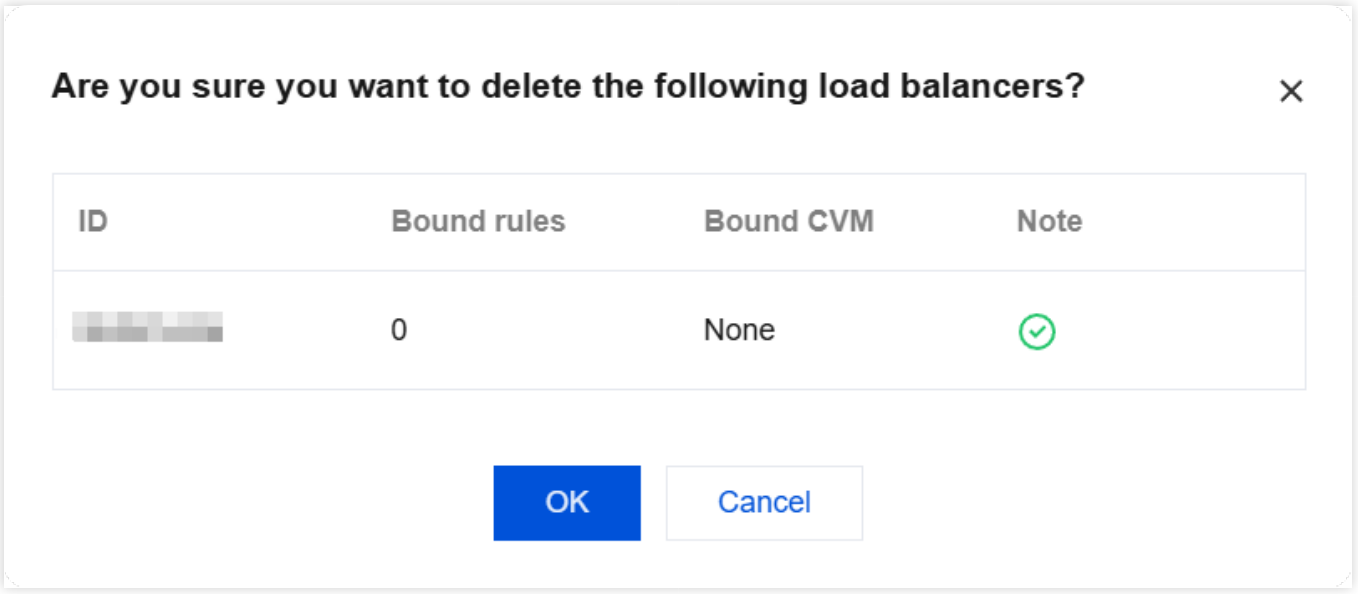
Click the instance ID to enter the Details page, where you can create a conversion rule.

Deleting a CLB Instance

When you confirm that the CLB instance no longer has traffic and no longer needs to be used, you can delete the instance through the CLB console or API.
Once an instance is deleted, it will be completely terminated and can not be recovered. We strongly recommend that you unbind all backend CVMs before deleting an instance and observe for a while before deleting it.

Deleting a CLB Instance through the Console

1. Log in to the CLB console.
2. Find the CLB instance you want to delete and click Delete in the rightmost operation column.
3. A final confirmation dialog box will pop up. After confirming that the operation safety prompt is normal, click OK to delete the instance.
   The following figure shows the final confirmation dialog box. We strongly recommend you confirm that the number of binding rules is 0, the number of bound CVM instances is None, and the operation safety prompt is shown in green before performing the deletion operation.



Configuring a CLB Security Group

After creating a CLB instance, you can configure the CLB security group to isolate public network traffic. This document describes how to configure CLB security groups in different modes.

Use Limits

- Each CLB can bind up to five security groups. To enhance the quota, contact the cloud platform's technical support personnel for assistance.
- A single CLB security group, when fully expanded, can contain up to 512 entries including outbound rules, inbound rules, and backend parameter templates (ipm/ipmg/ppm/ppmg).
- After a private network CLB binds an EIP address, the security group on the new CLB takes effect for traffic from the EIP address and private network CLB. Security groups on existing CLB instances take effect for traffic from private network CLB, but not for traffic from the EIP address.
- The feature of allowing traffic by default in the security group takes effect only for backend servers of the ENI or CVM type within this VPC. When binding PaaS services (such as CDB) as backend servers, the feature is not supported.

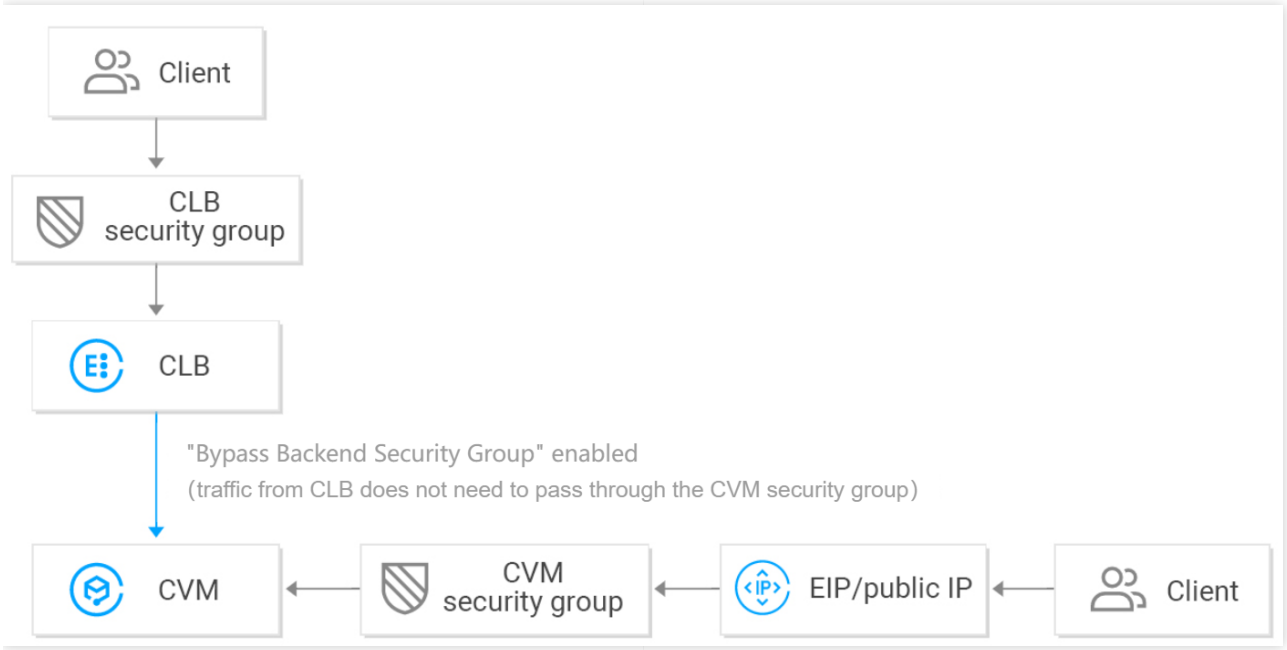
Background Information

A security group is a virtual firewall that filters stateful data packets and controls the inbound and outbound traffic at the instance level. For details, see Security Group Overview.
CLB security groups are bound to CLB instances, while CVM security groups are bound to CVM instances. The two restrict different objects. The CLB security group is configured primarily in the following two modes:

- Enabling the feature of allowing traffic by default in the security group
- Disabling the feature of allowing traffic by default in the security group

> Note:
>
> - In IPv4 CLB, the feature of allowing traffic by default in the security group is disabled by default. You can enable or disable it in the console.

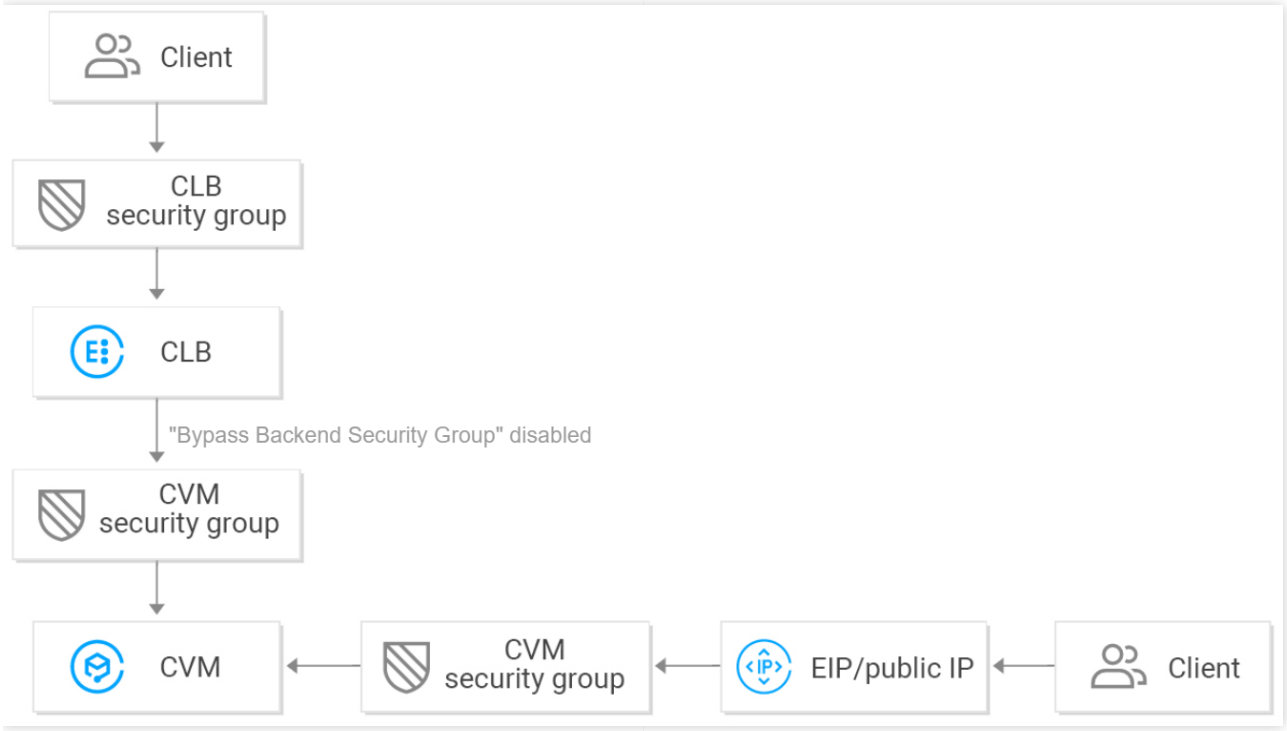
Enabling the Feature of Allowing Traffic by Default in the Security Group



After the feature of allowing traffic by default in the security group is enabled:

- The access traffic from CLB only needs to pass through the CLB security group, as the backend server allows traffic from CLB by default and does not need to open the port.
- Traffic from public IP addresses (including general public IP addresses and EIP addresses) still needs to pass through the CVM security group.
- If a CLB instance is not configured with a security group, all traffic is allowed, and only ports configured with listeners on the VIP of the CLB instance can be accessed. Therefore, the listening ports will allow traffic from all IP addresses.
- To deny traffic from a specified client IP address, you should deny its access in the CLB security group; denying access from an IP address in the CVM security group takes effect only for traffic from public IP addresses (including general public IP addresses and EIP addresses), but not for traffic from CLB.

Disabling the Feature of Allowing Traffic by Default in the Security Group



After the feature of allowing traffic by default in the security group is disabled:

- Business traffic passing through CLB will be double-checked by both the CLB security group and the CVM security group.
- Traffic from public IP addresses (including general public IP addresses and EIP addresses) still needs to pass through the CVM security group.
- If a CLB instance is not configured with a security group, only traffic passing through the CVM security group is allowed.
- To deny traffic from a specific client IP address, you can deny its access in either the CLB security group or the CVM security group.
  When the feature of allowing traffic by default in the security group is disabled, the CVM security group should be configured as follows to ensure the health check feature:

1. Configure the public network CLB.
   You need to allow the CLB VIP in the backend CVM security group, so that CLB can use the VIP to detect the health status of the backend CVM.
2. Configure the private network CLB.
   - For private network CLB, you need to allow the CLB VIP in the security group of the backend CVM (for health checks).

Operation Steps

In the following example about the configuration of the public network CLB security group, business traffic is only allowed to enter from CLB port 80, with services provided by CVM port 8080. Client IP addresses are not limited and access from any IP address is supported.

> Note:
>
> For the public network CLB used in this example, the CLB VIP needs to be allowed in the backend CVM security group for health checks. The current IP address 0.0.0.0/0 can be any IP address and already contains the CLB VIP.

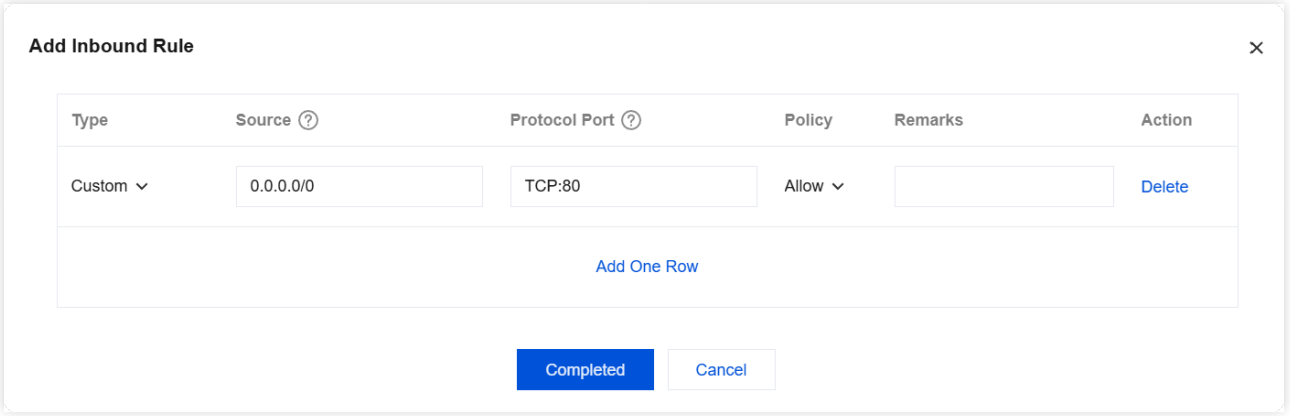
Step 1: Creating a CLB Instance and Listener and Binding a CVM Instance

For details, see Getting Started with CLB. An HTTP:80 listener is created and bound to a backend CVM instance whose service port is port 8080 in this example.

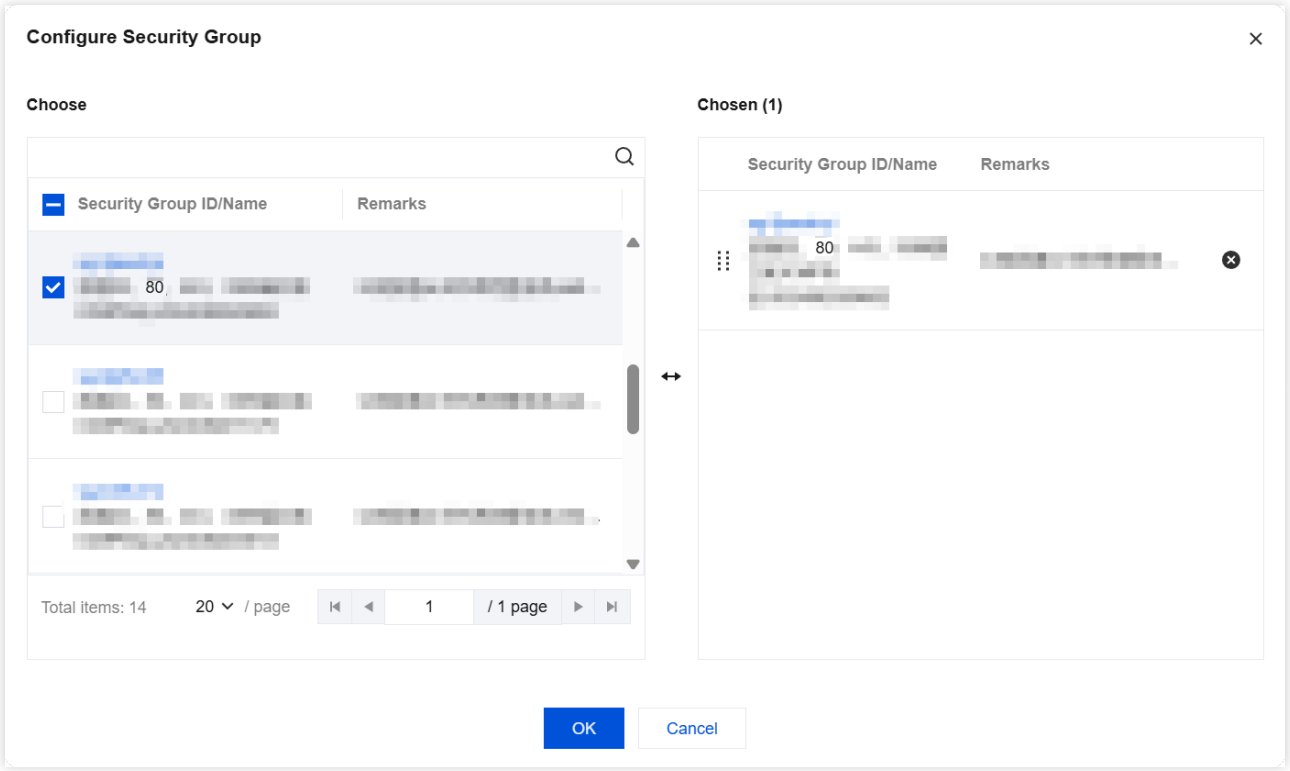
Step 2: Configuring a CLB Security Group

1. Configure a CLB security group rule.
   In the security group console, configure a security group rule. In the inbound rule, allow port 80 of all IP addresses (namely 0.0.0.0/0) and deny traffic from other ports.

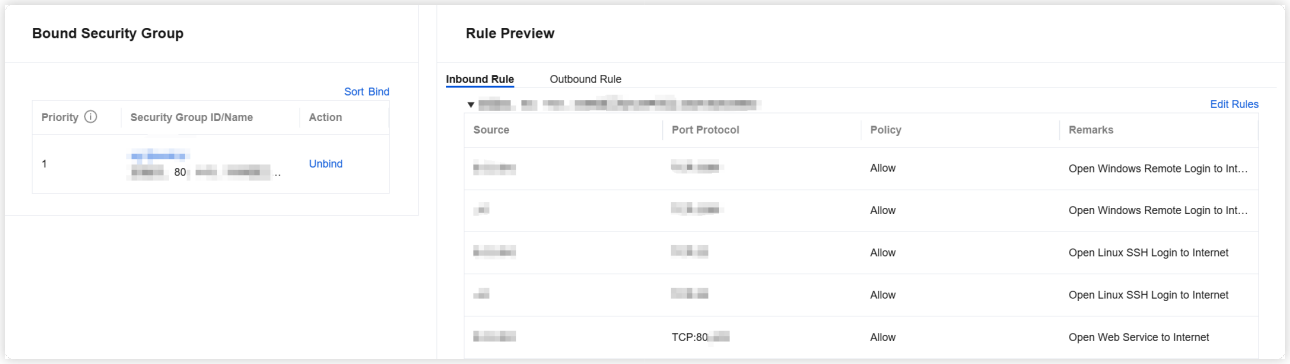
   > Note:
   >
   > - Security group rules are screened to take effect from top to bottom. If the new rule is effective, other rules will be denied by default. Therefore, pay attention to their order. For details, see Description of Security Group Rules.
   > - A security group has inbound and outbound rules. The above configuration is intended to restrict inbound traffic and is therefore an inbound rule, while the outbound rule does not need to be specially configured.



2. Bind the security group to a CLB instance.
   i. Log in to the CLB console.
   ii. On the LB Instance List page, click the ID of the target CLB instance.
   iii. On the Instance Details page, click the Security Group tab, and then click Bind in the Bound Security Groups module.
   iv. In the pop-up Configure Security Group window, select the security group bound to the CLB instance, and click OK.



The CLB security group configuration is completed, which only allows access to CLB from port 80.



Step 3: Configuring the Feature of Allowing Traffic by Default in the Security Group

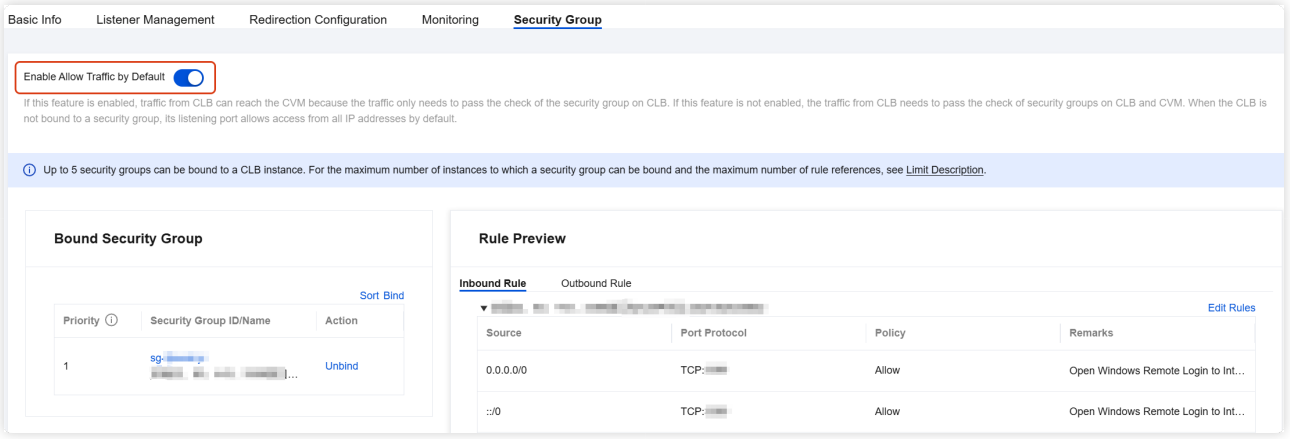
You can enable or disable the feature of allowing traffic by default in the security group with different configurations as follows:

- Method 1: Enable the feature of allowing traffic by default in the security group, so that the backend server does not need to open the port.
- Method 2: Disable the feature of allowing traffic by default in the security group, and allow the client IP address (0.0.0.0/0 in this example) in the CVM security group.

Method 1: Enabling the Feature of Allowing Traffic by Default in the Security Group

1. Log in to the CLB console.
2. On the LB Instance List page, click the ID of the target CLB instance.
3. On the Instance Details page, click the Security Group tab.

4. On the Security Group page, click to enable Enable Allow Traffic by Default.
5. After the feature is enabled, only the security group rules in the following rule preview will be verified for traffic from CLB instance, and the CVM security group will allow traffic from the CLB instance by default.

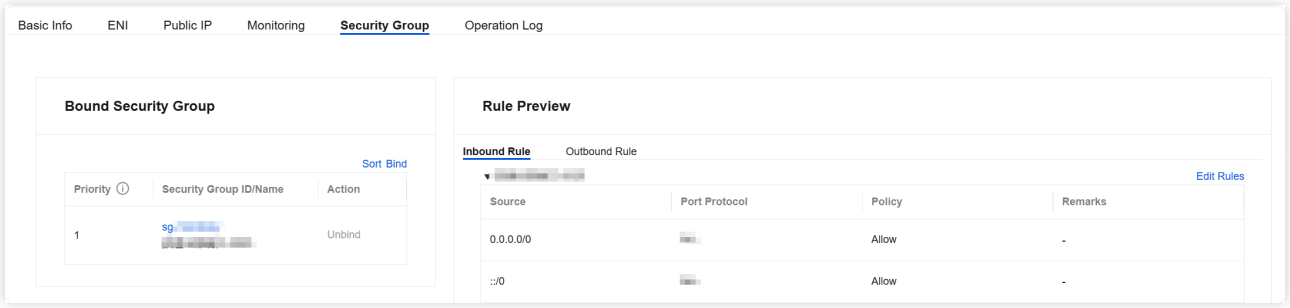


Method 2: Disabling the Feature of Allowing Traffic by Default in the Security Group

If Allow by Default is disabled, you need to allow the client IP address in the CVM security group. Business traffic is allowed to access CVM only from CLB port 80 and use services provided by CVM port 8080.

> Note:
>
> Traffic from a specified client IP address can be allowed, but that should be done in both the CLB security group and the CVM security group. If the CLB instance is not configured with a security group, that only needs to be done in the CVM security group.

1. Configure a CVM security group rule.
   A CVM security group can be configured to only allow access from service ports for traffic accessing the backend CVM.
   In the security group console, configure a security group policy. In the inbound rule, allow port 8080 of all IP addresses. To ensure remote host login and ping services, allow port 22, port 3389, and ICMP services in the security group.
2. Bind the security group to a CVM instance.
   i. In the CVM console, click the ID of the CVM instance bound to the CLB instance to go to the Details page.
   ii. Select the Security Group tab and click Bind in the Bound Security Groups module.
   iii. In the pop-up Configure Security Group window, select the security group to be bound to the CVM instance and click OK.


Basic Info
ENI
Public IP
Monitoring
Security Group
Operation Log
Bound Security Group
Sort
Bind
Priority
Security Group ID/Name
Action
1
Unbind
Rule Preview
Inbound Rule
Outbound Rule
Edit Rules
Source
Port Protocol
Policy
Remarks
0.0.0.0/0
Allow
-
::/0
Allow
-

Adjusting the Bandwidth Configuration of the Instance

The network bandwidth of private and public network CLB instances can be adjusted as needed.

Usage Notes

Private and public network IPv4 instances: Network bandwidth can be adjusted.

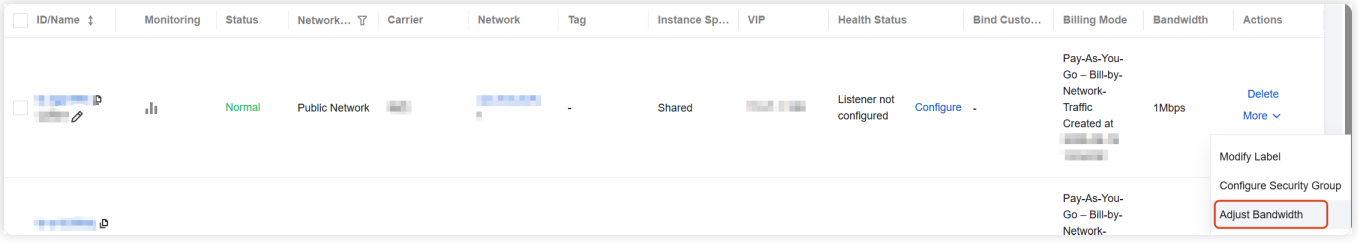
Bandwidth Limit

Instance Billing Mode	Network Billing Mode	Bandwidth Limit Configurable Range (Mbps)
Pay-as-you-go	Billing by traffic usage	0–2,048 (inclusive)

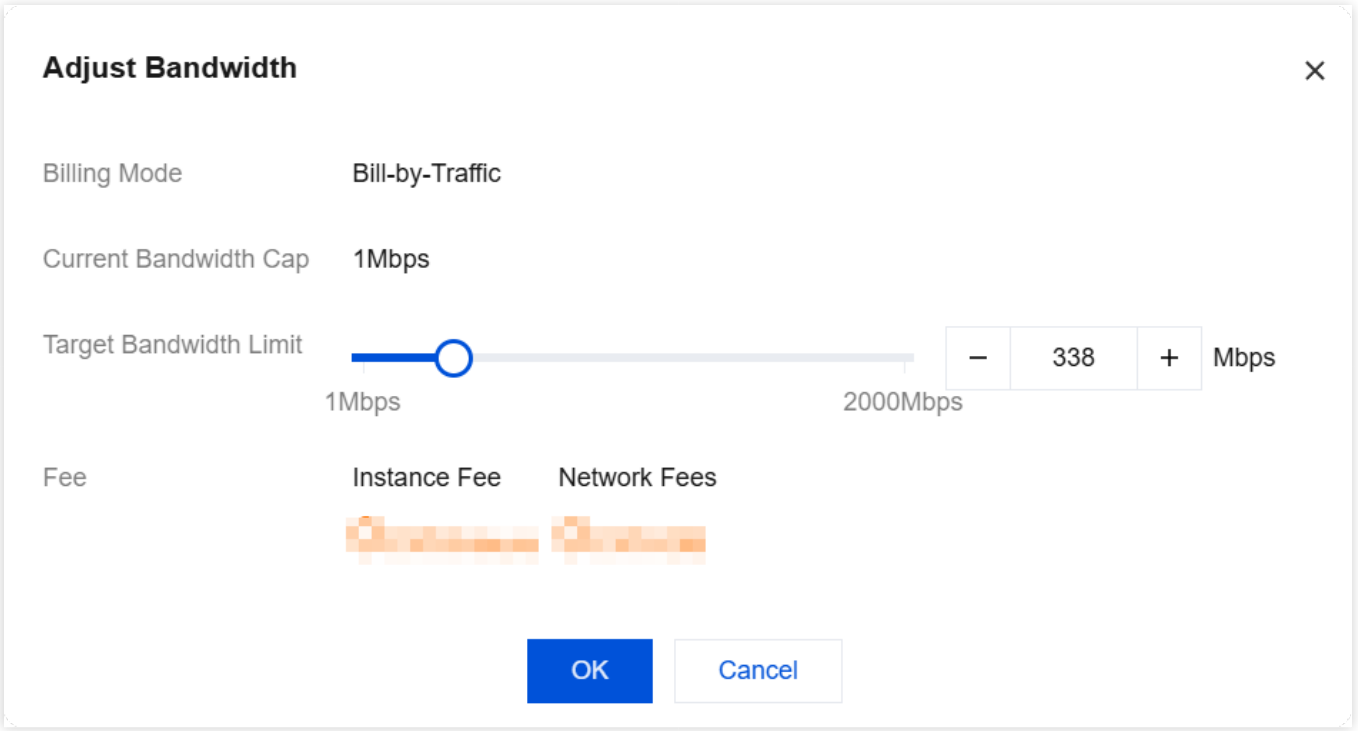
> Note:
>
> To enhance the bandwidth limit, contact the cloud platform's technical support personnel for assistance.

Adjusting Bandwidth

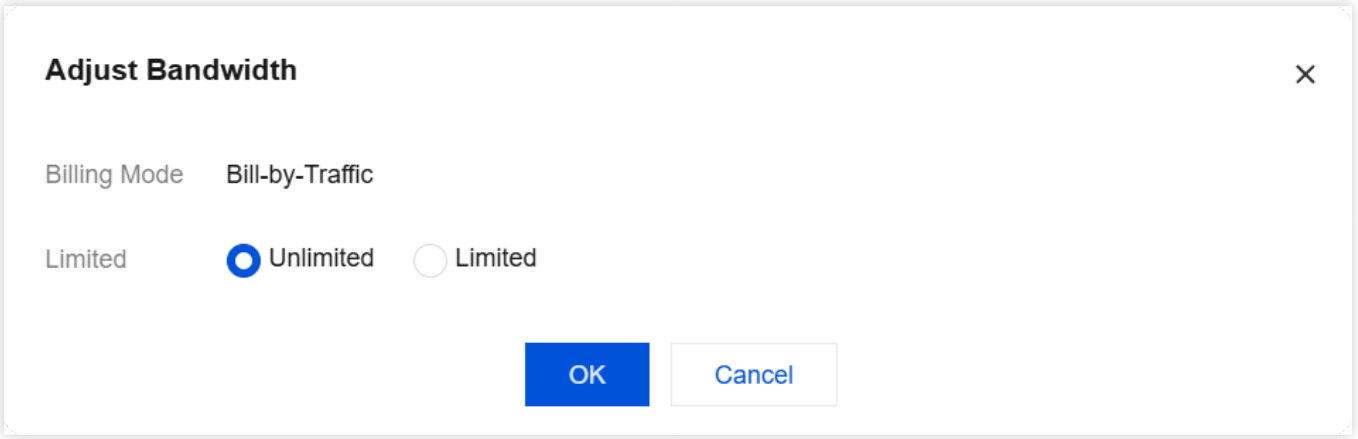
1. Log in to the cloud platform console and choose Cloud Products > Cloud Load Balancer to go to the CLB console.
2. On the LB Instance List page, select the region, locate the target CLB instance, and choose More > Adjust Bandwidth in the operation column.



3. In the pop-up dialog box, set the target bandwidth limit, and click OK.



4. Private network CLB instances support two options: unlimited bandwidth and limited bandwidth. Set it based on the actual situation.



5. After the setup is complete, the bandwidth value of the instance can be viewed on the list page.

CLB Listener

CLB Listener Overview

After you create a CLB instance, you need to configure a listener for the instance. The listener is responsible for monitoring requests on the CLB instance and distributing traffic to the backend servers based on the balancing policy.

The CLB listener needs to be configured:

1. Listening protocol and listening port. The listening port of the CLB, also known as the frontend port, is the port used to receive requests and forward requests to the backend server.
2. Monitoring policies, including balancing policies, session persistence, etc.
3. Health Check policies.
4. Bind the backend service and select the IP and port of the backend server. The service port is also called the backend port, which is the port used to receive requests.

Supported Protocol Types

The CLB listener can listen to the layer-4 and layer-7 requests on the CLB instance and distribute these requests to the backend servers, which then process the requests. The difference between layer-4 and layer-7 load balancing is mainly reflected in whether the traffic is forwarded based on the layer-4 protocol or the layer-7 protocol when the CLB user requests.

- Layer-4 Protocol: Transport layer protocol, which mainly accepts requests through VIP + Port and distributes traffic to the backend server.
- Layer-7 Protocol: Application layer protocol, which distributes traffic based on application layer information such as URL and HTTP header.
  Converge Cloud CLB supports request forwarding of the following protocols:
- TCP (transport layer)
- UDP (transport layer)
- HTTP (application layer)
- HTTPS (application layer)

Layer-4 listener

Protocols	Note	Use Cases
TCP	Connection-Oriented, Reliable	It is suitable for cases with high requirements

Protocols	Note	Use Cases
	Transport Layer Protocol - The source and terminal of the transmission need to be handshake three times to establish a connection before data transmission. - Support session persistence based on client IP (source IP). - You can see the client IP at the network layer. - The server can directly obtain the client IP.	on reliability and data accuracy but low requirements on transmission speed, such as file transfer, email sending and receiving, remote log-in, etc. For details, see Configuring TCP Listener.
UDP	Connectionless Transport Layer Protocol - The transmission source and terminal do not establish a connection, and there is no need to maintain the connection status. - Each UDP connection can only be point-to-point. - Supports one-to-one, one-to-many, many-to-one and many-to-many interactive communications. - Support session persistence based on client IP (source IP). - The server can directly obtain the client IP.	It is suitable for cases with high requirements on transmission efficiency and relatively low requirements on accuracy, such as instant messaging and online videos. For details, see Configuring UDP Listener.

If you use a layer-4 listener (that is, use layer-4 protocol forwarding), the CLB instance establishes a TCP connection with the backend instance on the listening port and directly forwards the request to the backend server. No data packets are modified during this process (transparent transmission mode), and the forwarding efficiency is extremely high.

Layer-7 Listener

Protocols	Note	Use Cases
HTTP	Application Layer Protocol - Supports forwarding based on request domain name and URL. - Supports cookie-based session persistence.	Applications that need to identify the content of requests, such as Web applications, App services. For details, see Configuring HTTP Listener.

Protocols	Note	Use Cases
HTTPS	Encrypted Application Layer Protocol - Supports forwarding based on request domain name and URL. - Supports cookie-based session persistence. - Unified certificate management service. CLB completes decryption operations. - Supports single authentication and two-way authentication.	HTTP applications that require encrypted transmission. For details, see Configuring HTTPS Listener.

Port Configuration

Listening port (frontend port)	Service port (backend port)	Note
The port on which the CLB is provided receives requests and forwards them to the backend server. Users can configure load balancing for ports 1 - 65,535, including 21 (FTP), 25 (SMTP), 80 (HTTP), 443 (HTTPS), etc.	The service port is the port that provides services for the CVM and receives and processes traffic from the CLB. In a CLB instance, the same CLB listening port can forward traffic to multiple ports of multiple CVMs.	In the same CLB instance: The listening port cannot be repeated. For example, you cannot create a listener TCP:80 and a listener HTTP:80 at the same time. Only ports for TCP and UDP protocols can be repeated. For example, you can create a listener for TCP:80 and a listener for UDP:80 at the same time. Service ports can be repeated within the same CLB instance. For example, the listener HTTP:80 and the listener HTTPS:443 can be bound to the same port of the same CVM at the same time.

Configuring TCP Listener

TCP Listener Introduction

You can add a TCP listener on the CLB instance to forward TCP protocol requests from the client's end. The TCP protocol is suitable for scenes that have high requirements for reliability and data accuracy but low requirements for transmission speed, such as file transfer, email sending and receiving, remote login, etc. For the listener of TCP protocol, its backend server can directly obtain the real IP of the client.

Configuring TCP Listener

Step 1: Open the Listener Management page

1. Log in to the CLB console.
2. In the left sidebar, select LB Instance List.
3. On the CLB Instance List page, click the instance ID that needs to be configured to enter the Instance Details page.
4. Click Listener Management to go to the page.

Step 2: Configure the listener

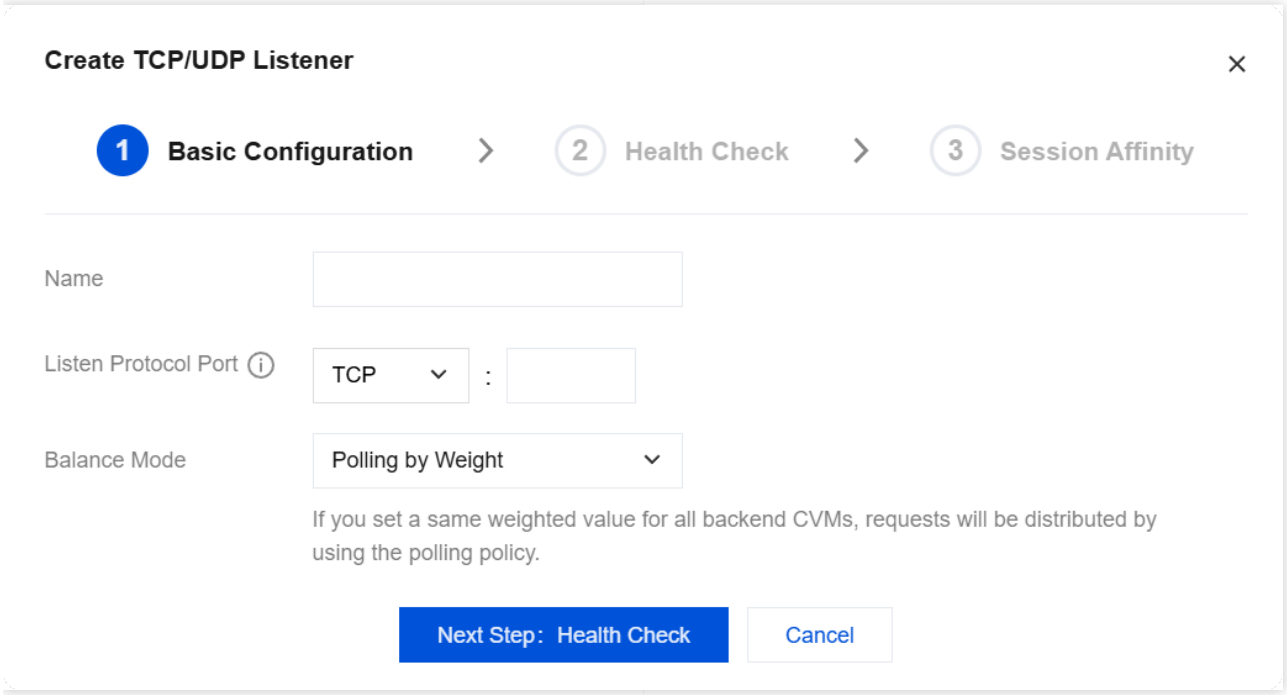
Under TCP/UDP Listeners, click Create, and configure a TCP listener in the dialog box.

1. Basic Configuration

Basic Configuration of Listener	Description	Example
Name	The Name of the Listener	test-tcp-80
Listen Protocol Port	The Listener's Protocol and Listening Port - Listening protocol: The protocols supported by CLB under this listener include TCP and UDP. In this example, TCP is selected. - Listening port: The port is used to receive requests and forward them to the backend server. The port range is 1-65535. - The listening ports in the same CLB instance cannot be repeated.	TCP:80
Balance Mode	In the TCP listener, the CLB supports two scheduling algorithms: Weighted Round-Robin (WRR) and Weighted Least Connections (WLC). - Polling by Weight: Distribute requests to different servers in turn according to the weight of the backend servers. The weighted round-robin algorithm is based on the number of newly created connections to schedule. The servers with a higher weight are more often (probably) to be polled, and servers with the same weight handle the same number of connections. - Min. Weighted Connections: The server load is estimated based on the number of active connections currently on the server. For the weighted least connections	Polling by Weight.

Basic Configuration of Listener	Description	Example
	algorithm, comprehensive scheduling is realized based on the server load and weight. When the weight values are the same, the backend server with fewer current connections will be selected more often (has a higher probability of being selected) in the round-robin scheduling.	

The basic configuration of creating a TCP listener is shown in the figure below.

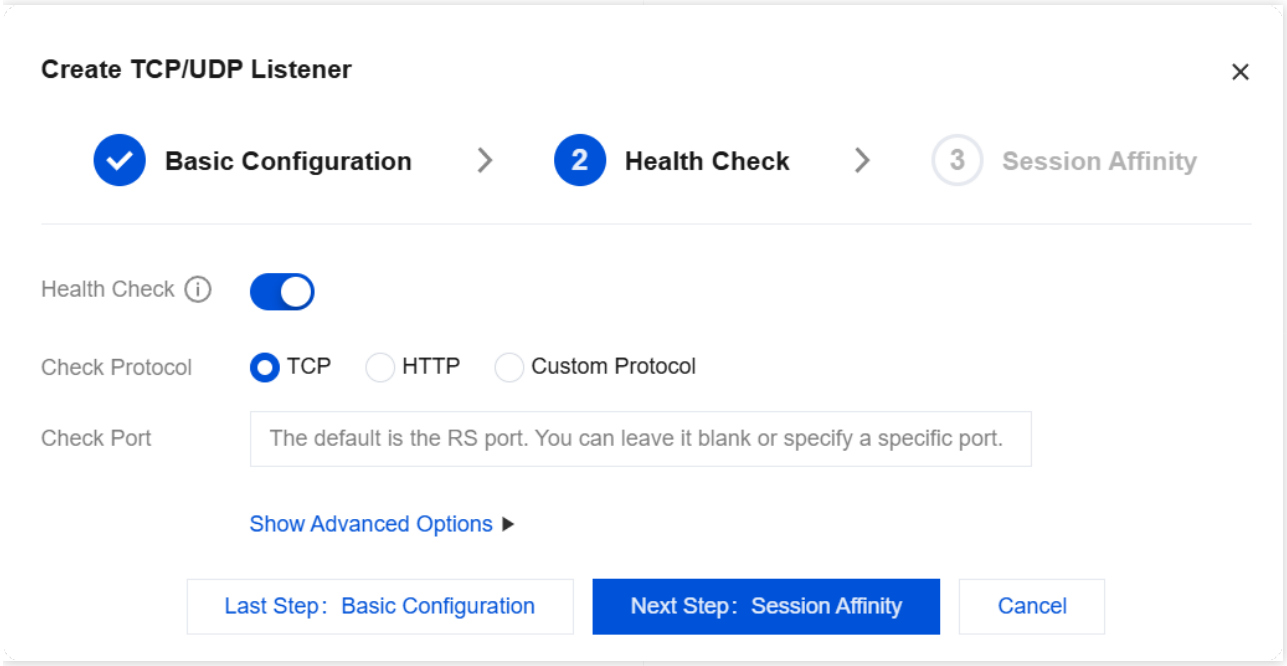


2. Health Check

Health Check Configuration	Description	Example
Health Check Status	Enable or disable health check. In the TCP listener, the CLB instance sends SYN packets to the specified server port to perform a health check.	Enable
Check Protocol	Selecting TCP indicates the configuration of a TCP health check.	TCP
Check port	Port number is not required. If it is not filled in, the backend server port will be used by default. It is recommended not to fill in the port number unless a specific port needs to be specified.	
Response Timeout	- The maximum timeout for health check responses. - If the backend CVM does not respond correctly within the timeout period, it is considered a health check exception. - Configurable range: 2 - 60 seconds. The default value is 2 seconds.	2s
Detection Interval	- Interval between CLB health checks. - Configurable range: 5-300 seconds, default value is 5 seconds.	5s

Health Check Configuration	Description	Example
Unhealthy Threshold	- If the health check results received for n consecutive times (n is the filled value) are failed, it is identified as unhealthy, and the console displays as Abnormal. - Configurable range: 2-10 times, default value is 3 times.	3 times
Health Threshold	- If the health check results received for n consecutive times (n is the filled value) are successful, it is identified as healthy, and the console displays as Healthy. - Configurable range: 2-10 times, default value is 3 times.	3 times

The specific configuration of the health check is shown in the following figure:

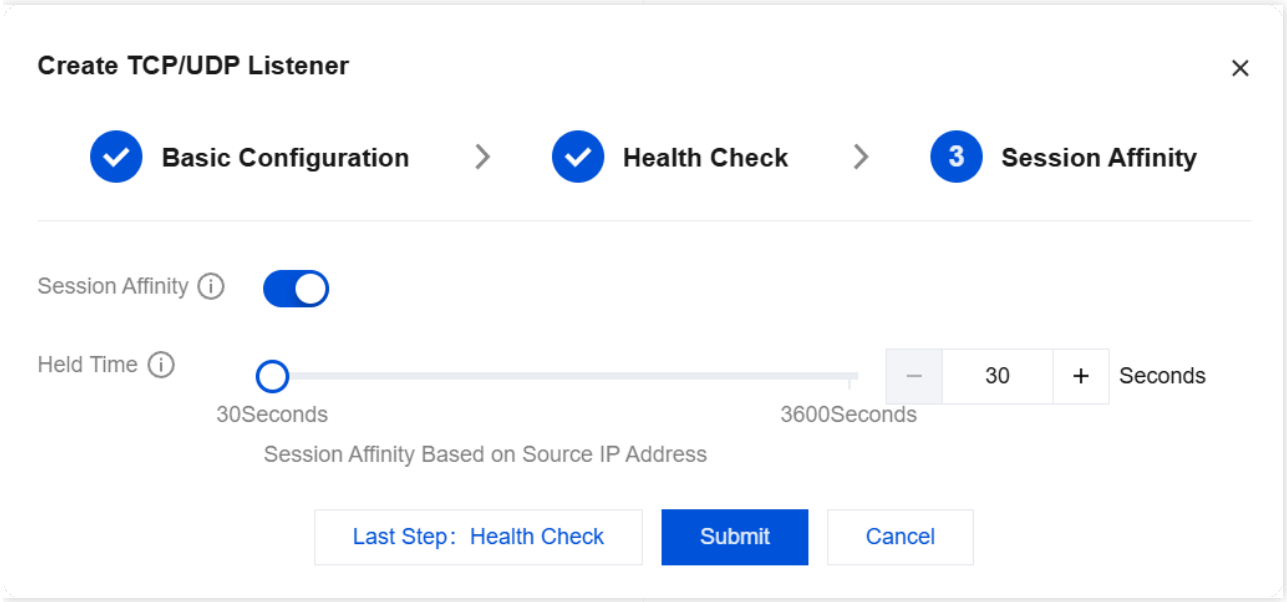


3. Session Persistence

Session Persistence Configuration	Description	Example
Session Affinity Status	Enable or disable session persistence. - After session persistence is enabled, the CLB listener will distribute access requests from the same client to the same backend server. - The TCP protocol is session persistence based on the client IP address, that is, access requests from the same IP address are forwarded to the same backend server. - WRR scheduling supports session persistence, but WLC scheduling does not support enabling the session persistence feature.	Enable
Session Held Time	Session Held Time - When the hold time is exceeded and there is no new request in the connection, the session will be automatically disconnected.	30s

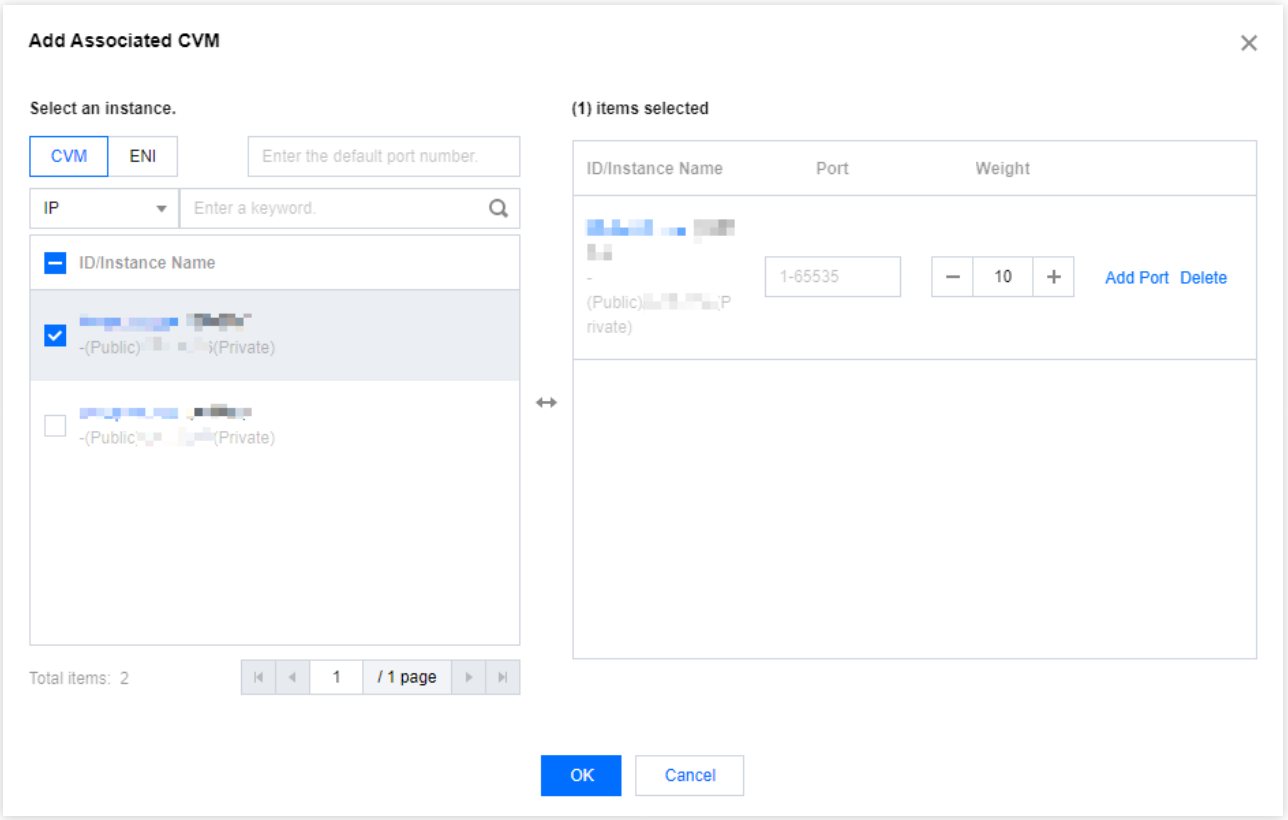
Session Persistence Configuration	Description	Example
	- Configurable range 30-3,600 seconds.	

The specific configuration of session persistence is shown in the following figure.



Step 3: Bind the backend CVM

1. On the Listener Management page, click the successfully created listener, such as the aforementioned TCP:80 listener, to view the bound backend service on the right side of the listener.
2. Click Bind, select the backend server to be bound in the pop-up, and configure the service port and weight.
   - Add Port Features: In the "Selected" Cloud Virtual Machine box on the right, click Add Port to add multiple ports of the same Cloud Virtual Machine, such as adding ports 80, 81, and 82 of CVM at the same time.
   - Default port feature: Fill in Default Port first, then select the CVM. The port of each CVM is the default port.



After completing Step 1 to Step 3, the TCP listener rules are configured.

Step 4: Modify/delete listeners (optional)

If you need to modify or delete a created listener, click the created listener on the Listener Management page, and then click the modify or delete icon to complete the operation.

Configuring UDP Listener

UDP Listener Introduction

You can add a UDP listener on the CLB instance to forward UDP protocol requests from clients. The UDP protocol is suitable for scenarios with high requirements for transmission efficiency and relatively low requirements for accuracy, such as instant messaging and online videos. The UDP protocol listener and the backend server can directly obtain the real IP of the client.

Configuring UDP Listener

Step 1: Open the Listener Management page

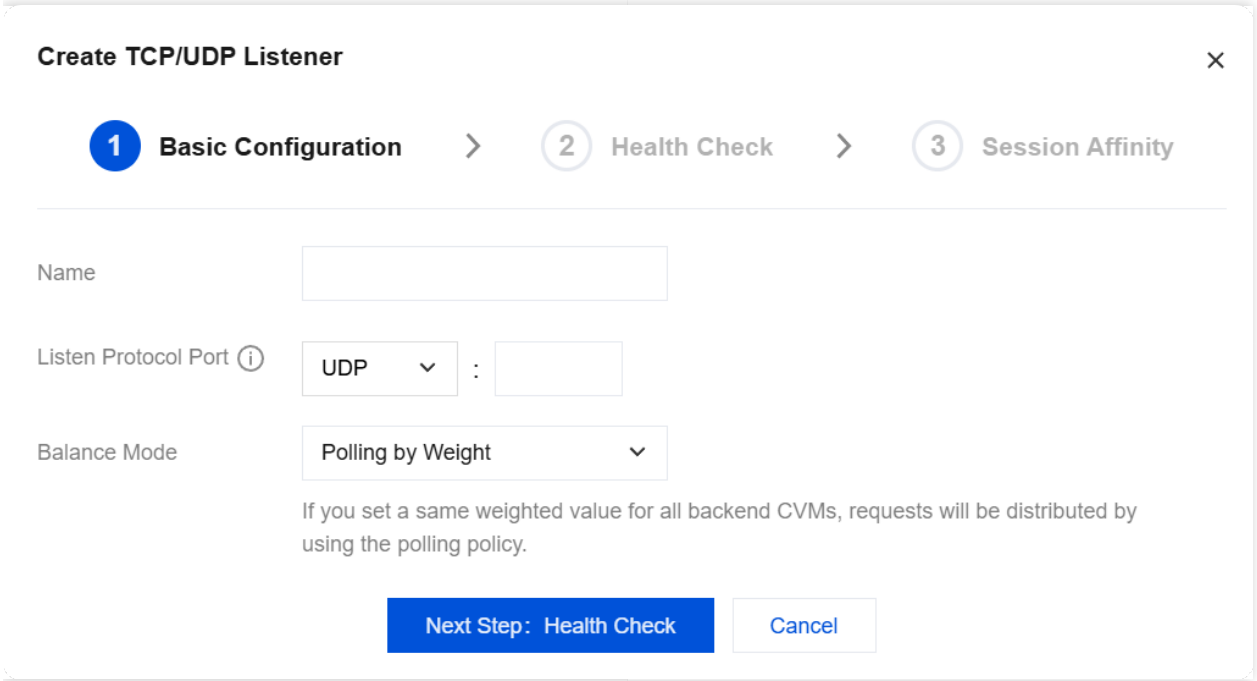
1. Log in to the CLB console.
2. In the left sidebar, select LB Instance List.
3. On the CLB Instance List page, click the instance ID that needs to be configured to enter the Instance Details page.
4. Click the Listener Management tab.

Step 2: Configure the listener

Under TCP/UDP Listener, click Create and configure the UDP listener in the pop-up box.

1. Basic Configuration

The specific basic configuration of creating a UDP listener is shown in the figure below.

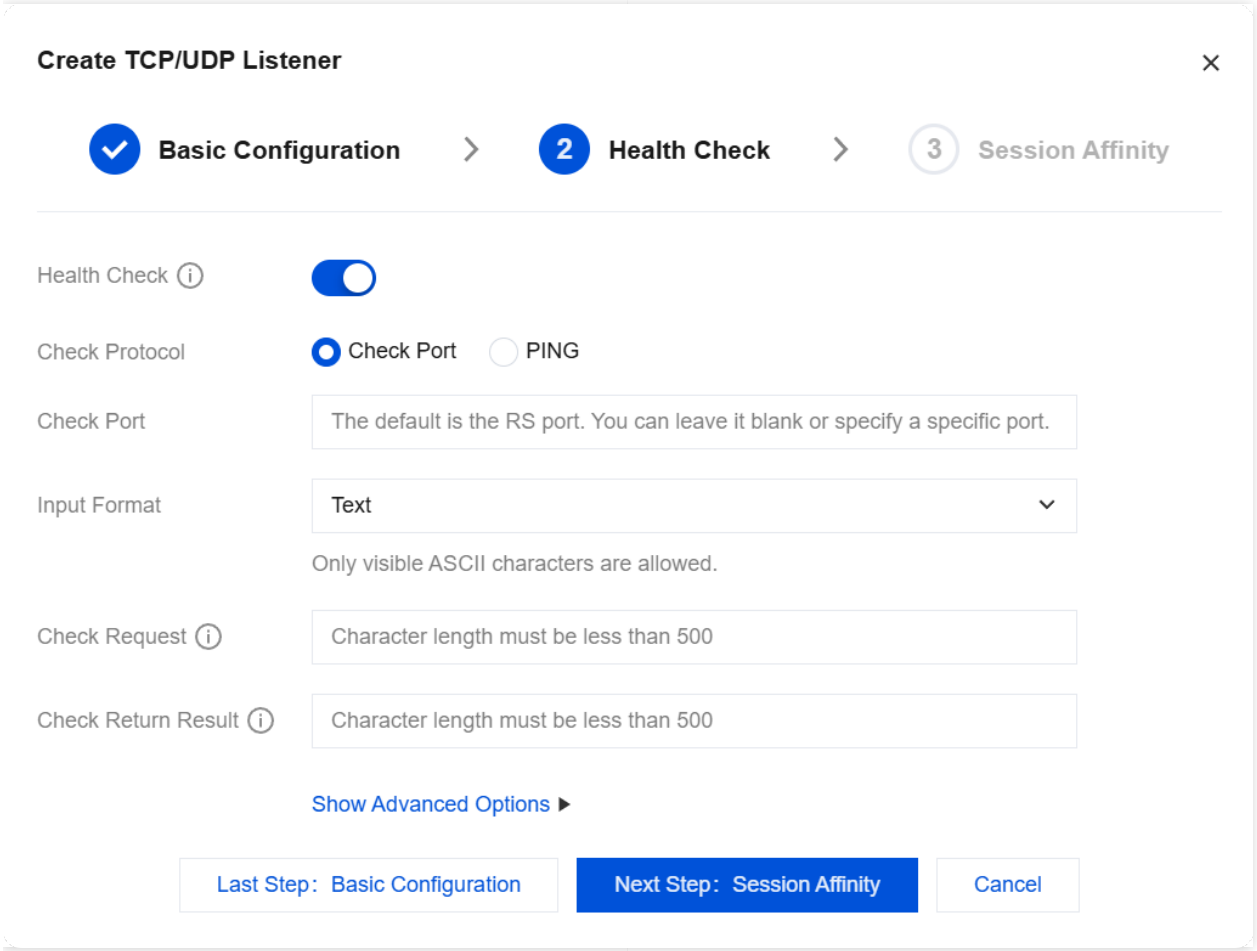


Basic Configuration of	Description	Example

Listener		
Name	The Name of the Listener	test-udp-8000
Listen Protocol Port	The listener's protocol and listening port. - Listener protocol: the protocols supported by CLB under this listener include TCP and UDP. In this example, UDP is selected. - Listener port: the port used to receive requests and forward them to the backend server. The port range is 1 - 65,535. The listening ports in the same CLB instance cannot be repeated.	UDP:8000
Balance Mode	In the UDP listener, CLB supports two scheduling algorithms: weighted round-robin (WRR) and weighted least connections (WLC). - Polling by Weight: Requests are distributed to different servers in turn based on the weight of the backend servers. Schedule by the weighted round-robin algorithm, based on the number of new connections. The server with a higher weight is rounded more often (high probability), and servers with the same weight handle the same number of connections. - Min. Weighted Connections: The server load is estimated based on the number of active connections currently on the server. For the weighted least connections algorithm, comprehensive scheduling is realized based on the server load and weight. When the weight values are the same, the backend server with fewer current connections will be selected more often (has a higher probability of being selected) in the round-robin scheduling.	Polling by Weight.

2. Health Check

The specific configuration of health check is shown in the figure below.

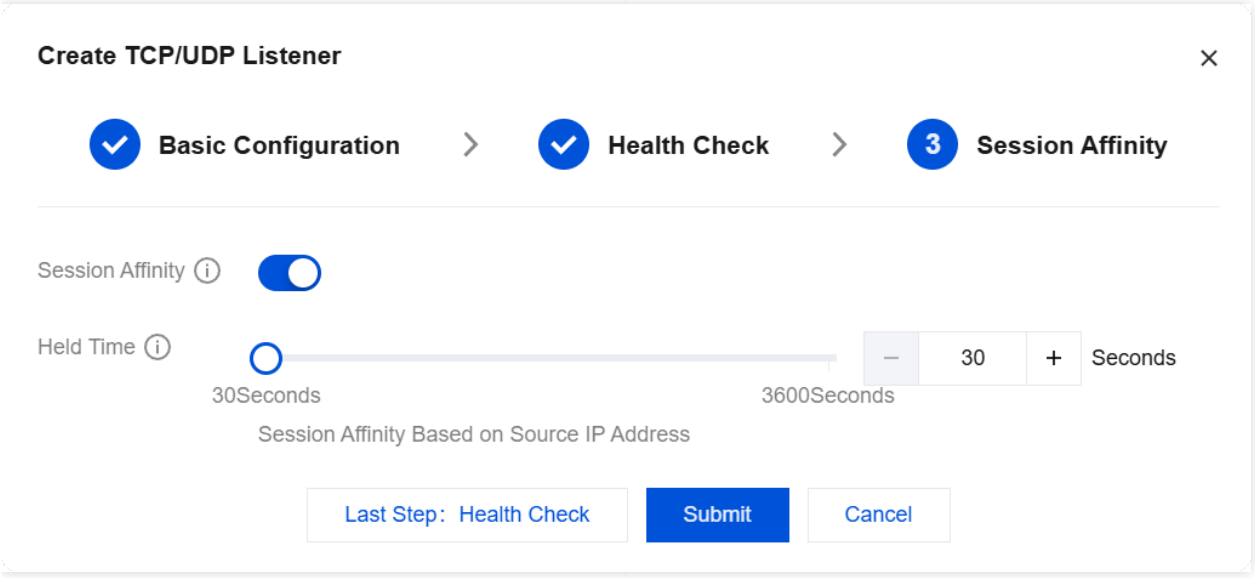


Parameter	Description
Health Check	You can either enable or disable the health check feature. It is recommended to enable the health check to help you automatically check and remove abnormal backend CVM ports.
Check Protocol	- Select Check Port, indicating that the VIP (the IP address that the CLB uses to provide services to clients) sends UDP detection messages to the backend CVM and obtains the status information of the backend CVM by pinging the IP address of the backend CVM. - Select PING to obtain the status information of the backend CVM by pinging the IP address of the backend CVM.
Check port	Port number is not required. If it is not filled in, the backend server port will be used by default. It is recommended not to fill in the port number unless a specific port needs to be specified.
Input Format	Text and hexadecimal are supported. If input format is text, text is converted into binary for request sending and compared with returned result. If input format is hexadecimal, hexadecimal is converted into binary for request sending and compared with returned result.
Check Request	Customize the health check request content. For example, a check request for detecting the DNS service is: F13E0100000100000000000003777777047465737403636F6D0774656E63656E7403636F6D0000010001.

Parameter	Description
Check Return Result	When you customize the health check request, you must configure the returned health check result. For example, the result returned by the DNS service detection is: F13E.
Health Check Status	Enable or disable health check. In the UDP listener, the CLB instance sends a ping to the server for health check.
Response Timeout	- The maximum timeout for health check responses. - If the backend CVM does not respond correctly within the timeout period, it is considered a health check exception. - Configurable range: 2 - 60 seconds. The default value is 2 seconds.
Detection Interval	- Interval between CLB health checks. - Configurable range: 5-300 seconds, default value is 5 seconds.
Unhealthy Threshold	- If the health check results received for n consecutive times (n is the filled value) are failed, it is identified as unhealthy, and the console displays as Abnormal. - Configurable range: 2-10 times, default value is 3 times.
Health Threshold	- If the health check results received for n consecutive times (n is the filled value) are successful, it is identified as healthy, and the console displays as Healthy. - Configurable range: 2-10 times, default value is 3 times.

3. Session Persistence

The specific configuration of session persistence is shown in the following figure.

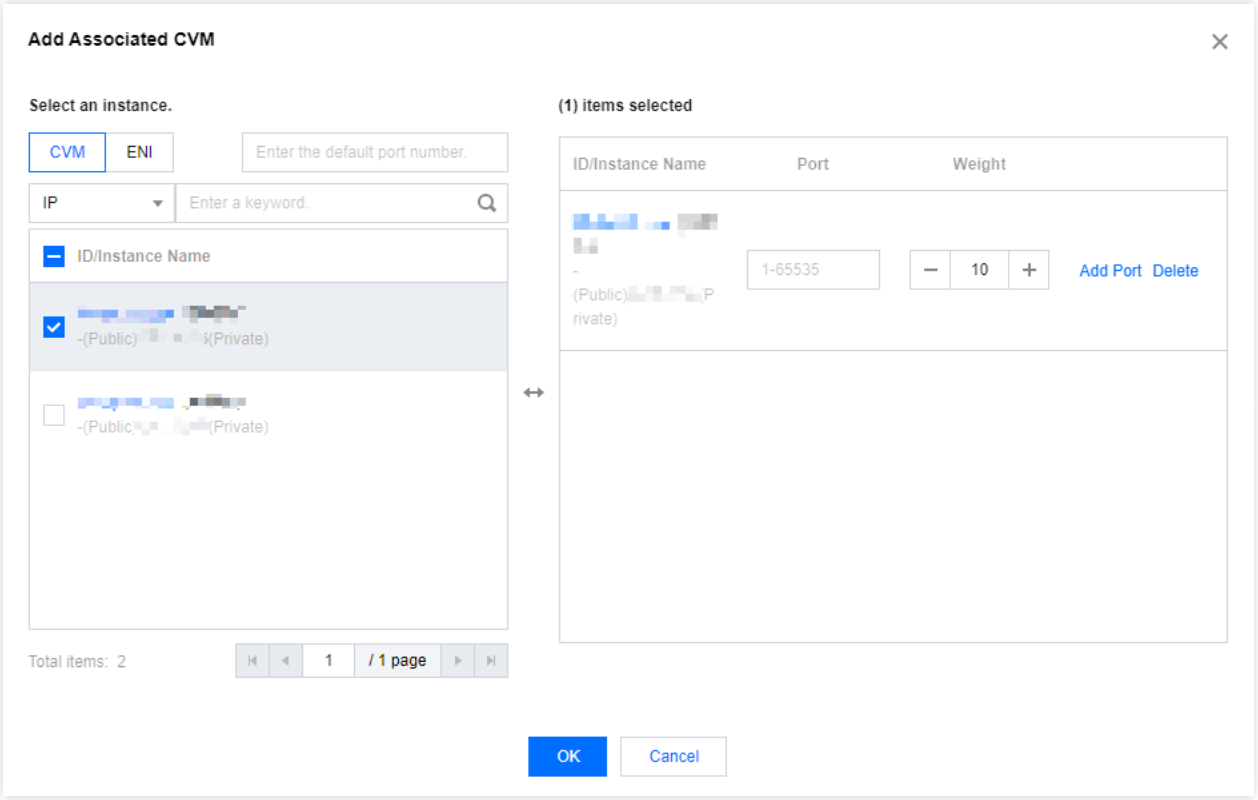


Session Persistence Configuration	Description	Example
Session Affinity Status	Enable or disable session persistence. - After session persistence is enabled, the Load Balance Listener will distribute access requests from the same client to the same backend server. - The UDP protocol is based on sessions persistence of the client IP address,	Enable

Session Persistence Configuration	Description	Example
	that is, forwarding access requests from the same IP address to the same backend server. - Weighted round-robin scheduling supports session persistence, while weighted least connections scheduling does not support session persistence.	
Session Held Time	- When the hold time is exceeded and there is no new request in the connection, the session will be automatically disconnected. - Configurable range 30-3,600 seconds.	30s

Step 3: Bind the backend CVM

1. On the Listener Management page, click the created listener, such as the aforementioned UDP:8000 listener, to view the bound backend service on the right side of the listener.
2. Click Bind, select the backend server to be bound in the pop-up, and configure the service port and weight.



Once you have completed Step 1 to Step 3, the UDP listener rules is configured.

Step 4: Modify/delete the listener

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

Configuring HTTP Listeners

HTTP Listener Introduction

You can add an HTTP listener to the CLB instance to forward HTTP protocol requests from clients. The HTTP protocol is suitable for applications that need to identify the content of the request, such as Web applications, App services, etc.

Configuring HTTP Listeners

Step 1: Open the Listener Management page

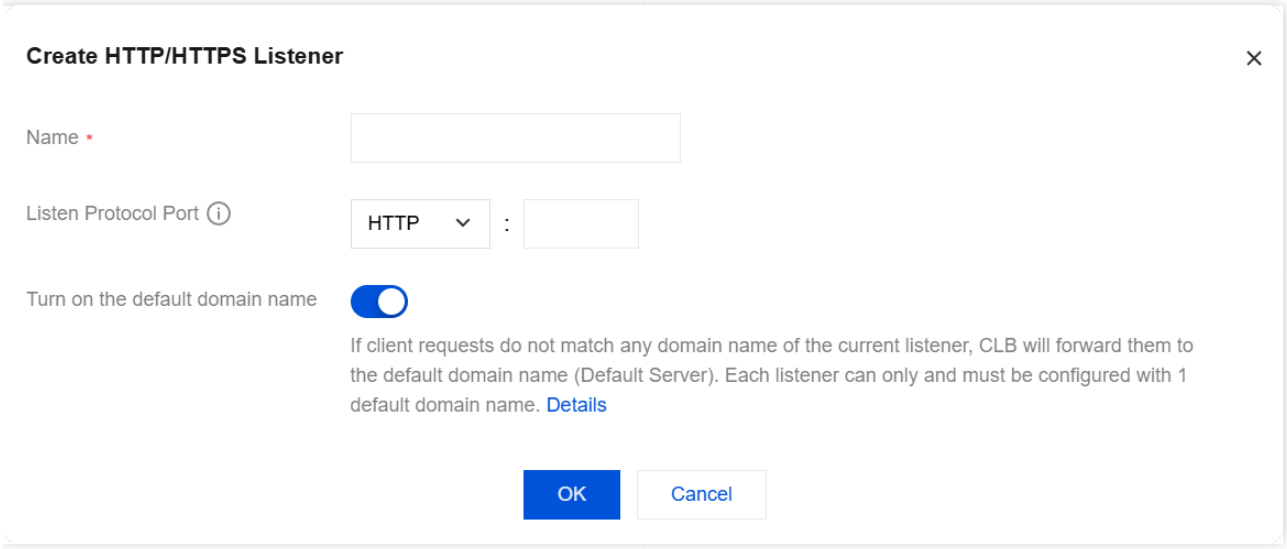
1. Log in to CLB control and click LB Instance List in the left sidebar.
2. On the CLB instance list page, click the target instance ID.
3. On the instance details page, click the Listener Management tab page.

Step 2: Configure the listener

In the HTTP/HTTPS listener module, click Create and configure the HTTP listener in the pop-up box.

1. Create a listener

Parameter	Description
Name	The name of the listener. In this example, it can be customized as Listener1.
Listen Protocol Port	The listener's protocol and listening port. - The listening protocols include HTTP and HTTPS. In this example, HTTP is selected. - The listening port is used to receive requests and forward them to the backend server. The port range is 1 - 65535. - The listening ports in the same CLB instance cannot be repeated.
Default Domain Name	You can choose to enable or disable it. - Enable: When the client request does not match any domain name of this listener, CLB will forward the request to the default domain name (Default Server). Each listener can only be configured with one default domain name. - If your Layer-7 listener has been configured with a default domain name, client requests that do not match other rules will be forwarded to the default domain name. - If your Layer-7 listener is not configured with a default domain name, client requests that do not match other rules will be forwarded to the first domain name loaded by the CLB. Since the loading order may be inconsistent with the console configuration order, it is not necessarily the first one configured by the console. - Disable: Client requests that do not match any domain name of this listener cannot be forwarded.

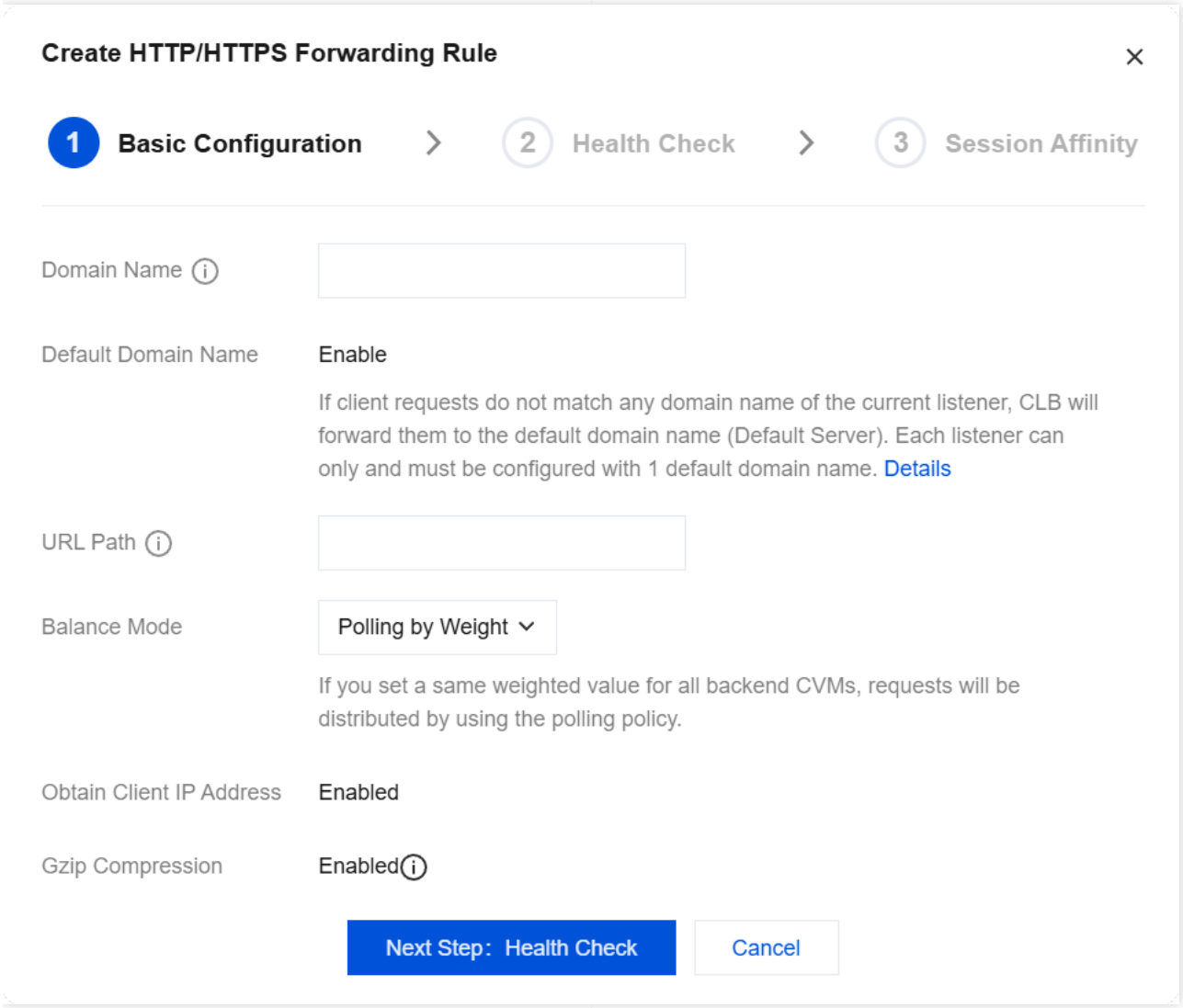


2. Create forwarding rules

Basic Configuration of Forwarding Rules	Description	Example
Domain name	Request a domain name. - Support precise domain names, such as www.example,com . Support wildcard domain names, such as *.example.com or www.example.* , in a single domain name * Can only appear once. - The character sets supported by irregular domain names are as follows: az``0-9``.``- . - Regular expressions are supported. The character sets that are not supported in regular expressions are as follows: "``{``};``\ ``` ~`````Space . - The domain name length limit is 1-120.	www.example.com
Default Domain Name	- When all domain names in the listener are not matched successfully, the system will direct the request to the default access domain name to make the default access controllable. - Only one default domain name can be configured for a listener.	Enable
URL Path	Request path. - The default is / , must be preceded by / .The length limit is 1 - 120. - Non-regular URL path starts with / . The supported character sets are as follows: az``AZ``0-9``.``-``/``=``? . - Support regular expressions. - Beginning with = indicates an exact match. - Beginning with ^~ indicates that the uri starts with a regular character string, not a regular match. - Beginning with ~ indicates a case-sensitive regular match. - Beginning with ~* indicates a case-insensitive regular match. - / means universal match; if there is no other match, any request will match - For the regular URL, unsupported character sets are as follows: "``{``}	/index

Basic Configuration of Forwarding Rules	Description	Example
	``;``\ ``` ~`````Space .	
Balance Mode	In the HTTP listener, CLB supports three scheduling algorithms: weighted round-robin (WRR), weighted least connections (WLC), and IP Hash. - Polling by Weight: Distributes requests to different servers in sequence based on the weights of the backend servers. The weighted round-robin algorithm schedules based on Create connection count, where servers with higher weights are polled more frequently (probability), and servers with the same weight handle an equal number of connections. - Min. Weighted Connections: Estimates the server load based on the number of active connections currently on the server. The weighted least connections are comprehensively scheduled based on the server load and weight. When the weight values are the same, the backend server with a smaller current number of connections has a higher probability of being rounded. - IP Hash: Based on the source IP address of the request, use the Hash key to find the corresponding server from the statically allocated hash table. If the server is available and not overloaded, the request is sent to the server; otherwise, it returns nothing.	Polling by Weight.
Obtain Client IP address	Enabled by default.	Enabled
Gzip Compression	Enabled by default.	Enabled

Select the HTTP listener for which you want to create a forwarding rule, and click the + sign on the right to create a rule. The basic configuration is as shown below.

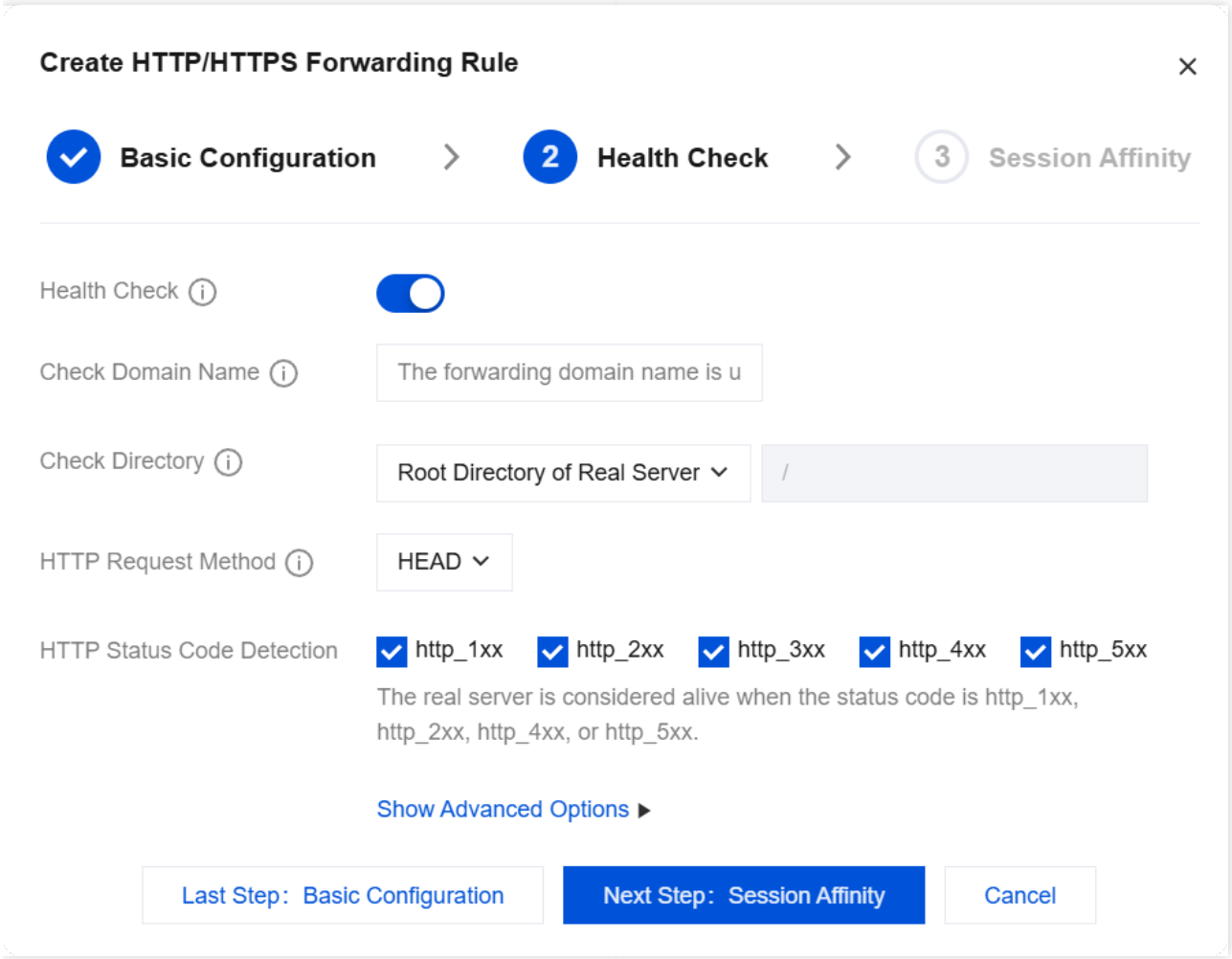


3. Health check

Health Check Configuration	Description	Example
Health Check	Enable or disable health check. In the HTTP listener, the CLB instance sends an HTTP request to the specified server port for health check.	Enable
Check Domain Name	Request domain - It is the forwarding domain name by default. - Supported character sets include: az``0-9``.``- , length limit is 1-120. - Regular expressions are not currently supported. - When the user fills in the domain name as a wildcard domain name, it is necessary to specify a fixed domain name (non-regular) as the health check domain name.	Use the default value (that is, www.example.com).
Check Directory	Request path - The default is / , must start with / . - Supported character sets include: az``0-9``.``- , length limit	Use the default value (that is, /).

Health Check Configuration	Description	Example
	is 1-120. - Regular expressions are not currently supported. - It is recommended to specify a fixed URL path (static page) for a health check.	
Detection Interval	- Interval between CLB health checks. - Configurable range: 5-300 seconds, default value is 5 seconds.	5s
Unhealthy Threshold	- If the health check results received for n consecutive times (n is the filled value) are failed, it is identified as unhealthy, and the console displays as Abnormal. - Configurable range: 2-10 times, default value is 3 times.	3 times
Health Threshold	- If the health check results received for n consecutive times (n is the filled value) are successful, it is identified as healthy, and the console displays as Healthy. - Configurable range: 2-10 times, default value is 3 times.	3 times
HTTP Request Method	HTTP request method for the health check. Options: GET or HEAD. The default option is GET. - If the HEAD method is used, the server only returns the HTTP header information, which can reduce backend overhead and improve request efficiency. The corresponding backend service needs to support HEAD. - If the GET method is used, the backend service only needs to support GET.	GET
HTTP Status Code Detection	When the status code is the selected status code, the backend server is considered alive, that is, the health check is normal. Optional: http_1xx, http_2xx, http_3xx, http_4xx, http_5xx.	Multiple selection: http_1xx, http_2xx, http_3xx, http_4xx

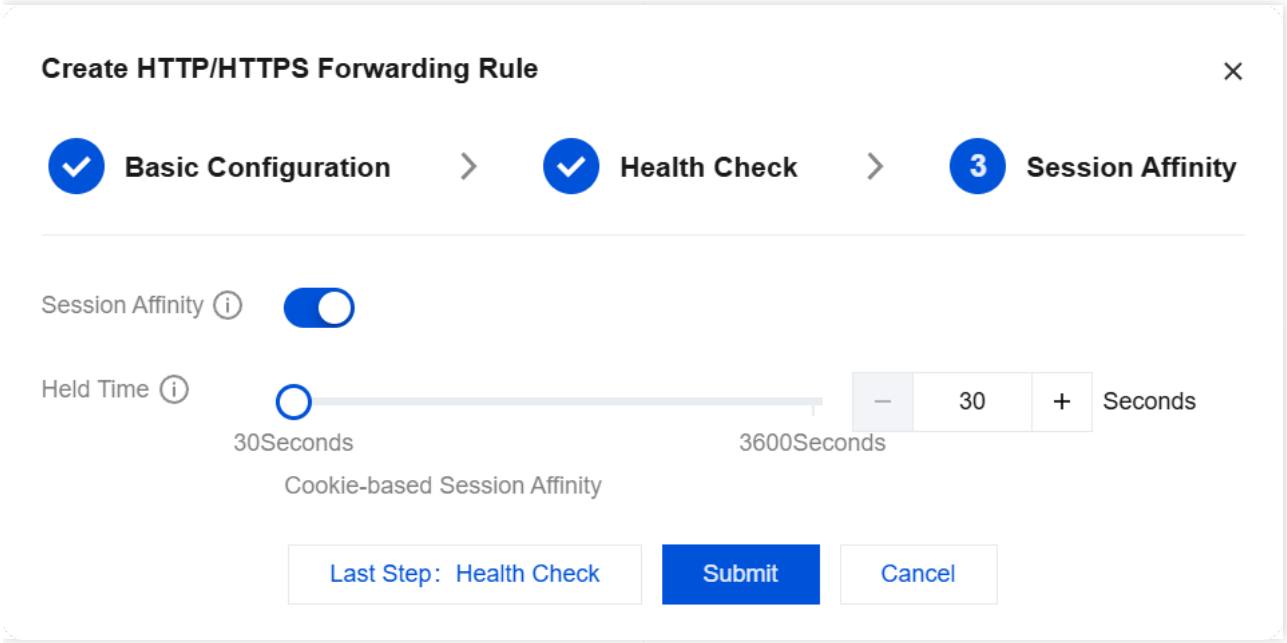
The specific configuration of health check is shown in the figure below.



4. Session persistence

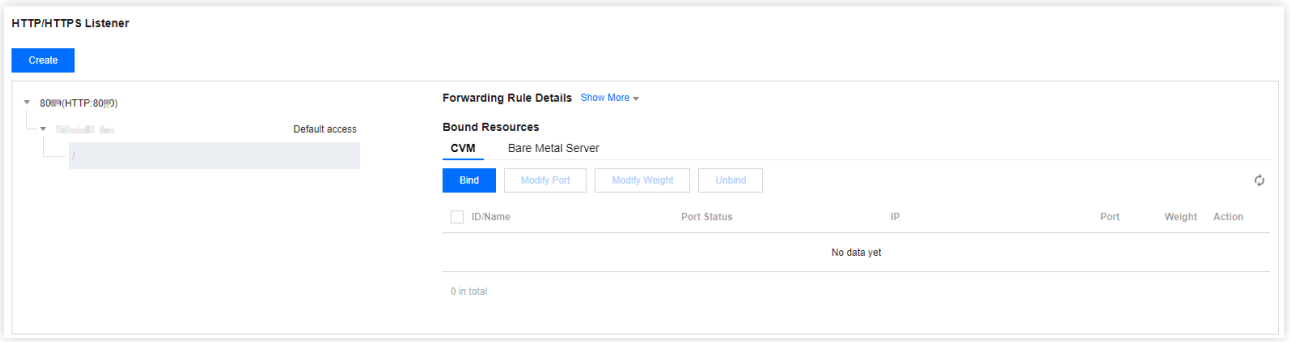
Session Persistence Configuration	Description	Example
Session Affinity Status	Enable or disable session persistence. - After session persistence is enabled, the CLB listener will distribute access requests from the same client to the same backend server. - The HTTP protocol is based on cookie session persistence, and the CLB embeds cookies into the client. - Weighted round-robin scheduling supports session persistence, while weighted least connections and IP Hash scheduling do not support session persistence.	Enable
Session Held Time	Session hold time. - When the hold time is exceeded and there is no new request in the connection, the session will be automatically disconnected. - Configurable range 30-3,600 seconds.	30s

The specific configuration of session persistence is shown in the following figure.

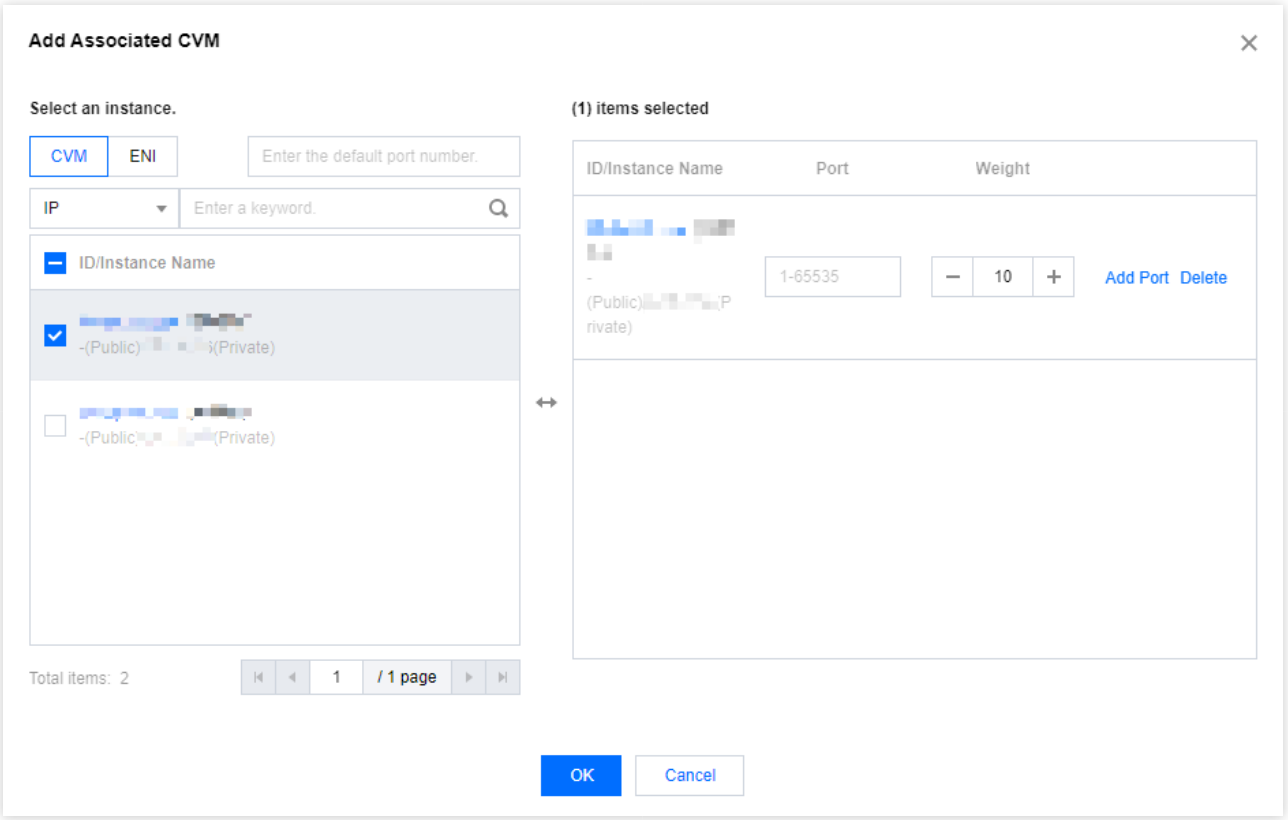


Step 3: Bind the backend CVM

1. On the Listener Management page, click the created listener, such as HTTP:80 listener above. Click the triangle icon on the left to expand the domain name and URL path, select a specific URL path, and you can view the backend services bound to the path on the right side of the listener.



2. Click Bind, select the backend server to be bound in the pop-up, and configure the service port and weight.
   - Add port feature: In the Selected CVM box on the right, click Add Port to add multiple ports to the same CVM.
   - Default port feature: Fill in Default Port first, then select the CVM. The port of each CVM is the default port.



Step 4: Modify/delete the listener

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

Configuring an HTTPS Listener

HTTPS Listener Introduction

You can add an HTTPS listener to your CLB instance to forward HTTPS requests from clients. The HTTPS protocol is suitable for HTTP applications that require encrypted transmission.

Configuring an HTTPS Listener

Step 1: Open the Listener Management page

1. Log in to the CLB console. In the left sidebar, click LB Instance List.
2. On the CLB instance list page, click the target instance ID.
3. On the instance details page, click the Listener Management tab page.

Step 2: Configure the listener

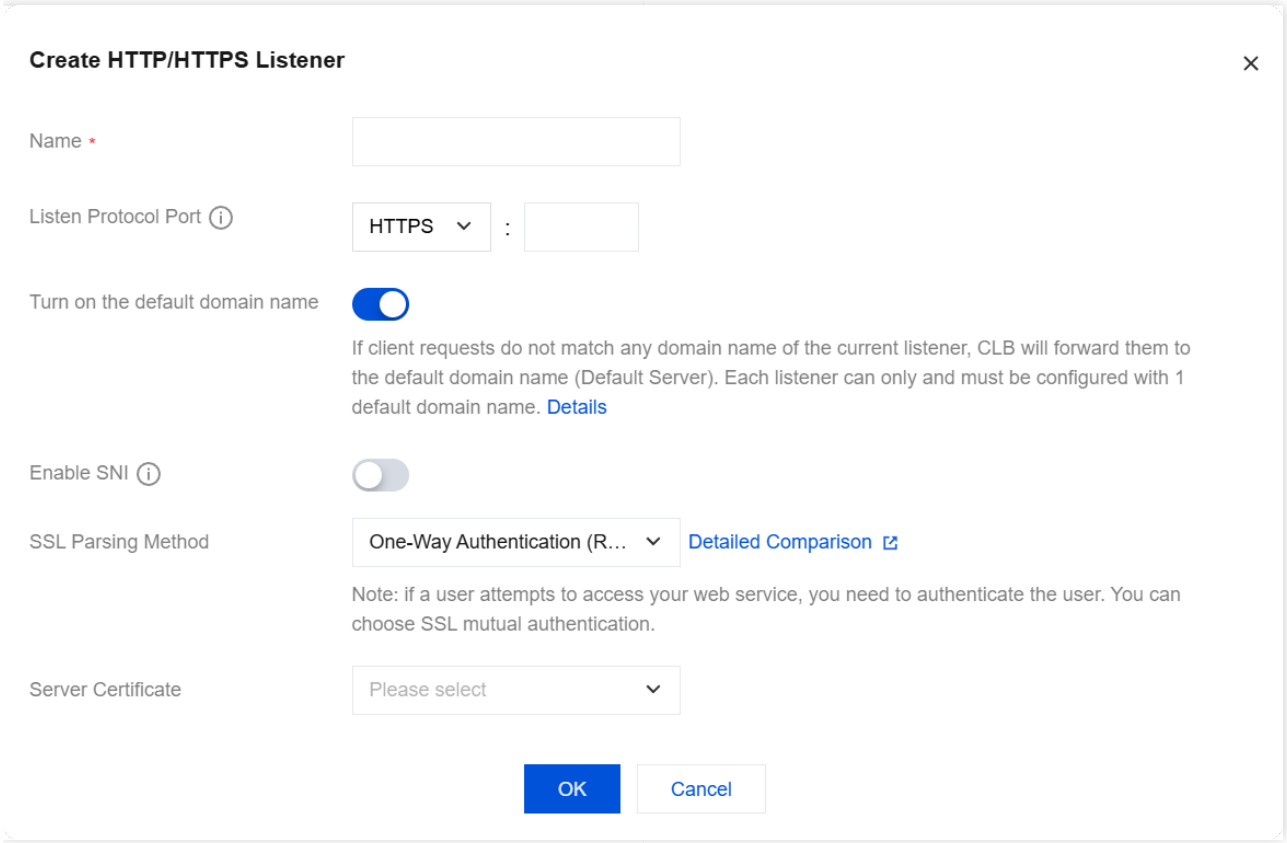
In the HTTP/HTTPS Listener module, click Create and configure an HTTPS listener in the pop-up.

1. Create a listener

Basic Configuration of Listener	Description	Example
Name	The name of the listener.	test-https-443
Listen Protocol Port	The listener's protocol and listening port. - Listener protocol: The protocols supported by CLB under this listener include HTTP and HTTPS. In this example, HTTPS is selected. - Listening port: The port is used to receive requests and forward them to the backend server. The port range is 1-65535. - The listening ports in the same CLB instance cannot be repeated.	HTTPS:443

Basic Configuration of Listener	Description	Example
Default Domain Name	You can choose to enable or disable it. - Enable: When the client request does not match any domain name of this listener, CLB will forward the request to the default domain name (Default Server). Each listener can only be configured and is required to be configured with one default domain name. - If your layer-7 listener has been configured with a default domain name, client requests that do not match other rules will be forwarded to the default domain name. - If your layer-7 listener is not configured with a default domain name, client requests that do not match other rules will be forwarded to the first domain name loaded by CLB. Since the loading order may be inconsistent with the console configuration order, the domain name is not necessarily the first one configured by the console. - Disable: Client requests that do not match any domain name of this listener cannot be forwarded.	Enable
Enabling SNI	Enabling SNI means that different certificates can be configured for different domain names under a listener. Not enabling SNI means that multiple domain names under the listener use the same certificate.	Enable
SSL Parsing Method	Support one-way authentication and two-way authentication.	One-way Authentication
Server Certificate	Select an existing certificate.	Selecting an Existing Certificate

The specific configuration for creating an HTTPS listener is shown in the figure below.

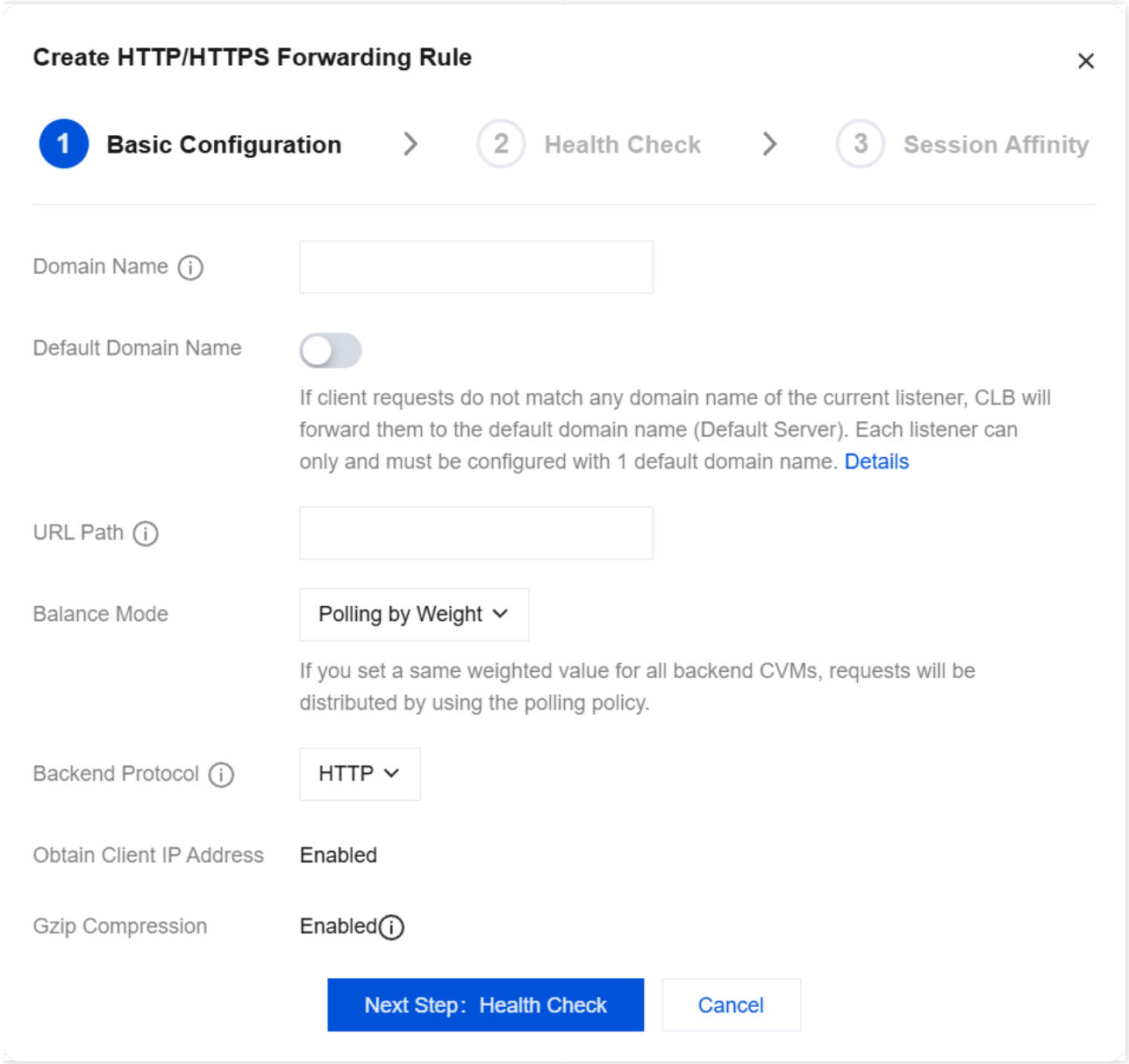


2. Create forwarding rules

Basic Configuration of Forwarding Rules	Description	Example
Domain name	Request a domain name. - Support precise domain names, such as www.example,com . Support wildcard domain names, such as *.example.com or www.example.* , in a single domain name * Can only appear once. - The character sets supported by irregular domain names are as follows: az``0-9``.``- . - Support regular expressions. The character sets that are not supported in regular expressions are as follows: "``{``} ``;``\`````~`````Space . - The domain name length limit is 1-120.	www.example.com
Default Domain Name	- When all domain names in the listener are not matched successfully, the system will direct the request to the default access domain name to make the default access controllable. - Only one default domain name can be configured for a listener.	Enable
URL Path	Request path. - The default is / , must be preceded by / .The length limit is 1-120 . - Non-regular URL path starts with / . The supported character sets are as follows: a-z``A-Z``0-9``.``-``/``=``? . - Support regular expressions.	/index

Basic Configuration of Forwarding Rules	Description	Example
	- Beginning with = indicates an exact match. - Beginning with ^~ indicates that the uri starts with a regular character string, not a regular match. - Beginning with ~ indicates a case-sensitive regular match. - Beginning with ~* indicates a case-insensitive regular match. - / means universal matching. Any request will be matched if there is no other match. - For the regular URL, unsupported character sets are as follows: "``{``} ``;``\ ``` ~`````Space .	
Balance Mode	In the HTTPS listener, CLB supports three scheduling algorithms: weighted round-robin (WRR), weighted least connections (WLC), and IP Hash. - Polling by Weight: Requests are distributed to different servers in turn based on the weight of the backend servers. Schedule by the weighted round-robin algorithm, based on the number of new connections. The server with a higher weight is rounded more often (high probability), and servers with the same weight handle the same number of connections. - Min. Weighted Connections: The server load is estimated based on the number of active connections currently on the server. For the weighted least connections algorithm, comprehensive scheduling is realized based on the server load and weight. When the weight values are the same, the backend server with fewer current connections will be selected more often (has a higher probability of being selected) in the round-robin scheduling. - IP Hash: Based on the source IP address of the request, use the Hash key to find the corresponding server from the statically allocated hash table. If the server is available and not overloaded, the request is sent to the server; otherwise, it returns nothing.	Polling by Weight.
Backend protocol	The backend protocol refers to the protocol used between CLB and the backend service: - If you select HTTP as the backend protocol, you need to deploy HTTP service for the backend service. - If you select HTTPS as the backend protocol, you need to deploy HTTPS service for the backend service. Encryption and decryption of HTTPS service will consume more resources on the backend service. The backend service needs to be configured with a corresponding SSL certificate. Traditional accounts do not support selecting HTTPS as the backend forwarding protocol.	HTTPS
Obtain Client IP address	Enabled by default.	Enabled
Gzip Compression	Enabled by default.	Enabled

Select the HTTPS listener for which you want to create a forwarding rule, and click Add Rule to create a rule. The basic configuration is as shown below.

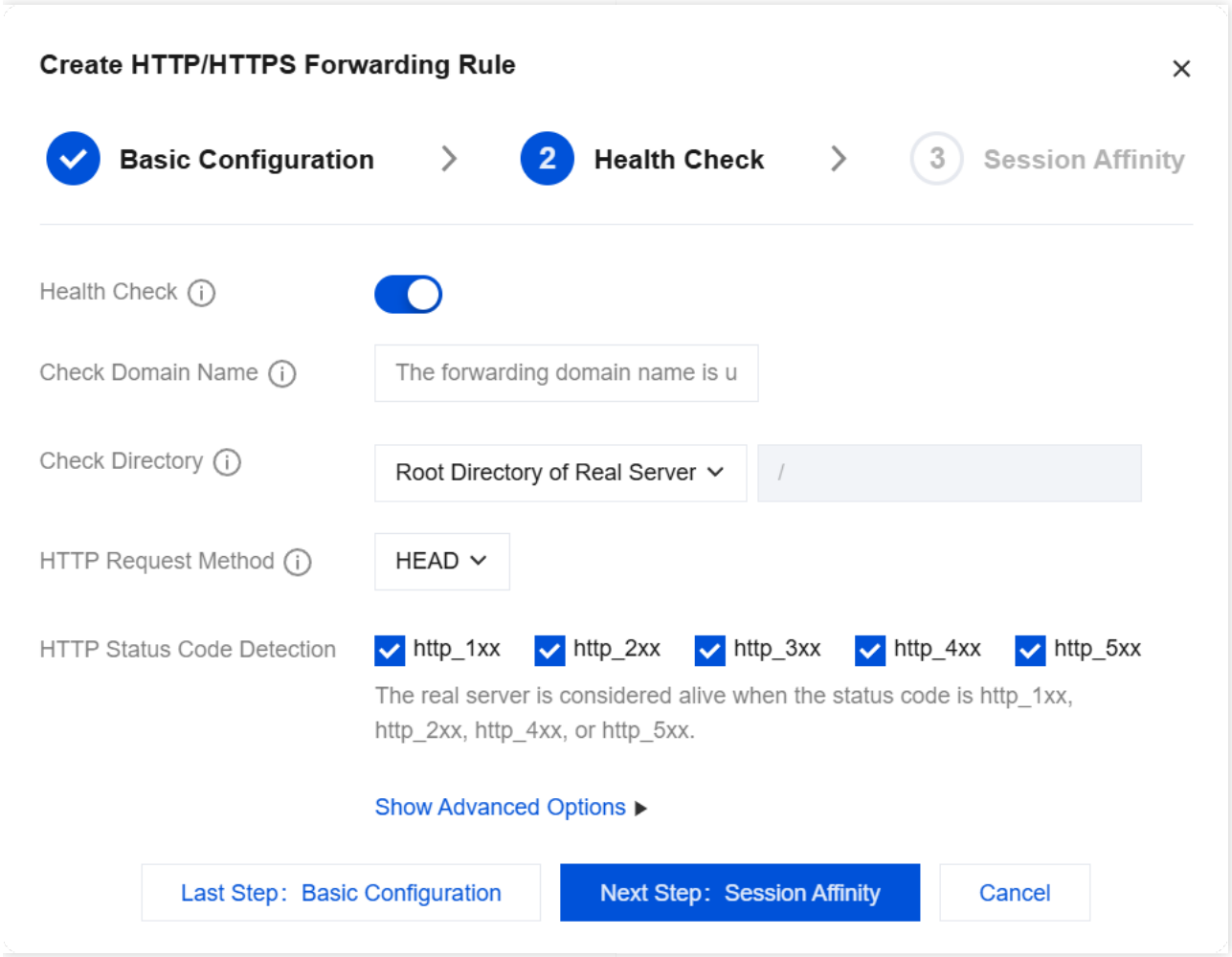


3. Health check

Health Check Configuration	Description	Example
Health Check	Enable or disable health check. In the HTTPS listener, the CLB instance sends HTTPS requests to the specified server port to perform a health check.	Enable
Check Domain Name	Request a domain name. - The default is the access domain name. - Supported character sets include: az``0-9``.``- , length limit is 1-120. - Regular expressions are not currently supported. - When the user fills in the domain name as a wildcard	Use the default value (that is, www.example.com).

Health Check Configuration	Description	Example
	domain name, it is necessary to specify a fixed domain name (non-regular) as the health check domain name.	
Check Directory	Request path. - The default is /, must start with /. - Supported character sets include: az``0-9``.``- , length limit is 1-120. - Regular expressions are not currently supported. - It is recommended to specify a fixed URL path (static page) for a health check.	Use the default value (that is, /).
Detection Interval	- Interval between CLB health checks. - Configurable range: 5-300 seconds, default value is 5 seconds.	5s
Unhealthy Threshold	- If the health check results received for n consecutive times (n is the filled value) are failed, it is identified as unhealthy, and the console displays as Abnormal. - Configurable range: 2-10 times, default value is 3 times.	3 times
Health Threshold	- If the health check results received for n consecutive times (n is the filled value) are successful, it is identified as healthy, and the console displays as Healthy. - Configurable range: 2-10 times, default value is 3 times.	3 times
HTTP Request Method	HTTP request method for the health check. Options: GET or HEAD. The default option is GET. - If the HEAD method is used, the server only returns the HTTP header information, which can reduce backend overhead and improve request efficiency. The corresponding backend service must support HEAD. - If the GET method is used, the backend service only needs to support GET.	GET
HTTP Status Code Detection	When the status code is the selected status code, the backend server is considered alive, that is, the health check is normal. Optional: http_1xx, http_2xx, http_3xx, http_4xx, http_5xx.	Multiple selection: http_1xx, http_2xx, http_3xx, http_4xx

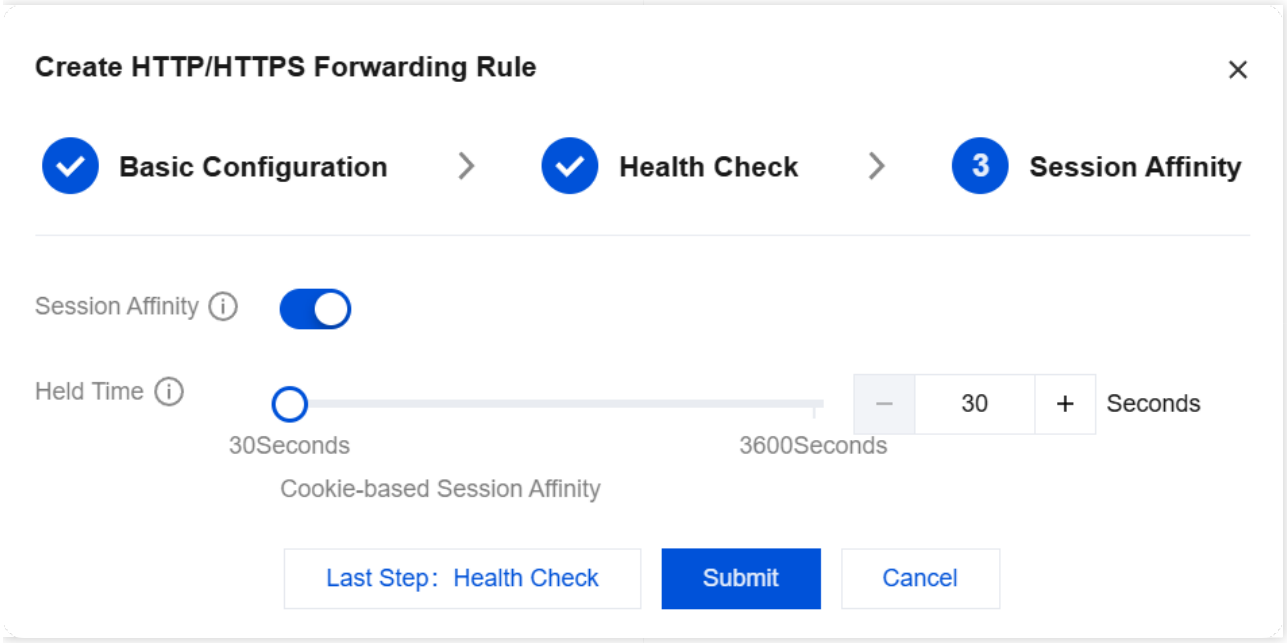
The specific configuration of the health checks is shown in the figure below.



4. Session persistence

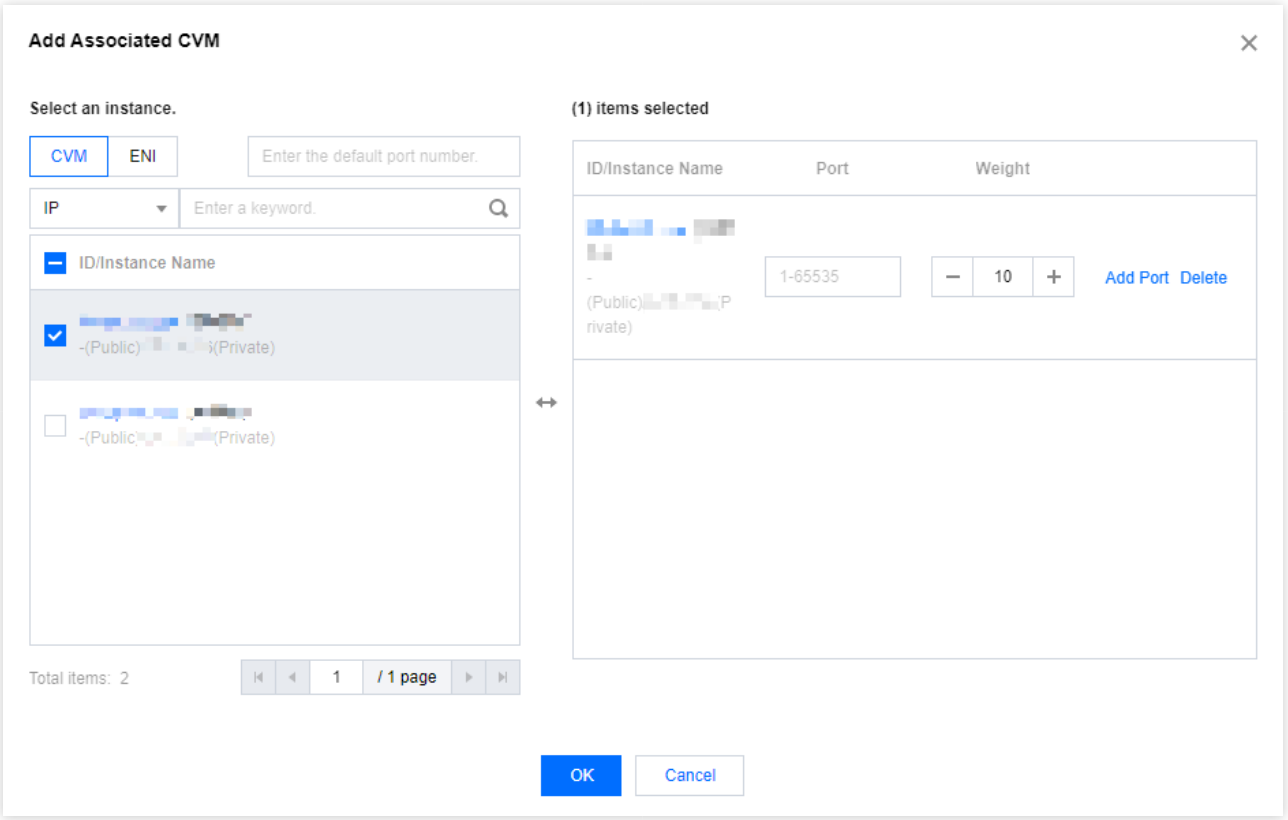
Session Persistence Configuration	Description	Example
Session Affinity Status	Enable or disable session persistence. - After session persistence is enabled, the CLB listener will distribute access requests from the same client to the same backend server. - The HTTPS protocol is based on Cookie session persistence, and the CLB embeds Cookies into the client. - Weighted round-robin scheduling supports session persistence, while weighted least connections and IP Hash scheduling do not support session persistence.	Enable
Session Held Time	Session hold time. - When the hold time is exceeded and there is no new request in the connection, the session will be automatically disconnected. - Configurable range 30-3,600 seconds.	30s

The specific configuration of session persistence is shown in the following figure.



Step 3: Bind the backend CVM

1. On the Listener Management page, click the created listener, click the triangle on the left side to expand the domain name and URL path, select a specific URL path, and view the backend services bound to the path on the right side of the listener.
2. Click Bind, select the backend server to be bound in the pop-up, and configure the service port and weight.
    - Add port feature: In the Selected CVM box on the right, click Add Port to add multiple ports of the same CVM.
    - Default port feature: Fill in Default Port first, then select the CVM. The port of each CVM is the default port.



After completing Step 1 to Step 3, the HTTPS listener rules are configured.

Step 4: Modify/delete listeners (optional)

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

Round-Robin Method

The round-robin scheduling is the algorithm for allocating traffic to the backend server by CLB. Different results can achieved according to different round-robin scheduling methods and weight settings of the backend servers.

Weighted round robin scheduling algorithm

The weighted round-robin scheduling algorithm requests scheduling of different servers in turn in a round-robin manner. The weighted round-robin scheduling algorithm can solve the problem of different performance among servers. It uses corresponding weights to represent the processing performance of the server and allocates requests to each server according to the weight and round-robin scheduling method. The weighted round-robin scheduling algorithm is scheduled based on the number of new connections. Servers with higher weights receive connections first. The higher the weight, the higher the number of times (probability) for its round-robin scheduling. Servers with the same weight handle the same number of connections.

- Advantages: It is simple and practical. It does not need to record the status of all current connections. It is a stateless scheduling.
- Disadvantages: Relatively simple, it is easy to cause load imbalance among servers when the request service time varies greatly or the time consumed by each request is inconsistent.
- Applicable Cases: The best load situation is when each request takes roughly the same amount of time to reach the backend. Commonly used for non-persistent connection services, such as HTTP.
- User Recommendation: If you know that the time each request takes to complete the task is roughly the same and the types of requests processed by the backend servers are the same or similar, we recommend that you choose the weighted round-robin scheduling method. When the request time difference is small, it is also recommended that you use the weighted round-robin scheduling method, because this implementation method consumes little energy, does not require traversal, and is more efficient.

Weighted Least Connection Scheduling Algorithm

In actual situations, the time that a client's request service stays on the server can vary greatly. As working hours extend, using simple round-robin scheduling or random balancing algorithms, the number of connection processes on each server may vary greatly, resulting in no true load balancing.
Least connection scheduling is a dynamic scheduling algorithm. In contrast to the round-robin scheduling algorithm, it estimates the server load by the number of active connections currently on the server. The

scheduler needs to record the number of connections established by each server. When a request is scheduled to a server, its number of connections increases by one; when the connection is terminated or times out, its number of connections decreases by one.
The weighted least connection scheduling is based on the least connection scheduling algorithm. It assigns different weights to each server according to the different processing capabilities of the servers so that they can accept service requests with corresponding weights. It is an improvement on the least connection scheduling algorithm.

> Note:
>
> Assume that the weight of each backend server is wi, the current number of connections is ci, and ci/wi is calculated in sequence. The backend server instance with the smallest value is used as the next allocated instance. If there are backend server instances with the same ci/wi, weighted round-robin scheduling is used.

- Advantages:This algorithm is suitable for long-time processing request services, such as FTP and other applications.
- Disadvantages: Due to API limitations, the minimum number of connections and session persistence features cannot be enabled at the same time.
- Applicable Cases: The case where the backend time taken by each request varies greatly. Commonly used in persistent connection services.
- User Recommendation: If the user needs to process different requests and the time taken by the requests to complete the backend differs greatly, such as a difference of orders of magnitude between 3ms and 3s, it is recommended to use the weighted least connection scheduling algorithm to achieve load balancing.

Source address hash scheduling algorithm

The source address hash scheduling algorithm (ip_hash) uses the hash key (Hash Key) to find the corresponding server from the statically allocated hash table according to the source IP address of the request. If the server is available and not overloaded, the request is sent to the server, otherwise it returns nothing.

- Advantages: A client's request can be mapped to the same backend server through a hash table. In cases where session persistence is not supported, ip_hash can be used to implement simple session persistence.
- User Recommendation: Perform a hash operation on the source address of the request and combine it with the weight of the backend server to dispatch the request to a matching server, so that requests from the same client IP are always dispatched to a specific server. This method is suitable for TCP protocols without cookie feature.

Balancing algorithm selection and weight configuration

In order to enable users to realize stable business handling of backend server clusters in different scenarios, the following is a case example of load balancing selection and weight configuration for your reference.

- Scenario 1:
  i. Assume that there are three backend servers with the same configuration (CPU/memory). Since their performance is consistent, the weights of all backend servers can be set to 10.
  ii. Now each backend server has established 100 TCP connections with the client, and 1 new backend server has been added.
  iii. In this scenario, it is recommended to use the least connection balancing method, which can quickly increase the load on the fourth backend server and reduce the pressure on the other three backend servers.
- Scenario 2:
  1.1. Suppose that you are new to cloud services and your website has been running for a short time with low traffic. It is recommended to purchase backend servers with the same configuration, where all the backend servers are identical access layer servers.
  1.2. In this scenario, the weights of all backend servers can be set to the default value of 10, and the weighted round-robin load balancing method can be used for traffic distribution.
- Scenario 3:
  1.1. Suppose that you have five servers that are used to host a simple static website, and the computing power of the five servers is in the ratio of 9:3:3:3:1 (based on CPU and memory).
  1.2. In this scenario, you can set the weight of the backend servers in the ratio of 90, 30, 30, 30, and 10. Since static website access mostly involves non-persistent connection requests, the weighted round-robin load balancing method can be used to allow the CLB instance to allocate requests according to their performance ratio.
- Scenario 4:
  1.1. Suppose that you have 10 backend servers that are used to handle massive web access requests, and you do not want to purchase additional backend servers to increase expenditure. However, a specific backend server often experiences high load, causing it to restart frequently.
  1.2. In this scenario, it is recommended to set weights for the backend servers based on their performance. Set a lower weight for a server with a high load. Additionally, the least-connection load balancing method can be used to allocate requests to the backend servers with fewer active connections, thus resolving the issue of high load on specific backend servers.
- Scenario 5:
  1.1. Suppose that you have three backend servers that are used to handle several persistent connection requests, and the computing power of these servers is in the ratio of 3:1:1 (based on CPU and memory).
  1.2. In this case, the best-performing server handles more requests. You do not want to overload this

server and want new requests to be distributed to the idle servers.

1.3. In this scenario, the least-connection load balancing method can be used, and the weights of busier servers can be appropriately reduced. This allows the CLB instance to allocate requests to the backend servers with fewer active connections, achieving load balancing.

- Scenario 6:

  1.1. Suppose that you want subsequent client requests to be allocated to the same server. In this case, using the weighted round-robin or weighted least-connection method cannot guarantee that requests from the same client will always be allocated to the same server.

  1.2. To meet the requirements of specific application servers and ensure that client sessions are "sticky" or "persistent", you can use the ip_hash load balancing method for traffic distribution in this scenario. This ensures that requests from the same client are always directed to the same backend server (except when the number of servers changes or the server becomes unavailable).

Health Examination

- Converge CloudCLB instance can periodically send Pings, connection attempts, or requests to backend servers to test the running status of the backend servers. These tests are called health checks.
- When a backend server instance is determined to be unhealthy, the CLB instance will not forward requests to that instance. However, health checks will be performed on all backend servers (regardless of whether they are assessed to be healthy or unhealthy). When the unhealthy instance returns to normal, the CLB instance will resume forwarding new requests to it.
- The auto scaling group periodically uses a similar method to determine the operating status of each instance in the group. For more information, see Auto scaling.

Layer-4 forwarding health check configuration

The health check mechanism of layer-4 forwarding: the CLB initiates an access request to the server port specified in the configuration. If the port access is normal, the backend server is considered to be running normally, otherwise it is considered to be running exceptionally.
For TCP business, use SYN packets for detection; for UDP business, use Ping for inspection.

Health check configuration	Note	Default Value
Response timeout	The maximum timeout for health check responses. If the backend CVM does not respond correctly within the timeout period, it is considered a health check exception. Configurable range: 2 - 60 seconds.	2 seconds
Detection interval	The interval for the CLB health check. Configurable range: 5 - 300 seconds.	5 seconds
Unhealthy Threshold	If the health check results received for n consecutive times (n is the filled value) fail, it is identified as unhealthy and the console displays exceptional. Configurable range: 2 - 10 times.	3 times
Health threshold	If the health check results received for n consecutive times (n is the filled value) are successful, it is identified as healthy and the console displays healthy. Configurable range: 2 - 10 times.	3 times

Layer-7 forwarding health check configuration

The health check mechanism of layer-7 forwarding is that the CLB sends HTTP requests to the backend server to detect the backend service. The CLB checks whether the service is normal based on the HTTP return value selected by the user. Assume that in a certain scenario, the HTTP return value is http_1xx , http_2xx , http_3xx , http_4xx and http_5xx . Users can edit http_1xx and http_2xx according to business needs as normal service status and set the return values of http_3xx is set to http_5xx as the exception status.

Health check configuration	Note	Default Value
Check the domain name	Request domain - Supported character sets include: az``0-9``.``- , length limit is 1 - 120. - Regular expressions are not currently supported. - When the user fills in the domain name as a wildcard domain name, it is necessary to specify a fixed domain name (non-regular) as the health check domain name.	Forwarding domain names
Check the path	Request path - Must be preceded by / . - Supported character sets include: az``0-9``.``- , length limit is 1 - 120. - Regular expressions are not currently supported. - It is recommended to specify a fixed URL path (static page) for health check.	/
Detection interval	- The interval for the CLB health check. - Configurable range: 5 - 300 seconds.	5 seconds
Unhealthy Threshold	- If the health check results received for n consecutive times (n is the filled value) fail, it is identified as unhealthy and the console displays exceptional. - Configurable range: 2 - 10 times.	3 times
Health threshold	- If the health check results received for n consecutive times (n is the filled value) are successful, it is identified as healthy and the console displays healthy. - Configurable range: 2 - 10 times.	3 times
HTTP Request Method	HTTP request method for health check, optional: GET or HEAD. - If the HEAD method is used, the server only returns the HTTP header information, which can reduce backend overhead and improve request efficiency. The corresponding	GET

Health check configuration	Note	Default Value
	backend service must support HEAD. - If the GET method is used, the backend service only needs to support GET.	
HTTP status code detection	When the status code is the selected status code, the backend server is considered alive, that is, the health check is normal. Optional: http_1xx, http_2xx, http_3xx, http_4xx, http_5xx.	http_1xx, http_2xx, http_3xx, http_4xx

Health check status

Based on the health check detection situation, the health check status of the backend service is as follows:

Status	Note	Whether to forward traffic
Health	Backend service is normal.	CLB forwards traffic to the healthy backend services.
Abnormal	Backend service exceptions	- CLB does not forward traffic to exceptional backend services. - Under a layer-4 listener or layer-7 URL rule, if CLB detects that all backend services are unhealthy, the all-dead and all-alive logic will be activated, that is, the request will be forwarded to all backend services.

Troubleshooting ideas for health check exceptions

Layer-4 troubleshooting

Under the TCP protocol, the CLB uses SYN packets for detection; under the UDP protocol, the CLB uses the ping command for detection.
Check the health status of the backend server port on the page. If it is unhealthy, the troubleshooting ideas are as follows:

- Determine whether the backend server has a security group configured that affects the service. For information on how to control access to backend servers to ensure normal service operation, see Backend CVM Security Group Configuration Instructions.
- Using netstat command to determine whether there is a process listening on the port of the backend server. If no process is found, restart the service.

Layer-7 troubleshooting

For layer-7 (HTTP/HTTPS protocol) services, when a health check exception occurs in a listener, you can troubleshoot it through the following aspects:

1. Since the layer-7 health check service of the CLB communicates with the backend CVM through the private network, you need to log in to the server to check whether the application server port is normally listening on the private network address. If it is not listening on that address, let the application server port listen on the private network to ensure normal communication between CLB system and the backend CVM.
   Assume that the front-end port of the CLB is 80, the back-end port of the CVM is also 80, and the private IP address of the CVM is: 1.1.1.10 .

   For Windows servers, use the following command:

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

   For Linux servers, use the following command:

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

   If you can see 1.1.1.10:80 Listening or 0.0.0.0:80 If the monitoring is successful, this configuration is normal.

2. Make sure that the backend server has opened the backend port you configured in the CLB listener. If it is a layer-4 CLB, as long as the backend port telnet responds, you can use telnet 1.1.1.10 80 to test. If it is a layer-7 CLB, the HTTP status code needs to be 200 or other normal status codes. The inspection method is as follows:

- In Windows, you can directly enter the private IP address in the browser in CVM to test whether it is normal. In this example, use http://1.1.1.10 is used. .
- Linux system can use curl -I command to check if the status is HTTP/1.1 200 OK In this example, use curl -I 1.1.1.10 Order.

3. Check whether there is a firewall or other security protection software inside the backend CVM. This type of software can easily block the local IP address of the CLB system, causing that system to be unable to communicate with the backend server.
   Check whether the server's private network firewall allows access to port 80. You can temporarily close the firewall for testing.

- For Windows systems, enter firewall.cpl to close.
- For Linux systems, enter /etc/init.d/iptables stop to close.

4. Check whether the CLB health check parameters settings are correct. It is recommended to set them according to the default values of the health check parameters provided in this document.
5. The detection file specified by the health check is recommended to be a simple HTML page, which is only used to check the return results. It is not recommended to use dynamic scripting languages such as PHP.
6. Check whether the backend has a high load, causing the CVM to respond slowly to external services.
7. Check the HTTP request method. If the HEAD method is used, the backend service must support HEAD. If the method is GET, the backend service must support GET.

Explanation of frequent health check detection

The frequency of health check probe packets is too high. The console is set to receive probe packets once every 5 seconds. In fact, the backend RS found that it received one or more health check requests within 1 second. The reason is:

- Currently, the problem of too high a health check frequency is mainly related to the implementation mechanism of the health detection of the CLB backend. Assume that 1 million client requests are distributed on 4 CLB backend physical machines and then transferred to the CVM. The health check detection is performed on the backend physical machines of the CLB separately. Therefore, the CLB instance sets a detection request every 5 seconds. In fact, each physical machine in the CLB backend sends a detection every 5 seconds. Therefore, multiple detection requests are received on the backend CVM. (Assuming that the cluster where the CLB instance is located has 8 physical machines, each machine sends a request once every 5 seconds, and the backend host may receive 8 times detection in 5 seconds.)
- The strengths of this implementation scheme are high efficiency, accurate detection, and avoidance of false elimination. For example, among the eight physical machines in the CLB instance cluster, if one fails, only one machine no longer forwards traffic, while the traffic of the other seven machines is normal.
  Therefore, if the detection frequency of your backend CVM is too high, you can solve it by setting a longer detection interval (for example, setting it to detect once every 15 seconds).

Session persistence

The session persistence allows requests from the same IP to be forwarded to the same backend server. By default, the CLB will route each request to a different backend server instance. However, you can use the session persistence feature to route requests for a particular user to the same backend server instance, which allows certain applications that require session persistence (such as shopping carts) to work properly.

Layer-4 session persistence

The layer-4 protocol (TCP/UDP) supports session persistence based on source IP. The session persistence time can be set to any integer value between 30 and 3,600 seconds. If there is no new request in the session after the time threshold is exceeded, the connection is disconnected. Session persistence is related to the balancing method:

- When the balancing method is round-robin scheduling, requests are distributed according to the weights of the backend servers, and session persistence based on the source IP is supported.
- When the balancing method is weighted least connections, comprehensive scheduling is performed based on server load and weight, and session persistence is not supported.

Layer-7 session persistence

The layer-7 protocol (HTTP/HTTPS) supports session persistence based on cookie insertion (CLB inserts cookies into a client). The session persistence time setting supports 30 - 3,600 seconds. Session persistence is related to the balancing method:

- When the balancing method is round-robin scheduling, requests are distributed according to the weights of the backend servers, and session persistence based on cookie insertion is supported.
- When the balancing method is weighted least connections, comprehensive scheduling is performed based on server load and weight, and session persistence is not supported.
- When the balancing method is "IP Hash", session persistence based on source IP is supported, but session persistence based on cookie insertion is not supported.

Connection timeout

The current HTTP connection timeout (keepalive_timeout) defaults to 75 seconds. If there is no data transmission in the session after the time threshold is exceeded, the connection will be disconnected.
The current TCP connection timeout period cannot be adjusted yet and the default is 900 seconds. If there

is no data transmission in the session after the time threshold is exceeded, the connection will be disconnected.

Configuring session persistence

1. Log in to the CLB console and click the ID of the CLB instance for which session persistence needs to be configured, to enter the CLB details page.
2. Select the Listener Management tab.
3. Click Modify next to the CLB listener for which session persistence needs to be configured.
4. Select whether to enable the session persistence feature, click the button to enable it, enter the persistence time, and click OK.

The relationship between persistent connection and session persistence

Scenario 1: HTTP layer-7 business

Assume that the client access is HTTP/1.1 protocol, and set Connection:keep-alive in the header information. Through CLB, I access the backend CVM again. If I do not enable session persistence at this time, can I access the same CVM the next time I access it?

Answer: cannot.

First, HTTP keep-alive means that the TCP connection will remain open after it is sent, so the browser can continue to send requests over the same connection. Keep-alives save the time required to establish a new connection for each request and also save bandwidth. The default timeout period of the CLB cluster is 75 seconds (if there is no new request refresh within 75 seconds, the TCP connection is disconnected by default).

HTTP keep-alive is established between the client and CLB. If the cookie session persistence is not enabled at this time, the next time you access, CLB will randomly select a CVM on the backend based on the round-robin scheduling policy, and the previous persistent connection will be wasted.

Therefore, it is recommended to enable session persistence.

When the cookie session duration is set to 1,000 seconds, the client initiates a request again. Since more than 75 seconds have passed since the last request, the TCP connection needs to be re-established. The application layer determines the cookie and finds the same CVM. The CVM accessed by the client is still the one accessed last time.

Scenario 2: TCP layer-4 business

Assume that the client initiates the access, the transport layer protocol is TCP, and persistent connection is enabled. However, session persistence based on source IP is not enabled. Can the same Client access the same machine next access,?

Answer: Uncertain.

First, according to the implementation mechanism of the layer-4, when TCP enables a persistent connection, if the persistent connection has not been disconnected, and the two previous and subsequent accesses are through the same connection, the same machine can be accessed. If the first connection is released due to other reasons (network restart, connection timeout) during the second access, the second access may be scheduled to other backend CVMs, and the default global timeout of the persistent connection is 900 seconds, that is, if there is no new request, it will be released.

Certificate configuration

Common Process for Certificate Application

- Generate a private key locally: openssl genrsa -out privateKey.pem 2048, where privateKey.pem is your private key file, and keep it properly.
- Generate a certificate request file: openssl req -new -key privateKey.pem -out server.csr, where server.csr is your certificate request file, which can be used to apply for a certificate.
- Get the content in the request file and go to the CA or other institution sites to apply for a certificate.

Certificate Format Instructions

Certificate Format Requirements

- The certificate that the user needs to apply for is: a certificate in PEM format under Linux environment. CLB does not support certificates in other formats. For certificates in other formats, see the Instructions on Converting Certificate Format to PEM below.
- If the certificate is issued by a root CA agency, the certificate you get is the only one and no additional certificates are required. The configured site can be considered trustworthy by browsers and other access devices.
- If the certificate is issued by an intermediate CA agency, the certificate file you get contains multiple certificates, and you need to manually merge the server certificate and the intermediate certificate and upload them together.
- If your certificate has a certificate chain, convert the certificate chain content into PEM format and upload it together with the certificate content.
- The splicing rule is: put the server certificate in the first copy, and the intermediate certificate in the second copy, with no blank lines in between.

Certificate format and certificate chain format examples

The following are examples of the certificate format and certificate chain format. Confirm that the format is correct before uploading:

1. Certificate issued by the root CA agency: The certificate format is PEM format in the Linux environment. For example:



The certificate rules are:

- [—-BEGIN CERTIFICATE—-, —-END CERTIFICATE—-] Beginning and ending; upload these contents together.
- Each line has 64 characters, and the last line has no more than 64 characters.

2. For certificates issued by an intermediate CA, the certificate chain typically looks like this:

———-BEGIN CERTIFICATE———-

———-END CERTIFICATE———-

———-BEGIN CERTIFICATE———-

———-END CERTIFICATE———-

———-BEGIN CERTIFICATE———-

———-END CERTIFICATE———-

The certificate chain rules are:

- There cannot be blank lines between certificates.
- Each certificate complies with the above 2.1 Certificate format requirements.

RSA Private Key Format Requirements

For example:



RSA private keys can include all private keys (RSA and DSA), public keys (RSA and DSA), and (x509) certificates. It stores DER format data encoded in Base64, surrounded by an ASCII header, thus suitable for text-mode transmission between systems.

RSA private key rules:

- [—-BEGIN RSA PRIVATE KEY—-, —-END RSA PRIVATE KEY—-] Beginning and ending; upload these contents together.
- Each line has 64 characters. The last line can be less than 64 characters.
  If you do not generate a usable private key in the format of [———-BEGIN PRIVATE KEY———-, ———-END PRIVATE KEY———-] according to the above scheme, you can convert it into a usable private key as follows:

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

Then upload the contents of new_server_key.pem along with the certificate.

Certificate Conversion to PEM Format Instructions

Currently, the CLB service only supports certificates in PEM format. Certificates in other formats need to be converted to PEM format before they can be uploaded to the CLB service. It is recommended to use the openssl tool for conversion. Below are several popular methods for converting popular certificate formats

to PEM format.

Converting certificate format from DER to PEM

The DER format is commonly found in the Java platform.

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

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

Convert P7B Format Certificates to PEM Format

The P7B format is commonly found in Windows Server and tomcat.

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

Get the content [——-BEGIN CERTIFICATE——-, ——-END CERTIFICATE——-] in outcertificat.cer and upload it as the certificate.

Private key conversion: Private keys can usually be exported in the IIS server.

Convert PFX Format Certificates to PEM Format

The PFX format is commonly found in Windows Server.

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

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

Convert CER/CRT Format Certificates to PEM Format

For certificates in CER/CRT format, you can convert them by directly changing the certificate file extension. For example, you can simply rename the "servertest.crt" certificate file to "servertest.pem".

Configuring Layer-7 Redirection

CLB supports layer-7 redirection. This feature allows users to configure redirection on the layer-7 HTTP/HTTPS listener.

> Note:
>
> Session persistence: If the client accesses example.com/bbs/test/123.html and session persistence has been enabled in 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.

Redirection Overview

1. Automatic redirection
   - Overview
     For an existing HTTPS:443 listener, an HTTP listener (port 80 by default) will be created by the system for forwarding. After the creation is successful, requests sent to HTTP:80 will be automatically redirected to HTTPS:443 .
   - Use case
     Forced HTTPS redirection, namely 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.
   - Solution strengths
     - Set-and-forget configuration: Forced HTTPS redirection can be implemented for a domain name with only one configuration operation.
     - 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
   - Introduction
     You can configure one-to-one 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 take your web workload offline temporarily (because of 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 page which displays a 404/503 error message, degrading the user experience and resulting in traffic waste.

Automatic Redirection

Converge Cloud CLB supports one-click forced redirection from HTTP to HTTPS.
Assume that a developer needs to configure the website https://www.example.com , so that its 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

An HTTPS:443 listener has been configured.

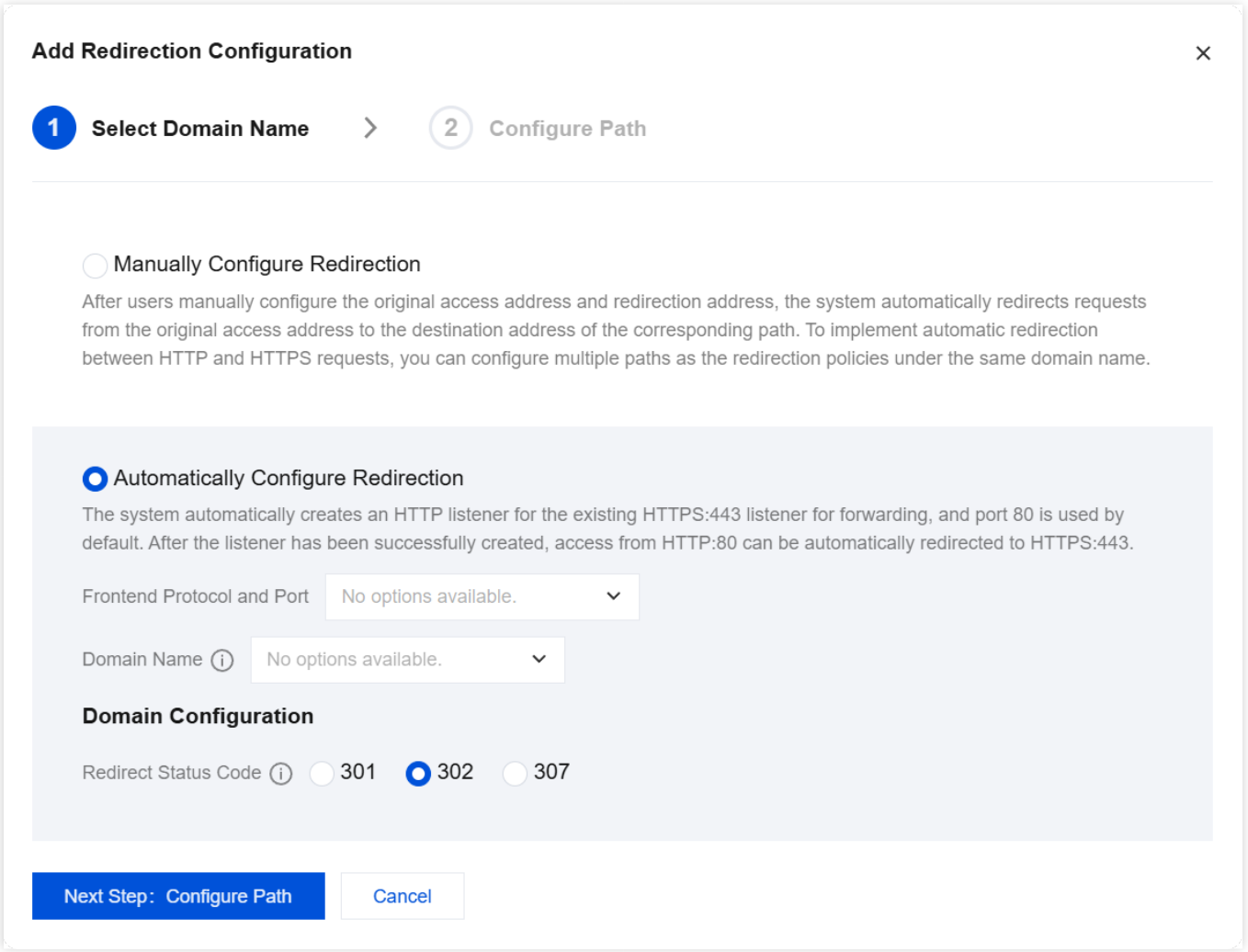
Operation Steps

1. Configure the CLB HTTPS listener in the CLB console and set up the web environment of https://example.com . For details, see Configuring an HTTPS Listener.

2. Configure the HTTPS listener.

3. In the Redirection Configuration tab of the CLB instance details, click Create Redirection Configuration.

4. Select Automatically Configure Redirection. Then, select the configured HTTPS listener and domain name, and a redirection status code in the Domain Name Configuration section. Click Next Step: Configure Path.



> Note:
>
> The status codes are 301 (Moved Permanently), 302 (Move Temporarily), and 307 (Temporary Redirect). For details, see HTTP/1.1 (RFC 7231).

5. Click Submit to complete the configuration.

6. After the redirection configuration is completed, you can see that 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

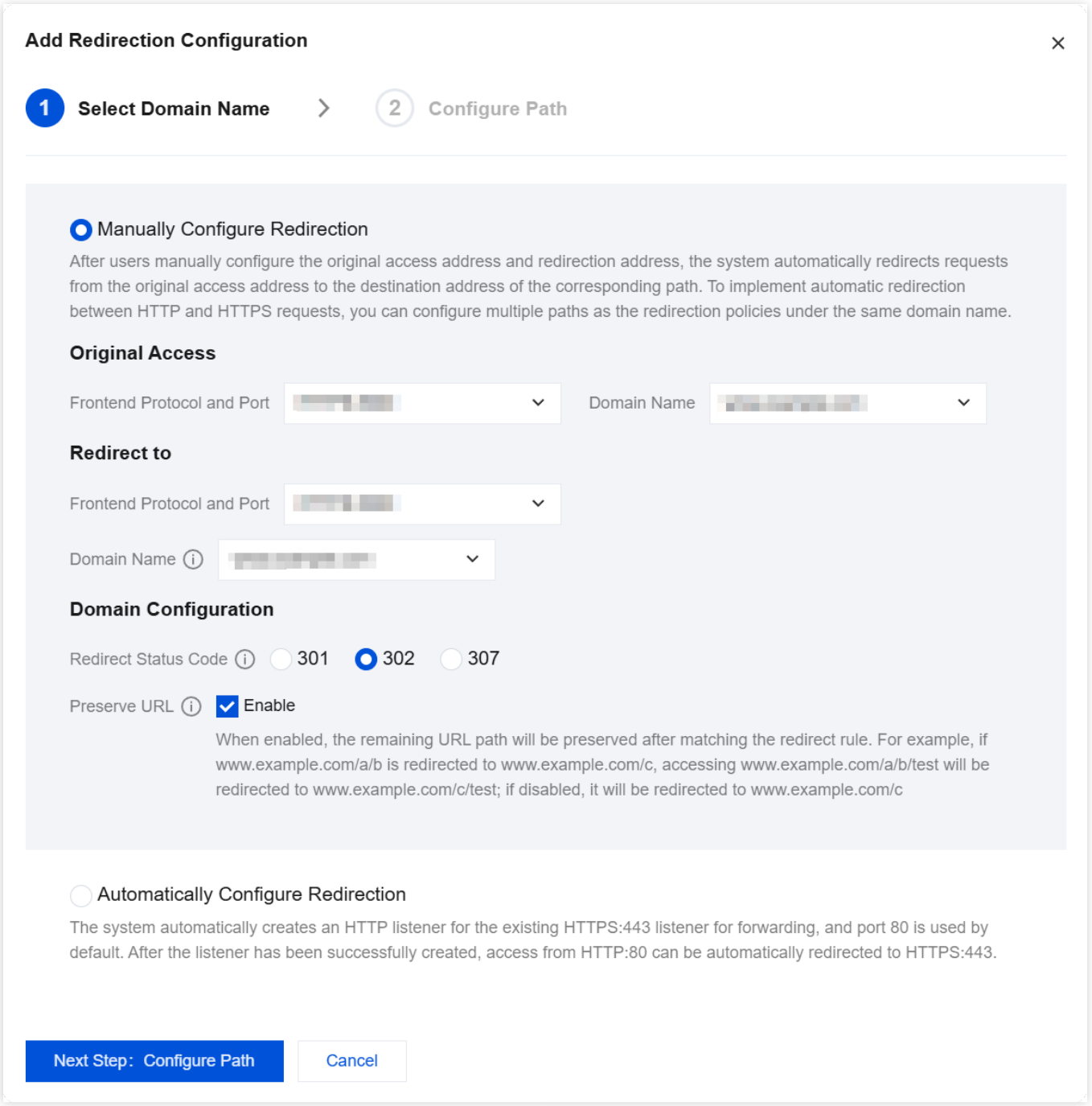
Converge Cloud CLB supports configuring one-to-one redirection.
For example, you used a forsale page for your promotional campaign. Now the campaign ends and you need to redirect the campaign page https://www.example.com/forsale to the new homepage https://www.new.com/index .

Prerequisites

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

Operation Steps

1. Configure the CLB HTTPS listener in the CLB console and set up the web environment of https://example.com . For details, see Configuring an HTTPS Listener.
2. Configure the HTTPS listener.
3. In the Redirection Configuration tab of the CLB instance details, click Create Redirection Configuration.
4. Select Manually Configure Redirection, and then select the originally accessed frontend protocol port HTTPS:443 and domain name https://www.example.com/forsale , as well as the frontend protocol port HTTPS:443 and domain name https://www.new.com/index after redirection. In Domain Configuration, select a redirection status code, and select or clear the URL reservation check box. Then click Next Step: Configure Path.



5. Select /forsale as the original access path and /index as the access path after redirection. Then, click Submit to complete the configuration.
6. After the redirection configuration is completed, you can see that in the HTTP:443 listener, https://www.example.com/forsale has been redirected to https://www.new.com/index .

Layer-7 personalized configuration

CLB supports personalized configuration feature, allowing you to set the configuration parameters of a single CLB instance, such as client_max_body_size, ssl_protocols, etc., to meet your personalized configuration needs.

> Note:
>
> - The number limit of personalized configurations is consistent with the tenant's public network CLB quota. For example, if the public network CLB quota is 200 by default, the number limit of personalized configurations is also 200.
> - The length of the personalized configuration is limited to 64 k.
> - Currently, only one personalized configuration can be bound to an instance.
> - Personalized configuration is only effective for the layer-7 HTTP/HTTPS listener of the CLB.

CLB personalized configuration parameters description

The current CLB personalized configuration supports the following fields:

Configuration fields	Default/ recommended values	Parameter Range	Description
ssl_protocols	TLSv1 TLSv1.1 TLSv1.2	TLSv1 TLSv1.1 TLSv1.2	The TLS protocol version to use.
ssl_ciphers	See below	See below	Cipher suite.
client_header_timeout	60s	[30-120]s	Get the timeout period in the Client request header. If the timeout period expires, return 408.
client_header_buffer_size	4k	[1-256]k	The default Buffer size for storing the Client request header.
client_body_timeout	60s	[30-120]s	The timeout for obtaining the client request body does not refer to the duration of obtaining the entire body, but refers to the timeout for a period of idle time without

Configuration fields	Default/ recommended values	Parameter Range	Description
			transmitting data. If the timeout occurs, 408 is returned.
client_max_body_size	60M	[1-10240]M	- The default configuration range is 1 M-256 M, you can configure it directly. - The maximum supported speed is 10,240 M, or 10 G. When the configuration range of client_max_body_size is greater than 256M, the value of proxy_request_buffering must be set to off.
keepalive_timeout	75s	[0-3600]s	Client-Server long connection persistence time: if set to 0, then long connection is disabled.
add_header	User-defined additions	-	Returns specific header fields to the client in the format of add_header xxx yyy.
more_set_headers	User-defined additions	-	Returns specific header fields to the client in the format of more_set_headers "A:B".
proxy_connect_timeout	4s	[4-120]s	Upstream backend connection timeout.
proxy_read_timeout	60s	[30-3600]s	Timeout for reading upstream backend response.
proxy_send_timeout	60s	[30-3600]s	The timeout for sending requests to the upstream backend.
keepalive_requests	100	[1-10000]	The maximum number of requests that can be sent on a long Client-Server connection.
proxy_buffer_size	4k	[1-32]k	For the size of the Server response header, by default, it is the size of the single buffer set in proxy_buffer. When using

Configuration fields	Default/ recommended values	Parameter Range	Description
			proxy_buffer_size, proxy_buffers must be set at the same time.
proxy_buffers	8?4k	[3-8]?[4-16]k	Number of buffers and buffer size.
proxy_request_buffering	on	on, off	- On means caching the client request body: CLB will cache the request and forward it to the backend CVM in blocks after receiving all requests. - Off means not caching the client request body: After receiving the request, CLB immediately forwards it to the backend CVM, which will cause certain performance pressure on the backend CVM.
proxy_set_header	X-Real-Port $remote_port	- X-Real-Port $remote_port - X-clb-stgw-vip $server_addr - Stgw-request-id $stgw_request_id - X-Forwarded-Port?$vport - X-Method $request_method - X-Uri?$uri	- X-Real-Port $remote_port indicates the client port. - X-clb-lbid $lbid represents the LBID of the CLB, which is the identifier of the CLB instance. - Stgw-request-id $stgw_request_id indicates the request ID (used internally by CLB). - X-Forwarded-Port indicates the port of the CLB listener. - X-Method indicates the client request method. - X-Uri represents the client request path URI.
send_timeout	60s	[1-3600]s	The timeout for the server to transmit data to the client is the interval between two consecutive data transmissions, not the entire request transmission time.

Configuration fields	Default/ recommended values	Parameter Range	Description
ssl_verify_depth	1	[1, 10]	Sets the verification depth in the client certificate chain.
proxy_redirect	http:// https://	http://?https://	When the response returned by the upstream server is a redirect or refresh request (For example, the HTTP response code is 301 or 302), proxy_redirect resets the http in the Location or Refresh field of the HTTP header to https to achieve a secure jump.
http2_max_field_size	4k	[1-256]k	Limit the maximum size of the HPACK compressed request header field (Size).
error_page	-	error_page?code?[?=?[response]]?uri	When a specific error code (Code) occurs, a predefined URI can be displayed, and the default status code (Response) is 302. The URI must be a path beginning with / .
proxy_ignore_client_abort	off	on, off	When the client actively disconnects from the CLB without waiting for a response result, configure whether the connection between the CLB and the backend server is disconnected.

ssl_ciphers configuration instructions

When configuring the ssl_ciphers cipher suite, the format must be consistent with the format used by OpenSSL. The algorithm list is one or more <cipher strings> Multiple algorithms are separated by ":". ALL means all algorithms, "!" means that the algorithm is not enabled, and "+" means that the algorithm is placed last.
The default encryption algorithm that is disabled 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-AES128-SHA256:ECDHE-RSA-AES128-SHA:ECDHE-ECDSA-AES128-SHA:ECDHE-RSA-AES256-SHA384:ECDHE-ECDSA-AES256-SHA384:ECDHE-RSA-AES256-SHA:ECDHE-ECDSA-AES256-SHA:AES128-GCM-SHA256:AES256-GCM-SHA384:AES128:AES256:AES:HIGH:!aNULL:!eNULL:!EXPORT:!DES:!RC4:!MD5:!PSK:!DHE:3DES;

Parameter Range:

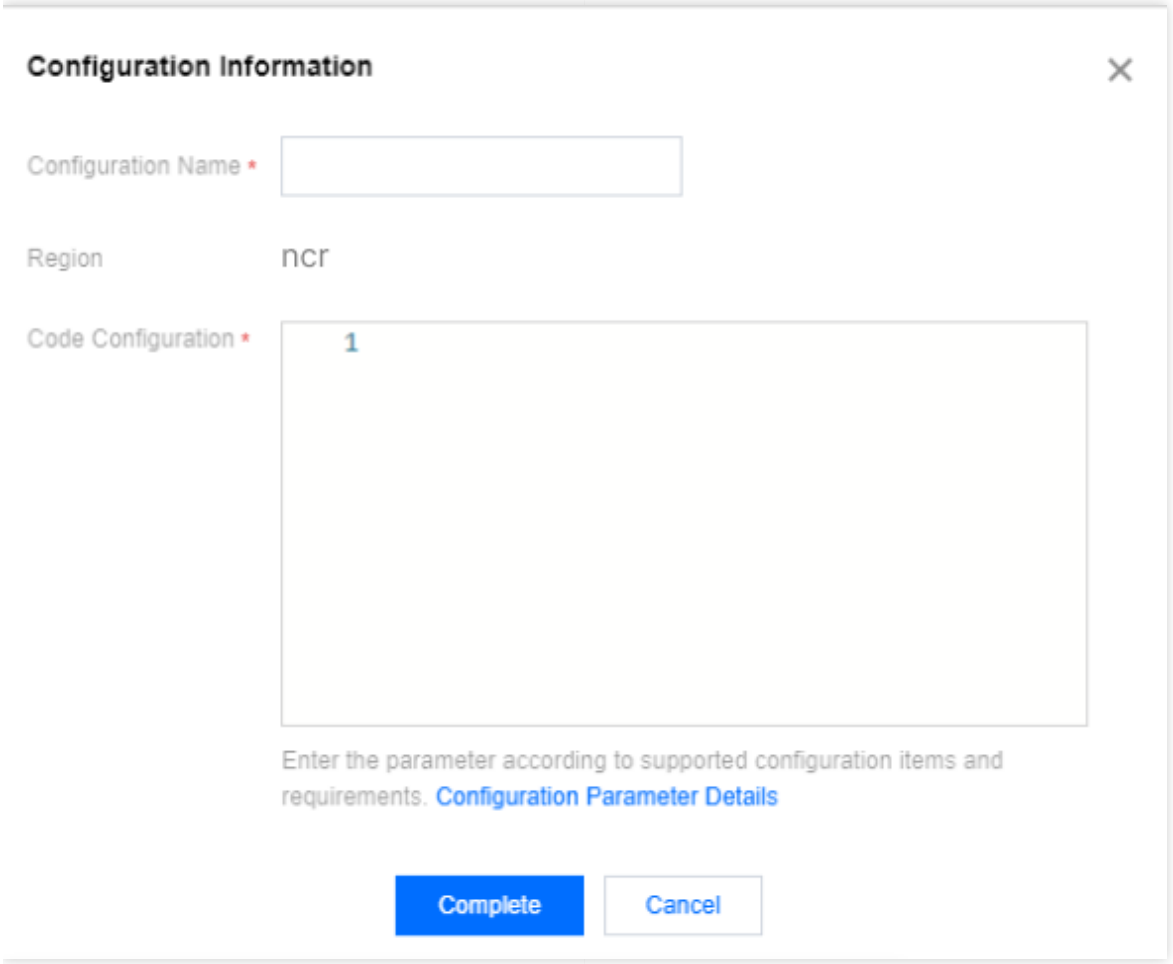
ECDH-ECDSA-AES128-SHA256:ECDH-RSA-AES256-SHA:ECDH-ECDSA-AES256-SHA:SRP-DSS-AES-256-CBC-SHA:SRP-AES-128-CBC-SHA:ECDH-RSA-AES128-SHA256:DH-RSA-AES128-SHA256:DH-RSA-CAMELLIA128-SHA:DH-DSS-AES256-GCM-SHA384:DH-RSA-AES256-SHA256:AES256-SHA256:SEED-SHA:CAMELLIA256-SHA:ECDH-RSA-AES256-SHA384:ECDH-ECDSA-AES128-GCM-SHA256:DH-RSA-AES128-SHA:DH-RSA-AES128-GCM-SHA256:DH-DSS-AES128-SHA:ECDH-RSA-AES128-SHA:DH-DSS-CAMELLIA256-SHA:SRP-AES-256-CBC-SHA:DH-DSS-AES128-SHA256:SRP-RSA-AES-256-CBC-SHA:ECDH-ECDSA-AES256-GCM-SHA384:ECDH-RSA-AES256-GCM-SHA384:DH-DSS-AES256-SHA256:ECDH-ECDSA-AES256-SHA384: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:AES128:AES256:AES:!aNULL:!eNULL:!EXPORT:!DES:!RC4:!MD5:!PSK:!DHE

CLB personalization configuration example

1. Log in to the CLB console and click Personalized Configuration in the left sidebar.
2. On the Personalization Configuration page, click Create.
3. In the pop-up Configuration Information dialog box, fill in the configuration name and code configuration item. The code configuration item ends with ;. After completing the configuration, click Finish.

> Note:
>
> Default configuration code example:
>
> 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;
```

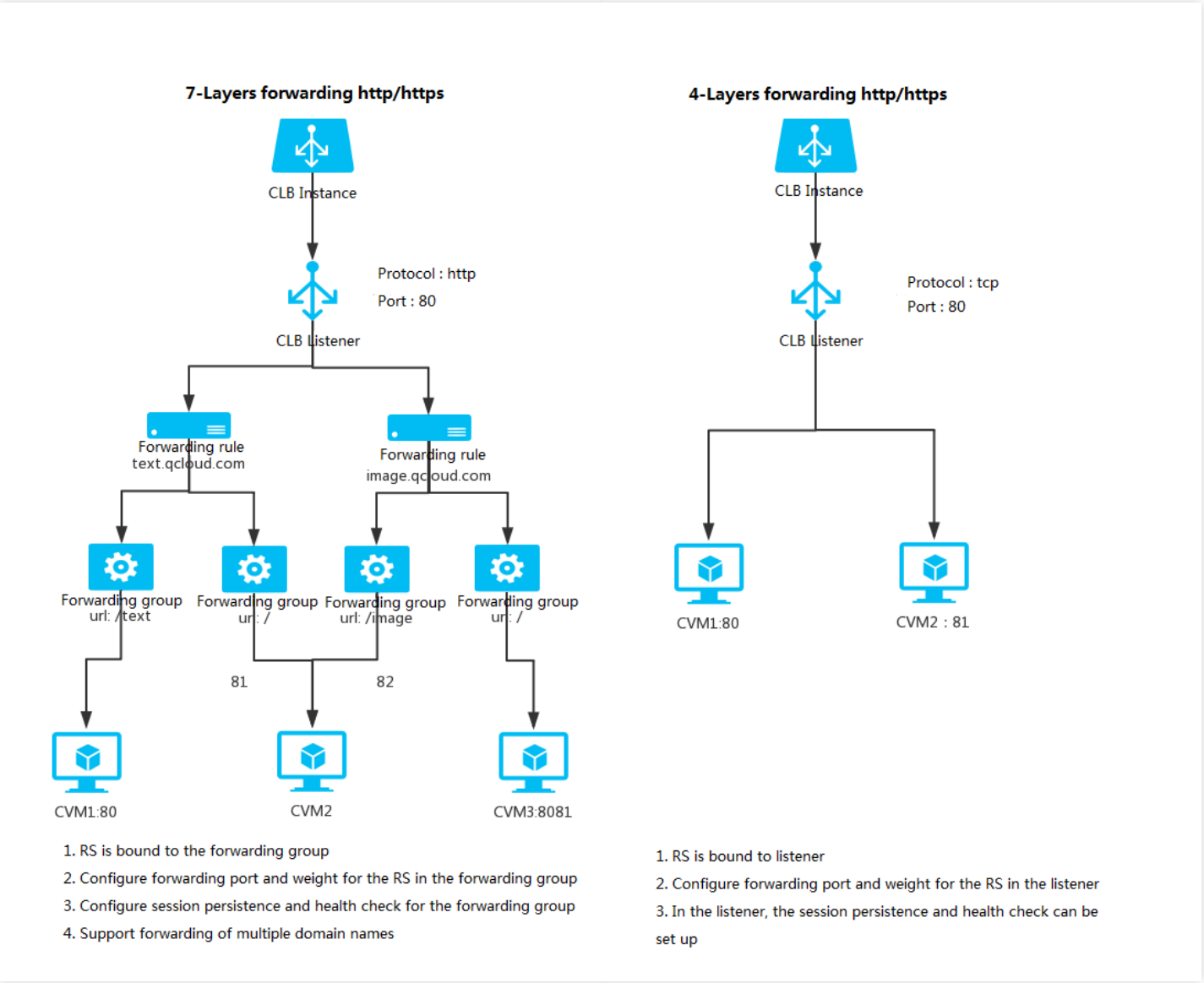


4. Return to the Personalization Configuration page and click Bind to Instance in the Operation column on the right.
5. In the Bind to Instance dialog box that pops up, select the CLB instance to be bound and click Submit.
6. After binding the instance, click the personalized configuration ID just configured on the Personalized Configuration page to enter the details page, and click the Bind to Instance tab to view the CLB instance just bound.

Layer-7 forwarding domain name and URL rule description

Business Flow Chart

The layer-7 and layer-4 business flow chart of the CLB are as follows:



When using the layer-7 forwarding HTTP/HTTPS protocol of the CLB , create a new forwarding rule in the listener of a CLB instance, and the user can add a corresponding domain name.

- When the user has only established one forwarding rule, accessing the VIP+ URL can correspond to the corresponding forwarding rule and access the service normally.

- When the user has established multiple forwarding rules, accessing VIP + URL at this time cannot ensure access to a specific domain name + URL. The user needs to directly access the domain name + URL to ensure that the specific forwarding rules are effective. That is, when the user configures multiple forwarding rules, the same VIP corresponds to multiple domain names. In this case, it is not recommended to access the service through VIP + URL, but to access the service through specific domain name + URL.

Forwarding rule configuration instructions

Domain name configuration rules

When configuring the domain name in the forwarding rule of the CLB layer-7 listener, you can use regular expressions but the length is limited to 1 - 120.

- The character sets supported by irregular domain names are as follows:
  az``0-9``.``-
- Wildcard domain name, currently only supports *.example.com or www.example.* . And only one occurrence of < 5 > * </5 > is supported in a single domain name.
- Regular expressions are supported. The character sets that are not supported in regular expressions are as follows:
  "``{``}``;``\ ``` ~`````Space .
- Examples of regular domain names supported by the CLB are as follows:
  ~^www\d+\.example\.com$

Health check configuration rules

- When the domain name filled in by the user is a wildcard domain name, you need to specify a fixed domain name (non-regular) as the health check domain name. The character sets supported by the health check domain name configuration are as follows:
  a-z``0-9``.``-
- When configuring the health check path for the load balancing layer-7 listener, the default is / , must be preceded by / The length limit is 1 - 120. Regular expressions are not supported yet. It is recommended to specify a fixed URL path (static page) for health check. The character sets supported by the health check path configuration are as follows:
  a-z``A-Z``0-9``.``-``/``=``?

Domain Name Matching Rules

1. Do not configure a domain name in the forwarding rule, fill in the IP instead, and configure multiple URLs in the forwarding group. The service is accessed through VIP + URL.
2. Configure the full domain name in the forwarding rule and configure multiple URLs in the forwarding

group. The service is accessed through the domain name + URL.

3. If a wildcard domain name is configured in the forwarding rule and multiple URLs are set in the forwarding group, access is granted by matching the requested domain name and the URL. This configuration can be used when users want different domain names to point to the same URL. Taking example.test.com as an example, the format is as shown below:

- example.test.com exactly matches the domain name example.test.com .
- *.test.com matches all domain names ending with test.com .
- example.test.* matches all domain names starting with example.test .

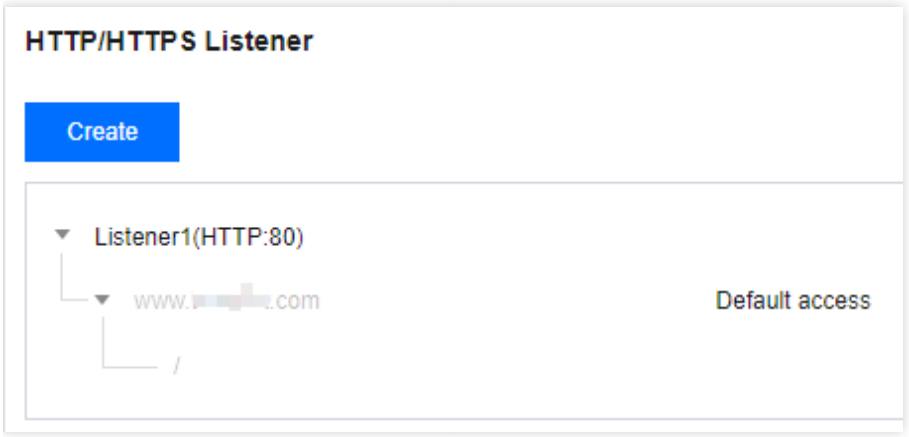
4. Configure the domain name in the forwarding rule and configure the fuzzy matching URL in the forwarding group. With prefix matching, you can add the wildcard $ at the end for a full match. For example, the user configures the forwarding group URL ~* \.(gif|jpg|bmp)$, hoping to match any file ending with gif, jpg, or bmp.

Default domain policy in forwarding domain name

You can choose to turn the default domain name on or off.

- Enable: When the client request does not match any domain name of this listener, CLB will forward the request to the default domain name (Default Server). Each listener can only be configured with one default domain name.
  - If your layer-7 listener has been configured with a default domain name, client requests that do not match other rules will be forwarded to the default domain name.
  - If your layer-7 listener is not configured with a default domain name, client requests that do not match other rules will be forwarded to the first domain name loaded by the CLB. Since the loading order may be inconsistent with the console configuration order, it is not necessarily the first one configured by the console.
- Disable: When the client request does not match any domain name of this listener, the request will not be forwarded.

For example, in CLB1's HTTP:80 , two domain names are configured under the listener: www.test1.com , 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 .



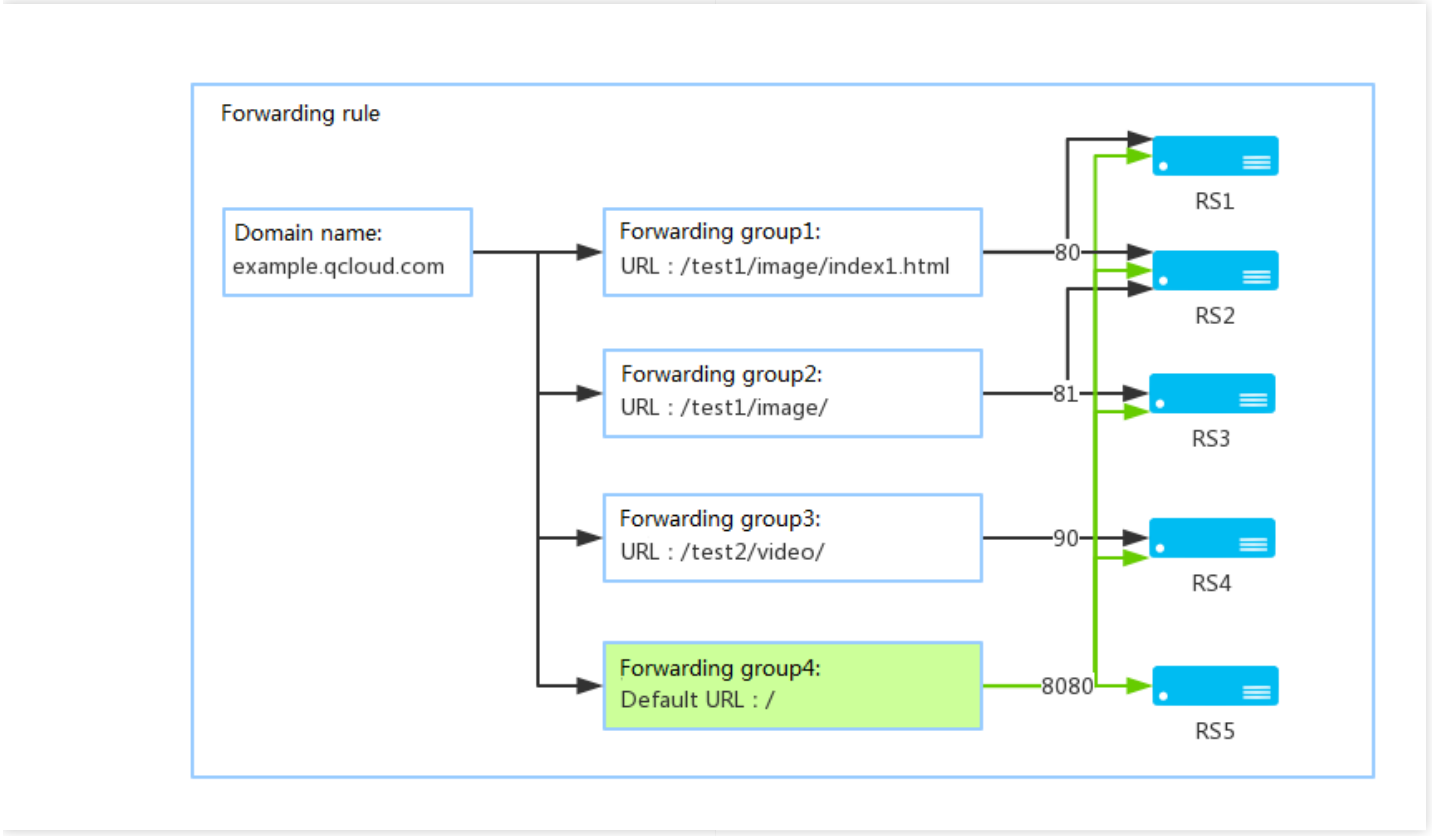
Forwarding group URL matching rules description

URL Configuration Rules

CLB layer-7 listener forwards a path URL,with / by default. The URL must be preceded by /, with a length range of 1 - 120. For the forwarding path URL of a CLB layer-7 listener, the default URL should start with a forward slash '/'. The length must be between 1 to 120 characters.

- URL supports regular expressions, which can be understood by the following explanations:
- Beginning with = indicates an exact match.
- Beginning with ^~ means that the URL starts with a regular character string, not a regular match.
- Beginning with ~ indicates a case-sensitive regular match.
- Beginning with ~* indicates a case-insensitive regular match.
- / means universal matching. Any request will be matched if there is no other match.
- Non-regular URL path starts with / supported character sets are as follows:
  az``AZ``0-9``.``-``/``=``? .
- For regular URL, unsupported character sets are as follows:
  "``{``}``;``\ ``` ~`````Space

URL Matching Rule Examples



1. Matching rules: priority to exact matching, then fuzzy matching according to the rules.

For example, after configuring the forwarding rules and forwarding groups according to the figure above, the following requests will be matched to different forwarding groups in turn:

2. example.qloud.com/test1/image/index1.html Since the URL rule set by forwarding group 1 is exactly matched, the request will be forwarded to the backend CVM associated with forwarding group 1, that is, port 80 of RS1 and RS2 in the figure.
3. example.qloud.com/test1/image/hello.html Since this request cannot exactly match the first rule, it will continue to match the rules in forwarding group 2 and finds that the fuzzy match is successful. Therefore, the request will be forwarded to the backend CVM associated with forwarding group 2, namely port 81 of RS2 and RS3 in the figure.
4. example.qloud.com/test2/video/mp4/ Since this request cannot be exactly matched to the first two rules, it will continue to match downwards until it finds a rule in forwarding group 3 that can be fuzzily matched. Therefore, the request will be forwarded to the backend CVM associated with forwarding group 3, which is port 90 of RS4 in the figure.
5. example.qloud.com/test3/hello/index.html Since this request cannot match the rules in the first three forwarding groups, it will match the most common rule configured by the user Default URL . At this time, Nginx forwards the request to the backend application server, such as FastCGI (php), tomcat (jsp), and Nginx exists as a reverse proxy server.
6. example.qloud.com/test2/ Since the request cannot be matched exactly to the rules in the first three

forwarding groups, it will match the user-configured general rule default URL .

7. If the service cannot run normally in the URL rule set by the user, it will not redirect to other pages after a successful match.
   For example, the client requests example.qloud.com/test1/image/index1.html matches the URL rule of forwarding group 1, but the backend server of forwarding group 1 is running abnormally. When a 404 page appears, the page will display 404 when the user visits it, and will not jump to other pages.
8. It is recommended that users set the Default URL, direct it to a stable service page (such as a static page, homepage), and bind it to all backend CVMs. At this time, if all rules are not matched successfully, the system will direct the request to the page where the Default URL is located, otherwise a 404 problem may occur.
9. If the user has not set a Default URL and all forwarding rules do not match, accessing the service will return a 404 error.

CLB supports SNI multi-domain certificates

Server Name Indication (SNI) is used to improve server and client SSL/TLS. It mainly solves the problem that a server can only use one certificate. Supporting SNI means that the server supports binding multiple certificates. When the client uses SNI, it needs to specify the domain name to connect to before establishing an SSL/TLS connection with the server. The server will return a suitable certificate based on this domain name.

Application Scenario

Converge Cloud The layer-7 HTTPS listener of CLB supports SNI, that is, it supports binding multiple certificates. Different domain names in the listening rules can use different certificates. For example, in the same CLB HTTPS:443 listener, *.test.com uses certificate 1 to forward requests from this domain name to a group of servers; *.example.com uses certificate 2 to forward requests from this domain name to another set of servers.

Prerequisites

CLB instance has been created.

Directions

1. Log in to the CLB console.
2. See Configure HTTPS listener directions to configure the listener and enable SNI when configuring the HTTPS listener.
3. When adding forwarding rules in the listener, configure different server certificates for different domain names, click Next, and continue to complete the configuration of health check and session persistence.

Backend CVM
Backend CVM Overview

What is a backend CVM?

The backend CVM is a CVM instance that is bound to the CLB instance after the CLB instance is created and processes the corresponding forwarding requests. In the configuration CLB listener When you need to bind a CVM instance as the backend server, CLB uses different round-robin scheduling, forwarding the request to the backend CVM, which is processed by CVM to ensure the smooth and reliable operation of the application. You can bind CVM instances to one or more availability zones in the region where the CLB instance is located to increase the robustness of the application and shield single points of failure.

Notes

When adding backend servers, we recommend that you:

- Install a Web server (such as Apache or IIS) on all CVM instances to be added to the CLB, and keep running applications consistently.
- It is recommended that you enable session persistence. This feature enables the CLB to maintain a TCP connection for a longer time and reuse it for multiple requests, which can reduce the load on the Web server and improve the throughput of the CLB.
- Ensure that the security group of the backend instance has inbound rules for the CLB listener port and health check port. For details, see Access control of backend servers .

Add, modify, and unbind backend servers

CLB routes requests to healthy backend CVM instances. If you need to add or remove backend servers for the first time or based on business needs, follow the instructions below.

> - Unbinding a backend server will disassociate the CLB instance from the CVM instance, and the CLB instance will immediately stop forwarding requests to it.
> - Unbinding a backend server will not affect the lifecycle of CVM, and you can add it to the backend server cluster again.
> - If a CLB instance is associated with an auto scaling group, the CVMs in the group will be automatically added to the CLB backend CVMs. The CVM instances removed from the auto scaling group will be automatically deleted from the CLB backend CVMs.

Add a CLB backend CVM

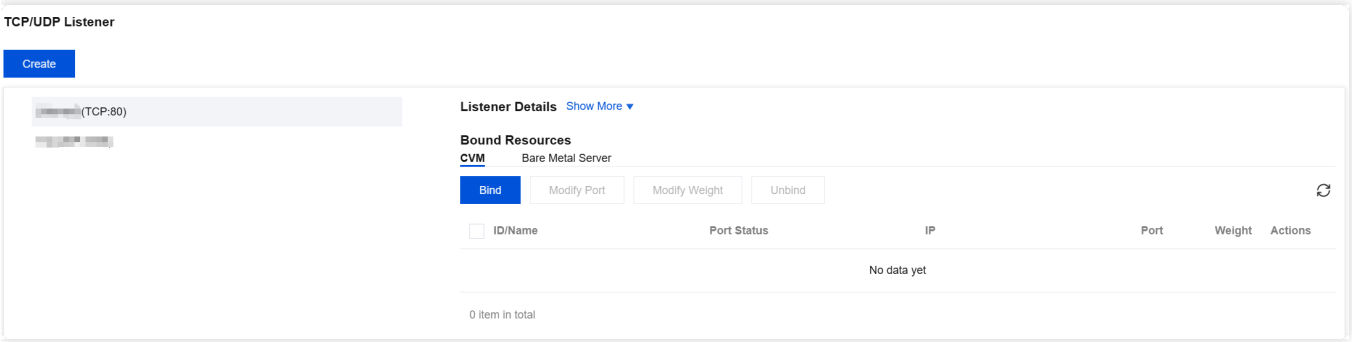
1. Log in to the CLB console. In the list, click the corresponding CLB instance ID to enter the CLB details page.

2. In the listener corresponding to the instance, add the corresponding backend CVM.

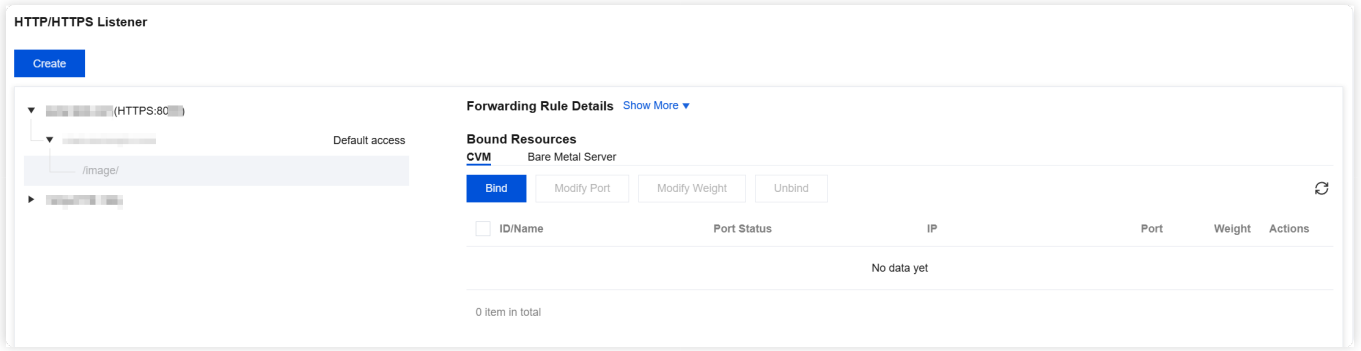
   > Description:
   >
   > IPv4 CLB supports binding to CVMs and BMSs in the same way;

   - TCP/UDP Listener

Select the listener that needs to bind a backend CVM, and click Bind.

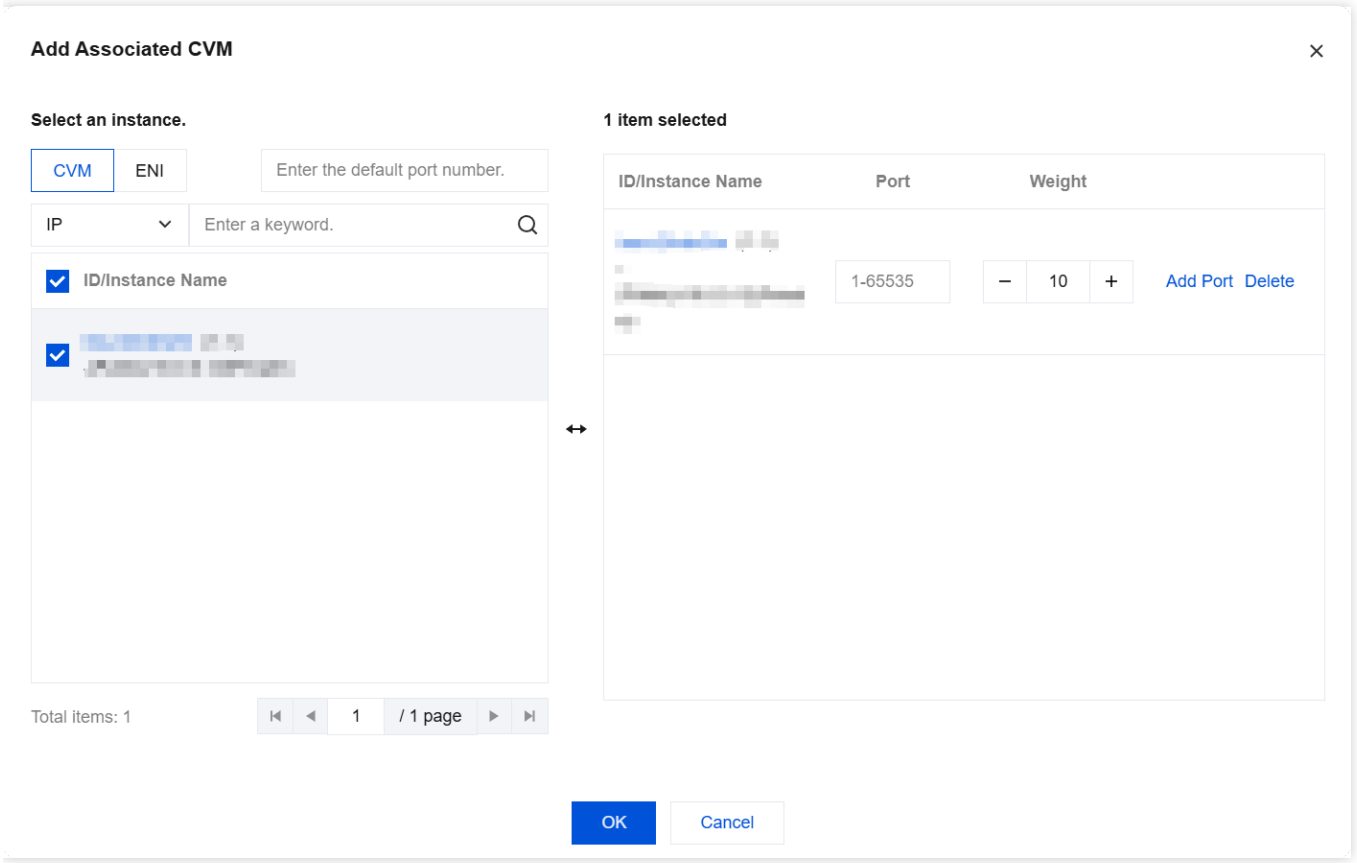


- HTTP/HTTPS Listener
  Select the listener that needs to bind a backend CVM, and click Bind.



3. In the pop-up, select the CVM to be associated, fill in the relevant port and weight for the CVM that need to be forwarded, and click OK to complete the association between the CVM and the CLB.

> Description:
>
> The pop-up only displays optional CVMs that are in the same Region, have the same Network Environment, are not isolated, are not expired, and have a Bandwidth (Peak) greater than 0.

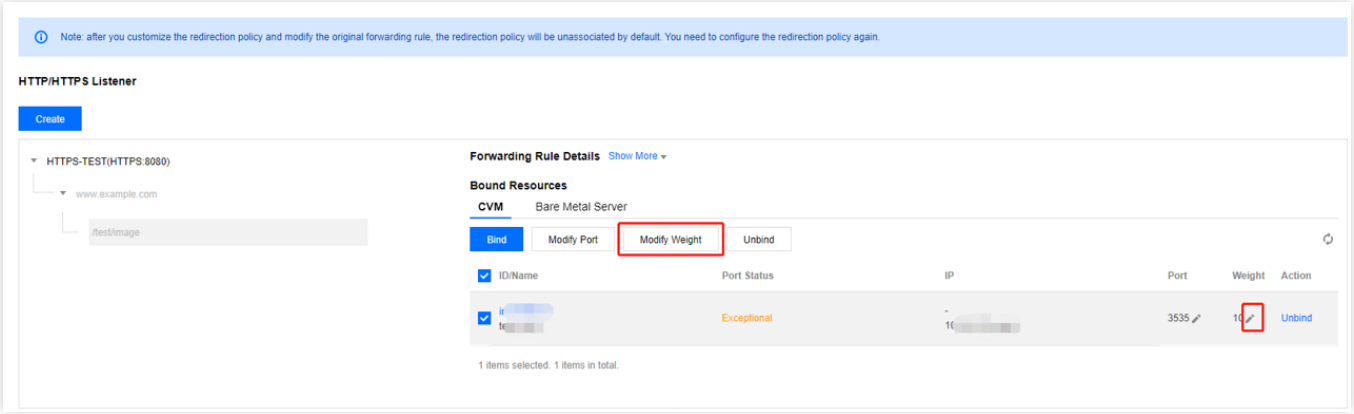


4. If you need to bind CVMs in batches and the preset port values are the same, you can first enter the preset port value in Default Port, then check the relevant CVMs and set the weight value, and click OK to complete the binding.

Modify the weight of the CLB backend server

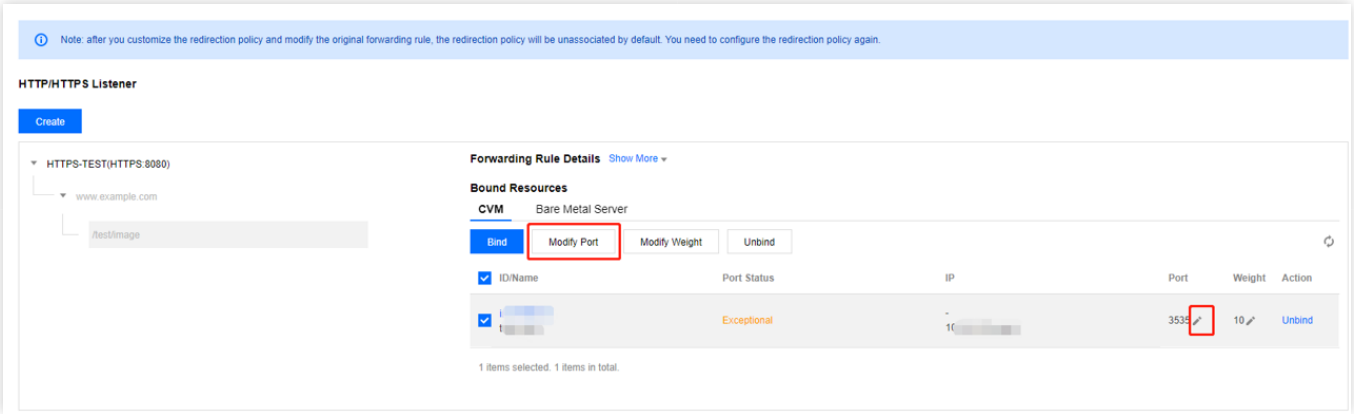
The backend server weight determines the relative number of requests forwarded to the CVM. When binding a backend CVM, you need to preset the weight. If you need to modify the weight later, see the following instructions. For more information about weights of CLB backend servers, see CLB round-robin scheduling method.

1. Log in to the CLB console. In the list, click the corresponding CLB instance ID to enter the CLB details page.
2. After selecting the instance and listener rules, select the relevant server in the server list, click the edit icon, enter the modified weight value, and click OK to complete the modification of the server weight.
3. If you need to modify in batch, click Modify Weight after selecting all relevant servers, enter the modified weight value, and click OK to complete the batch modification.



Modify the CLB backend server port

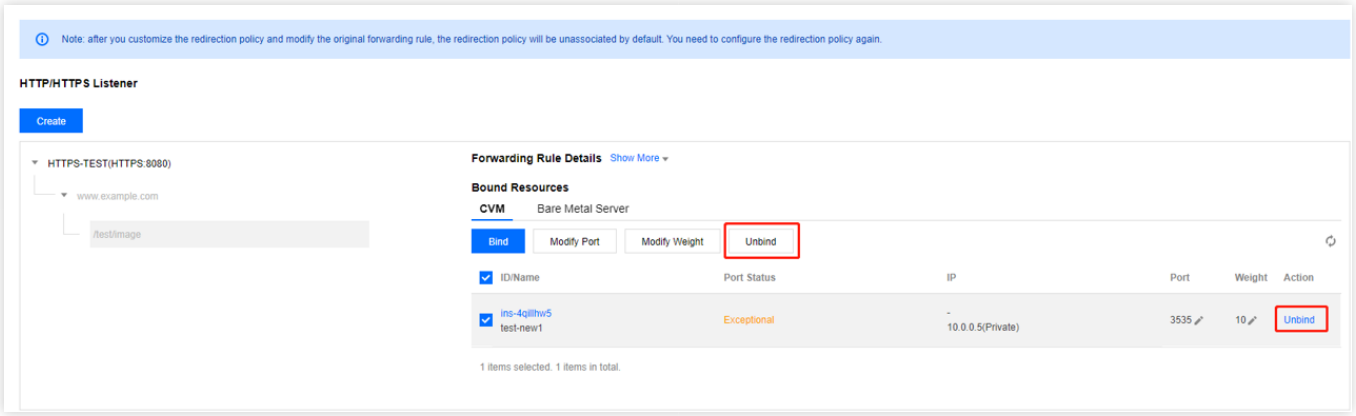
1. Log in to the CLB console. In the list, click the corresponding CLB instance ID to enter the CLB details page.
2. After selecting the instance and listener rules, select the relevant server in the server list, click Edit, enter the modified port value, and click OK to complete the modification of the server port.
3. If you need to modify in batch, select all related servers, click Modify Port, enter the modified port, and click OK to complete the batch modification.



Unbind the CLB backend server

1. Log in to the CLB console. In the list, click the corresponding CLB instance ID to enter the CLB details page.

2. After selecting the listener and rule, select the CVM to be unbound in the CM list on the right, click Unbind, and click OK in the pop-up box.
3. If you need to unbind in batches, select all the CVMs that need to be unbound, click Unbind, and then click OK in the pop-up box.



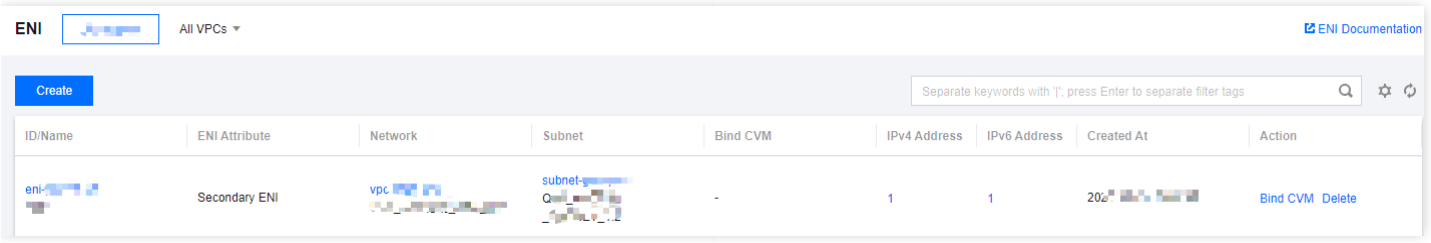
Binding Elastic Network Interface

Introduction to ENI

Elastic Network Interface (ENI) is a virtual network interface that can be bound to a CVM instance in a virtual private cloud. Elastic network interface can be freely migrated between CVMs in the same virtual private cloud and availability zone. Elastic network interface can be used to build high-availability clusters, perform low-cost failover, and implement refined network management.
The backend service of CLB supports CVM and ENI, that is, CLB supports binding CVM and ENI. CLB uses private network communication with backend services. When CLB is bound to multiple CVMs and ENIs, access traffic will be forwarded to the private network IP of the CVM and the private network IP of the ENI.

Prerequisites

The ENI must first be bound to a specific CVM before the CLB can bind to that ENI. The CLB only forwards traffic and does not handle business logic, so you need CVM instances to process user requests. Please go to the VPC Console to bind the required ENI to a Cloud Host.

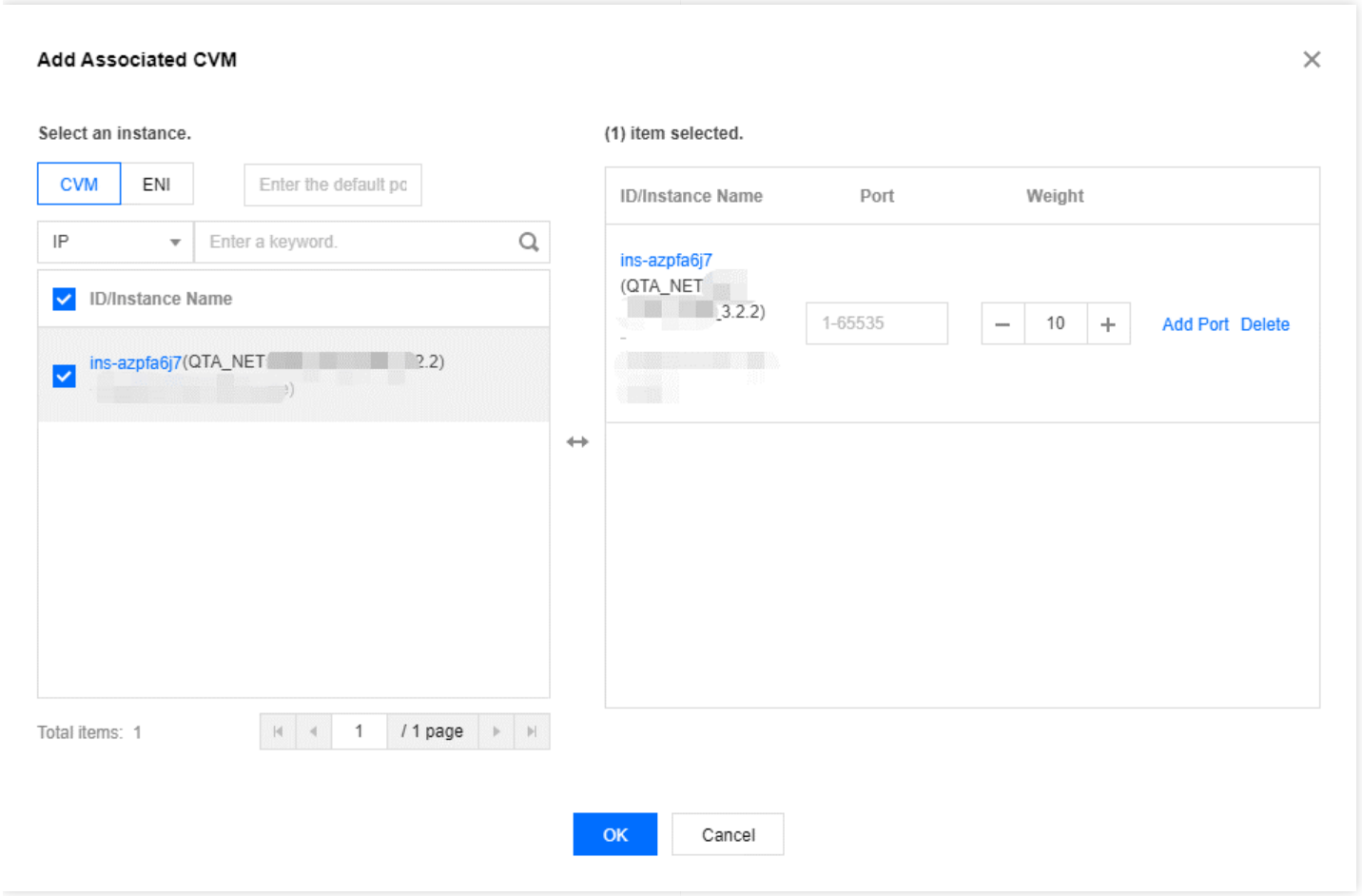


Operation step

1. You need to configure the CLB listener first. For details, see CLB listener overview.
2. Click the created listener, expand the domain name and URL path, select the specific URL path, and view the bound backend service on the right side of the listener.
3. Click Bind, select the backend server to be bound in the pop-up box, and configure the service port and weight. When binding the backend service, you can select Cloud Virtual Machine or Elastic Network Interface.
    - CVM: You can bind the primary private network IP address of the primary network card of all CVMs in the same private network as CLB.
    - ENI: You can bind all ENI IP addresses in the same virtual private cloud as the CLB, except for

the primary private network IP of the CVM's primary ENI, such as the secondary private network IP of the primary ENI and the private network IP of the secondary ENI. For details on ENI IP types, see ENI-Related Concepts.

> Description:
>
> IPv4 CLB supports binding to CVMs or ENIs.



4. Click OK to complete the binding of the backend ENI.

Backend CVM security group configuration instructions

Introduction to CVM security groups

The backend CVM instance of the CLB can perform access control through security group and act as a firewall.
You can associate one or more security groups with backend CVMs and add one or more rules to each security group to control traffic access permissions for different servers. You can modify the rules of a security group at any time, and the new rules will automatically apply to all instances associated with the security group. In virtual private cloud environment, you can also use network ACL to perform access control.

CVM security group configuration instructions

In the CVM security group, the Client IP and service port must be allowed.
If you use CLB to forward business traffic to CVM, to ensure the health check function, you need to make the following configuration on the CVM security group:

1. Public Network CLB: You need to allow access to the CLB VIP in the security group of the backend CVM. The CLB uses the VIP to detect the health status of the backend CVM.
2. Private Network CLB: CLB belongs to a VPC network. You need to allow access to the CLB VIP in the security group of the backend CVM (for health check).

CVM security group configuration example

The following example shows the configuration of the CVM security group when accessing the CVM through the CLB.

- Use case 1:
  Public network CLB. The listener is configured as TCP:80 and the backend service port is 8080. To allow only specific client IP addresses (ClientA IP address and ClientB IP address) to access the CLB instance, configure the inbound rules for the backend server security group as follows:

  ClientA IP + 8080 allow
  ClientB IP + 8080 allow

CLB VIP + 8080 allow
0.0.0.0/130618222416261120 + 8080 drop

- Use case 2:
  Public network CLB. The listener is configured as HTTP:80 and the backend service port is 8080. To allow normal access from all client IP addresses, configure the inbound rules for the backend server security group as follows:

  0.0.0.0/130618222416261120 + 8080 allow

- Use case 3:
  Application-based private network CLB. The network type is VPC, and the CVM security group should allow traffic from the CLB instance VIP for health check. A TCP:80 listener is configured for the CLB instance, with the backend service port set to 8080. The goal is to allow only specific client IP addresses (ClientA IP address and ClientB IP address) to access the CLB instance VIP, and also restrict the client IP addresses to only access the backend hosts bound to the CLB instance.

  i. The inbound rules of the backend server security group are configured as follows:

  ClientA IP + 8080 allow
  ClientB IP + 8080 allow
  CLB VIP + 8080 allow
  0.0.0.0/130618222416261120 + 8080 drop

  ii. The outbound rules of the server security group used as the Client are configured as follows:

  CLB VIP + 8080 allow
  0.0.0.0/130618222416261120 + 8080 drop

- Use case 4: blocklist
  If you need to configure a blocklist for certain client IP addresses to deny their access, you can achieve this by configuring the security group associated with the cloud service. The security group rules need to be configured as follows:
  1.1. Add the client IP address and port that need to be denied access to the security group, and select the option to deny access from that IP address in the policy section.
  1.2. After completing the settings, add another security group rule to allow access from all IP addresses on this port by default.
  After the configuration is complete, the security group rules are as follows:

  clientA IP + port drop
  clientB IP + port drop

0.0.0.0/130618222416261120 + port accept

> Note:

- The above configuration steps need to be performed in a specific order. Reversing the order will cause the blocklist configuration to fail.
- Security groups are stateful. Therefore, the above configurations are all for inbound rules, and no special configuration is required for outbound rules.

CVM security group operation guide

Managing backend server security groups using the console

1. Log in to the CLB console and click the corresponding CLB instance ID to enter the CLB details page.
2. On the CLB-bound CVM page, click the corresponding backend server ID to enter the CVM details page.
3. Click the Security Group tab to bind/unbind the security group.

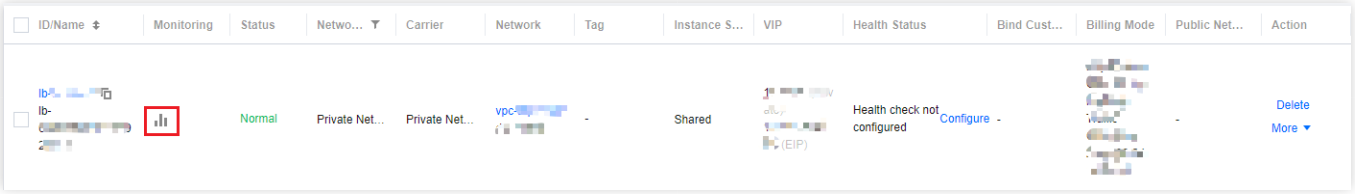
Monitoring and Alarms
Obtain monitoring data

Cloud Monitor provides data collection and display functions for CLB and backend instances. Using Cloud Monitor, you can CLB statistics, verify whether the system is running properly, and create corresponding alarms. For more information about Cloud Monitoring, see Cloud Monitor product documentation.

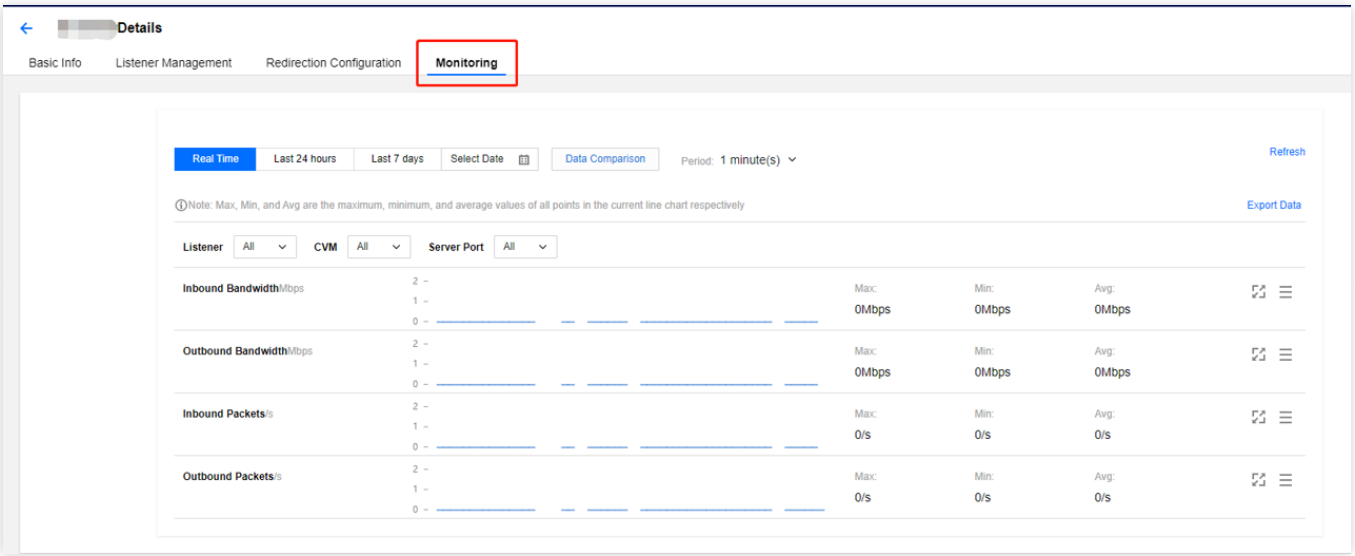
Converge Cloud provides cloud monitoring to all users by default. You do not need to enable it manually. As long as you use the CLB, Cloud Monitor can help you collect relevant monitoring data. You can view the monitoring data of the CLB in the following ways:

CLB Console

1. Log in to the CLB console and click the monitoring icon next to the CLB instance ID to quickly view the performance data of each instance through the monitoring floating window.



2. Click the CLB instance ID to enter the CLB details page, and click the Monitoring tab to view the monitoring data of the current CLB instance.



Cloud Monitor Console

Log in to Cloud Monitor Console, click CLB under the Cloud Services Monitoring module in the left sidebar, click the CLB instance ID to enter the monitoring details page, where you can view the monitoring data of the CLB instance. Expand the instance to view the monitoring information of the listener and backend server.

Monitoring metric description

Cloud Monitor collects raw data from the CLB instances under running status and displays the data in easy-to-read icons. Statistics are saved for one month by default. You can observe the running status of the instance for one month to better understand the running status of the application service.

We recommend that you view the monitoring of CLB through the cloud monitor console. Click Cloud Services Monitor > CLB, click the CLB instance ID, go to the monitoring details page, view the monitoring data of the CLB instance, and expand the instance to view the monitoring information of the listener, backend server, etc.

CLB instance dimension

Metrics	Unit	Note
Inbound Bandwidth	bps	The bandwidth used by clients to access the CLB function through the public network during the statistical period.
Outbound Bandwidth	bps	The bandwidth used by CLB to access the public network during the statistical period.
Number of packages	per second	The number of request packets received by the CLB per second during the statistical period.
Number of outgoing packets	per second	The number of data packets sent by the CLB per second during the statistical period.

Layer-4 listener (TCP/UDP) dimension

The layer-4 listener allows you to view the monitoring metrics at the following three dimensions:

- Listener dimension
- Backend CVM dimension
- Port dimension of backend CVM

Metrics	Unit	Note
Number of Connections	per	The number of connections on the listener during the statistical period.

Metrics	Unit	Note
Number of New Connections	per	The number of new connections created on the listener during the statistical period.
Inbound Bandwidth	bps	The bandwidth used by clients to access the CLB function through the public network during the statistical period.
Outbound Bandwidth	bps	The bandwidth used by CLB to access the public network during the statistical period.
Number of packages	per second	The number of request packets received by the CLB per second during the statistical period.
Number of outgoing packets	per second	The number of data packets sent by the CLB per second during the statistical period.

Layer-7 listener (HTTP/HTTPS) dimension

The layer-7 listener allows you to view the monitoring metrics at the following five dimensions:

- Listener dimension
- Domain name dimension
- URL forwarding path dimension
- Backend CVM dimension
- Port dimension of backend CVM

Metrics	Unit	Note
Number of Connections	per	The number of connections on the listener during the statistical period.
Number of New Connections	per	The number of new connections created on the listener during the statistical period.
Inbound Bandwidth	bps	The bandwidth used by clients to access the CLB function through the public network during the statistical period.
Outbound Bandwidth	bps	The bandwidth used by CLB to access the public network during the statistical period.
Number of packages	per second	The number of request packets received by the CLB per second during the statistical period.

Metrics	Unit	Note
Number of outgoing packets	per second	The number of data packets sent by the CLB per second during the statistical period.
Average Response Time	ms	The average response time of CLB during the statistical period.
Maximum response time	ms	The maximum response time of CLB during the statistical period.
Number of response timeouts	per	The number of CLB response timeouts during the statistical period.
Requests per second	per	The number of CLB requests per second during the statistical period is QPS.
2xx status codes	per	The number of 2xx status codes returned by the backend server during the statistical period.
3xx status codes	per	The number of 3xx status codes returned by the backend server during the statistical period.
4xx status codes	per	The number of 4xx status codes returned by the backend server during the statistical period.
5xx status codes	per	The number of 5xx status codes returned by the backend server during the statistical period.
404 status code	per	The number of 404 status codes returned by the backend server during the statistical period.
502 status code	per	The number of 502 status codes returned by the backend server during the statistical period.
3xx status code returned by CLB	per	The number of 3xx status codes returned by CLB during the statistical period.
4xx status code returned by CLB	per	The number of 4xx status codes returned by CLB during the statistical period.
5xx status code returned by CLB	per	The number of 5xx status codes returned by CLB during the statistical period.
404 status code returned by CLB	per	The number of 404 status codes returned by CLB during the statistical period.
The 502 status code returned by CLB	per	The number of 502 status codes returned by CLB during the statistical period.

> Note:
>
> If you need to view the monitoring data of a CVM under a listener, log in to the CLB console and click the monitoring icon next to the CLB instance ID. You can quickly browse the performance data of each instance through the monitoring floating window.

Configure Alarm Policy

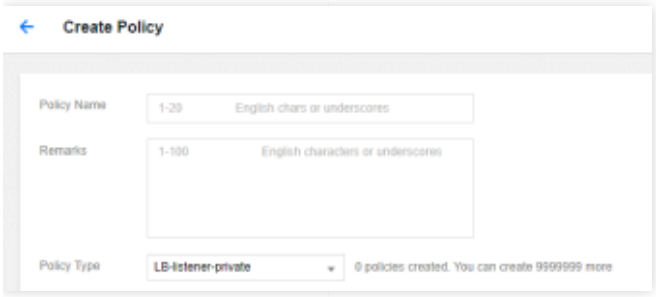
You can create alarms to trigger alarms when the cloud service status reaches the set conditions and send relevant messages to specified user groups. The created alarm will determine whether to trigger relevant notifications based on the situation of the monitored metrics relative to the given threshold at intervals.

When a cloud service status change causes an alarm to be triggered, designated users can take corresponding preventive or remedial measures in a timely manner. Therefore, creating alerts properly can help you improve the robustness and reliability of your application. For more information about alarms, see Create an alarm policy.

The detailed steps to create an alarm policy are as follows:

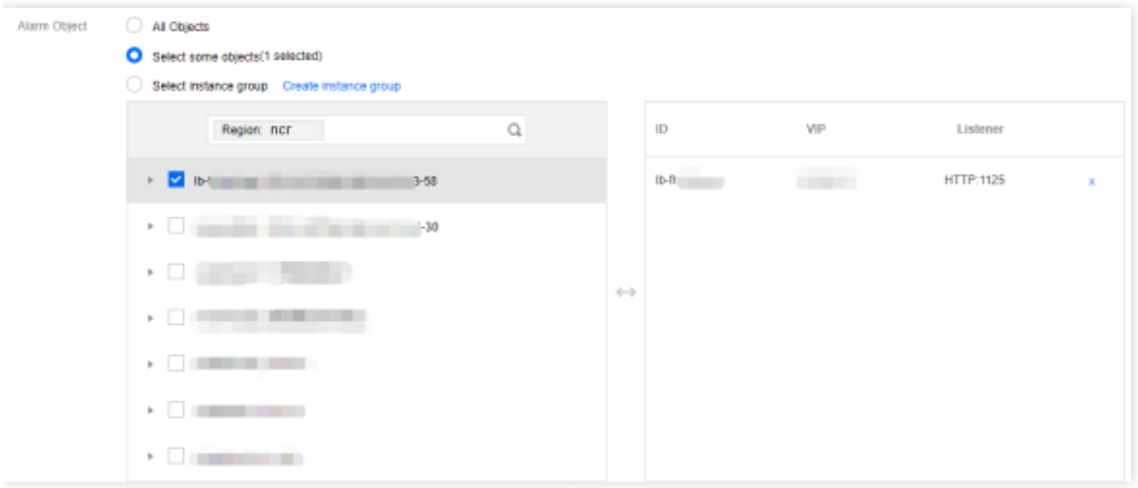
1. Log in to the cloud console and click Cloud Monitor to enter the cloud monitor console.
2. On the left menu, select Alarm Configuration > Alarm Policy to enter the alarm policy configuration page.
3. Click Add to configure the alarm policy.
4. For Configuring basic options, the configuration instructions are as follows.

- Policy Name: Enter the policy name.
- Notes: Fill in the policy notes.
- Policy Type: Select a monitoring item.



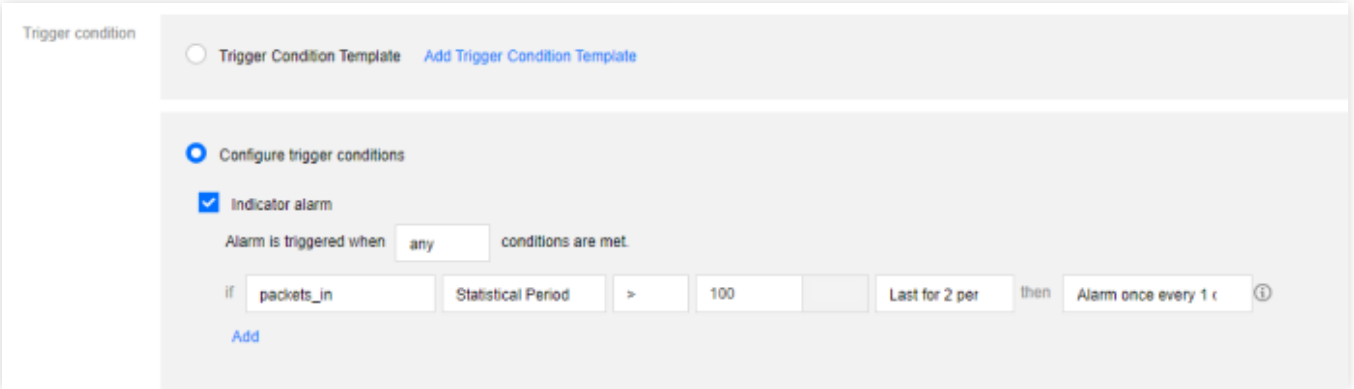
5. Configure the alarm object.

- After you select all objects, the alarm policy is bound to all instances under the current account.
- After you select some objects, the alarm policy is bound to the instances selected.
- By selecting an instance group, the alarm policy will be bound to the user-selected group of instances.

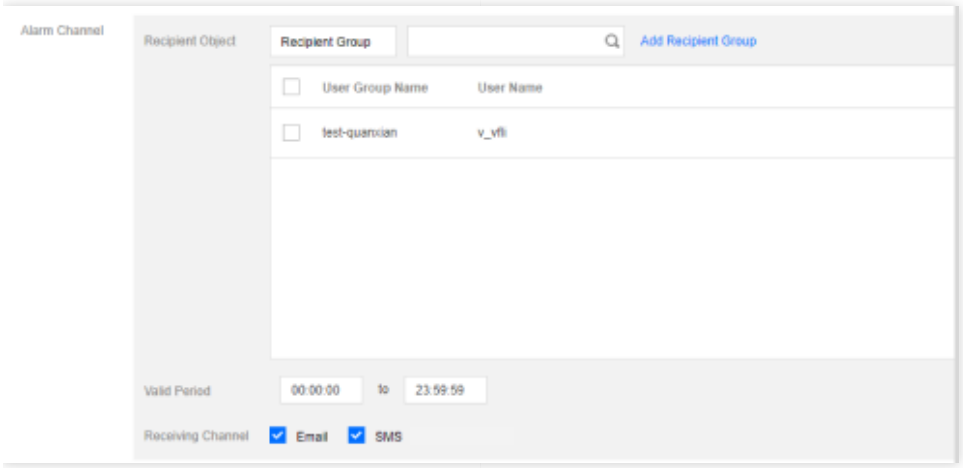


6. Set the alarm trigger conditions. There are two methods: trigger condition template and configure trigger condition. You can choose one of the trigger conditions.

- Trigger Condition Template
  Open the trigger condition template and select a configured template from the drop-down list. For detailed configuration, see the Cloud Monitor documentation. If the newly created template is not displayed, click Refresh on the right to refresh the trigger alarm template selection list.
- Configure Trigger Condition
  The alarm trigger condition is a semantic condition composed of metrics, statistical cycles, comparison relationships, thresholds, duration cycles, and repeated notifications.
  For example, if the specified metric is the number of packets , the statistical period is one minute , the comparison relationship is > , threshold is 100 per second , the duration is for 2 cycles , repeat notification is once every day , this indicates: The incoming packet data is collected once every minute; if the incoming packet volume of a CLB instance listener exceeds 100 packets per second for two consecutive times, an alarm is triggered, and once daily.



7. Configure Alarm Channels
   Configure the alarm receiving group, effective period, and receiving channel (email, object) as required.



8. You can set an existing policy as the default alarm policy, and newly purchased CLB will be automatically associated with the default policy.

- There is only one default policy per project for each policy type.
- The default alarm policy cannot be deleted.

Alarm Metric Description

Alarm Description

You can create alarms for instance metrics that you are interested in, so that when the running status of the CLB instance reaches a certain condition, alarm information can be sent to the user groups that are interested in it in a timely manner. This ensures that you can detect exceptional conditions in a timely manner and take appropriate measures to maintain the stability and reliability of the system. For details, see Alarm Overview.
The CLB alarm policies include the following types:

- Public network listener
- Private network listener
- Server port
- Server port dimension

Public Network Monitor/Private Network Listener

Currently, both public network CLB and private network CLB support alarms at the listener dimension. The specific metrics are as follows:

Metrics	Unit	Note
Inbound Bandwidth	bps	The bandwidth used by clients to access the CLB function through the public network during the statistical period.
Outbound Bandwidth	bps	The bandwidth used by CLB to access the public network during the statistical period.
Number of packages	per second	The number of request data packets received by the private CLB per second during the statistical period.
Number of outgoing packets	per second	The number of data packets sent by the CLB per second during the statistical period.

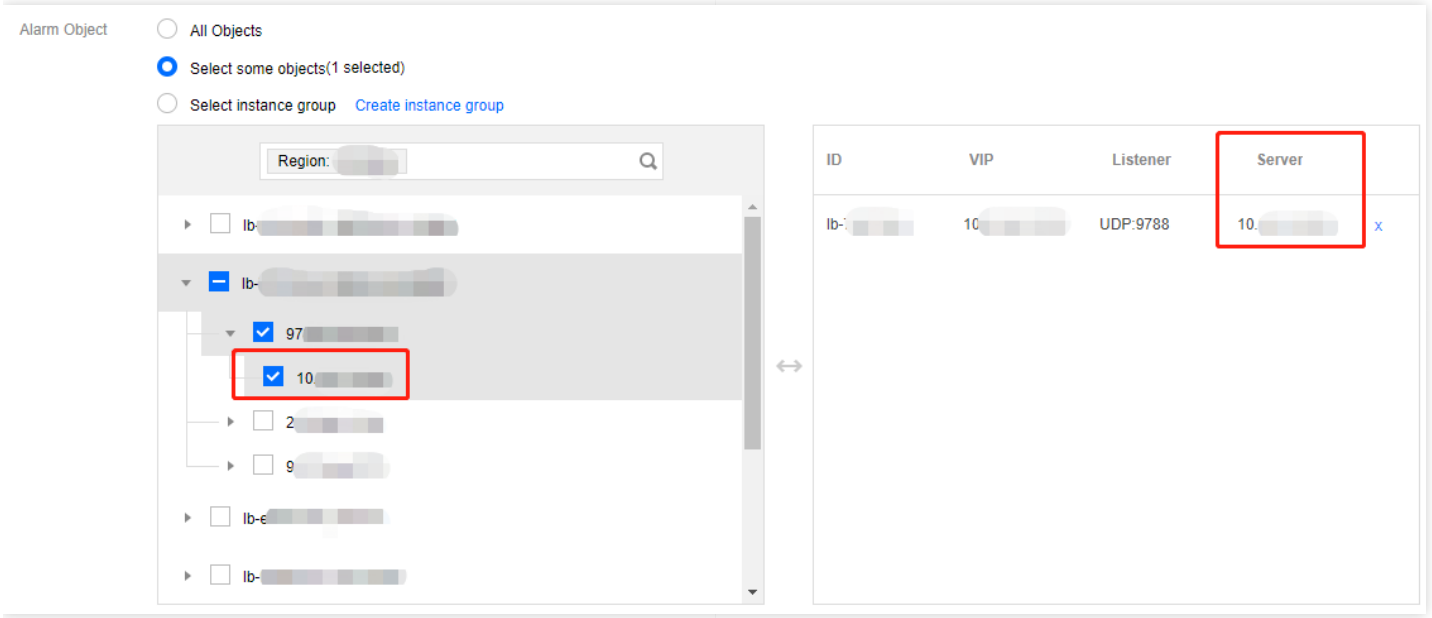
Server port

CLB supports alarms at the server port dimension:

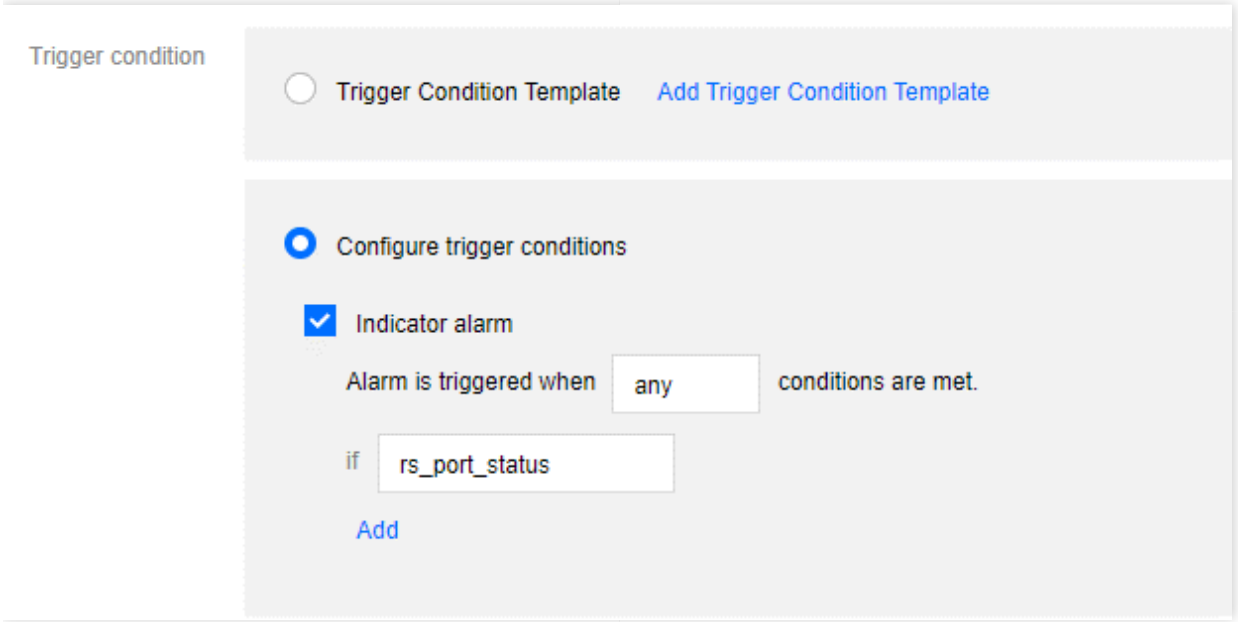
You can configure an abnormal alarm for a port of a backend CVM bound to a listener, and send an alarm

as long as the port is exceptional.

Alarm object configuration:



Trigger condition configuration:



> Note:
>
> The backend server port exception means: the CLB detects that the port of the backend server is unavailable. A few network jitters can also trigger port exceptions.

Certificate Management
Management Certificates

When configuring the HTTPS listener of the CLB service, you can directly use the certificate from the SSL certificate service or upload the required server certificate and SSL certificate issued by a third party to the CLB service.

Certificate Requirements

CLB only supports certificates in PEM format. Before uploading your certificate, make sure your certificate, certificate chain, and private key meet the required formats. For certificate requirements, see Certificate Requirements and Certificate Format Conversion.

Configuring Certificates

There are two types of certificate configuration for HTTPS listeners:

- If SNI is not enabled, the certificate is configured at the listener dimension, and all domain names under the listener use the same certificate. For details, see Configure certificates at the listener dimension.
- If SNI is enabled, the certificate is configured at the domain name dimension. Different certificates can be configured for different domain names under this listener. For details, see Configure certificates at the domain dimension.

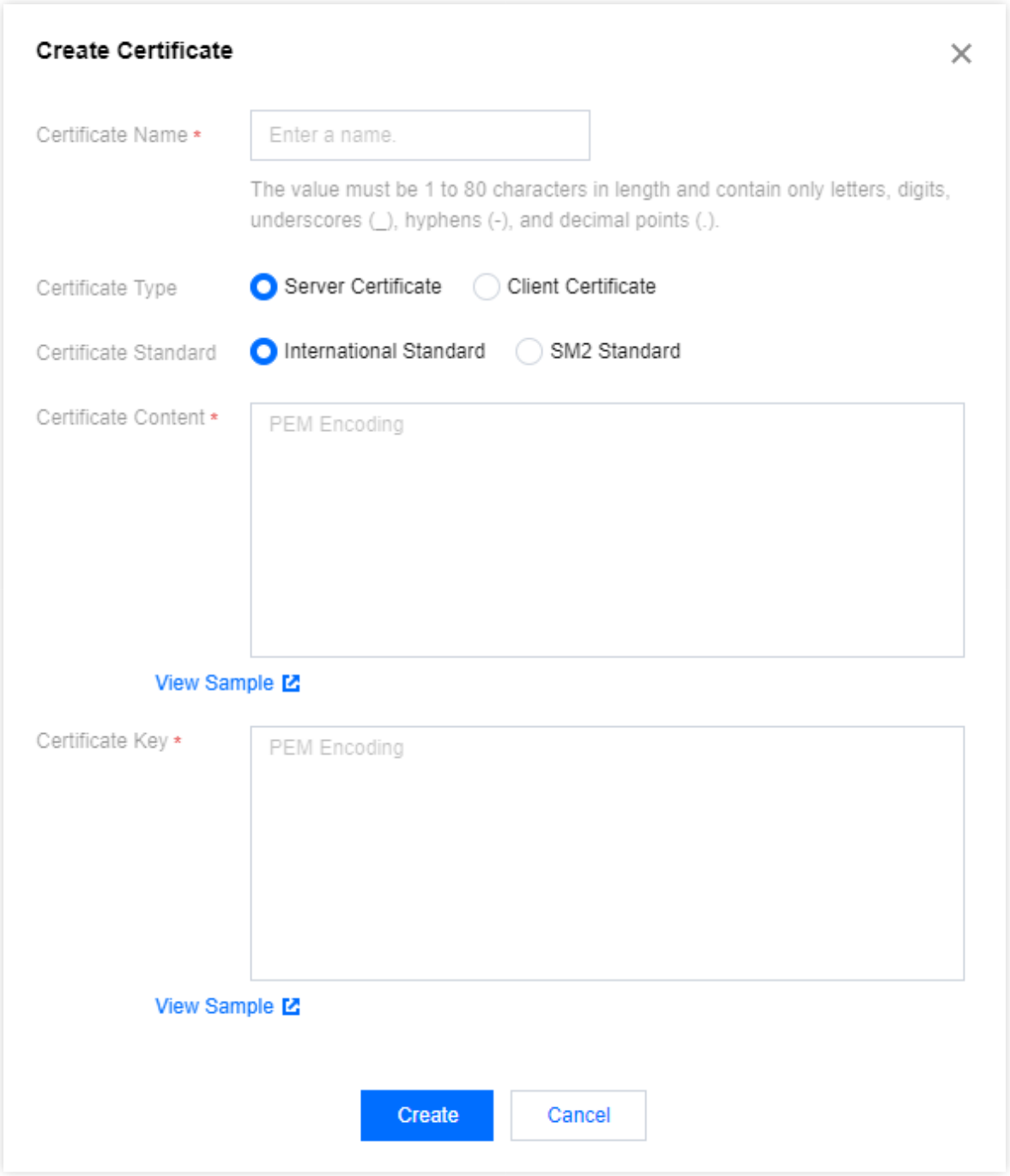
Batch certificate renewal

To avoid the impact of certificate expiration on your services, renew the certificate before it expires.

> Note:
>
> After certificate update, the system will not delete the old certificate, but will generate a new certificate. All CLB instances using the certificate will automatically update the certificate.

1. Log in to the CLB console.
2. In the left sidebar, click Certificate Management.
3. In the certificate list on the Certificate Management page, click Update in the Action column to the right of the target certificate.

4. In the Create Certificate dialog box that pops up, enter the certificate content and key content of the new certificate and click Create.



View the CLB associated with the certificate

1. Log in to the CLB console.
2. In the left sidebar, click Certificate Management.
3. In the certificate list on the Certificate Management page, click the target certificate ID.
4. On the Basic Information page, view the CLB instance to which the certificate is associated.

Basic Info

Name	2-16
ID	
Certificate Type	Server Certificate
Certificate Specifications	RSA
Certificate Content	-----BEGIN CERTIFICATE-----
Load balancer associated.	
Primary Domain Name	
Alternate Domain Name	
Uploaded At	20 15:27:16
Enabled At	20. 14:36:33
Expired At	20 4:36:33

Certificate Requirements and Certificate Format Conversion

Common Certificate Application Process

- Generate a private key locally: openssl genrsa -out privateKey.pem 2048, where privateKey.pem is your private key file, and keep it properly.
- Generate a certificate request file: openssl req -new -key privateKey.pem -out server.csr, where server.csr is your certificate request file, which can be used to apply for a certificate.
- Get the content in the request file and go to the CA or other institution sites to apply for a certificate.

Certificate Format Description

Certificate Format Requirements

- The certificate that the user needs to apply for is: a certificate in PEM format under Linux environment. CLB does not support certificates in other formats. For certificates in other formats, see the Instructions on Converting Certificate Format to PEM below.
- If the certificate is issued by a root CA agency, the certificate you get is the only one and no additional certificates are required. The configured site can be considered trustworthy by browsers and other access devices.
- If the certificate is issued by an intermediate CA agency, the certificate file you get contains multiple certificates, and you need to manually merge the server certificate and the intermediate certificate and upload them together.
- If your certificate has a certificate chain, convert the certificate chain content into PEM format and upload it together with the certificate content.
- The splicing rule is: put the server certificate in the first copy, and the intermediate certificate in the second copy, with no blank lines in between.

  > Usually, when an organization issues a certificate, corresponding instructions will be provided. Please make sure to review them.

Certificate format and certificate chain format examples

The following are examples of the certificate format and certificate chain format. Confirm that the format is correct before uploading:

1. Certificate issued by the root CA agency: The certificate format is PEM format in the Linux environment. For example:



The certificate rules are:

- [———-BEGIN CERTIFICATE———-, ———-END CERTIFICATE———-] Beginning and ending; upload these contents together.
- Each line has 64 characters, and the last line has no more than 64 characters.

2. Certificate chain issued by the intermediate agency:

———-BEGIN CERTIFICATE———-

———-END CERTIFICATE———-

———-BEGIN CERTIFICATE———-

———-END CERTIFICATE———-

———-BEGIN CERTIFICATE———-

———-END CERTIFICATE———-

The certificate chain rules are:

- There cannot be blank lines between certificates.
- Each certificate complies with the above 2.1 Certificate format requirements.

RSA Private Key Format Requirements

For example:



RSA private keys can include all private keys (RSA and DSA), public keys (RSA and DSA), and (x509) certificates. It stores DER format data encoded in Base64, surrounded by an ASCII header, thus suitable for text-mode transmission between systems.

RSA private key rules:

- [———-BEGIN RSA PRIVATE KEY———-, ———-END RSA PRIVATE KEY———-] Beginning and ending; upload these contents together.
- Each line has 64 characters. The last line can be less than 64 characters.
  If you do not generate a usable private key in the format of [———-BEGIN PRIVATE KEY———-, ———-END PRIVATE KEY———-] according to the above scheme, you can convert it into a usable private key as follows:

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

Then upload the contents of new_server_key.pem along with the certificate.

Certificate Conversion to PEM Format Instructions

Currently, the CLB service only supports certificates in PEM format. Certificates in other formats need to be converted to PEM format before they can be uploaded to the CLB service. It is recommended to use the openssl tool for conversion. Below are several popular methods for converting popular certificate formats

to PEM format.

Converting certificate format from DER to PEM

The DER format is commonly found in the Java platform.

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

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

Converting P7B Format Certificates to PEM Format

The P7B format is commonly found in Windows Server and tomcat.

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

Get the content [——-BEGIN CERTIFICATE——-, ——-END CERTIFICATE——-] in outcertificat.cer and upload it as the certificate.

Private key conversion: Private keys can usually be exported in the IIS server.

Converting PFX Format Certificates to PEM Format

The PFX format is commonly found in Windows Server.

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

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

Converting CER/CRT Format Certificates to PEM Format

For certificates in CER/CRT format, you can convert them by directly changing the certificate file extension. For example, you can simply rename the "servertest.crt" certificate file to "servertest.pem".

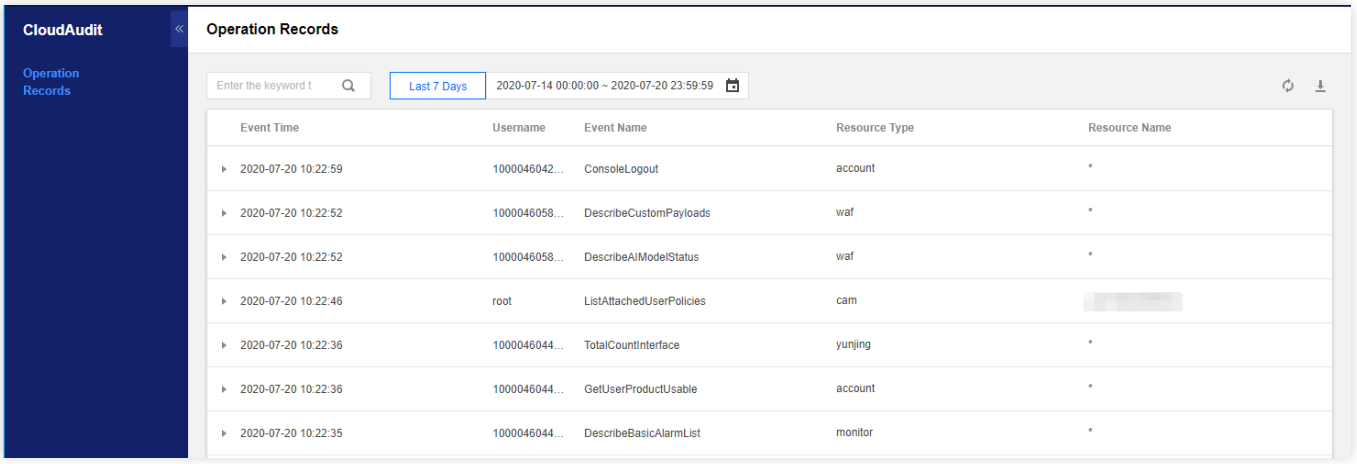
Log Management
View operation logs

You can find it in the CloudAudit Console to query and download CLB operation records.

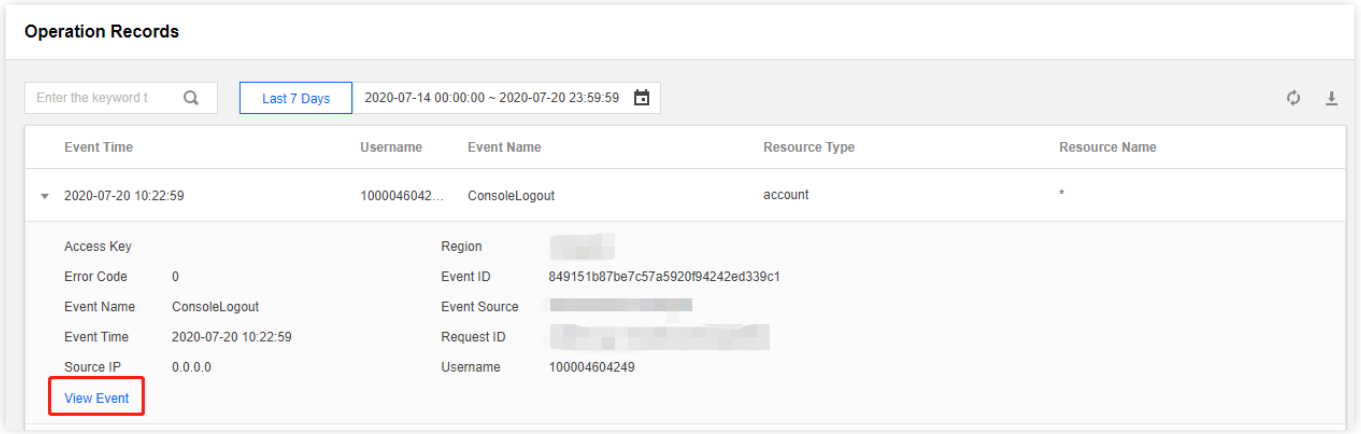
CloudAudit It is a service that supports supervision, compliance checking, operation review and risk review of your Converge Cloud account. CloudAudit provides an event history of Converge Cloud account activities, including operations performed through the Converge Cloud management console, API services, command line interfaces, and other cloud services. This event history simplifies security analysis, resource change tracking, and troubleshooting.

Directions

1. Log in to the CloudAudit Console.
2. On the Operation Records page, you can query operation records based on username, resource type, resource name, event source, event ID, etc. By default, only some data is displayed. You can click [Load More] at the bottom of the page to get more records.



3. If you want to learn more about a single operation record, click the on the left side of the record to view details, including the access key, error code, event ID, etc. You can also click [View Event] for related event information.


Operation Records
Enter the keyword t
Last 7 Days
2020-07-14 00:00:00 ~ 2020-07-20 23:59:59
Event Time
Username
Event Name
Resource Type
Resource Name
2020-07-20 10:22:59
1000046042...
ConsoleLogout
account
*
Access Key
Region
Error Code
0
Event ID
849151b87be7c57a5920f94242ed339c1
Event Name
ConsoleLogout
Event Source
Event Time
2020-07-20 10:22:59
Request ID
Source IP
0.0.0.0
Username
100004604249
View Event

Configuring Access Logs

The CLB access log collects detailed information about each client request, including request time, request path, client IP and port, return code, response time, and other information. Access logs can help you understand client requests, assist in troubleshooting, analyze and sort out user behavior, etc. Currently, access logs can be stored in CLS, with minute-level log reporting and online multi-rule retrieval supported.
The access log of CLB is mainly used for troubleshooting to help businesses quickly locate problems. Access log features include log reporting, log storage and query.

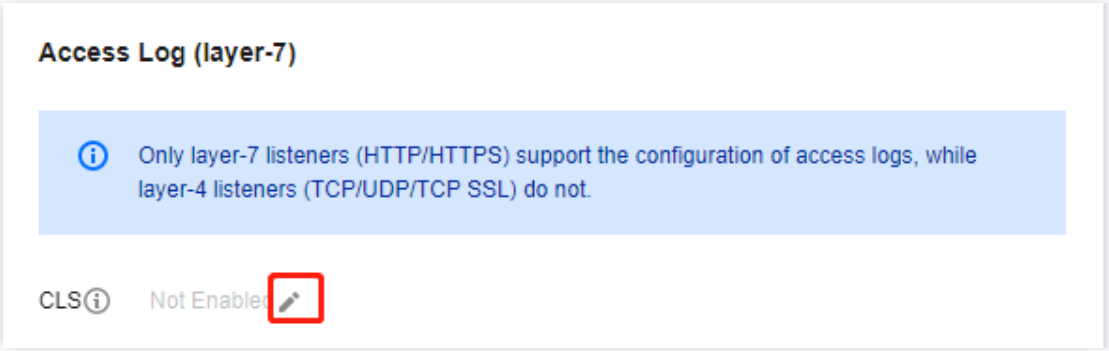
> Note:
>
> - Only the IPv4 version of the CLB service supports the configuration of access logs.
> - Only layer-7 CLB supports the configuration of access logs, while layer-4 CLB does not support it.
> - Both public and private network types of CLB support the configuration of access logs.

Step 1: Enable access log storage in CLS

1. Log in to the CLB console and click LB Instance List in the left sidebar.

2. On the LB Instance List page, click the target CLB ID.

3. In the Access Log (layer-7) module on the Basic Information page, click the pencil icon to enable the Cloud Log Service (CLS).



4. In the Authorization dialog box that pops up, click Go to Cloud Access Management, and then click Agree to Authorization.

5. After authorization is successful, return to the instance details page, click the pencil icon, and in the pop-up Modify CLS log storage location dialog box, turn on Enable log, select the logset and log topic for storing access logs, and click Submit. If you have not created a logset or log topic, create the relevant resources and then select a specific storage location.

> Note:
>
> It is recommended to select the log topic with the CLB logo under the clb_logset log set. The difference between a log topic with the CLB logo and a normal log topic is:
>
> - Log topics with the CLB logo are automatically indexed by default; ordinary log topics need to be indexed manually, otherwise search is not supported.
> - Log topics with the CLB logo support dashboards by default; ordinary log topics require manual configuration of dashboards.

6. After the configuration is completed, click the logset or log topic to jump to the search and analysis page of the CLS console.

7. (Optional) If you want to close the access log, click the pencil icon again, close it in the pop-up "Modify CLS log storage location" dialog box, and submit it.

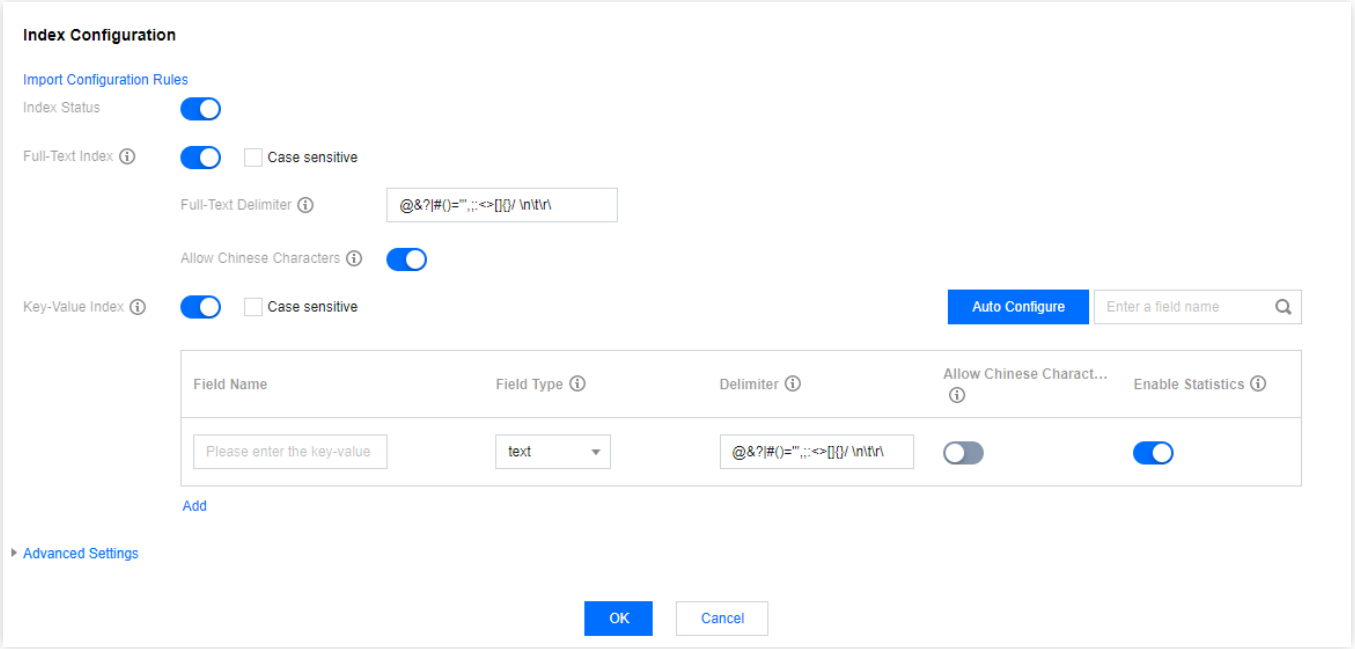
Step 2: Configure the log topic index

> Note:
>
> The log topic of the access log configured for a single instance must be indexed; otherwise, the log cannot be searched.

The recommended indexes are as follows:

Key-value index	Field Type	Word segmentation
server_addr	text	No need to configure word segmentation
server_name	text	No need to configure word segmentation
http_host	text	No need to configure word segmentation
status	long	-
vip_vpcid	long	-

1. Log in to the CLS console and click Log Topic in the left sidebar.
2. On the Log Topic page, click the target log topic ID.
3. On the log topic details page, click the Index Configuration tab and click Edit in the upper-right corner to add an index. After completing the configuration, click OK.



Step 3: View access logs

1. Log in to the CLS console and click Search and Analysis in the left sidebar.
2. On the Search and Analysis page, select the logset, log topic, and time range, and click Search and Analysis to search for access logs reported by CLB to CLS.

Cloud Access Management Overview

If you use CLB, CVM, database and other services that are managed by different people with your cloud account key, the following issues may occur:

- The risk of your key being compromised is high since multiple users are sharing it.
- The access permission of other users is not under control. They can introduce security risks caused by misoperations.
  CAM (CAM) is used to manage resource access permissions under cloud accounts. With CAM, you can control the operational permissions of sub-accounts through identity management and policy management.

For example, your account has multiple CLB instances deployed in different projects. To strengthen permission control and authorize resources, you can bind an authorization policy to the administrator of project A. The policy stipulates that only the administrator can operate the CLB resources under project A.

If you do not need to manage the access to CLB-related resources for your sub-accounts, you can skip this section. Skipping these sections does not affect your understanding and use of the rest of the documentation.

Basic Concepts of CAM

The root account authorizes sub-accounts by binding policies. The policy settings can be precise to the API, Resource, User/User Group, Allow/Deny, and Condition dimensionality.

1. Account
   - Root account: The basic entity for cloud resource ownership, resource usage metering and billing. You can log in to the Converge Cloud service.
   - Sub-account: An account created by the root account, has a certain identity ID and identity credentials, and can log in to the Converge Cloud console. The root account can create multiple sub-accounts (users). Sub-accounts do not own resources by default and must be authorized by their root account.
   - Identity credentials: Two types of identity credentials are available: login credentials and access certificates. Login credentials refer to the user login name and password, and access certificates refer to Cloud API keys (SecretId and SecretKey).
2. Resource and Permission
   - Resource: It is an object that is operated in cloud services, such as a CVM instance, a COS bucket, or a VPC instance.

- Permission: It is an authorization that allows or forbids users to perform certain operations. By default, a root account has full access to all resources under it, while a sub-account does not have access to any resources under its root account.
- Policy: It is a syntax rule that defines and describes one or more permissions. The root account performs authorization by associating policies with users/user groups.

For more information, see the CAM Overview.

Documentation

Target	Link
Understand the Relationship Between Policy and User	Policy management
Understand the Basic Policy Structure	Policy syntax

Authorization Definition

The types of CLB resources that can be authorized in CAM are

Resource Type	Resource Description Method in Authorization Policies
CLB Instance	qcs::clb:$region::clb/$loadbalancerid
CLB Listener	qcs::clb:$region::listener/$loadbalancerlistenerid
CLB backend server	qcs::cvm:$region:$account:instance/$cvminstanceid

Where:

- All $region should be the ID of a region and can be empty.
- All $account should be the AccountId of the resource owner, or "*".
- All $loadbalancerid should be the ID of a loadbalancer, or "*".

  And so on.

API in CAM that can authorize CLB

In CAM, the following actions can be authorized for a CLB resource.

Instance related

API Operations	Resource Description	API Description
DescribeLoadBalancers	Query the list of CLB instances	* authenticate only the API.
CreateLoadBalancer	Purchase a CLB	qcs:$projectid:clb:$region:$account:clb/*
DeleteLoadBalancers	Delete a CLB	qcs::clb:$region:$account:clb/$loadbalancerid
ModifyLoadBalancerAttributes	Modify CLB attribute information	qcs::clb:$region:$account:clb/$loadbalancerid
ModifyForwardLBName	Change the name of the CLB	qcs::clb:$region:$account:clb/$loadbalancerid

Listener related

API Operations	Resource Description	API Description
DeleteLoadBalancerListeners	Delete CLB listeners	qcs::clb:$region:$account:clb/$loadbalancerid``qcs::clb:$region:$account:listener/$loadbalancerlistenerid
DescribeLoadBalancerListeners	Get the list of CLB listeners	qcs::clb:$region:$account:clb/$loadbalancerid``qcs::clb:$region:$account:listener/*
ModifyLoadBalancerListener	Modify the CLB listener attributes	qcs::clb:$region:$account:clb/$loadbalancerid``qcs::clb:$region:$account:listener/$loadbalancerlistenerid
CreateLoadBalancerListeners	Creating a Load Balance Listener	qcs::clb:$region:$account:clb/$loadbalancerid
DeleteForwardLBListener	Delete CLB listeners (layer-4 and layer-7)	qcs::clb:$region:$account:clb/$loadbalancerid``qcs::clb:$region:$account:listener/$loadbalancerlistenerid
ModifyForwardLBSeventhListener	Modify the attributes of the CLB layer-7 listener	qcs::clb:$region:$account:clb/$loadbalancerid``qcs::clb:$region:$account:listener/$loadbalancerlistenerid
ModifyForwardLBFourthListener	Modify the attributes of the CLB layer-4 listener	qcs::clb:$region:$account:clb/$loadbalancerid``qcs::clb:$region:$account:listener/$loadbalancerlistenerid
DescribeForwardLBListeners	Query the CLB listener list	qcs::clb:$region:$account:clb/$loadbalancerid``qcs::clb:$region:$account:listener/*
CreateForwardLBSeventhLayerListeners	Create a layer-7 CLB listener	qcs::clb:$region:$account:clb/$loadbalancerid
CreateForwardLBFourthLayerListeners	Create a layer-4 CLB listener	qcs::clb:$region:$account:clb/$loadbalancerid

Load balancing domain name + URL related

API Operations	Resource Description	API Description
ModifyForwardLBRulesDomain	Modify the domain name of the CLB listener forwarding rule	qcs::clb:$region:$account:clb/$loadbalancerid``qcs::clb:$region:$account:listener/$loadbalancerlistenerid
CreateForwardLBListenerRules	Create a CLB listener forwarding rule	qcs::clb:$region:$account:clb/$loadbalancerid``qcs::clb:$region:$account:listener/$loadbalancerlistenerid
DeleteForwardLBListenerRules	Delete the layer-7 CLB listener rule	qcs::clb:$region:$account:clb/$loadbalancerid``qcs::clb:$region:$account:listener/$loadbalancerlistenerid
DeleteRewrite	Delete the redirection relationship between CLB forwarding rules	qcs::clb:$region:$account:clb/$loadbalancerid``qcs::clb:$region:$account:listener/$sourceloadbalancerlistenerid``qcs::clb:$region:$account:listener/$targetloadbalancerlistenerid
ManualRewrite	Manually add a redirection relationship for a CLB forwarding rule	qcs::clb:$region:$account:clb/$loadbalancerid``qcs::clb:$region:$account:listener/$sourceloadbalancerlistenerid``qcs::clb:$region:$account:listener/$targetloadbalancerlistenerid
AutoRewrite	Automatically generate redirection relationships for CLB forwarding rules	qcs::clb:$region:$account:clb/$loadbalancerid``qcs::clb:$region:$account:listener/$loadbalancerlistenerid

Backend server related

API Operations	Resource Description	API Description
ModifyLoadBalancerBackends	Modify the backend server weight of the CLB	qcs::clb:$region:$account:clb/$loadbalancerid``qcs::cvm:$region:$account:instance/$loadbalancerlistenerid

API Operations	Resource Description	API Description
DescribeLoadBalancerBackends	Get the list of backend servers bound to the CLB	qcs::clb:$region:$account:clb/$loadbalancerid``qcs::clb:$region:$account:listener/*
DeregisterInstancesFromLoadBalancer	Unbind the backend server	qcs::clb:$region:$account:clb/$loadbalancerid``qcs::cvm:$region:$account:instance/$cvminstanceid
RegisterInstancesWithLoadBalancer	Bind the backend server to the CLB	qcs::clb:$region:$account:clb/$loadbalancerid``qcs::cvm:$region:$account:instance/$cvminstanceid
DescribeLBHealthStatus	Query the health status of the CLB	qcs::clb:$region:$account:clb/$loadbalancerid``qcs::clb:$region:$account:listener/*
ModifyForwardFourthBackendsPort	Modify the port of the CVM in the forwarding rule of the layer-4 listener	qcs::clb:$region:$account:clb/$loadbalancerid``qcs::clb:$region:$account:listener/$loadbalancerlistenerid``qcs::cvm:$region:$account:instance/$cvminstanceid
ModifyForwardFourthBackendsWeight	Modify the weight of the CVM in the forwarding rule of the layer-4 listener	qcs::clb:$region:$account:clb/$loadbalancerid``qcs::clb:$region:$account:listener/$loadbalancerlistenerid``qcs::cvm:$region:$account:instance/$cvminstanceid
RegisterInstancesWithForwardLBSeventhListener	Bind the CVM to the forwarding rules of the CLB layer-7 listener	qcs::clb:$region:$account:clb/$loadbalancerid``qcs::clb:$region:$account:listener/$loadbalancerlistenerid``qcs::cvm:$region:$account:instance/$cvminstanceid

API Operations	Resource Description	API Description
RegisterInstancesWithForwardLBFourthListener	Bind the CVM to the forwarding rules of the CLB layer-4 listener	qcs::clb:$region:$account:clb/$loadbalancerid``qcs::clb:$region:$account:listener/$loadbalancerlistenerid``qcs::cvm:$region:$account:instance/$cvminstanceid
DeregisterInstancesFromForwardLBFourthListener	Unbind the CVM from the forwarding rule of the CLB layer-4 listener	qcs::clb:$region:$account:clb/$loadbalancerid``qcs::clb:$region:$account:listener/$loadbalancerlistenerid``qcs::cvm:$region:$account:instance/$cvminstanceid
DeregisterInstancesFromForwardLB	Unbind the CVM from the forwarding rule of the CLB layer-7 listener	qcs::clb:$region:$account:clb/$loadbalancerid``qcs::clb:$region:$account:listener/$loadbalancerlistenerid``qcs::cvm:$region:$account:instance/$cvminstanceid
ModifyForwardSeventhBackends	Modify the weight of the CVM in the forwarding rule of the layer-7 listener	qcs::clb:$region:$account:clb/$loadbalancerid``qcs::cvm:$region:$account:instance/$cvminstanceid
ModifyForwardSeventhBackendsPort	Modify the port of the CVM in the forwarding rule of the layer-7 listener	qcs::clb:$region:$account:clb/$loadbalancerid``qcs::clb:$region:$account:listener/$loadbalancerlistenerid``qcs::cvm:$region:$account:instance/$cvminstanceid
DescribeForwardLBBackends	Query the CLB CVM list	qcs::clb:$region:$account:clb/$loadbalancerid``qcs::cvm:$region:$account:instance/*

API Operations	Resource Description	API Description
DescribeForwardLBHealthStatus	Query the health check status of the CLB	qcs::clb:$region:$account:clb/*
ModifyLoadBalancerRulesProbe	Modify the health check and forwarding path of the CLB listener forwarding rule	qcs::clb:$region:$account:clb/$loadbalancerid``qcs::clb:$region:$account:listener/$loadbalancerlistenerid

Policy Examples

Full Read-Write Policy of CLB

- Grant a sub-account full management permission (including creation and management) for the CLB service.

- Policy Name: CLB ResourceFullAccess

```
{
  "version": "2.0",
  "statement": [{
      "action": [
          "name/clb:*"
      ],
      "resource": "*",
      "effect": "allow"
  }]
}
```

Read-Only Policy of CLB

- Grant a sub-account read-only access to the CLB (that is, the ability to view all resources under all CLBs), but the sub-account cannot create, update, or delete them. In the console, the premise of operating a resource is that it can be viewed, so it is recommended you open CLB full read permissions for sub-accounts.

- Policy Name: CLB ResourceReadOnlyAccess

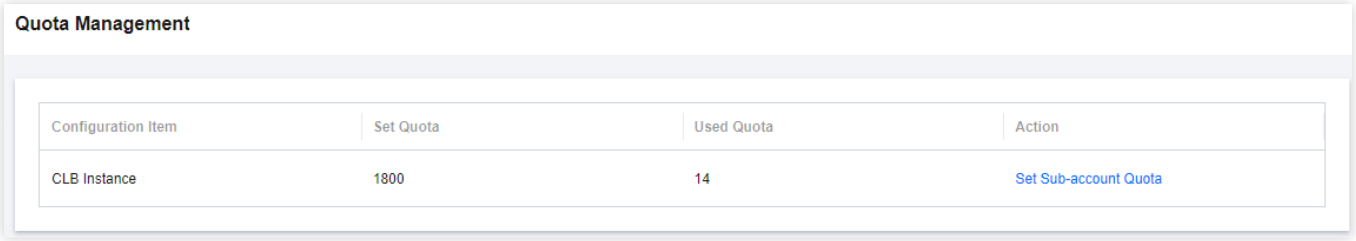
```
{
  "version": "2.0",
  "statement": [{
      "action": [
          "name/clb:Describe*"
      ],
      "resource": "*",
      "effect": "allow"
  }]
}
```

Quota Management

CLB supports the root account to manage sub-account quotas and adjust sub-account quotas. Sub-accounts can only view quotas.

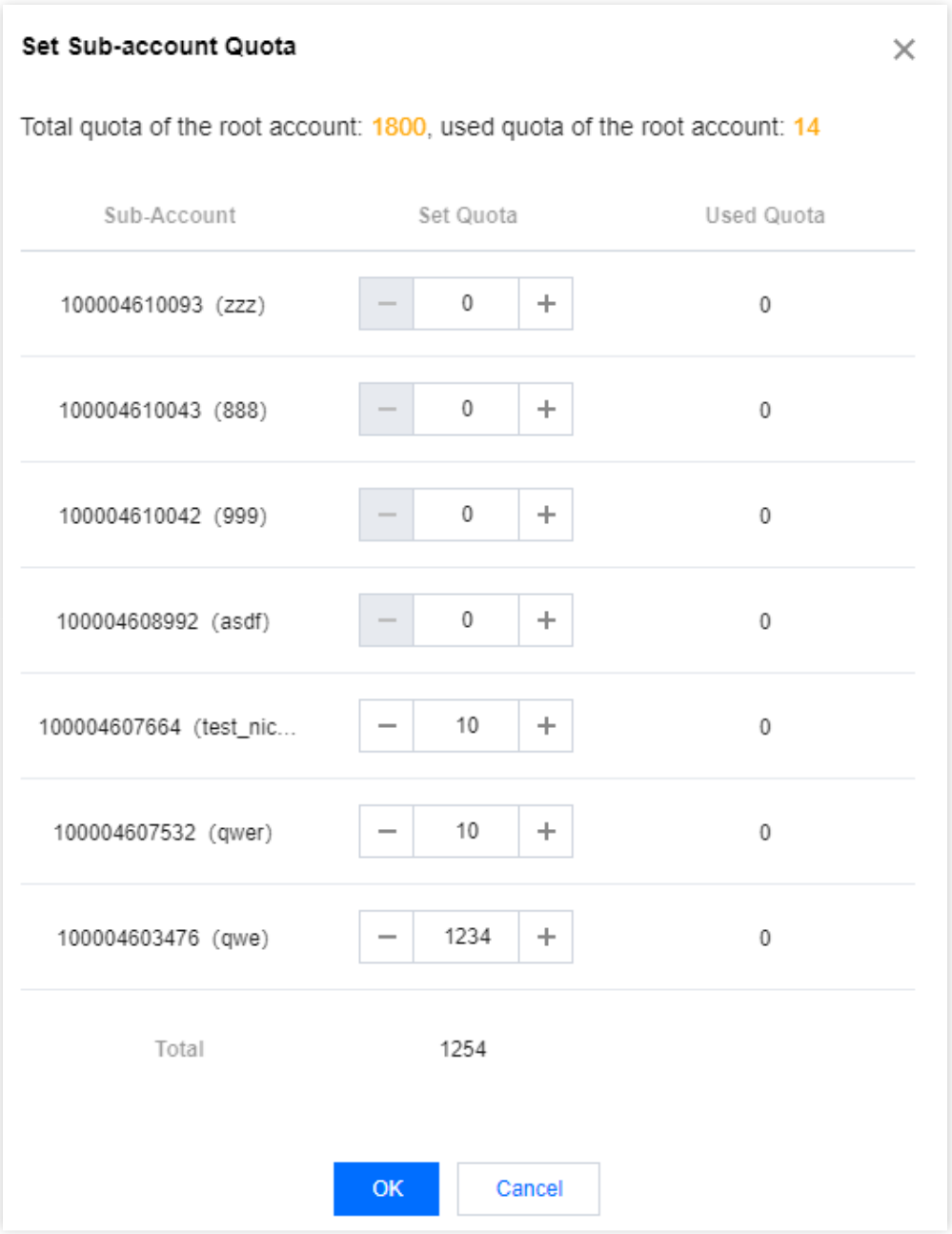
Viewing Sub-account Quotas

1. Log in to the CLB console.
2. In the left sidebar, click Quota Management to view the quota of the sub-account.



Managing Sub-account Quotas

1. Use the root account to log in to the CLB console.
2. In the left sidebar, click Quota Management.
3. On the Quota Management page, click Set Sub-Account Quota in the Action column on the right.
4. In the Set Sub-account Quota dialog box that pops up, set the quota for the sub-account and click OK.


Set Sub-account Quota
Total quota of the root account: 1800, used quota of the root account: 14
Sub-Account
Set Quota
Used Quota
100004610093 (zzz)
0
0
100004610043 (888)
0
0
100004610042 (999)
0
0
100004608992 (asdf)
0
0
100004607664 (test_nic...
10
0
100004607532 (qwer)
10
0
100004603476 (qwe)
1234
0
Total
1254
OK
Cancel

Best Practice
Method Description of CLB Enabling Gzip Configuration and Detection

In Public Network CLB instance, HTTP/HTTPS protocol supports users to enable the gzip compression feature by default. Enabling the gzip feature to compress web pages can effectively reduce the amount of data transmitted over the network and increase the access speed of client browsers. During use, pay attention to the following matters:

Notes

- GZIP support should also be enabled for the backend CVM instances.
  For common Nginx service containers, GZIP must be enabled in the configuration file (nginx.conf by default) and the service must be restarted.

  ```
  gzip on;
  ```

- The file types currently supported by the CLB are as follows. You can specify the file type for compression in the gzip_types configuration item

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

> Description:
>
> The GZIP support for the above file types must be enabled in the CLB backend CVM business software.

- The client request must include a compression request header.
  For compression to be enabled, the client request must carry the following header:

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

Sample of Backend CVM Enabling GZIP Process Support

Sample of CVM Running Environment: Debian 6

1. Use vim to open the Nginx configuration file according to the user path:

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

2. Find the following code:

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

The syntax details of the above code is as follows:

- gzip: Enable or disable the gzip module.
  Syntax: gzip on/off
  Scope: http, server, location
- gzip_min_length: specifies the minimum number of bytes for a page to be eligible for compression. The page's byte size is obtained from the Content-Length header. The default value is 1k.
  Syntax: gzip_min_length length
  Scope: http, server, location
- gzip_buffers: specifies the number of units of memory that the system allocates to store the compressed data stream for GZIP. If the size is set to 16k, memory will be allocated in 16k units, with 4 times the original data size being requested.
  Syntax: gzip_buffers number size
  Scope: http, server, location
- gzip_http_version: specifies the minimum HTTP version required to use GZIP. Setting this parameter to HTTP/1.0 means that GZIP compression can be used for HTTP 1.0 and later versions, thus ensuring upward compatibility with HTTP/1.1. Since Converge Cloud now fully supports HTTP/1.1 across the network; no changes are needed.
  Syntax: gzip_http_version 1.0 | 1.1;
  Scope: http, server, location
- gzip_comp_level: specifies the GZIP compression level, ranging from 1 to 9. A level of 1 provides the lowest compression ratio and the fastest processing speed, while a level of 9 provides the highest compression ratio and the slowest processing speed (faster transmission, but more CPU-intensive).
  Syntax: gzip_comp_level 1..9
  Scope: http, server, location
- gzip_types: Match the MIME type for compression. By default, the text/html type is compressed. In

addition, gzip under Nginx does not compress static resource files such as javascript and images by default. You can specify the MIME type that needs to be compressed through gzip_types. If it is not a set value, no compression is performed. For example, if you need to compress json format data, you need to add application/json type data in this statement.
The supported types are as follows:

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

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

If you have modified the configuration, save the file and exit, then go to the Nginx bin file directory and execute the following command to reload Nginx:

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

Use the curl command to test whether gzip is successfully enabled:

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

Getting Started Guide for HTTPS Forwarding Configuration

CLB Capability Description

The CLB has achieved a significant improvement in HTTPS performance through deep optimization of the protocol stack and server. At the same time, we have greatly reduced the cost of certificates through international cooperation on certificates. CLB can bring significant benefits to the business in the following aspects:

1. Using HTTPS does not slow down client access.
2. The SSL encryption and decryption performance of a single server in the cluster is up to 65,000 cps for a full handshake. Compared with high-performance CPUs, it has been improved by at least 3.5 times, saving server costs, greatly improving business operations and service capabilities during sudden traffic increases, and enhancing computing-based anti-attack capabilities.
3. Supports multiple protocol offloading and conversion. Reduce the pressure on the business to adapt to various client protocols. The business backend only needs to support HTTP1.1 to use various versions of protocols such as HTTP2, SPDY, SSL3.0 and TLS1.2.
4. One-stop SSL certificate application, monitoring, and replacement. We have started dialogues with international certificate vendors Comodo and Symantec to explore cooperation and significantly reduce the certificate application process and costs.
5. Anti-CC and WAF features. It can effectively prevent slow connections, high-frequency targeted attacks, SQL injections, web page Trojans and other application layer attacks.

HTTP, HTTPS Header Identifier

CLB proxies HTTPS. Whether it is an HTTP or HTTPS request, when CLB forwards it to the backend CVM, it is an HTTP request. At this time, developers cannot tell whether the front-end request is HTTP or HTTPS.

When CLB forwards the request to the backend CVM, the header will be embedded with X-Client-Proto:

X-Client-Proto: http (front-end is HTTP request)
X-Client-Proto: https (front-end is HTTPS request)

Getting Started

Assume that the user needs to configure the website https://example.com . Developers want users to type

www.example.com directly when they enter a URL in a browser. It can be accessed securely via HTTPS protocol.

At this time, the www.example.com request forwarding process entered by the user is as follows:

1. The request is transmitted via HTTP protocol, accesses port 80 of the CLB listener through VIP, and is forwarded to port 8080 of the backend CVM.
2. By configuring the rewrite operation on the backend server's Nginx, the request goes through port 8080 and is rewritten to https://example.com page.
3. At this time, the browser sends https://example.com again to the corresponding HTTPS zone. The request accesses port 443 of the CLB listener through VIP and is forwarded to port 80 of the backend CVM.
   At this point, the request forwarding is completed.

This operation rewrites the user's HTTP request into a more secure HTTPS request without the browser user's awareness. To implement the above request forwarding operation, users can configure the backend server as follows:

```
server {

    listen 8080;
    server_name www.example.com;

    location / {

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

}
}
```

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

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

server {
    listen      443 ssl;
   server_name    www.example.com;
```

```
    [....]
}
```

How To Obtain the Client's Real IP?

CLB's layer-4 (TCP/UDP) and layer-7 (HTTP/HTTPS) services both support obtaining the client's real IP address directly on the backend CVM without requiring additional configuration.

- For layer-4 CLB, the source IP obtained on the backend CVM is the client IP.
- For layer-7 CLB, you can use X-Forwarded-For or remote_addr field to directly obtain the client IP.

> Note:
> For CLB, no additional configuration is required on the backend CVM to obtain the client IP.
> For other layer-7CLB services that have SNAT, you need to configure it on the backend CVM and then use X-Forwarded-For to obtain the client's real IP address.

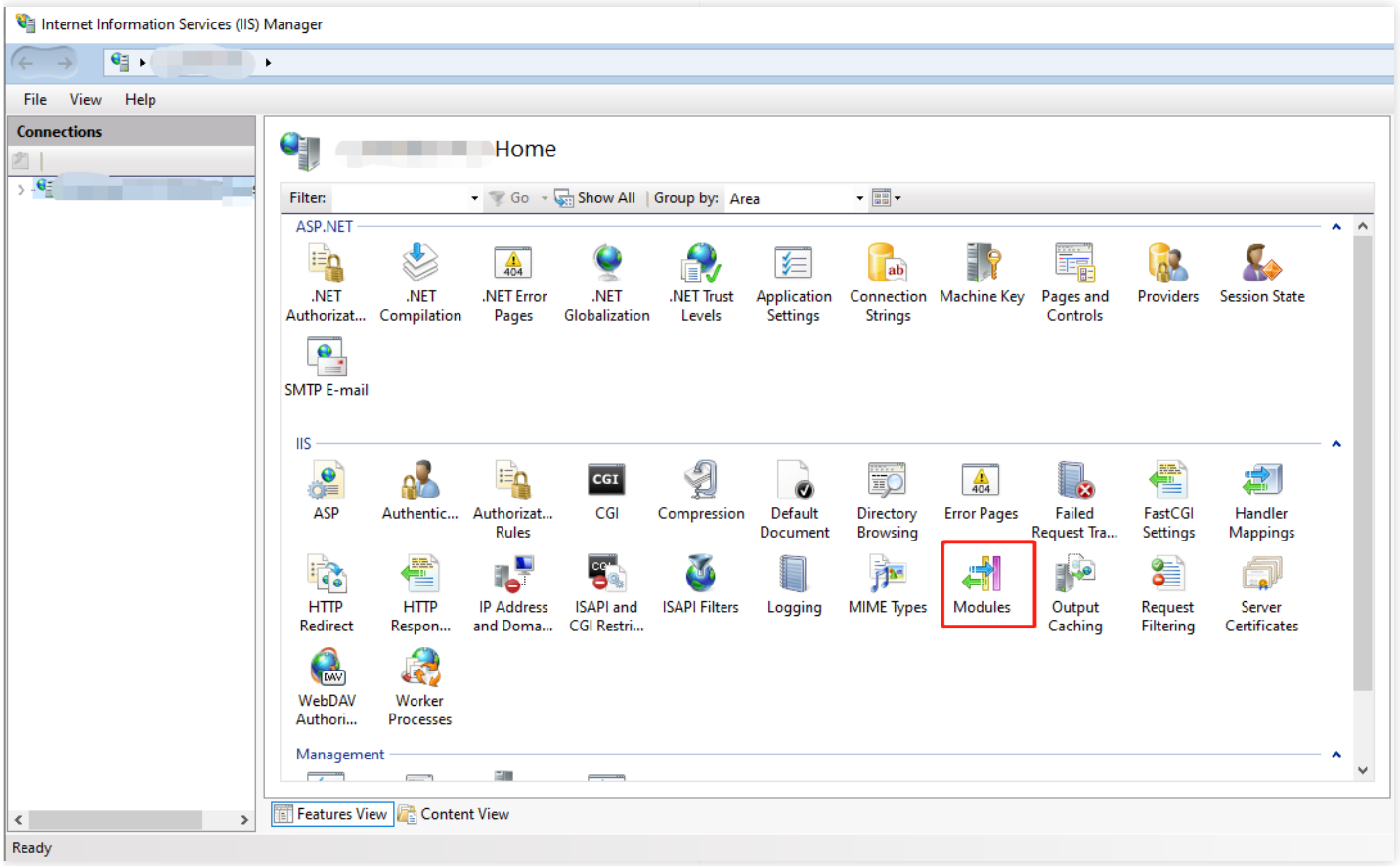
The following section introduces common application server configuration solutions.

IIS 6 Configuration Scheme

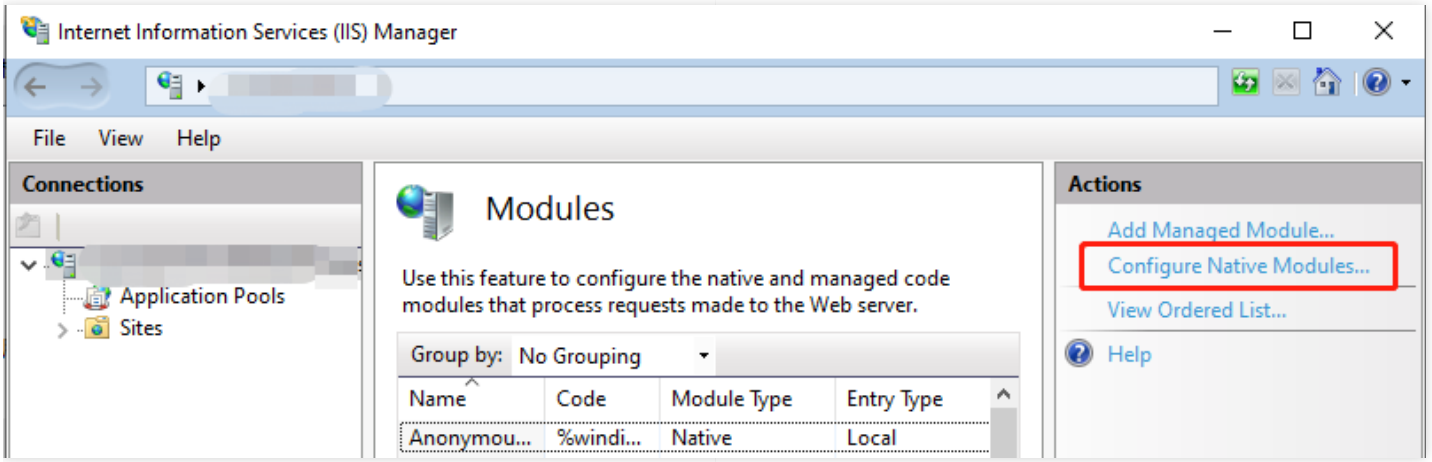
1. Download and install the plugin F5XForwardedFor module, copy F5XForwardedFor. dll from x86\ Release or x64\ Release directory to a directory, assuming C:\ISAPIFilters , depending on your server operating system version, and ensure that the IIS process has read access to that directory.
2. Open IIS Manager, find the currently open website, right-click on the website, and select Attribute to open the attribute page.
3. On the property page, switch to ISAPI Filter, click Add, and the Add window pops up.
4. In the Add window, fill in F5XForwardedFor in Filter Name and fill in the full path of F5XForwardedFor.dll in Executable File and click OK.
5. Restart the IIS server and wait for the configuration to take effect.

IIS 7 Configuration Scheme

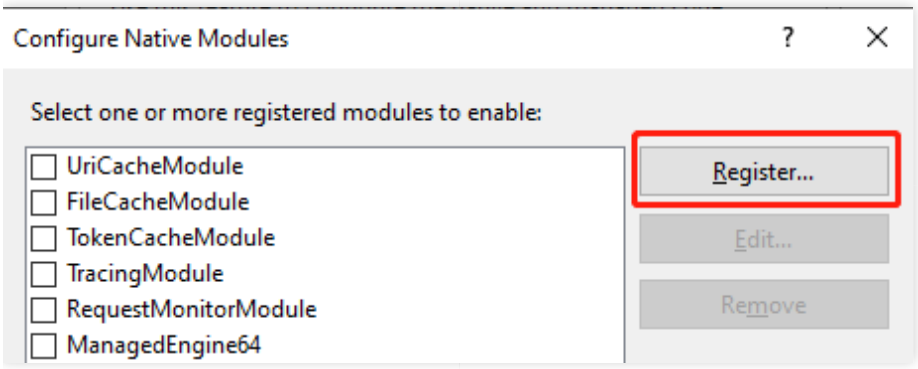
1. Download and install the plugin F5XForwardedFor module, copy F5XFFHttpModule. dll and F5XFFHttpModule. ini in the x86\ Release or x64\ Release directory to a directory, assuming C: \F5XForwardedFor , according to your server operating system version, and ensure that the IIS process has read access to the directory.
2. Select IIS Server and double-click the Module feature.



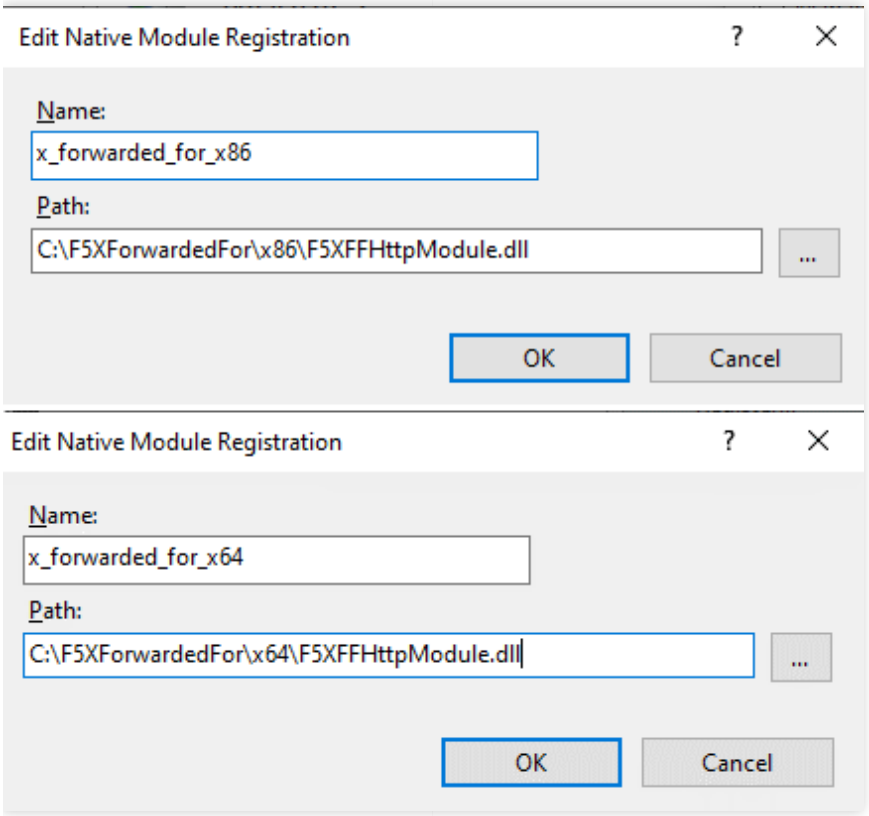
3. Click Configure Native Module.



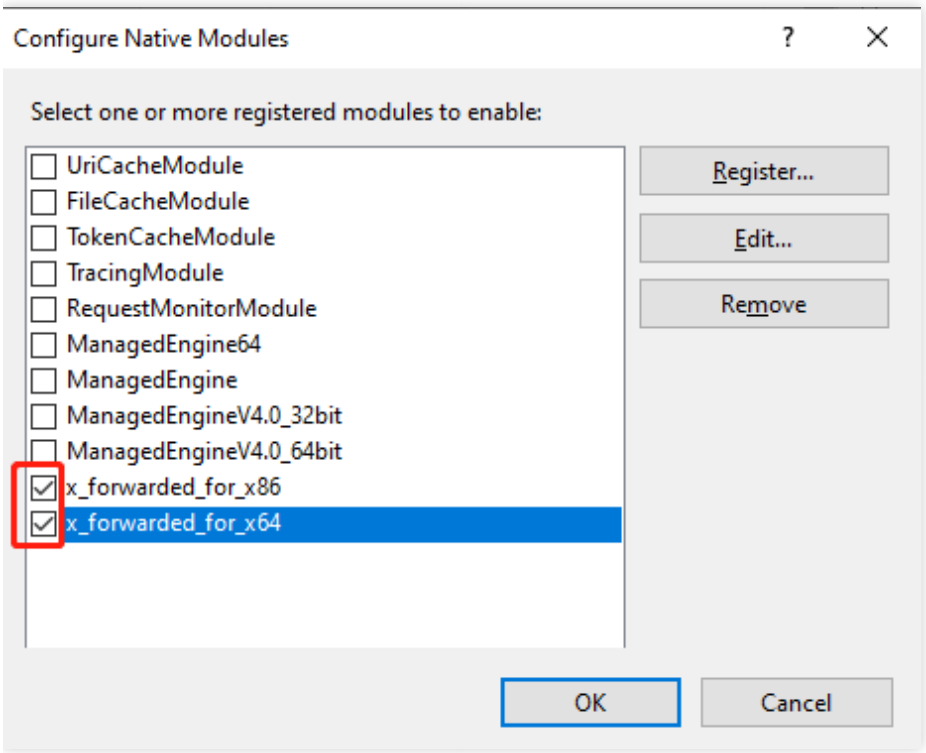
4. Click on Register in the pop-up window.



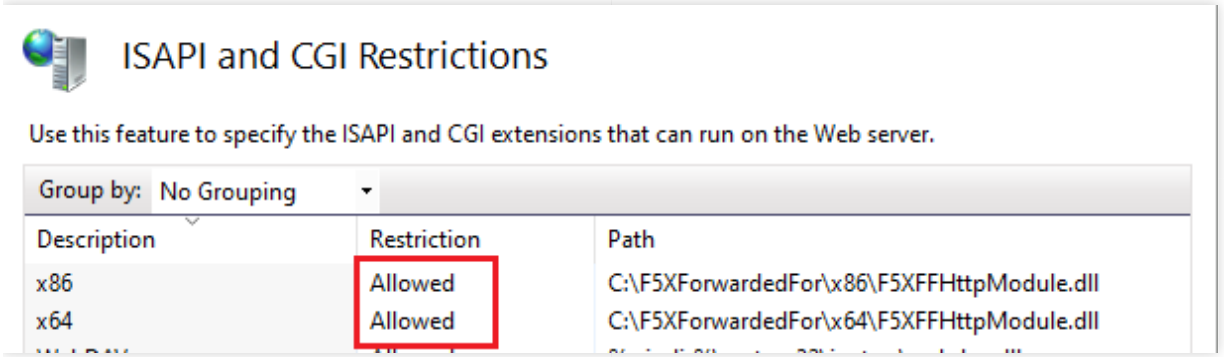
5. Add the downloaded DLL file as shown below:



6. After addition, check and click OK.



7. Add the two DLLs above in ISAPI and CGI Restrictions and set the restrictions to allow.



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

Apache Configuration Solution

1. Install the Apache third-party module mod_rpaf.

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

2. Modify Apache configuration /etc/httpd/conf/httpd.conf , add at the end:

```
LoadModule rpaf_module modules/mod_rpaf-2.0.so
RPAFenable On
RPAFsethostname On
RPAFproxy_ips IP address (This IP address is not the public IP provided by the CLB. You can check the Apache log for the specific IP address. Usually there are two IP addresses that need to be wri tten down.)
RPAFheader X-Forwarded-For
```

3. After addition, restart Apache.

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

Nginx configuration solution

1. When Nginx is used as a server, http_realip_module is used to obtain the real IP address of the client. The default installed Nginx does not have this module installed. You need to recompile Nginx and add --with-http_realip_module.

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

2. Modify nginx.conf.

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

Modify the following parts:

```
fastcgi connect_timeout 300;
fastcgi send_timeout 300;
fastcgi read_timeout 300;
fastcgi buffer_size 64k;
```

```
fastcgi buffers 4 64k;
fastcgi busy_buffers_size 128k;
fastcgi temp_file_write_size 128k;

set_real_ip_from IP address; (This IP address is not the public IP provided by the CLB. You can check t
real_ip_header X-Forwarded-For;
```

3. Restart Nginx.

   ```
   service nginx restart
   ```

Instructions on Multi Availability Zone High Availability Config

CLB Multi Availability Zone High Availability

CLB supports multi-availability zone disaster recovery capabilities for IPv4 private and public networks and EIPs. For example, in SZ Zone 1 and Financial District Two availability zones (within the same region), multiple clusters will be deployed separately to achieve cross-availability zone disaster recovery within the same region. This feature ensures that when an entire availability zone fails, the CLB can switch frontend access traffic to another availability zone within the same region within 10 seconds to restore service capability.

Specific Cases and Questions

Question 1: There are CLB test1 in SZ Financial District A and B. What is the policy for client public network inbound traffic?

In the IDCs of SZ District 1 and District 2, there is a pair of IP resource pools, which can be understood as equal IP resources. Developers do not need to understand which of zone 1 and zone 2 is the primary cluster and which is the replica cluster. The two clusters have the same load capacity. When developers purchase a CLB and bind it to a CVM, two sets of rules are generated and written into two clusters, and high availability is now achieved.

Question 2: There are CLBs test1 in SZ Financial District A and B, and the backend is bound to 100 servers in each availability zone A and B. During business operations, 1 million HTTP persistent connections are established (TCP connections are not closed). At this time, when the CLB cluster of Financial District A is completely paralyzed and unavailable, how do the businesses look like?

When the CLB in Financial District A loses its service capability, all current persistent connections are disconnected, but non-persistent connections are not affected. The disaster recovery architecture will automatically bind all 100 servers in zones A/B to the lCLB in financial zone B within 10 seconds, and business capabilities will be restored immediately without manual intervention.

Question 3: What types of CLBs are supported by the multi-availability zone disaster recovery capability?

IPv4 private and public network CLB and EIP support Cross-AZ Disaster Recovery.

SSL certificate format requirements and format conversion instructions

Certificate Format Instructions

Certificate Format Requirements

- The certificate that the user needs to apply for is: a certificate in PEM format under Linux environment. CLB does not support certificates in other formats. For certificates in other formats, see the Instructions on Converting Certificate Format to PEM below.
- If the certificate is issued by a root CA agency, the certificate you get is the only one and no additional certificates are required. The configured site can be considered trustworthy by browsers and other access devices.
- If the certificate is issued by an intermediate CA agency, the certificate file you get contains multiple certificates, and you need to manually merge the server certificate and the intermediate certificate and upload them together.
- If your certificate has a certificate chain, convert the certificate chain content into PEM format and upload it together with the certificate content.
- The splicing rule is: put the server certificate in the first copy, and the intermediate certificate in the second copy, with no blank lines in between. > Note: Generally, the institution has corresponding instructions when they issue the certificate. Pay attention to checking them.

Certificate format and certificate chain format examples

The following are examples of the certificate format and certificate chain format. Confirm that the format is correct before uploading:

1. Certificate issued by the root CA agency: The certificate format is PEM format in the Linux environment. For example:



The certificate rules are:

- [—-BEGIN CERTIFICATE—-, —-END CERTIFICATE—-] Beginning and ending; upload these contents together.
- Each line has 64 characters, and the last line has no more than 64 characters.

2. For certificates issued by an intermediate CA, the certificate chain typically looks like this:

———-BEGIN CERTIFICATE———-

———-END CERTIFICATE———-

———-BEGIN CERTIFICATE———-

———-END CERTIFICATE———-

———-BEGIN CERTIFICATE———-

———-END CERTIFICATE———-

The certificate chain rules are:

- There cannot be blank lines between certificates.
- Each certificate complies with the above 1.1 Certificate format requirements.

2. RSA Private Key Format Requirements

For example:



RSA private keys can include all private keys (RSA and DSA), public keys (RSA and DSA), and (x509) certificates. It stores DER format data encoded in Base64, surrounded by an ASCII header, thus suitable for text-mode transmission between systems.

RSA private key rules:

- [—-BEGIN RSA PRIVATE KEY—-, —-END RSA PRIVATE KEY—-] Beginning and ending; upload these contents together.
- Each line has 64 characters. The last line can be less than 64 characters.
  If you do not generate a usable private key in the format of [———-BEGIN PRIVATE KEY———-, ———-END PRIVATE KEY———-] according to the above scheme, you can convert it into a usable private key as follows:

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

Then upload the contents of new_server_key.pem along with the certificate.

Certificate Conversion to PEM Format Instructions

Currently, the CLB only supports certificates in PEM format. Certificates in other formats need to be converted to PEM format before they can be uploaded to the CLB. It is recommended to use the openssl tool for conversion. Below are several popular methods for converting popular certificate formats to PEM

format.

Converting certificate format from DER to PEM

The DER format is commonly found in the Java platform.

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

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

Convert P7B Format Certificates to PEM Format

The P7B format is commonly found in Windows Server and tomcat.

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

Get the content [——-BEGIN CERTIFICATE——-, ——-END CERTIFICATE——-] in outcertificat.cer and upload it as the certificate.

Private key conversion: Private keys can usually be exported in the IIS server.

Convert PFX Format Certificates to PEM Format

The PFX format is commonly found in Windows Server.

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

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

Convert CER/CRT Format Certificates to PEM Format

For certificates in CER/CRT format, you can convert them by directly changing the certificate file extension. For example, you can simply rename the "servertest.crt" certificate file to "servertest.pem".

Balancing Algorithm Selection and Weight Configuration Instance

Comparative Analysis of CLB Algorithms

Weighted Round-Robin Scheduling

- Principle
  The round-robin scheduling algorithm allocates requests to different servers in a round-robin method. Specifically, it selects server i by executing i = (i + 1) mod n for each scheduling operation. The weighted round-robin scheduling algorithm addresses the issue of varying server performance by assigning a weight to each server to represent its processing performance. Requests are allocated based on these weights and in a round-robin method. Servers with higher weights receive new connections first than those with lower weights, servers with higher weights get more connections than those with lower weights, and servers with equal weights get equal connections.
- Advantages
  The algorithm excels in simplicity and practicality. As it does not need to track the status of all current connections, it operates as a stateless scheduling mechanism.
- Disadvantages
  The weighted round-robin scheduling algorithm is relatively simple, but it is not suitable for situations where the request service time varies greatly or the time consumed by each request is inconsistent. In this case, the round-robin scheduling algorithm is likely to cause a load imbalance between servers.
- Applicable scenarios
  Load balancing performs best when the backend processing time for each request is roughly the same. It is often used for non-persistent connection services, such as HTTP.
- User recommendations
  When users know that the backend processing time for each request is roughly the same, such as when the RS processes requests of the same or similar types, it is recommended to choose the weighted round-robin method. The weighted round-robin method is also recommended when request times differ slightly, because this implementation method has low overhead, does not require traversal, and is highly efficient.

Weighted Least-Connection Scheduling

- Principle
  In real-world scenarios, the duration for which each client request resides on a server may vary significantly. Over time, using simple round-robin or random load balancing algorithms may result in

a highly uneven number of connection processes across servers, which does not achieve true load balancing. The least-connection scheduling algorithm dynamically estimates the load on a server based on the number of currently active connections. Contrary to the round-robin scheduling algorithm, the scheduler needs to keep track of the number of established connections on each server. When a request is scheduled to a server, its connection number is incremented by 1. When a connection terminates or times out, its connection number is decremented by 1. The weighted least-connection scheduling algorithm is an improvement upon the basic least-connection scheduling algorithm. It assigns different weights to servers based on their processing capabilities, allowing each server to handle a proportional number of requests according to its assigned weight.

1. Assume that the weight of each RS is WI, the current number of connections is CI and calculate CI/WI in turn. The RS with the smallest value is the next RS to be allocated.
2. If there are RS with the same CI/WI, these RS are scheduled using weighted round-robin scheduling.

- Advantages
  This balancing algorithm is suitable for requests that require prolonged processing time, such as FTP applications.
- Disadvantages
  Due to API limitations, the least connection and session persistence features cannot be enabled at the same time.
- Applicable scenarios
  This algorithm is applicable to scenarios where the backend processing time for each request differs significantly. It is commonly used for persistent connection services.
- User recommendations
  If users need to process different requests where the backend processing time varies significantly (differing by orders of magnitude, such as 3 ms versus 3s), it is recommended to use the weighted least-connection scheduling algorithm for load balancing.

Source Address Hash Scheduling Algorithm ip_hash

- Principle
  This algorithm uses the source IP address of the request as the hash key to look up the corresponding server from a statically allocated hash table. If the server is available and not overloaded, the request is sent to that server. Otherwise, null is returned.
- Advantages
  ip_hash can achieve partial session persistence by remembering the source IP address, ensuring that requests from a particular client are consistently mapped to the same RS through the hash table. Therefore, ip_hash can be used for scheduling in scenarios where session persistence is not supported.
- User recommendations

Perform a hash operation on the source address of the request and dispatch the request to a matching server based on the backend server's weight. This ensures that requests from the same client IP address are always dispatched to a specific server. This method is suitable for TCP protocols without the cookie feature in CLB.

Balancing Algorithm Selection and Weight Configuration Instance

Among the new features that will be released soon in CLB, layer-7 forwarding will support the balancing method with the least connections. In order to enable the RS cluster to stably undertake business in different cases, we provide several examples of CLB selection and weight configuration for users' reference.

- Scenario 1
  Suppose that there are three RSs with the same configuration (CPU/memory). Since their performance is the same, the user can set the weight for each RS to 10. Assume that each RS has established 100 TCP connections with the client and a new RS is added. In this scenario, it is recommended to use the least-connection load balancing method. This configuration can quickly increase the load on the fourth RS and reduce the pressure on the other three RSs.
- Scenario 2
  Suppose that a user is new to cloud services and the website has been running for a short time with low traffic. It is recommended to purchase RSs with the same configuration, as they serve as identical access layer servers. In this scenario, the user can set the weight for each RS server to 10 and use the weighted round-robin load balancing method for traffic distribution.
- Scenario 3
  A user has five servers that are used to host a simple static website, and the computing power of the five servers is in the ratio of 9:3:3:3:1 (based on CPU and memory). In this scenario, the user can set the weight of the RSs in the ratio of 90, 30, 30, 30, and 10. Since static website access mostly involves non-persistent connection requests, the weighted round-robin load balancing method can be used to allow the CLB instance to allocate requests among the RSs according to their performance ratio.
- Scenario 4
  A user has 10 RSs that are used to handle massive web access requests, and does not want to purchase additional RSs to increase expenditure. However, a specific RS often experiences high load, causing it to restart frequently. In this scenario, it is recommended to set weights for the RSs based on their performance. Set a lower weight for an RS with a high load. Additionally, the least-connection load balancing method can be used to allocate requests to the RSs with fewer active connections, thus resolving the issue of high load on specific RSs.
- Scenario 5
  A user has three RSs that are used to handle several persistent connection requests, and the

computing power of these servers is in the ratio of 3:1:1 (based on CPU and memory). In this case, the best-performing server handles more requests. The user does not want to overload this server and wants new requests to be distributed to the idle servers. In this scenario, the least-connection load balancing method can be used, and the weights of busier servers can be appropriately reduced. This allows the CLB instance to allocate requests to the RSs with fewer active connections, achieving load balancing.

- Scenario 6
  A user wants subsequent client requests to be allocated to the same server. However, using the weighted round-robin or weighted least-connection scheduling method cannot guarantee that requests from the same client will always go to the same server. To meet the requirements of specific application servers and ensure that client sessions are "sticky" or "persistent", the user can use the ip_hash load balancing method for traffic distribution in this scenario. This method ensures that requests from the same client are always directed to the same RS (except when the number of servers changes or the server becomes unavailable).

Purchase Guide
Billing Overview

This document gives a full overview of the billing options for Converge Cloud Load Balancer (CLB) instances. CLB is a cloud service that distributes traffic across multiple servers to improve application performance and avoid single points of failure. By intelligently balancing the load, CLB ensures that your application remains fast, stable, and highly available, even during periods of high demand. It uses a cluster-based setup to achieve up to 99.95% availability.

Key Advantages of Converge Cloud Load Balancer (CLB)

1. High Performance - With the ability to support hundreds of millions of concurrent connections and handle millions of requests per second, a CLB cluster ensures performance at scale—perfect for businesses running e-commerce, social networking, or gaming platforms with over 10 million daily visits.

2. High Availability - CLB is built on a high-availability cluster model, providing up to 99.95% uptime. Even with just one physical server remaining, it can handle tens of millions of connections. The system continuously monitors health status, removing faulty components and keeping the rest stable to ensure consistent service.

3. High Elasticity - CLB enables elastic scaling of your application by adjusting resources in real time through Auto Scaling, which adds or removes CVMs based on load. With built-in monitoring and accurate per-second billing, you can run efficiently without manual effort or over-provisioning—minimizing waste and maximizing value.

4. High Security and Stability - CLB includes built-in protection against common network threats like DDoS and web attacks, powered by the BGP Anti-DDoS system. It filters malicious traffic within seconds, helping prevent IP blocking and bandwidth overload.
   A built-in synproxy mechanism that protects backend servers from crashes during attacks, ensuring service stability. All public CLB instances come with Anti-DDoS Basic by default, with options to upgrade to Pro, Advanced, or add Web Application Firewall (WAF) for stronger protection.

5. Low Costs - CLB helps reduce costs and complexity by removing the need for physical load balancers and routine manual tasks—cutting hardware and labor expenses by up to 99%. Choose from flexible billing options based on your usage.

Billing Items

Converge CLB currently provides a Shared Instance that supports self-service management, automated fault recovery, and protection against network attacks. It is suitable for use cases such as enterprises, residential communities, e-commerce platforms, and online games. The service charges for CLB consist of three components: the instance fee, the network fee, and the cross-region binding fee.

bill-by-IP accounts

Instance Type	Network Type	Instance Type	Public Network Fee	Cross-Region Binding Fee
Shared	Public Network	Yes	Yes	Optional
	Private Network		No	

Optimizing Performance with Converge CLB Shared Instance

🛠️ Simple Configuration
Easy to set up and manage, making it ideal for beginners and small to medium-sized deployments.

Cost-Effective
You share the load balancing infrastructure with other users, which helps reduce costs without compromising performance.

Common Use Cases
Well-suited for standard web applications, e-commerce platforms, company websites, and internal tools.

Secure and Isolated
Although resources are shared, your traffic and data remain secure and logically isolated from others.

Billing Modes

Standard Account Type (bill-by-IP accounts)

In this account type, bandwidth and traffic are assigned to public IP addresses or Cloud Load Balancers (CLBs), rather than directly to Cloud Virtual Machines (CVMs). As a result, CVMs do not have direct access to public network bandwidth or traffic resources.

Billing Mode	Network Billing Mode	Best For

Billing Mode	Network Billing Mode	Best For
Pay-As-You-Go	Traffic-based billing	Bill by the outgoing data volume
	Bandwidth packages	Bill by the higher one between inbound bandwidth peak and outbound bandwidth peak

Billing
Instance Pricing

Instance Pricing

Converge Cloud Load Balancer (CLB), an "instance" refers to an individual CLB service unit that you create and configure to distribute traffic across backend servers (such as CVMs). It's the actual deployed load balancer that handles incoming requests based on your selected settings (e.g., region, protocol, forwarding rules).

A Converge CLB instance continues to incur charges for the instance fee even when it is idle—that is, when there are no incoming access requests and no real servers are associated with it.

Converge CLB offers two types of instances:

- CLB supports TCP, UDP, HTTP, and HTTPS protocols, allowing you to flexibly forward traffic based on domain names and URL paths.
- Classic CLB offers easy configuration but lacks support for HTTP/HTTPS over private networks.

Product Type	CLB		Classic CLB	
	Public Network	Private Network	Public Network	Private Network
Layer-7 forwarding (HTTP/ HTTPS)	✓	✓	✓	×
Layer-4 forwarding (TCP/UDP)	✓	✓	✓	✓
Encrypted Layer-4 forwarding (TCP SSL)	✓	✓	×	×

Billing cycle (Pay-As-You-Go)

- The instance fee is charged daily but calculated based on usage.
- Billing begins when the CLB instance is successfully deployed and stops once the termination process is triggered.
- The minimum billing unit is one hour.

Key Considerations:

When a pay-as-you-go CLB instance is created, an hourly fee—equivalent to 1/24 of the daily rate—is automatically deducted from your account balance. Please ensure your account has enough funds to cover

this cost.

Network Pricing

Network Pricing

In Converge Cloud Load Balancer (CLB), network use refers to the data transferred between the CLB instance and the internet or other Converge Cloud resources. This includes both inbound (incoming) and outbound (outgoing) traffic. Only public network CLBs incur network fees. Private network CLB instances do not incur network fees.

Outbound Bandwidth (Purchased)	Inbound Bandwidth (Allocated)	Description
≤ 10 Mbps	10 Mbps	Minimum inbound bandwidth allocation
> 10 Mbps	Equal to purchased outbound bandwidth	Inbound matches outbound bandwidth

All network billing modes:

- If the user purchases 10 Mbps or less of outbound bandwidth, Converge Cloud provides 10 Mbps of public network inbound bandwidth by default.
- If the user purchases greater than 10 Mbps, Converge Cloud allocates public network inbound bandwidth equal to the purchased outbound bandwidth.

Billing cycle (Pay-As-You-Go)
For pay-as-you-go CLB instances, the network billing modes include traffic-based billing and bandwidth-based billing.

A. Traffic-based billing (Pay-by-Traffic)

In Converge Cloud, data transferred over the public network is calculated in gigabytes (GB) and billed on an hourly basis. This pricing model is well-suited for businesses with fluctuating or unpredictable traffic and is most cost-effective when bandwidth utilization is below 10%. You can set a bandwidth cap to control costs—any traffic beyond the cap is dropped without additional charges.

This model is best suited for:

- Startups and small applications
- Testing or development environments
- Seasonal or campaign-based traffic

It offers flexibility, accurate billing, and helps avoid overpaying for unused bandwidth.

Key Considerations:

- Traffic charges are based on the outbound traffic sent from the CLB instance to the public network.
- To avoid unexpected costs from traffic spikes, you can set a bandwidth limit—any traffic beyond the cap is discarded and not billed.
- Traffic is measured using binary units, where 1 TB = 1,024 GB and 1 GB = 1,024 MB.

INCLUDE PRICING

B. Bandwidth-based Billing (Pay-by-Bandwidth)

Bandwidth-based billing charges you based on the fixed outbound bandwidth (in Mbps) that you pre-select, regardless of how much traffic is actually used. This model is offered by Converge Cloud for CLB users who need stable and predictable network performance.

It is especially recommended for scenarios where traffic remains steady or high over time, such as live streaming, gaming platforms, enterprise websites, or real-time communication services. Since the cost does not fluctuate with usage, this billing model offers budget predictability, which is ideal for organizations managing fixed monthly operational costs.

While it may not be as cost-effective for low or variable traffic, it ensures reliable bandwidth availability, helping maintain performance consistency during peak demand.

Purchase Methods

Purchase Methods

Converge Cloud allows you to purchase a CLB instance on the official purchase page or via API.

1. Log in to the Converge Cloud console and go to the CLB purchase page.
2. Select the following CLB configuration items as needed:

Parameter	Description
Billing mode	Supports pay-as-you-go billing.
Region	Select a region.
Instance type	Supports the CLB instance type only.
Network type	Cloud Load Balancer (CLB) supports two network types: public and private. • In a public network, CLB handles traffic coming from the internet. • In a private network, CLB distributes traffic within the Converge Cloud internal network. Private network CLB instances do not support certain settings, such as: Elastic IP (EIP), IP version, ISP selection, instance specifications, network billing options, and bandwidth limits—so these options are hidden by default. Support for network types depends on the billing method: • Under the pay-as-you-go model, both public and private networks are supported.
Elastic IP	• If you do not select an Elastic IP (EIP), Converge Cloud will provide a public network CLB instance with a fixed public IP address that cannot be modified. • If you choose to use an EIP, Converge Cloud will assign an Elastic IP along with a private network CLB instance that functions similarly to a public CLB. Note: Only pay-as-you-go public CLB instances support the option to bind an EIP.
IP version	Cloud Load Balancer (CLB) supports three IP versions: IPv4, IPv6, and IPv6 NAT64. Note that only pay-as-you-go CLB instances currently support IPv6, which is still in beta testing.

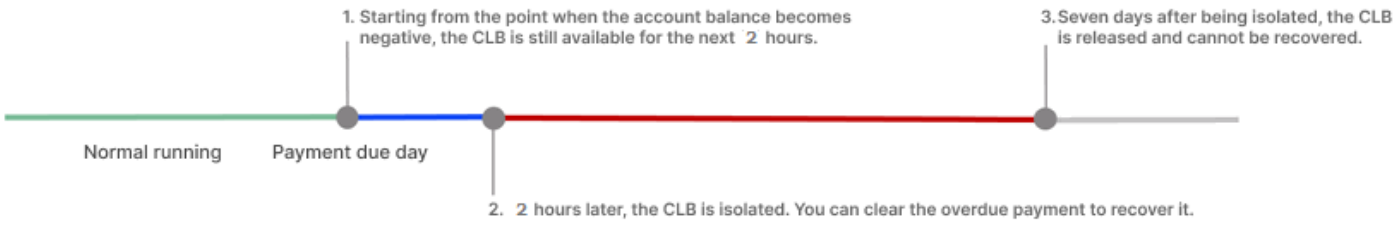
Parameter	Description
Network	Load Balancer (LB) supports both classic network and Virtual Private Cloud (VPC) environments • The classic network is a shared public network where private IP addresses are automatically assigned by Converge Cloud. Users cannot define IP ranges or specific addresses. • A VPC is an isolated, customizable network within Converge Cloud. It allows users to set custom IP ranges, assign specific IPs, and define routing rules. For greater flexibility and control, using a VPC is recommended—especially for applications that require tailored network configurations.
ISP	
Primary/ Secondary availability zone	The primary availability zone (AZ) is the main zone that actively handles incoming traffic. The secondary AZ remains on standby and only takes over if the primary AZ becomes unavailable, ensuring service continuity and high availability.
Instance specification	Shared Instances provide performance based on their defined specifications. Each instance can handle up to 50,000 concurrent connections, support 5,000 new connections per second, and process 5,000 queries per second, ensuring reliable service for typical workloads.
Network billing mode	Cloud Load Balancer supports multiple network billing options, including bandwidth-based billing (monthly or hourly), traffic-based billing, and bandwidth packages. For pay-as-you-go instances, the available billing modes are: hourly bandwidth, traffic-based, and bandwidth package.
Bandwidth cap	For shared CLB instances, the maximum public network bandwidth is capped at 2 Gbps. For the private network, it's recommended to keep the bandwidth limit within 5 Gbps to ensure stable performance. For CLB instances that support Load Capacity Units (LCU), the bandwidth cap is determined by the instance specification selected during setup.
Project	Select a project.
Tag	Select a tag key and value.
Instance name	You can assign a name with up to 60 characters, including letters, numbers, hyphens (-), underscores (_), and dots (.). If no name is provided, Converge Cloud will generate one automatically.

3. Once you've completed the configuration, review the quantity and pricing details, then click “Buy Now” to proceed.
    - If you're using the pay-as-you-go billing option, a confirmation window will appear—simply click “OK” to finalize the purchase.
4. After completing your purchase, the CLB instance becomes active, and you can start configuring and using it right away.

Payment Overdue

Payment Overdue

Payment Overdue Timeline: Important Dates and Notices



- To avoid continued charges, make sure to delete any unused pay-as-you-go resources.
- Please note that due to ongoing usage updates, there may be minor differences between your actual usage and displayed account balance.

Balance Alert

- The system checks your current balance and how much you've used in the past 24 hours to estimate when your funds might run out. If it looks like your balance will be used up within 5 days, an alert is sent to the account owner and any collaborators who subscribed to alerts, through email, SMS, or the Message Center.

Overdue Payment Alert

- Pay-as-you-go resources are charged by the hour. If your account balance goes negative (as shown in Point 1 in the figure), an alert is sent to the Converge Cloud account owner and any collaborators who’ve subscribed to notifications via email, SMS, or the Message Center.

Overdue Policies

- Once your account balance goes negative, your pay-as-you-go load balancers will keep running for 2 more hours, and billing will continue during that time. After 2 hours (see Point 2 in the figure), the load balancers will be suspended, and billing will stop.
- If a CLB is isolated and suspended, it will be permanently deleted after seven days and can’t be recovered.

Key Considerations:

- A CLB instance remains bound to its associated CVM unless manually unbound. Isolation of the CVM does not affect this binding.

Troubleshooting
UDP Health Check Exception

This section covers troubleshooting issues related to Cloud Load Balancer. This aims to ensure that you, our valued customers, can easily resolve any problems with load balancing to ensure greater availability and distribution of workloads in your environment.

UDP Health Check Exception:
In the CLB UDP health check, the results are different from the actual status of the server port.

If the PING command is successful and an error message port XX unreachable is not returned within the response timeout period, the real server is considered healthy. Otherwise, the real server is considered unhealthy.

There are two possible causes for this error:
The response timeout period is too short for the real server to return an ICMP message reply or a port unreachable to the health check nodes, leading to an inaccurate health check result.
The real server restricts the rate at which ICMP messages are generated. In this case, when a service exception occurs, port XX unreachable cannot be returned, and the CLB instance considers that the health check is successful, as no ICMP response is received. Therefore, the health check result is inconsistent with the actual server health.

Troubleshooting steps:

1. Check if the response timeout is too short. Log into the CLB Console, and increase the timeout duration of the UDP listener.

   Note:

   UDP health checks are different from other health checks. If the health check timeout period is too short, the health check result will switch back and forth between healthy and unhealthy.

2. If the error persists, check whether the real server restricts the frequency of ICMP messages. Log in to the real server, and run the following commands to check the rate limit.

   ```
   sysctl -q net.ipv4.icmp_ratelimit
   sysctl -q net.ipv4.icmp_ratemask
   ```

3. Check whether the return value of the first command is 0 or 1000 (default). We recommend using the default value.

4. If the error persists, run the following command to disable the rate limit of the ICMP.

Note:
Note that if the rate limit of ICMP port unreachable messages is lifted, when the real server is connected to the public network and encounters UDP port scanning attack, it will keep returning port unreachable messages

Run the command `net.ipv4.icmp_ratemask` in step 2 to query the rate mask

Keep the first three digits of the returned rate mask unchanged, and subtract 8 from the last digit. Fo r example, if the mask returned is “6168”, replace "xxxx" with 6160; if it is 1819, replace "xxxx" with 181 1.

sysctl -w net.ipv4.icmp_ratemask=xxxx

Step 1. Configure a listener

Log in to the CLB Console and click Instance management in the left sidebar.

Select your region in the top-left corner of the CLB instance list and click Configure listener in the Operation column of the target instance.

Under TCP/UDP/TCP SSL/QUIC listener, click Create, and configure the UDP listener in the pop-up window.

3.1 Configure basic parameters

Parameter	Description	Example
Name	Listener name.	test-udp-8000
Listening protocol and port	Listening protocol: In this case, select UDP. Listening port: The port used to receive requests and forward them to real servers. The port number ranges from 1 to 65535. Port 4789 is reserved for the system and is not yet unavailable. A listening port must be unique in the same CLB instance.	UDP:8000
Balancing method	CLB supports two scheduling algorithms for UDP listeners: weighted round robin (WRR) and weighted least connections (WLC) WRR: Requests are distributed to real servers in sequence based on their weights. This algorithm performs scheduling based on the number of new connections. Servers with higher weights are more likely to be scheduled and servers with the	WRR

Parameter	Description	Example
	same weight process the same number of connections. WLC: Loads of servers are estimated based on the number of active connections to the servers. This algorithm performs scheduling based on server loads and weights. For servers with the same weight, those have less loads are more likely to be scheduled. Note: If WLC is selected, the listener does not support session persistence.	
Schedule by QUIC ID	Once this feature is enabled, CLB will schedule client requests by QUIC ID, so requests with the same QUIC Connection ID will be scheduled to the same real server. If a request doesn't have a QUIC Connection ID, it will be downgraded to normal WRR scheduling, i.e., scheduling according to the quadruple (source IP address + destination IP address + source port + destination port).	Enabled
Configure ProxyProtocol	After checking, you can enable the ProxyProtocol configuration. It supports carrying the client's source address to the real server through the ProxyProtocol protocol.	Check and use

3.2 Configure health check

3.3 Configure session persistence

Parameter	Description	Example
Session persistence	After session persistence is enabled, a CLB listener will distribute access requests from the same client to the same real server. UDP session persistence is implemented based on the client IP address. The 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
Hold Time	Session persistence duration. Session persistence is terminated if there are no new requests in the connection within the specified duration. Value range: 30-3600 seconds	30 seconds

Step 2. Bind a real server

On the Listener management page, click the created listener UDP:8000 to view the bound real servers on the right of the listener.
Click Bind, select the target real server, and configure the server port and weight in the pop-up window.

Note:
If you set the default port first and then select real servers, the port of every real server is the default port.

Step 3. Configure a security group (optional)

You can configure a CLB security group to isolate public network traffic.

Step 4. Modify or delete a listener (optional)

If you need to modify or delete a created listener, click the listener on the Listener management page and click for modification or for deletion.

API Documentation
Load balancer(clb) version(2018-03-17)
API overview

API version

V3

CLB APIs

Interface name	Interface function
AutoRewrite	AutoRewrite
BatchDeregisterTargets	BatchDeregisterTargets
BatchModifyTargetWeight	BatchModifyTargetWeight
BatchRegisterTargets	BatchRegisterTargets
CreateListener	CreateListener
CreateLoadBalancer	CreateLoadBalancer
CreateRule	CreateRule
DeleteCert	DeleteCert
DeleteListener	DeleteListener
DeleteLoadBalancer	DeleteLoadBalancer
DeleteRewrite	DeleteRewrite
DeleteRule	DeleteRule
DeregisterTargets	DeregisterTargets
DescribeAppIdLabel	DescribeAppIdLabel

Interface name	Interface function
DescribeCerts	DescribeCerts
DescribeCustomizedConfigAssociateList	DescribeCustomizedConfigAssociateList
DescribeCustomizedConfigList	DescribeCustomizedConfigList
DescribeIspInfo	DescribeIspInfo
DescribeListenerListByCertId	DescribeListenerListByCertId
DescribeListeners	DescribeListeners
DescribeLoadBalancers	DescribeLoadBalancers
DescribeRewrite	DescribeRewrite
DescribeSubUinQuotas	DescribeSubUinQuotas
DescribeTargetHealth	DescribeTargetHealth
DescribeTargets	DescribeTargets
DescribeTaskStatus	DescribeTaskStatus
InquiryPriceModifyLoadBalancer	InquiryPriceModifyLoadBalancer
ManualRewrite	ManualRewrite
ModifyCertAlias	ModifyCertAlias
ModifyDomainAttributes	ModifyDomainAttributes
ModifyListener	ModifyListener
ModifyLoadBalancerAttributes	ModifyLoadBalancerAttributes
ModifyRule	ModifyRule
ModifyTargetPort	ModifyTargetPort
ModifyTargetWeight	ModifyTargetWeight
RegisterTargets	RegisterTargets
ReplaceCert	ReplaceCert
SetCustomizedConfigForLoadBalancer	SetCustomizedConfigForLoadBalancer
SetSubUinQuotas	SetSubUinQuotas

Interface name	Interface function
UpLoadCert	UpLoadCert

Other APIs

Interface name	Interface function
DescribeLoadBalancerListByCertId	DescribeLoadBalancerListByCertId
SetLoadBalancerSecurityGroups	SetLoadBalancerSecurityGroups
SetSecurityGroupForLoadbalancers	SetSecurityGroupForLoadbalancers

CallingMethod signaturev1

Converge Cloud authenticates each access request, i.e. each request needs to include authentication information (Signature) in the common parameters to verify the identity of the requester.
The Signature is generated by the security credentials which include SecretId and SecretKey. If you don't have the security credentials yet, go to the API Key page to apply for them; otherwise, you cannot invoke the API.

1. Applying for Security Credentials

Before using the API for the first time, go to the API Key page to apply for security credentials.
Security credentials consist of SecretId and SecretKey:

- SecretId is used to identify the API requester.
- SecretKey is used to encrypt the signature string and verify it on the server.
- You must keep your security credentials private and avoid disclosure.

You can apply for the security credentials through the following steps:

1. Log in to the Converge Cloud Console.
2. Go to the Cloud API Key page.
3. On the API Key Management page, click Create Key to create a SecretId/SecretKey pair.

Note: Each account can have up to two pairs of SecretId/SecretKey.

2. Generating a Signature

With the SecretId and SecretKey, a signature can be generated. The following describes how to generate a signature:

Assume that the SecretId and SecretKey are:

- SecretId: AKIDz8krbsJ5yKBZQpn74WFkmLPx3EXAMPLE
- SecretKey: Gu5t9xGARNpq86cd98joQYCN3EXAMPLE

Note: This is just an example. For actual operations, please use your own SecretId and SecretKey.

Take the Cloud Virtual Machine's request to view the instance list (DescribeInstances) as an example. When

you invoke this API, the request parameters may be as follows:

Parameter name	Description	Parameter value
Action	Method name	DescribeInstances
SecretId	Key ID	AKIDz8krbsJ5yKBZQpn74WFkmLPx3EXAMPLE
Timestamp	Current timestamp	1465185768
Nonce	Random positive integer	11886
Region	Region where the instance is located	luzon
InstanceIds.0	ID of the instance to query	ins-09dx96dg
Offset	Offset	0
Limit	Allowed maximum output	20
Version	API version number	2017-03-12

2.1. Sorting Parameters

First, sort all the request parameters in an ascending lexicographical order (ASCII code) by their names. Notes: (1) Parameters are sorted by their names instead of their values; (2) The parameters are sorted based on ASCII code, not in an alphabetical order or by values. For example, InstanceIds.2 should be arranged after InstanceIds.12. You can complete the sorting process using a sorting function in a programming language, such as the ksort function in PHP. The parameters in the example are sorted as follows:

```
{
    'Action' : 'DescribeInstances',
    'InstanceIds.0' : 'ins-09dx96dg',
    'Limit' : 20,
    'Nonce' : 11886,
    'Offset' : 0,
    'Region' : 'luzon',
    'SecretId' : 'AKIDz8krbsJ5yKBZQpn74WFkmLPx3EXAMPLE',
    'Timestamp' : 1465185768,
    'Version': '2017-03-12',
}
```

When developing in another programming language, you can sort these sample parameters and it will work as long as you obtain the same results.

2.2. Concatenating a Request String

This step generates a request string.
Format the request parameters sorted in the previous step into the form of "parameter name"="parameter value". For example, for the Action parameter, its parameter name is "Action" and its parameter value is "DescribeInstances", so it will become Action=DescribeInstances after formatted.
Note: The "parameter value" is the original value but not the value after URL encoding.

Then, concatenate the formatted parameters with "&". The resulting request string is as follows:

```
Action=DescribeInstances&InstanceIds.0=ins-09dx96dg&Limit=20&Nonce=11886&Offset=0&Region=luzon&SecretId=AKIDz8krbsJ5yKBZQpn74WFkmLPx3EXAMPLE&Timestamp=1465185768&Version=2017-03-12
```

2.3. Concatenating the Signature Original String

This step generates a signature original string.
The signature original string consists of the following parameters:

1. HTTP method: POST and GET modes are supported, and GET is used here for the request. Please note that the method name should be in all capital letters.
2. Request host: the domain name of the request to view the list of instances (DescribeInstances) is cvm.convergecloud.com. The actual request domain name varies by the module to which the API belongs. For more information, see the instructions of the specific API.
3. Request path: The request path in the current version of Cloud API is fixed to / .
4. Request string: the request string generated in the previous step.

The concatenation rule of the signature original string is: Request method + request host + request path + ? + request string

The concatenation result of the example is:

```
GETcvm.convergecloud.com/?Action=DescribeInstances&InstanceIds.0=ins-09dx96dg&Limit=20&Nonce=11886&Offset=0&Region=luzon&SecretId=AKIDz8krbsJ5yKBZQpn74WFkmLPx3EXAMPLE&Timestamp=1465185768&Version=2017-03-12
```

2.4. Generating a Signature String

This step generates a signature string.
First, use the HMAC-SHA1 algorithm to sign thesignature original stringobtained in the previous step, and then encode the generated signature using Base64 to obtain the final signature.

The specific code is as follows with the PHP language being used as an example:

```
$secretKey = 'Gu5t9xGARNpq86cd98joQYCN3EXAMPLE';
$srcStr = 'GETcvm.convergecloud.com/?Action=DescribeInstances&InstanceIds.0=ins-09dx96dg&Limit
=20&Nonce=11886&Offset=0&Region=luzon&SecretId=AKIDz8krbsJ5yKBZQpn74WFkmLPx3EXAMPL
E&Timestamp=1465185768&Version=2017-03-12';
$signStr = base64_encode(hash_hmac('sha1', $srcStr, $secretKey, true));
echo $signStr;
```

The final signature is:

```
EliP9YW3pW28FpsEdkXt/+WcGeI=
```

When developing in another programming language, you can sign and verify the original in the example above and it works as long as you get the same results.

3. Encoding a Signature String

The generated signature string cannot be directly used as a request parameter and must be URL encoded.

For example, if the signature string generated in the previous step is EliP9YW3pW28FpsEdkXt/+WcGeI= , the final signature string request parameter (Signature) is EliP9YW3pW28FpsEdkXt%2f%2bWcGeI%3d, which will be used to generate the final request URL.

**Note: If your request method is GET, or the request method is POST and the Content-Type is application/x-www-form-urlencoded, then all the request parameter values need to be URL encoded (except the parameter key and the symbol of =) when sending the request. Non-ASCII characters need to be encoded with UTF-8 before URL encoding. **

Note: The network libraries of some programming languages automatically URL encode all parameters, in which case there is no need to URL encode the signature string; otherwise, two rounds of URL encoding will cause the signature to fail.

Note: Other parameter values also need to be encoded using RFC 3986. Use %XY in percent-encoding for special characters such as Chinese characters, where "X" and "Y" are hexadecimal characters (0-9 and uppercase A-F), and using lowercase will cause an error.

4. Signature Failure

The following situational error codes for signature failure may occur. Please resolve the errors accordingly.

Error code	Error description
AuthFailure.SignatureExpire	The signature is expired

Error code	Error description
AuthFailure.SecretIdNotFound	The key does not exist
AuthFailure.SignatureFailure	Signature error
AuthFailure.TokenFailure	Token error
AuthFailure.InvalidSecretId	Invalid key (not a Cloud API key type)

5. Signature Demo

When calling API 3.0, you are recommended to use the corresponding Converge Cloud SDK 3.0, Swhich encapsulates the signature process, enabling you to focus on only the specific APIs provided by the product when developing. See SDK Center for more information. Currently, the following programming languages are supported:

- Python
- Java
- PHP
- Go
- Node

To further explain the signing process, we will use a programming language to implement the process described above. The request domain name, API and parameter values in the sample are used here. This goal of this example is only to provide additional clarification for the signature process, please see the SDK for actual usage.

The final output URL might be: https://cvm.convergecloud.com/?Action=DescribeInstances&InstanceIds.0=ins-09dx96dg&Limit=20&Nonce=11886&Offset=0&Region=luzon&SecretId=AKIDz8krbsJ5yKBZQpn74WFkmLPx3EXAMPLE&Signature=EliP9YW3pW28FpsEdkXt%2F%2BWcGeI%3D&Timestamp=1465185768&Version=2017-03-12

Note: The key in the example is fictitious, and the timestamp is not the current time of the system, so if this URL is opened in the browser or called using commands such as curl, an authentication error will be returned: Signature expired. In order to get a URL that can work properly, you need to replace the SecretId and SecretKey in the example with your real credentials and use the current time of the system as the Timestamp.

Note: In the example below, even if you use the same programming language, the order of the parameters in the URL may be different for each execution. However, the order does not matter, as long as all the parameters are included in the URL and the signature is calculated correctly.

Note: The following code is only applicable to API 3.0. It cannot be directly used in other signature processes. Even with an older API, signature calculation errors may occur due to the differences in details. Please refer to the corresponding documentation.

Java

```
import java.io.UnsupportedEncodingException;
import java.net.URLEncoder;
import java.util.Random;
import java.util.TreeMap;
import javax.crypto.Mac;
import javax.crypto.spec.SecretKeySpec;
import javax.xml.bind.DatatypeConverter;

public class CloudAPIDemo {
    private final static String CHARSET = "UTF-8";

    public static String sign(String s, String key, String method) throws Exception {
        Mac mac = Mac.getInstance(method);
        SecretKeySpec secretKeySpec = new SecretKeySpec(key.getBytes(CHARSET), mac.getAlgorithm());
        mac.init(secretKeySpec);
        byte[] hash = mac.doFinal(s.getBytes(CHARSET));
        return DatatypeConverter.printBase64Binary(hash);
    }

    public static String getStringToSign(TreeMap<String, Object> params) {
        StringBuilder s2s = new StringBuilder("GETcvm.convergecloud.com/?");
        // When signing, the parameters need to be sorted in lexicographical order. TreeMap is used here
to guarantee the correct order.
        for (String k : params.keySet()) {
            s2s.append(k).append("=").append(params.get(k).toString()).append("&");
        }
        return s2s.toString().substring(0, s2s.length() - 1);
    }

    public static String getUrl(TreeMap<String, Object> params) throws UnsupportedEncodingException
{
        StringBuilder url = new StringBuilder("https://cvm.convergecloud.com/?");
        // There is no requirement for the order of the parameters in the actual request URL.
        for (String k : params.keySet()) {
            // The request string needs to be URL encoded. As the Key is all in English letters, only the valu
e is URL encoded here.
            url.append(k).append("=").append(URLEncoder.encode(params.get(k).toString(), CHARSET)).app
end("&");
        }
        return url.toString().substring(0, url.length() - 1);
```

```
    }

    public static void main(String[] args) throws Exception {
        TreeMap<String, Object> params = new TreeMap<String, Object>(); // TreeMap enables automati
c sorting
        // A random number should be used when actually calling, for example: params.put("Nonce", new
Random().nextInt(java.lang.Integer.MAX_VALUE));
        params.put("Nonce", 11886); // Common parameter
        // The current time of the system should be used when actually calling, for example:   params.put
("Timestamp", System.currentTimeMillis() / 1000);
        params.put("Timestamp", 1465185768); // Common parameter
        params.put("SecretId", "AKIDz8krbsJ5yKBZQpn74WFkmLPx3EXAMPLE"); // Common parameter
        params.put("Action", "DescribeInstances"); // Common parameter
        params.put("Version", "2017-03-12"); // Common parameter
        params.put("Region", "luzon"); // Common parameter
        params.put("Limit", 20); // Business parameter
        params.put("Offset", 0); // Business parameter
        params.put("InstanceIds.0", "ins-09dx96dg"); // Business parameter
        params.put("Signature", sign(getStringToSign(params), "Gu5t9xGARNpq86cd98joQYCN3EXAMPLE
", "HmacSHA1")); // Common parameter
        System.out.println(getUrl(params));
    }
}
```

Python

Note: If running in a Python 2 environment, the following requests dependency package must be installed first: pip install requests .

```
# -*- coding: utf8 -*-
import base64
import hashlib
import hmac
import time

import requests

secret_id = "AKIDz8krbsJ5yKBZQpn74WFkmLPx3EXAMPLE"
secret_key = "Gu5t9xGARNpq86cd98joQYCN3EXAMPLE"

def get_string_to_sign(method, endpoint, params):
    s = method + endpoint + "/?"
    query_str = "&".join("%s=%s" % (k, params[k]) for k in sorted(params))
    return s + query_str

def sign_str(key, s, method):
```

```
    hmac_str = hmac.new(key.encode("utf8"), s.encode("utf8"), method).digest()
    return base64.b64encode(hmac_str)

if __name__ == '__main__':
    endpoint = "cvm.convergecloud.com"
    data = {
        'Action' : 'DescribeInstances',
        'InstanceIds.0' : 'ins-09dx96dg',
        'Limit' : 20,
        'Nonce' : 11886,
        'Offset' : 0,
        'Region' : 'luzon',
        'SecretId' : secret_id,
        'Timestamp' : 1465185768, # int(time.time())
        'Version': '2017-03-12'
    }
    s = get_string_to_sign("GET", endpoint, data)
    data["Signature"] = sign_str(secret_key, s, hashlib.sha1)
    print(data["Signature"])
    # n actual invocation would occur here, which may incur fees after success
    # resp = requests.get("https://" + endpoint, params=data)
    # print(resp.url)
```

signaturev3

Converge Cloud The API will authenticate each access request, meaning that each request must include signature information (Signature) in the public request parameters to verify the requester's identity.
The signature information is generated by the security credentials, which include SecretId and SecretKey . If the user does not have security credentials, please apply for them on the Cloud API Key page, otherwise they cannot call the Cloud API interfaces.

1. Apply for security credentials.

Before using the Cloud API for the first time, please go to the Cloud API Key page to apply for a security credential.
Security credentials include SecretId and SecretKey:

- SecretId The security credentials are used to identify the identity of the API caller.
- SecretKey The secret key is used to encrypt and sign the string for server-side verification of the signature.
- Users must strictly protect their security credentials and avoid disclosure them.

The specific steps to apply for security credentials are as follows:

1. Log in to Converge Cloud Administration Center Console
2. Go to the console page for cloud API keys
3. On the Cloud API Key page, click New to create a pair of SecretId/SecretKey

Note: Developer accounts can have up to two pairs SecretId / SecretKey.

2. TC3-HMAC-SHA256 signature method

Note: For GET method, only Content-Type: application/x-www-form-urlencoded protocol format is supported. For POST method, Content-Type: application/json and Content-Type: multipart/form-data are currently supported. By default, all business interfaces are supported in json format. Only specific business interfaces are supported in multipart format. In this case, the interface cannot be called in json format. Refer to the specific business interface document description.

The following takes Cloud Virtual Machine querying Region-1 instance list as an example to introduce the signature calculation process step by step. We only use two parameters for the query instance list: Limit and Offset, called using the GET method.

Assume that the SecretId and SecretKey of the user are AKIDz 8krbsJ 5yKBZQpn 74WFkmLPx 3EXAMPLE and Gu 5t9xGARNpq 86cd 98joQYCN3EXAMPLE, respectively.

2.1. splice specification request string

Concatenate CanonicalRequest strings in the following format:

```
CanonicalRequest =
    HTTPRequestMethod + '\n' +
    CanonicalURI + '\n' +
    CanonicalQueryString + '\n' +
    CanonicalHeaders + '\n' +
    SignedHeaders + '\n' +
    HashedRequestPayload
```

- HTTPRequestMethod: HTTP request methods (GET, POST), GET in this example;
- CanonicalURI: URI parameter, API 3.0 fixed to forward slash (/);
- CanonicalQueryString: Query string in the URL that initiates the HTTP request. For POST requests, it is fixed as an empty string. For GET requests, it is the question mark (?) in the URL. The following string content, this example value is: Limit=10&Offset=0. Note: CanonicalQueryString needs to be URL encoded.
- CanonicalHeaders: The header information of participating signature, at least including host and content-type, can also be added with customized header to improve the uniqueness and security of its own request. Splicing rules: 1) head key and value are uniformly converted to lowercase, and the first and last spaces are removed, and spliced according to the format of key:value\n;2) multiple heads are spliced according to the dictionary sorting of head key (lowercase). In this example: content-type: application/x-www-form-urlencoded\nhost: cvm.convergecloud.com\n
- SignedHeaders: The header information involved in the signature indicates which headers participate in the signature of this request, and the header content contained in CanonicalHeaders is one-to-one corresponding. content-type and host are required headers. constructed as: 1) The head key is uniformly converted to lowercase; 2) Multiple header keys (lowercase) are concatenated according to dictionary sorting and separated by semicolons (;). In this case: content-type;host
- HashedRequestPayload: The hash value of the request body, calculated as Lowercase(HexEncode(Hash.SHA256(RequestPayload))), SHA256 hash the entire payload of the HTTP request body, then hexadecimal encoding, and finally converting the encoding string to lowercase letters. Note: For GET requests, RequestPayload is fixed as an empty string. For POST requests, RequestPayload is the payload of the HTTP request body.

According to the above rules, the resulting canonical request string in the example is as follows (for clarity, \nnewlines are replaced by a new line printed separately):

```
GET
```

```
/
Limit=10&Offset=0
content-type:application/x-www-form-urlencoded
host:cvm.convergecloud.com

content-type;host
e3b0c44298fc1c149afbf4c8996fb92427ae41e4649b934ca495991b7852b855
```

2.2. Concatenate the string to be signed

Concatenate the string to be signed in the following format:

```
StringToSign =
  Algorithm + \n +
  RequestTimestamp + \n +
  CredentialScope + \n +
  HashedCanonicalRequest
```

- Algorithm: Signature algorithm, currently fixed as TC3-HMAC-SHA256;
- RequestTimestamp: Request timestamp, i.e. the X-TC-Timestamp value of the request header. The request in the example above is 1539084154;
- CredentialScope: Credential range in the format Date/service/tc3_request, containing date, requested service and termination string (tc3_request). **Date is the date of UTC standard time, the value needs to be consistent with UTC standard time converted by common parameter X-TC-Timestamp **;service is the product name, which must be consistent with the product domain name called, such as cvm. The above example request, the value is 2018-10-09/cvm/tc3_request;
- HashedCanonicalRequest: The hash value of the canonical request string obtained by concatenating the above steps is calculated by Lowercase(HexEncode(Hash.SHA256(CanonicalRequest))).

note:

> 1. Date must be calculated from timestamp X-TC-Timestamp and time zone UTC+0. Adding local time zone information to the system, such as East Zone 8, will cause calls to succeed during the day and night, but calls to fail in the early hours of the morning. Assuming the timestamp is 1551113065, the time in the East Eight District is 2019-02-26 00:44:25, but the calculated Date with UTC+0 should be 2019-02-25 instead of 2019-02-26.
> 2. Timestamp must be the current system time and ensure that the system time and standard time are synchronized. If the difference exceeds five minutes, it will fail. If it is not synchronized with the standard time for a long time, it may cause the request to fail after running for a period of time (return signature expiration error).

According to the above rules, the string to be signed in the example is as follows (for clarity,\nnewlines are

replaced by new lines printed):

```
TC3-HMAC-SHA256
1539084154
2018-10-09/cvm/tc3_request
91c9c192c14460df6c1ffc69e34e6c5e90708de2a6d282cccf957dbf1aa7f3a7
```

2.3. calculate the signature

1. Compute the derived signature key, pseudocode as follows

```
SecretKey = "Gu5t9xGARNpq86cd98joQYCN3EXAMPLE"
SecretDate = HMAC_SHA256("TC3" + SecretKey, Date)
SecretService = HMAC_SHA256(SecretDate, Service)
SecretSigning = HMAC_SHA256(SecretService, "tc3_request")
```

- SecretKey: Original SecretKey;
- Date: That is, the Date field information in Credential, as shown in the example above, is 2018-10-09.;
- Service: That is, the Service field information in Credential, as shown in the example above, is cvm;

2. Calculate the signature, pseudocode as follows

```
Signature = HexEncode(HMAC_SHA256(SecretSigning, StringToSign))
```

- SecretSigning: That is, the derivative signature key calculated above;
- StringToSign: That is, the character string to be signed calculated in step 2;

2.4. Splicing Authorization

Splice as follows Authorization:

```
Authorization =
    Algorithm + ' ' +
    'Credential=' + SecretId + '/' + CredentialScope + ', ' +
    'SignedHeaders=' + SignedHeaders + ', '
    'Signature=' + Signature
```

- Algorithm: Signature method, fixed as TC3-HMAC-SHA256;
- SecretId: SecretId in key pair;
- CredentialScope: See above, Scope of Voucher;
- SignedHeaders: See above, header information for participating signatures;
- Signature: signature value

According to the above rules, the values obtained in the example are:

```
TC3-HMAC-SHA256 Credential=AKIDEXAMPLE/Date/service/tc3_request, SignedHeaders=content-type;host, Signature=5da7a33f6993f0614b047e5df4582db9e9bf4672ba50567dba16c6ccf174c474
```

The final complete invocation information is as follows:

```
https://cvm.convergecloud.com/?Limit=10&Offset=0

Authorization: TC3-HMAC-SHA256 Credential=AKIDz8krbsJ5yKBZQpn74WFkmLPx3EXAMPLE/2018-10-09/cvm/tc3_request, SignedHeaders=content-type;host, Signature=5da7a33f6993f0614b047e5df4582db9e9bf4672ba50567dba16c6ccf174c474
Content-Type: application/x-www-form-urlencoded
Host: cvm.convergecloud.com
X-TC-Action: DescribeInstances
X-TC-Version: 2017-03-12
X-TC-Timestamp: 1539084154
X-TC-Region: luzon
```

3. signing failed

According to the actual situation, there are the following signature failure error codes, please handle according to the actual situation

error code	error description
AuthFailure.SignatureExpire	signature expiration
AuthFailure.SecretIdNotFound	key is not present
AuthFailure.SignatureFailure	signature error
AuthFailure.TokenFailure	token error
AuthFailure.InvalidSecretId	Illegal key (not cloud API key type)

4. Signature Demo

Java

```
import java.io.BufferedReader;
import java.io.InputStream;
```

```
import java.io.InputStreamReader;
import java.net.URL;
import java.text.SimpleDateFormat;
import java.util.Date;
import java.util.Map;
import java.util.TimeZone;
import java.util.TreeMap;
import javax.crypto.Mac;
import javax.crypto.spec.SecretKeySpec;
import javax.net.ssl.HttpsURLConnection;
import javax.xml.bind.DatatypeConverter;

import org.apache.commons.codec.digest.DigestUtils;

public class CloudAPITC3Demo {
    private final static String CHARSET = "UTF-8";
    private final static String ENDPOINT = "cvm.convergecloud.com";
    private final static String PATH = "/";
    private final static String SECRET_ID = "AKIDz8krbsJ5yKBZQpn74WFkmLPx3EXAMPLE";
    private final static String SECRET_KEY = "Gu5t9xGARNpq86cd98joQYCN3EXAMPLE";
    private final static String CT_X_WWW_FORM_URLENCODED = "application/x-www-form-urlencode
d";
    private final static String CT_JSON = "application/json";
    private final static String CT_FORM_DATA = "multipart/form-data";

    public static byte[] sign256(byte[] key, String msg) throws Exception {
        Mac mac = Mac.getInstance("HmacSHA256");
        SecretKeySpec secretKeySpec = new SecretKeySpec(key, mac.getAlgorithm());
        mac.init(secretKeySpec);
        return mac.doFinal(msg.getBytes(CHARSET));
    }

    public static void main(String[] args) throws Exception {
        String service = "cvm";
        String host = "cvm.convergecloud.com";
        String region = "luzon";
        String action = "DescribeInstances";
        String version = "2017-03-12";
        String algorithm = "TC3-HMAC-SHA256";
        String timestamp = "1539084154";
        //String timestamp = String.valueOf(System.currentTimeMillis() / 1000);
        SimpleDateFormat sdf = new SimpleDateFormat("yyyy-MM-dd");
        // Pay attention to the time zone, otherwise it is easy to make mistakes.
        sdf.setTimeZone(TimeZone.getTimeZone("UTC"));
        String date = sdf.format(new Date(Long.valueOf(timestamp + "000")));

        // ************* Step 1: Assemble the specification request string *************
```

```
        String httpRequestMethod = "GET";
        String canonicalUri = "/";
        String canonicalQueryString = "Limit=10&Offset=0";
        String canonicalHeaders = "content-type:application/x-www-form-urlencoded\n" + "host:" + host
+ "\n";
        String signedHeaders = "content-type;host";
        String hashedRequestPayload = DigestUtils.sha256Hex("");
        String canonicalRequest = httpRequestMethod + "\n" + canonicalUri + "\n" + canonicalQueryStri
ng + "\n"
                + canonicalHeaders + "\n" + signedHeaders + "\n" + hashedRequestPayload;
        System.out.println(canonicalRequest);

        // ************* Step 2: Stitching the string to be signed *************
        String credentialScope = date + "/" + service + "/" + "tc3_request";
        String hashedCanonicalRequest = DigestUtils.sha256Hex(canonicalRequest.getBytes(CHARSET));
        String stringToSign = algorithm + "\n" + timestamp + "\n" + credentialScope + "\n" + hashedCan
onicalRequest;
        System.out.println(stringToSign);

        // ************* Step 3: Calculate the signature *************
        byte[] secretDate = sign256(("TC3" + SECRET_KEY).getBytes(CHARSET), date);
        byte[] secretService = sign256(secretDate, service);
        byte[] secretSigning = sign256(secretService, "tc3_request");
        String signature = DatatypeConverter.printHexBinary(sign256(secretSigning, stringToSign)).toLow
erCase();
        System.out.println(signature);

        // ************* Step 4: Splicing Authorization *************
        String authorization = algorithm + " " + "Credential=" + SECRET_ID + "/" + credentialScope + ", "
                + "SignedHeaders=" + signedHeaders + ", " + "Signature=" + signature;
        System.out.println(authorization);

        TreeMap<String, String> headers = new TreeMap<String, String>();
        headers.put("Authorization", authorization);
        headers.put("Host", host);
        headers.put("Content-Type", CT_X_WWW_FORM_URLENCODED);
        headers.put("X-TC-Action", action);
        headers.put("X-TC-Timestamp", timestamp);
        headers.put("X-TC-Version", version);
        headers.put("X-TC-Region", region);
    }
}
```

Python

```
# -*- coding: utf-8 -*-
```

```
import hashlib, hmac, json, os, sys, time
from datetime import datetime

# key parameters
secret_id = "AKIDz8krbsJ5yKBZQpn74WFkmLPx3EXAMPLE"
secret_key = "Gu5t9xGARNpq86cd98joQYCN3EXAMPLE"

service = "cvm"
host = "cvm.convergecloud.com"
endpoint = "https://" + host
region = "luzon"
action = "DescribeInstances"
version = "2017-03-12"
algorithm = "TC3-HMAC-SHA256"
timestamp = 1539084154
date = datetime.utcfromtimestamp(timestamp).strftime("%Y-%m-%d")
params = {"Limit": 10, "Offset": 0}

# ************* Step 1: Assemble the specification request string *************
http_request_method = "GET"
canonical_uri = "/"
canonical_querystring = "Limit=10&Offset=0"
ct = "x-www-form-urlencoded"
payload = ""
if http_request_method == "POST":
    canonical_querystring = ""
    ct = "json"
    payload = json.dumps(params)
canonical_headers = "content-type:application/%s\nhost:%s\n" % (ct, host)
signed_headers = "content-type;host"
hashed_request_payload = hashlib.sha256(payload.encode("utf-8")).hexdigest()
canonical_request = (http_request_method + "\n" +
                     canonical_uri + "\n" +
                     canonical_querystring + "\n" +
                     canonical_headers + "\n" +
                     signed_headers + "\n" +
                     hashed_request_payload)
print(canonical_request)

# ************* Step 2: Stitching the string to be signed *************
credential_scope = date + "/" + service + "/" + "tc3_request"
hashed_canonical_request = hashlib.sha256(canonical_request.encode("utf-8")).hexdigest()
string_to_sign = (algorithm + "\n" +
                  str(timestamp) + "\n" +
                  credential_scope + "\n" +
                  hashed_canonical_request)
print(string_to_sign)
```

```
# ************* Step 3: Calculate the signature *************
# Compute signature digest function
def sign(key, msg):
    return hmac.new(key, msg.encode("utf-8"), hashlib.sha256).digest()
secret_date = sign(("TC3" + secret_key).encode("utf-8"), date)
secret_service = sign(secret_date, service)
secret_signing = sign(secret_service, "tc3_request")
signature = hmac.new(secret_signing, string_to_sign.encode("utf-8"), hashlib.sha256).hexdigest()
print(signature)

# ************* Step 4: Splicing Authorization *************
authorization = (algorithm + " " +
                 "Credential=" + secret_id + "/" + credential_scope + ", " +
                 "SignedHeaders=" + signed_headers + ", " +
                 "Signature=" + signature)
print(authorization)

# Common parameters added to request header
headers = {
    "Authorization": authorization,
    "Host": host,
    "Content-Type": "application/%s" % ct,
    "X-TC-Action": action,
    "X-TC-Timestamp": str(timestamp),
    "X-TC-Version": version,
    "X-TC-Region": region,
}
```

request

1. Service Address

Region refers to the geographical area of a physical data center. Converge Cloud ensures complete isolation between different regions, providing maximum stability and fault tolerance. To reduce access latency and improve download speed, it is recommended to choose the region closest to your customers.

You can view the complete region list by using the API interface "DescribeRegions".

2. Communications Protocol

All the Converge Cloud APIs communicate via HTTPS, providing highly secure communication tunnels.

3. Request Methods

Supported HTTP request methods:

- POST (recommended)
- GET

PThe Content-Type types supported by POST requests:

- application/json (recommended). The TC3-HMAC-SHA256 signature algorithm must be used.
- application/x-www-form-urlencoded. The HmacSHA1 or HmacSHA256 signature algorithm must be used.
- multipart/form-data (only supported by certain APIs). You must use TC3-HMAC-SHA256 to calculate the signature.

The size of a GET request packet is up to 32 KB. The size of a POST request is up to 1 MB when the HmacSHA1 or HmacSHA256 signature algorithm is used, and up to 10 MB when TC3-HMAC-SHA256 is used.

4. Character Encoding

Only UTF-8 encoding is used.

result

Response for Successful Requests

For example, when calling CAM API (version: 2017-03-12) to view the status of instances (DescribeInstancesStatus), if the request has succeeded, you may see the response as shown below:


```
{
    "Response": {
        "TotalCount": 0,
        "InstanceStatusSet": [],
        "RequestId": "b5b41468-520d-4192-b42f-595cc34b6c1c"
    }
}
```


- The API will return Response, which contains RequestId, as long as it processes the request. It does not matter if the request is successful or not.
- RequestId is the unique ID of an API request. Contact us with this ID when an exception occurs.
- Except for the fixed fields, all fields are action-specified. For the definitions of action-specified fields, see the corresponding API documentation. In this example, TotalCount and InstanceStatusSet are the fields specified by the API DescribeInstancesStatus. 0 TotalCount means that the requester owns 0 CVM instance so the InstanceStatusSet is empty.

Response for Failed Requests

If the request has failed, you may see the response as shown below:


```
{
    "Response": {
        "Error": {
            "Code": "AuthFailure.SignatureFailure",
            "Message": "The provided credentials could not be validated. Please check your signature is cor
rect."
        },
        "RequestId": "ed93f3cb-f35e-473f-b9f3-0d451b8b79c6"
    }
}
```


- The presence of the Error field indicates that the request has failed. A response for a failed request will include Error, Code and Message fields.

- Code is the code of the error that helps you identify the cause and solution. There are two types of error codes so you may find the code in either common error codes or API-specified error codes.
- Message explains the cause of the error. Note that the returned messages are subject to service updates. The information the messages provide may not be up-to-date and should not be the only source of reference.
- RequestId is the unique ID of an API request. Contact us with this ID when an exception occurs.

Common Error Codes

If there is an Error field in the response, it means that the API call failed. The Code field in Error indicates the error code. The following table lists the common error codes that all actions can return.

Error Code	Description
AuthFailure.InvalidSecretId	Invalid key (not a API key type).
AuthFailure.MFAFailure	MFA failed.
AuthFailure.SecretIdNotFound	The key does not exist.
AuthFailure.SignatureExpire	Signature expired.
AuthFailure.SignatureFailure	Signature error.
AuthFailure.TokenFailure	Token error.
AuthFailure.UnauthorizedOperation	The request does not have CAM authorization.
DryRunOperation	DryRun Operation. It means that the request would have succeeded, but the DryRun parameter was used.
FailedOperation	Operation failed.
InternalError	Internal error.
InvalidAction	The API does not exist.
InvalidParameter	Incorrect parameter.
InvalidParameterValue	Invalid parameter value.
LimitExceeded	Quota limit exceeded.
MissingParameter	A parameter is missing.
NoSuchVersion	The API version does not exist.

Error Code	Description
RequestLimitExceeded	The number of requests exceeds the frequency limit.
ResourceInUse	Resource is in use.
ResourceInsufficient	Insufficient resource.
ResourceNotFound	The resource does not exist.
ResourceUnavailable	Resource is unavailable.
UnauthorizedOperation	Unauthorized operation.
UnknownParameter	Unknown parameter.
UnsupportedOperation	Unsupported operation.
UnsupportedProtocol	hHTTPS request method error. Only GET and POST requests are supported.
UnsupportedRegion	API does not support the requested region.

PublicParameter

Common parameters are parameters used to identify users and interface authentication purposes. If not necessary, these parameters are not described in the separate interface document of each interface, but these parameters need to be carried in each request before the request can be normally initiated.

signature method v3

When using TC3-HMAC-SHA256 signature method, the common parameters need to be uniformly placed in the HTTP Header request header, as follows:

parameter name	type	required	described
X-TC-Action	String	Yes	The name of the operation interface. Refer to the description of the public parameter Action in the input parameters of the interface documentation for the value. For example, the value for the query instance list interface of the cloud server is DescribeInstances.
X-TC-Region	String	Yes	region parameters, Data used to identify which region you wish to operate on. The region value accepted by the interface refers to the description of the input parameter common parameter Region in the interface document. Note: Some interfaces do not need to pass this parameter, which is specifically stated in the interface documentation. In this case, even passing this parameter will not take effect.
X-TC-Timestamp	Integer	Yes	The current UNIX timestamp, which records the time when the API request was initiated. for example 1529223702. Note: If the time difference from the server exceeds 5 minutes, it will cause a signature expiration error.
X-TC-Version	String	Yes	Version of API for operation. Refer to the description of the public parameter Version in the interface document for the value. For example, the version of the cloud server. 2017-03-12.
Authorization	String	Yes	HTTP The standard authentication header field is "Authorization". For example: TC3-HMAC-SHA256 Credential=AKIDEXAMPLE/Date/service/tc3_request, SignedHeaders=content-type;host, Signature=fe5f80f77d5fa3beca038a248ff027d0445342fe2855ddc963176630326f1024 Here, - TC3-HMAC-SHA256: Signature method, currently fixed as this value, - Credential: Credential: Signature credential; AKIDEXAMPLE is the SecretId; Date is a date in UTC time, and this value must match the value of X-TC-Timestamp (a common parameter) in UTC time format; service is the name of the product/service, and is generally a domain name prefix. For example,cvm; - SignedHeaders: The headers that contains the authentication information; content-type and host are the required headers; - Signature: Signature digest.
X-TC-Token	String	No	The token used for temporary certificates needs to be used in conjunction with a temporary key. Temporary keys and tokens need to be obtained by calling the access management service interface. Long-term keys do not require tokens.

signature method v1

When using HmacSHA1 and HmacSHA256 signature methods, common parameters need to be placed uniformly in the request string, as follows

parameter name	type	required	described
Action	String	Yes	The name of the interface to be operated. The value refers to the explanation of the public parameter Action in the interface documentation. For example, for the cloud server's query instance list interface, the value is DescribeInstances .
Region	String	Yes	The region parameter is used to identify which region's data you want to operate on. The region value accepted by the interface is specified in the input parameter of the public parameter Region in the interface documentation. Note: Some interfaces do not require this parameter to be passed, which will be specifically mentioned in the interface documentation. Even if this parameter is passed, it will not take effect in this case.
Timestamp	Integer	Yes	The current UNIX timestamp, which can record the time when the API request is initiated. For example, 1529223702. If the difference between the current time is too large, it will cause a signature expiration error.
Nonce	Integer	Yes	A random positive integer combined with Timestamp is used to prevent replay attacks.
SecretId	String	Yes	The SecretId is an identifier applied for the cloud API key to identify the user's identity. A SecretId corresponds to a unique SecretKey , and the SecretKey is used to generate the request signature Signature .
Signature	String	Yes	Request signature is used to verify the legitimacy of this request. Users need to calculate it based on the actual input parameters. The specific calculation method can be found in the interface authentication documentation.
Version	String	Yes	The version of the API being operated on. The value refers to the description of the public parameter Version in the interface documentation. For example, the version of the cloud server. 2017-03-12.
SignatureMethod	String	No	Signature method, currently supports HmacSHA256 and HmacSHA1. HmacSHA256 algorithm is used to verify signatures only if this parameter is specified as HmacSHA 256, otherwise HmacSHA1 is used to verify signatures.
Token	String	No	Token used for temporary certificate needs to be used together with temporary key. The temporary key and Token need to be obtained from the Access Management service invocation interface. Long-term keys do not require tokens.

zone list

A region (Region) refers to the geographical area of a physical data center. Converge Cloud Translation: To deliver verification that completely isolates different regions, ensuring maximum stability and fault tolerance between regions. In order to reduce access latency and improve download speed, it is recommended that you choose the region closest to your customers.

You can view the complete region list by using the API interface Query Region List.

CLB APIs
AutoRewrite

1. Interface Description

Interface request domain name: clb.api3.convergecloud.com.

An HTTPS:443 listener needs to be created first, along with a forwarding rule. When this API is called, an HTTP:80 listener will be created automatically if it did not exist and a forwarding rule corresponding to Domains (specified in the input parameter) under the HTTPS:443 listener will also be created. After successful creation, access requests to an HTTP:80 address will be redirected to an HTTPS:443 address automatically.
This is an async API. After it returns a successful result, you need to call the DescribeTaskStatus API with the returned RequestID as an input parameter to check whether this task is successful.

Default interface request rate limit: 20 times per second.

Interface update time: 2024-02-18 17:59:55.

The interface only verifies the signature but does not authenticate.

2. Input Parameters

The following list of request parameters only lists interface request parameters and some public parameters, For a complete list of public parameters see public request parameters.

Parameter name	Required	Allow NULL	Type	Description
Action	Yes	No	String	Public parameter, The value of this interface: AutoRewrite
Version	Yes	No	String	Public parameter, The value of this interface: 2018-03-17
Region	No	No	String	Public parameter,Regional information A list of supported geographies is available through the DescribeRegions interface
LoadBalancerId	Yes	No	String	CLB instance ID Example: lb-r6nx1iby
ListenerId	Yes	No	String	HTTPS:443 listener ID Example: lbl-lmeeyb1q
Domains	No	No	Array of String	The domain name to be redirected under the HTTPS:443 listener Example: ["www.domain1.com"]
TakeUrls	No	No	Array of Bool	Whether the matched URL is carried in redirection Example: [true]

Parameter name	Required	Allow NULL	Type	Description
RewriteCodes	No	No	Array of Int64	Redirection status code. Valid values: 301, 302, and 307. Example: [301]

3. Output Parameters

Parameter name	Type	Description
RequestId	String	Unique request id, Each request will return. The RequestId of the request needs to be provided when locating the problem.

4. Error Code

The following lists only the error codes related to the interface business logic, For other error codes, see public ErrorCode.

Error Code	Description
InvalidParameter	Wrong parameter.
InvalidParameterValue	Incorrect parameter value
InternalError	Internal Error
UnauthorizedOperation	Unauthorized operation
ResourceInsufficient	Insufficient resources
LimitExceeded	Quota limit reached
FailedOperation	Failed to operate.
InvalidParameter.LBIdNotFound	CLB instance ID error.
InvalidParameter.ListenerIdNotFound	Listener ID error.
InvalidParameter.LocationNotFound	Unable to find eligible forwarding rules.
InvalidParameter.PortCheckFailed	Listener port check due to port conflicts or other reasons.
InvalidParameter.FormatError	Incorrect parameter format
InvalidParameterValue.Length	Invalid parameter length.

BatchDeregisterTargets

1. Interface Description

Interface request domain name: clb.api3.convergecloud.com.

This API is used to batch unbind real servers of the layer-4 and layer-7 VPC-based CLB instances. Up to 500 resources can be unbound in a batch.

Default interface request rate limit: 20 times per second.

Interface update time: 2022-07-28 10:26:30.

The interface only verifies the signature but does not authenticate.

2. Input Parameters

The following list of request parameters only lists interface request parameters and some public parameters, For a complete list of public parameters see public request parameters.

Parameter name	Required	Allow NULL	Type	Description
Action	Yes	No	String	Public parameter, The value of this interface: BatchDeregisterTargets
Version	Yes	No	String	Public parameter, The value of this interface: 2018-03-17
Region	No	No	String	Public parameter,Regional information A list of supported geographies is available through the DescribeRegions interface
LoadBalancerId	Yes	No	String	CLB ID. Example: lb-j3ix99s2
Targets	Yes	No	Array of BatchTarget	Unbound targets. Example:View

3. Output Parameters

Parameter	Type	Description

name		
FailListenerIdSet	Array of String	ID of the listener failed to unbind. Example: ["lbl-4ibsji93"]
RequestId	String	Unique request id, Each request will return. The RequestId of the request needs to be provided when locating the problem.

4. Error Code

The following lists only the error codes related to the interface business logic, For other error codes, see public ErrorCode.

Error Code	Description
InvalidParameter	Wrong parameter.
InternalError	Internal Error
UnauthorizedOperation	Unauthorized operation
ResourceInsufficient	Insufficient resources
LimitExceeded	Quota limit reached
FailedOperation	Failed to operate.
InvalidParameter.LBIdNotFound	CLB instance ID error.
InvalidParameter.ListenerIdNotFound	Listener ID error.
InvalidParameter.LocationNotFound	Unable to find eligible forwarding rules.
InvalidParameter.PortCheckFailed	Listener port check due to port conflicts or other reasons.
InvalidParameter.ProtocolCheckFailed	Listener protocol check failed because, for example, the protocol used is incompatible with the corresponding operation.
InvalidParameter.FormatError	Incorrect parameter format
InvalidParameterValue.Length	Invalid parameter length.
InvalidParameterValue.Duplicate	Duplicate parameter values.
InvalidParameter.RegionNotFound	Invalid region.
MissingParameter	Missing parameter error.

Error Code	Description
InvalidParameterValue.Range	Invalid parameter value range.
InvalidParameterValue	Incorrect parameter value

BatchModifyTargetWeight

1. Interface Description

Interface request domain name: clb.api3.convergecloud.com.

This API (BatchModifyTargetWeight) is used to batch modify the forwarding weights of real servers bound to CLB listeners. It is supported by layer-4 and layer-7 CLB listeners, but not by Classic CLB.
This is an async API. After it returns a successful result, you need to call the DescribeTaskStatus API with the returned RequestID as an input parameter to check whether this task is successful.

Default interface request rate limit: 20 times per second.

Interface update time: 2021-04-27 20:46:14.

The interface only verifies the signature but does not authenticate.

2. Input Parameters

The following list of request parameters only lists interface request parameters and some public parameters, For a complete list of public parameters see public request parameters.

Parameter name	Required	Allow NULL	Type	Description
Action	Yes	No	String	Public parameter, The value of this interface: BatchModifyTargetWeight
Version	Yes	No	String	Public parameter, The value of this interface: 2018-03-17
Region	No	No	String	Public parameter,Regional information A list of supported geographies is available through the DescribeRegions interface
LoadBalancerId	Yes	No	String	CLB instance ID Example: lb-dx98lwo0
ModifyList	Yes	No	Array of RsWeightRule	List of weights to be modified in batches. For example, "ModifyList": [{"ListenerId": "lbl-di3kx902", "LocationId":"", "Domain": "", "Url": "", "Targets": [{"Type":"CVM", "InstanceId": "ins-xxxxxxx", "Port": "443", "Weight": 10, "EniIp":""}], "Weight": 10}]. Example:View

3. Output Parameters

Parameter name	Type	Description
RequestId	String	Unique request id, Each request will return. The RequestId of the request needs to be provided when locating the problem.

4. Error Code

The following lists only the error codes related to the interface business logic, For other error codes, see public ErrorCode.

Error Code	Description
InvalidParameter	Wrong parameter.
InvalidParameterValue	Incorrect parameter value
InternalError	Internal Error
UnauthorizedOperation	Unauthorized operation
FailedOperation	Failed to operate.
InvalidParameter.FormatError	Incorrect parameter format

Error Code	Description
InvalidParameterValue.Length	Invalid parameter length.

BatchRegisterTargets

1. Interface Description

Interface request domain name: clb.api3.convergecloud.com.

This API is used to batch bind CVM instances or ENIs. Up to 500 resources can be bound in a batch. It supports cross-region binding and binding based on layer-4 and layer-7 (TCP/UDP/HTTP/HTTPS) protocols. It supports VPC-based CLB instances only.

Default interface request rate limit: 20 times per second.

Interface update time: 2022-07-28 10:21:19.

The interface only verifies the signature but does not authenticate.

2. Input Parameters

The following list of request parameters only lists interface request parameters and some public parameters, For a complete list of public parameters see public request parameters.

Parameter name	Required	Allow NULL	Type	Description
Action	Yes	No	String	Public parameter, The value of this interface: BatchRegisterTargets
Version	Yes	No	String	Public parameter, The value of this interface: 2018-03-17
Region	No	No	String	Public parameter,Regional information A list of supported geographies is available through the DescribeRegions interface
LoadBalancerId	Yes	No	String	CLB ID. Example: lb-j3ix99s2
Targets	Yes	No	Array of BatchTarget	Bound targets. Example:View

3. Output Parameters

Parameter name	Type	Description
FailListenerIdSet	Array	ID of the listener failed to bind. Empty indicates all listeners are bound successfully.

Parameter name	Type	Description
	of String	Example: ["lbl-4ibsji93"]
RequestId	String	Unique request id, Each request will return. The RequestId of the request needs to be provided when locating the problem.

4. Error Code

The following lists only the error codes related to the interface business logic, For other error codes, see public ErrorCode.

Error Code	Description
InvalidParameter	Wrong parameter.
InvalidParameterValue	Incorrect parameter value
InternalError	Internal Error
UnauthorizedOperation	Unauthorized operation
LimitExceeded	Quota limit reached
FailedOperation	Failed to operate.
InvalidParameter.LBIdNotFound	CLB instance ID error.
InvalidParameter.ListenerIdNotFound	Listener ID error.
InvalidParameter.LocationNotFound	Unable to find eligible forwarding rules.
InvalidParameter.PortCheckFailed	Listener port check due to port conflicts or other reasons.
InvalidParameter.ProtocolCheckFailed	Listener protocol check failed because, for example, the protocol used is incompatible with the corresponding operation.
InvalidParameter.FormatError	Incorrect parameter format
InvalidParameterValue.Length	Invalid parameter length.
InvalidParameter.RegionNotFound	Invalid region.
MissingParameter	Missing parameter error.

CreateListener

1. Interface Description

Interface request domain name: clb.api3.convergecloud.com.

This API is used to create a listener under a CLB instance.
This is an async API. After it returns a successful result, you need to call the DescribeTaskStatus API with the returned RequestID as an input parameter to check whether this task is successful.

Default interface request rate limit: 20 times per second.

Interface update time: 2021-08-22 15:35:31.

The interface only verifies the signature but does not authenticate.

2. Input Parameters

The following list of request parameters only lists interface request parameters and some public parameters, For a complete list of public parameters see public request parameters.

Parameter name	Required	Allow NULL	Type	Description
Action	Yes	No	String	Public parameter, The value of this interface: CreateListener
Version	Yes	No	String	Public parameter, The value of this interface: 2018-03-17
Region	No	No	String	Public parameter,Regional information A list of supported geographies is available through the DescribeRegions interface
LoadBalancerId	Yes	No	String	CLB instance ID Example: lb-cuxwji93
Ports	Yes	No	Array of Int64	Ports for which listeners are to be created. Each port corresponds to a new listener. Example: ["7569","7570"]
Protocol	Yes	No	String	Listener protocol: TCP
ListenerNames	No	No	Array of String	List of names for listeners to be created. The names correspond sequentially to the array of ports on a one-to-one basis. If naming is not immediately necessary, this parameter need not be provided. Example: ["lis0","lis1"]
HealthCheck	No	No	HealthCheck	A health check parameter, applicable only to TCP/UDP/TCP_SSL listeners Example:View
Certificate	No	No	CertificateInput	Certificate information, applicable only to TCP_SSL listeners and HTTPS listeners without the SNI feature enabled Example:View
SessionExpireTime	No	No	Int64	Session persistence time. Unit: sec. Value range: 30-3,600. Default: 0, indicating disabled. This parameter is applicable only to TCP/UDP listeners. Example: 60
Scheduler	No	No	String	Listener forwarding mode. Valid values: WRR (weighted round robin) and LEAST_CONN (least connections) Default: WRR. This parameter is applicable only to TCP/UDP/TCP_SSL listeners. Example: WRR
SniSwitch	No	No	Int64	Whether to enable the SNI feature. This parameter applies only to HTTPS listeners. 0: disable; 1: enable. Example: 1
TargetType	No	No	String	Target real server type. NODE: Binding a general node; TARGETGROUP: Binding a target group. Example: NODE
DefaultSerSwitch	No	No	Bool	Whether the listener supports setting a default domain name. Default value: true. Example: true

3. Output Parameters

Parameter name	Type	Description
ListenerIds	Array of String	Unique identification array of created listeners Example: ["lbl-d1ubsydq","lbl-4udz130k"]
RequestId	String	Unique request id, Each request will return. The RequestId of the request needs to be provided when locating the problem.

4. Error Code

The following lists only the error codes related to the interface business logic, For other error codes, see public ErrorCode.

Error Code	Description
InvalidParameter	Wrong parameter.
InvalidParameterValue	Incorrect parameter value
InternalError	Internal Error

Error Code	Description
UnauthorizedOperation	Unauthorized operation
ResourceInsufficient	Insufficient resources
LimitExceeded	Quota limit reached
FailedOperation	Failed to operate.
InvalidParameter.FormatError	Incorrect parameter format
InvalidParameterValue.Length	Invalid parameter length.
MissingParameter	Missing parameter error.
InvalidParameter.PortCheckFailed	Listener port check due to port conflicts or other reasons.

CreateLoadBalancer

1. Interface Description

Interface request domain name: clb.api3.convergecloud.com.

This API is used to create a CLB instance. (This API only supports purchasing pay-as-you-go CLB instances. For monthly subscription CLB instances, purchase through the console.) To use the CLB service, you should purchase one or more CLB instances. After this API is successfully called, the unique ID of the CLB instance will be returned. The type of CLB instances is divided into public network and private network. For details, see the product type in the product description.
Note: (1) Apply for CLB instances in the specified AZ and adopt cross-zone disaster recovery. (2) The default purchase quota for an account in each region is: 100 for public network and 100 for private network.
This API is an asynchronous API. After the API returns a call success message, the DescribeLoadBalancers API can be used to query the status of the CLB instance (such as creating or normal) to determine whether the creation is successful.

Default interface request rate limit: 20 times per second.

Interface update time: 2024-02-22 11:26:37.

The interface only verifies the signature but does not authenticate.

2. Input Parameters

The following list of request parameters only lists interface request parameters and some public parameters, For a complete list of public parameters see public request parameters.

Parameter name	Required	Allow NULL	Type	Description
Action	Yes	No	String	Public parameter, The value of this interface: CreateLoadBalancer
Version	Yes	No	String	Public parameter, The value of this interface: 2018-03-17
Region	No	No	String	Public parameter,Regional information A list of supported geographies is available through the DescribeRegions interface
LoadBalancerType	Yes	No	String	CLB instance network type: OPEN: Public network; INTERNAL: Private network. Example: OPEN
Forward	No	No	Int64	The type of CLB instance. 1: General CLB instance. Currently, only 1 is supported. Example: 1
LoadBalancerName	No	No	String	CLB instance name, which takes effect only when one instance is to be created. Rule: 1-50 English letters, digits, dashes (-), or underscores (_).Note: If this name is the same as that of an existing CLB instance, the system will automatically generate a name for this newly created instance. Example: lb-1
VpcId	No	No	String	ID of the network specified for the target backend load balancing device. For example, vpc-j9x73ko1. This ID can be obtained through the DescribeVpcEx API. If this parameter is not specified, the classic network ("0") is used by default. Example: vpc-30xqjl93
SubnetId	No	No	String	A subnet ID must be specified when you purchase a private network CLB instance in a VPC, and the VIP of this instance will be generated in this subnet. Example: subnet-dva8jj93
ProjectId	No	No	Int64	The ID of the project to which the CLB instance belongs. This parameter can be obtained through the DescribeProject API. If this parameter is not specified, the default project will be used. Example: 0
AddressIPVersion	No	No	String	IP version, applicable only to public network CLB. Valid values: IPV4, IPV6, and IPv6FullChain. Default: IPV4. Example: IPV4
Number	No	No	Uint64	Number of CLB instances to create. Default: 1. Example: 1
MasterZoneId	No	No	String	Primary AZ ID for cross-AZ disaster recovery settings. For example, 100001 or ap-region1-1. Note: The primary AZ loads traffic while the secondary AZ does not load traffic by default. The primary AZ is used only if the latter is unavailable. The list of primary AZs in a region can be queried through the DescribeMasterZones API. Example: ap-region1-1
ZoneId	No	No	String	AZ ID. For example, ap-region1-1. Applicable only to public network CLB. It specifies an AZ for creating CLB instances. This parameter is not used on Private Cloud. Example: ap-region1-1
AnycastZone	No	No	String	Anycast release domain, applicable only to public network CLB. Valid values: ZONE_A and ZONE_B. This parameter is supported only by non-migration bandwidth users. (Deprecated) Example: ZONE_A
InternetAccessible	No	No	InternetAccessible	CLB network billing mode, applicable only to public network CLB instances. Example:View
VipIsp	No	No	String	Applicable only to public network CLB. CMCC
Tags	No	No	Array of TagInfo	Tags the CLB instance when you purchase it. Example:View
ZhiTong	No	No	Bool	Whether passthrough is supported. This parameter is not supported on Private Cloud. Example: true
Vip	No	No	String	Apply for CLB instances for a specified VIP. The TgwGroupName parameter must also be specified. Example: 139.X.X.X
TgwGroupName	No	No	String	Name of the Tgw exclusive cluster Example: tg-1
IsDDos	No	No	Bool	Whether anti-DDoS package binding is supported. This parameter is not supported on Private Cloud. Example: true
BandwidthPackageId	No	No	String	Bandwidth package ID, which is required when the network billing mode is Bandwidth Package Example: bwp-pnbeji93
ExclusiveCluster	No	No	ExclusiveCluster	Exclusive cluster information Example:View
SlaveZoneId	No	No	String	Secondary AZ ID for cross-AZ disaster recovery settings. For example, 100001 or ap-region1-1. Note: The primary AZ loads traffic, while the secondary AZ does not load traffic by default. The latter is used only if the primary AZ is unavailable, and the switch is automatic. The list of primary and secondary AZs in a region can be queried through the DescribeMasterZones API. Example: ap-region1-1
TgwSetLabels	No	No	Array of String	Parameter specific to Private Cloud. A layer-4 cluster tag can be entered for private network CLB application to specify a layer-4 exclusive cluster. Example: ["l4-label1"]
StgwSetLabels	No	No	Array of String	Parameter specific to Private Cloud. A layer-7 cluster tag can be entered for private network CLB application to specify a layer-7 exclusive cluster. Example: ["l7-label1"]
EipAddressId	No	No	String	Unique ID of an EIP, in the format of eip-11112222, which is applicable only for binding EIP to private network CLB. Example: eip-1111jj93
ClusterIds	No	No	Array of String	ID of a local dedicated cluster Example: ["cluster-gbo2jj93"]
IPv6DecoupleVpc	No	No	Int64	Whether to apply for an IPv6 address from the VPC side. 1 indicates decoupling from the VPC and not applying for an address from the VPC side. 0 indicates not decoupling from the VPC and applying for an address from the VPC side. Example: 0

3. Output Parameters

Parameter name	Type	Description
LoadBalancerIds	Array of String	Array of unique CLB instance IDs. Example: ["lb-6efswuxa"]
RequestId	String	Unique request id, Each request will return. The RequestId of the request needs to be provided when locating the problem.

4. Error Code

The following lists only the error codes related to the interface business logic, For other error codes, see public ErrorCode.

Error Code	Description
InvalidParameter	Wrong parameter.
InvalidParameterValue	Incorrect parameter value
InternalError	Internal Error
UnauthorizedOperation	Unauthorized operation
ResourceInsufficient	Insufficient resources
LimitExceeded	Quota limit reached
FailedOperation	Failed to operate.
InvalidParameter.FormatError	Incorrect parameter format
InvalidParameterValue.Length	Invalid parameter length.
MissingParameter	Missing parameter error.
InvalidParameterValue.Range	Invalid parameter value range.

CreateRule

1. Interface Description

Interface request domain name: clb.api3.convergecloud.com.

This API (CreateRule) is used to create a forwarding rule under an existing layer-7 CLB listener, where real servers must be bound to the rule instead of the listener.
This is an async API. After it returns a successful result, you need to call the DescribeTaskStatus API with the returned RequestID as an input parameter to check whether this task is successful.

Default interface request rate limit: 20 times per second.

Interface update time: 2020-01-10 20:35:19.

The interface only verifies the signature but does not authenticate.

2. Input Parameters

The following list of request parameters only lists interface request parameters and some public parameters, For a complete list of public parameters see public request parameters.

Parameter name	Required	Allow NULL	Type	Description
Action	Yes	No	String	Public parameter, The value of this interface: CreateRule
Version	Yes	No	String	Public parameter, The value of this interface: 2018-03-17
Region	No	No	String	Public parameter,Regional information A list of supported geographies is available through the DescribeRegions interface
LoadBalancerId	Yes	No	String	CLB instance ID Example: lb-cuxwji93
ListenerId	Yes	No	String	Listener ID Example: lbl-4fbxji93*
Rules	Yes	No	Array of RuleInput	Information of the new forwarding rule Example:View

3. Output Parameters

Parameter name	Type	Description
LocationIds	Array of String	Unique identification array of created forwarding rules Example: ["loc-ho6lvh8m"]
RequestId	String	Unique request id, Each request will return. The RequestId of the request needs to be provided when locating the problem.

4. Error Code

The following lists only the error codes related to the interface business logic, For other error codes, see public ErrorCode.

Error Code	Description
InvalidParameter	Wrong parameter.
InvalidParameterValue	Incorrect parameter value
InternalError	Internal Error
UnauthorizedOperation	Unauthorized operation
ResourceInsufficient	Insufficient resources
LimitExceeded	Quota limit reached
FailedOperation	Failed to operate.
InvalidParameter.FormatError	Incorrect parameter format
InvalidParameterValue.Length	Invalid parameter length.
MissingParameter	Missing parameter error.

DeleteCert

1. Interface Description

Interface request domain name: clb.api3.convergecloud.com.

Delete a certificate on the tenant-side CLB

Default interface request rate limit: 20 times per second.

Interface update time: 2021-08-05 17:43:25.

The interface is both signature verification and authentication.

2. Input Parameters

The following list of request parameters only lists interface request parameters and some public parameters, For a complete list of public parameters see public request parameters.

Parameter name	Required	Allow NULL	Type	Description
Action	Yes	No	String	Public parameter, The value of this interface: DeleteCert
Version	Yes	No	String	Public parameter, The value of this interface: 2018-03-17
Region	No	No	String	Public parameter,Regional information A list of supported geographies is available through the DescribeRegions interface
CertIds	Yes	No	Array of String	Certificate ID List Example: ["GpW7fKkH"]

3. Output Parameters

Parameter name	Type	Description
RequestId	String	Unique request id, Each request will return. The RequestId of the request needs to be provided when locating the problem.

4. Error Code

The following lists only the error codes related to the interface business logic, For other error codes, see public ErrorCode.

Error Code	Description
UnauthorizedOperation	Unauthorized operation
FailedOperation	Failed to operate.
InternalError	Internal Error

DeleteListener

1. Interface Description

Interface request domain name: clb.api3.convergecloud.com.

This API is used to delete listeners (layer-4 and layer-7) under a CLB instance.
This is an async API. After it returns a successful result, you need to call the DescribeTaskStatus API with the returned RequestID as an input parameter to check whether this task is successful.

Default interface request rate limit: 20 times per second.

Interface update time: 2020-01-10 20:37:25.

The interface only verifies the signature but does not authenticate.

2. Input Parameters

The following list of request parameters only lists interface request parameters and some public parameters, For a complete list of public parameters see public request parameters.

Parameter name	Required	Allow NULL	Type	Description
Action	Yes	No	String	Public parameter, The value of this interface: DeleteListener
Version	Yes	No	String	Public parameter, The value of this interface: 2018-03-17
Region	No	No	String	Public parameter,Regional information A list of supported geographies is available through the DescribeRegions interface
LoadBalancerId	Yes	No	String	CLB instance ID Example: lb-cuxwji93
ListenerId	Yes	No	String	The listener ID to delete Example: lbl-4udzji93

3. Output Parameters

Parameter name	Type	Description
RequestId	String	Unique request id, Each request will return. The RequestId of the request needs to be provided when locating the problem.

4. Error Code

The following lists only the error codes related to the interface business logic, For other error codes, see public ErrorCode.

Error Code	Description
InvalidParameter	Wrong parameter.
InvalidParameterValue	Incorrect parameter value
InternalError	Internal Error
UnauthorizedOperation	Unauthorized operation
FailedOperation	Failed to operate.
InvalidParameter.FormatError	Incorrect parameter format
InvalidParameterValue.Length	Invalid parameter length.

DeleteLoadBalancer

1. Interface Description

Interface request domain name: clb.api3.convergecloud.com.

This API (DeleteLoadBalancer) is used to delete one or more specified CLB instances.
This is an async API. After it returns a successful result, you need to call the DescribeTaskStatus API with the returned RequestID as an input parameter to check whether this task is successful.

Default interface request rate limit: 20 times per second.

Interface update time: 2020-01-10 20:37:48.

The interface is both signature verification and authentication.

2. Input Parameters

The following list of request parameters only lists interface request parameters and some public parameters, For a complete list of public parameters see public request parameters.

Parameter name	Required	Allow NULL	Type	Description
Action	Yes	No	String	Public parameter, The value of this interface: DeleteLoadBalancer
Version	Yes	No	String	Public parameter, The value of this interface: 2018-03-17
Region	No	No	String	Public parameter,Regional information A list of supported geographies is available through the DescribeRegions interface
LoadBalancerIds	Yes	No	Array of String	Array of IDs of the CLB instances to be deleted. Array length limit: 20 Example: ["lb-hsb93u5o"]

3. Output Parameters

Parameter name	Type	Description
RequestId	String	Unique request id, Each request will return. The RequestId of the request needs to be provided when locating the problem.

4. Error Code

The following lists only the error codes related to the interface business logic, For other error codes, see public ErrorCode.

Error Code	Description
InvalidParameter	Wrong parameter.
InvalidParameterValue	Incorrect parameter value
InternalError	Internal Error
UnauthorizedOperation	Unauthorized operation
ResourceInsufficient	Insufficient resources
LimitExceeded	Quota limit reached
FailedOperation	Failed to operate.
InvalidParameter.LBIdNotFound	CLB instance ID error.
InvalidParameter.FormatError	Incorrect parameter format
InvalidParameterValue.Length	Invalid parameter length.

DeleteRewrite

1. Interface Description

Interface request domain name: clb.api3.convergecloud.com.

This API (DeleteRewrite) is used to delete the redirection relationship between specified forwarding rules.
This is an async API. After it returns a successful result, you need to call the DescribeTaskStatus API with the returned RequestID as an input parameter to check whether this task is successful.

Default interface request rate limit: 20 times per second.

Interface update time: 2020-01-10 20:29:51.

The interface only verifies the signature but does not authenticate.

2. Input Parameters

The following list of request parameters only lists interface request parameters and some public parameters, For a complete list of public parameters see public request parameters.

Parameter name	Required	Allow NULL	Type	Description
Action	Yes	No	String	Public parameter, The value of this interface: DeleteRewrite
Version	Yes	No	String	Public parameter, The value of this interface: 2018-03-17
Region	No	No	String	Public parameter,Regional information A list of supported geographies is available through the DescribeRegions interface
LoadBalancerId	Yes	No	String	CLB instance ID Example: lb-51jjji93
SourceListenerId	Yes	No	String	Source listener ID Example: lbl-r4iaji93
TargetListenerId	Yes	No	String	Target listener ID Example: lbl-0nonji93
RewriteInfos	Yes	No	Array of RewriteLocationMap	Redirection relationship between forwarding rules Example:View

3. Output Parameters

Parameter name	Type	Description
RequestId	String	Unique request id, Each request will return. The RequestId of the request needs to be provided when locating the problem.

4. Error Code

The following lists only the error codes related to the interface business logic, For other error codes, see public ErrorCode.

Error Code	Description
InvalidParameter	Wrong parameter.
InvalidParameterValue	Incorrect parameter value
InternalError	Internal Error
UnauthorizedOperation	Unauthorized operation
ResourceInsufficient	Insufficient resources
LimitExceeded	Quota limit reached
FailedOperation	Failed to operate.
InvalidParameter.LBIdNotFound	CLB instance ID error.
InvalidParameter.SomeRewriteNotFound	Some redirection rules do not exist.
InvalidParameter.FormatError	Incorrect parameter format
InvalidParameterValue.Length	Invalid parameter length.

DeleteRule

1. Interface Description

Interface request domain name: clb.api3.convergecloud.com.

This API (DeleteRule) is used to delete a forwarding rule under a layer-7 CLB instance listener.
This is an async API. After it returns a successful result, you need to call the DescribeTaskStatus API with the returned RequestID as an input parameter to check whether this task is successful.

Default interface request rate limit: 20 times per second.

Interface update time: 2021-04-06 21:37:30.

The interface only verifies the signature but does not authenticate.

2. Input Parameters

The following list of request parameters only lists interface request parameters and some public parameters, For a complete list of public parameters see public request parameters.

Parameter name	Required	Allow NULL	Type	Description
Action	Yes	No	String	Public parameter, The value of this interface: DeleteRule
Version	Yes	No	String	Public parameter, The value of this interface: 2018-03-17
Region	No	No	String	Public parameter,Regional information A list of supported geographies is available through the DescribeRegions interface
LoadBalancerId	Yes	No	String	CLB instance ID Example: lb-cuxw2rm0
ListenerId	Yes	No	String	CLB listener ID Example: lbl-4fbxq45k
LocationIds	No	No	Array of String	Array of IDs of the forwarding rules to be deleted Example: ["loc-5lc6ji93"]
Domain	No	No	String	Domain name of the forwarding rule to be deleted. This parameter will not take effect when the LocationIds parameter is provided. Example: foo.net
Url	No	No	String	Forwarding path of the forwarding rule to be deleted. This parameter will not take effect when the LocationIds parameter is provided. Example: /bar2
NewDefaultServerDomain	No	No	String	A default domain name must be configured for the listener. When you need to delete the default domain name, you can use this parameter to specify a new default domain name. Example: www.default.example.com

3. Output Parameters

Parameter name	Type	Description
RequestId	String	Unique request id, Each request will return. The RequestId of the request needs to be provided when locating the problem.

4. Error Code

The following lists only the error codes related to the interface business logic, For other error codes, see public ErrorCode.

Error Code	Description
InvalidParameter	Wrong parameter.
InvalidParameter.FormatError	Incorrect parameter format
InvalidParameterValue	Incorrect parameter value
UnauthorizedOperation	Unauthorized operation
InvalidParameterValue.Length	Invalid parameter length.
FailedOperation	Failed to operate.
InternalError	Internal Error

DeregisterTargets

1. Interface Description

Interface request domain name: clb.api3.convergecloud.com.

This API (DeregisterTargets) is used to unbind one or more real servers from a CLB listener or forwarding rule. For layer-4 listeners, only the listener ID needs to be specified. For layer-7 listeners, the forwarding rule also needs to be specified through LocationId or Domain+Url.
This is an async API. After it returns a successful result, you need to call the DescribeTaskStatus API with the returned RequestID as an input parameter to check whether this task is successful.

Default interface request rate limit: 20 times per second.

Interface update time: 2020-01-10 20:36:52.

The interface only verifies the signature but does not authenticate.

2. Input Parameters

The following list of request parameters only lists interface request parameters and some public parameters, For a complete list of public parameters see public request parameters.

Parameter name	Required	Allow NULL	Type	Description
Action	Yes	No	String	Public parameter, The value of this interface: DeregisterTargets
Version	Yes	No	String	Public parameter, The value of this interface: 2018-03-17
Region	No	No	String	Public parameter,Regional information A list of supported geographies is available through the DescribeRegions interface
LoadBalancerId	Yes	No	String	CLB instance ID. For example, lb-j9x73ko1. Example: lb-cuxwji93
ListenerId	Yes	No	String	Listener ID. For example, lbl-j9x73ko1. Example: lbl-d1ubji93
LocationId	No	No	String	Forwarding rule ID. For example, loc- j9x73ko1. Either this parameter or Domain+Url is required to unbind a layer-7 forwarding rule from servers. Example: loc-j9x7ji93
Domain	No	No	String	Domain name of the target rule. This parameter will not take effect when the LocationId parameter is provided. Example: www.aaa.com
Url	No	No	String	The URL of the target rule. This parameter will not take effect when the LocationId parameter is provided. Example: /home
Targets	Yes	No	Array of Target	List of real servers to be unbound. Array length limit: 20 Example:View

3. Output Parameters

Parameter name	Type	Description
RequestId	String	Unique request id, Each request will return. The RequestId of the request needs to be provided when locating the problem.

4. Error Code

The following lists only the error codes related to the interface business logic, For other error codes, see public ErrorCode.

Error Code	Description
InvalidParameter	Wrong parameter.
InvalidParameterValue	Incorrect parameter value

Error Code	Description
InternalError	Internal Error
UnauthorizedOperation	Unauthorized operation
FailedOperation	Failed to operate.
InvalidParameter.FormatError	Incorrect parameter format
InvalidParameterValue.Length	Invalid parameter length.
MissingParameter	Missing parameter error.

DescribeAppIdLabel

1. Interface Description

Interface request domain name: clb.api3.convergecloud.com.

This API is used to query users and cluster tags bound to them.

Default interface request rate limit: 20 times per second.

Interface update time: 2021-03-31 15:26:57.

The interface is both signature verification and authentication.

2. Input Parameters

The following list of request parameters only lists interface request parameters and some public parameters, For a complete list of public parameters see public request parameters.

Parameter name	Required	Allow NULL	Type	Description
Action	Yes	No	String	Public parameter, The value of this interface: DescribeAppIdLabel
Version	Yes	No	String	Public parameter, The value of this interface: 2018-03-17
Region	No	No	String	Public parameter,Regional information A list of supported geographies is available through the DescribeRegions interface

3. Output Parameters

Parameter name	Type	Description
OwnerLabelSet	Array of OwnerLabel	User and tag correspondence Example:View
RequestId	String	Unique request id, Each request will return. The RequestId of the request needs to be provided when locating the problem.

4. Error Code

The following lists only the error codes related to the interface business logic, For other error codes, see public ErrorCode.

Error Code	Description
FailedOperation	Failed to operate.
MissingParameter	Missing parameter error.
UnauthorizedOperation	Unauthorized operation
InternalError	Internal Error
InvalidParameter	Wrong parameter.
InvalidParameterValue	Incorrect parameter value

DescribeCerts

1. Interface Description

Interface request domain name: clb.api3.convergecloud.com.

Query the list of available certificates

Default interface request rate limit: 20 times per second.

Interface update time: 2021-12-17 15:08:53.

The interface is both signature verification and authentication.

2. Input Parameters

The following list of request parameters only lists interface request parameters and some public parameters, For a complete list of public parameters see public request parameters.

Parameter name	Required	Allow NULL	Type	Description
Action	Yes	No	String	Public parameter, The value of this interface: DescribeCerts
Version	Yes	No	String	Public parameter, The value of this interface: 2018-03-17
Region	No	No	String	Public parameter,Regional information A list of supported geographies is available through the DescribeRegions interface
Offset	No	No	Uint64	Data offset. Default: 0. Example: 0
Limit	No	No	Uint64	Number of certificates returned. Default: 20 Example: 20
SearchKey	No	No	String	Search keyword Example: lbname
CertType	No	No	String	Certificate type (CA: Client certificate; SVR: Server certificate) Example: SVR
CertIdList	No	No	Array of String	Certificate ID list Example: ["GpW7fKkH"]

Parameter name	Required	Allow NULL	Type	Description
WithCert	No	No	Bool	Whether to obtain certificate content at the same time Example: true

3. Output Parameters

Parameter name	Type	Description
CertList	Array of CertList	Certificate list Example:View
TotalCount	Uint64	Total certificates Example: 200
RequestId	String	Unique request id, Each request will return. The RequestId of the request needs to be provided when locating the problem.

4. Error Code

The following lists only the error codes related to the interface business logic, For other error codes, see public ErrorCode.

Error Code	Description
UnauthorizedOperation	Unauthorized operation
FailedOperation	Failed to operate.
InternalError	Internal Error
InvalidParameterValue	Incorrect parameter value
InvalidParameterValue.Duplicate	Duplicate parameter values.
InvalidParameter	Wrong parameter.
MissingParameter	Missing parameter error.
InvalidParameterValue.Length	Invalid parameter length.

DescribeCustomizedConfigAssociateList

1. Interface Description

Interface request domain name: clb.api3.convergecloud.com.

This API is used to query the locations, servers, or CLB instances bound to a configuration. If there are domains, the result will be filtered by domain.

Default interface request rate limit: 20 times per second.

Interface update time: 2020-03-16 09:51:04.

The interface only verifies the signature but does not authenticate.

2. Input Parameters

The following list of request parameters only lists interface request parameters and some public parameters, For a complete list of public parameters see public request parameters.

Parameter name	Required	Allow NULL	Type	Description
Action	Yes	No	String	Public parameter, The value of this interface: DescribeCustomizedConfigAssociateList
Version	Yes	No	String	Public parameter, The value of this interface: 2018-03-17
Region	No	No	String	Public parameter,Regional information A list of supported geographies is available through the DescribeRegions interface
UconfigId	No	No	String	Configuration ID Example: pz-fi018waq
Offset	No	No	Int64	Start position of the binding list. Default: 0. Example: 0
Limit	No	No	Int64	Number of binding lists to pull. Default: 20. Example: 3
Domain	No	No	String	Search for Domains Example: a.cc
UconfigIds	No	No	Array of String	Configuration ID list in CLB dimension. Filtering by domain is not supported. Example: [pz-n651fsue]

3. Output Parameters

Parameter name	Type	Description

Parameter name	Type	Description
BindList	Array of BindDetailItem	Binding relationship list Example:View
TotalCount	Int64	Total number of binding relationships Example: 1
RequestId	String	Unique request id, Each request will return. The RequestId of the request needs to be provided when locating the problem.

4. Error Code

The following lists only the error codes related to the interface business logic, For other error codes, see public ErrorCode.

Error Code	Description
InvalidParameter	Wrong parameter.
InvalidParameterValue	Incorrect parameter value
InternalError	Internal Error
UnauthorizedOperation	Unauthorized operation
FailedOperation	Failed to operate.
InvalidParameter.FormatError	Incorrect parameter format
InvalidParameterValue.Length	Invalid parameter length.
MissingParameter	Missing parameter error.

DescribeCustomizedConfigList

1. Interface Description

Interface request domain name: clb.api3.convergecloud.com.

Pull the list of custom configurations. The configurations of the specified type under the user's AppId will be returned.

Default interface request rate limit: 20 times per second.

Interface update time: 2021-04-27 20:50:03.

The interface is both signature verification and authentication.

2. Input Parameters

The following list of request parameters only lists interface request parameters and some public parameters, For a complete list of public parameters see public request parameters.

Parameter name	Required	Allow NULL	Type	Description
Action	Yes	No	String	Public parameter, The value of this interface: DescribeCustomizedConfigList
Version	Yes	No	String	Public parameter, The value of this interface: 2018-03-17
Region	No	No	String	Public parameter,Regional information A list of supported geographies is available through the DescribeRegions interface
ConfigType	Yes	No	String	Configuration type. CLB: CLB dimension; SERVER: Domain name dimension; LOCATION: Rule dimension. Example: CLB
Offset	No	No	Int64	Pagination offset. Default: 0 Example: 0
Limit	No	No	Int64	Number of results. Default: 20 Example: 20
ConfigName	No	No	String	Specify the name of configurations to query. Fuzzy match is supported. Example: custom
UconfigIds	No	No	Array of String	Configuration ID Example: ["pz-j9x73ko1"]

3. Output Parameters

Parameter name	Type	Description
ConfigList	Array of ConfigListItem	Configuration list Example:View
TotalCount	Int64	Number of configurations Example: 1
RequestId	String	Unique request id, Each request will return. The RequestId of the request needs to be provided when locating the problem.

4. Error Code

The following lists only the error codes related to the interface business logic, For other error codes, see public ErrorCode.

Error Code	Description
InvalidParameter	Wrong parameter.
InvalidParameterValue	Incorrect parameter value
InternalError	Internal Error
UnauthorizedOperation	Unauthorized operation
ResourceInsufficient	Insufficient resources
LimitExceeded	Quota limit reached
FailedOperation	Failed to operate.
InvalidParameter.LBIdNotFound	CLB instance ID error.
InvalidParameter.SomeRewriteNotFound	Some redirection rules do not exist.
InvalidParameter.ListenerIdNotFound	Listener ID error.
InvalidParameter.LocationNotFound	Unable to find eligible forwarding rules.
InvalidParameter.PortCheckFailed	Listener port check due to port conflicts or other reasons.
InvalidParameter.RewriteAlreadyExist	The forwarding rule has been bound to a redirection relationship.
InvalidParameter.ProtocolCheckFailed	Listener protocol check failed because, for example, the protocol used is incompatible with the corresponding operation.
InvalidParameter.FormatError	Incorrect parameter format
InvalidParameterValue.Length	Invalid parameter length.
InvalidParameterValue.Duplicate	Duplicate parameter values.
InvalidParameter.RegionNotFound	Invalid region.
MissingParameter	Missing parameter error.

DescribeIspInfo

1. Interface Description

Interface request domain name: clb.api3.convergecloud.com.

Query the ISPs supported in the current region

Default interface request rate limit: 20 times per second.

Interface update time: 2021-04-16 11:23:56.

The interface is both signature verification and authentication.

2. Input Parameters

The following list of request parameters only lists interface request parameters and some public parameters, For a complete list of public parameters see public request parameters.

Parameter name	Required	Allow NULL	Type	Description
Action	Yes	No	String	Public parameter, The value of this interface: DescribeIspInfo
Version	Yes	No	String	Public parameter, The value of this interface: 2018-03-17
Region	No	No	String	Public parameter,Regional information A list of supported geographies is available through the DescribeRegions interface

3. Output Parameters

Parameter name	Type	Description
IspSet	Array of IspSet	ISP information list Example:View
RequestId	String	Unique request id, Each request will return. The RequestId of the request needs to be provided when locating the problem.

4. Error Code

The following lists only the error codes related to the interface business logic, For other error codes, see public ErrorCode.

Error Code	Description
UnauthorizedOperation	Unauthorized operation
InvalidParameter.RegionNotFound	Invalid region.
InternalError	Internal Error

DescribeListenerListByCertId

1. Interface Description

Interface request domain name: clb.api3.convergecloud.com.

This API is used to obtain the bound listener by certificate ID.

Default interface request rate limit: 20 times per second.

Interface update time: 2024-11-22 10:35:55.

The interface only verifies the signature but does not authenticate.

2. Input Parameters

The following list of request parameters only lists interface request parameters and some public parameters, For a complete list of public parameters see public request parameters.

Parameter name	Required	Allow NULL	Type	Description
Action	Yes	No	String	Public parameter, The value of this interface: DescribeListenerListByCertId
Version	Yes	No	String	Public parameter, The value of this interface: 2018-03-17
Region	No	No	String	Public parameter,Regional information A list of supported geographies is available through the DescribeRegions interface
CertIds	Yes	No	Array of String	Array of certificate ID. Example: ["ke93l2xn","84jbxme2"]

3. Output Parameters

Parameter name	Type	Description
CertSet	Array of CertIdRelatedWithListener	Result set. Example:View

Parameter name	Type	Description
RequestId	String	Unique request id, Each request will return. The RequestId of the request needs to be provided when locating the problem.

4. Error Code

The following lists only the error codes related to the interface business logic, For other error codes, see public ErrorCode.

Error Code	Description
InvalidParameter	Wrong parameter.
InternalError	Internal Error
InvalidParameterValue	Incorrect parameter value
UnauthorizedOperation	Unauthorized operation
FailedOperation	Failed to operate.
InvalidParameter.FormatError	Incorrect parameter format
InvalidParameterValue.Length	Invalid parameter length.

DescribeListeners

1. Interface Description

Interface request domain name: clb.api3.convergecloud.com.

This API (DescribeListeners) is used to obtain the list of listeners by CLB ID, listener protocol, or listener port. If no filter is specified, all listeners for the CLB instance will be returned.

Default interface request rate limit: 20 times per second.

Interface update time: 2024-07-22 15:10:58.

The interface only verifies the signature but does not authenticate.

2. Input Parameters

The following list of request parameters only lists interface request parameters and some public parameters, For a complete list of public parameters see public request parameters.

Parameter name	Required	Allow NULL	Type	Description
Action	Yes	No	String	Public parameter, The value of this interface: DescribeListeners
Version	Yes	No	String	Public parameter, The value of this interface: 2018-03-17
Region	No	No	String	Public parameter,Regional information A list of supported geographies is available through the DescribeRegions interface
LoadBalancerId	Yes	No	String	CLB instance ID Example: lb-aniqji93
ListenerIds	No	No	Array of String	Array of CLB listener IDs to query Example: ["lbl-8cnlji93","lbl-we2d"]
Protocol	No	No	String	The protocol type of the listener to query. Valid value: TCP Example: TCP
Port	No	No	Int64	Port of listener to query Example: 23
Offset	No	No	Int64	Display start offset. The default value is 0. Example: 10

Parameter name	Required	Allow NULL	Type	Description
Limit	No	No	Int64	Display limit. If not specified, all will be obtained. Example: 10
Filters	No	No	Array of Filter	Filter array Example:View
SearchKey	No	No	String	Name fuzzy search Example: jfhadsihfgiegs0dfhje2

3. Output Parameters

Parameter name	Type	Description
Listeners	Array of Listener	Listener list Example:View
TotalCount	Uint64	Total number of listeners Example: 3
RequestId	String	Unique request id, Each request will return. The RequestId of the request needs to be provided when locating the problem.

4. Error Code

The following lists only the error codes related to the interface business logic, For other error codes, see public ErrorCode.

Error Code	Description
FailedOperation	Failed to operate.
InvalidParameter	Wrong parameter.
InvalidParameter.FormatError	Incorrect parameter format
InvalidParameter.LBIdNotFound	CLB instance ID error.
UnauthorizedOperation	Unauthorized operation
InternalError	Internal Error
InvalidParameterValue	Incorrect parameter value

Error Code	Description
InvalidParameterValue.Length	Invalid parameter length.
InvalidParameterValue.InvalidFilter	Filter parameter input error.

DescribeLoadBalancers

1. Interface Description

Interface request domain name: clb.api3.convergecloud.com.

Query the list of CLB instances in a region

Default interface request rate limit: 20 times per second.

Interface update time: 2020-01-10 19:40:46.

The interface only verifies the signature but does not authenticate.

2. Input Parameters

The following list of request parameters only lists interface request parameters and some public parameters, For a complete list of public parameters see public request parameters.

Parameter name	Required	Allow NULL	Type	Description
Action	Yes	No	String	Public parameter, The value of this interface: DescribeLoadBalancers
Version	Yes	No	String	Public parameter, The value of this interface: 2018-03-17
Region	No	No	String	Public parameter,Regional information A list of supported geographies is available through the DescribeRegions interface
LoadBalancerIds	No	No	Array of String	CLB instance ID. There can be up to 20 IDs. Example: ["lb-rbw5jl93"]
LoadBalancerType	No	No	String	CLB instance network type: OPEN: Public network; INTERNAL: Private network. Example: OPEN
Forward	No	No	Int64	CLB instance type. 1: general CLB instance. If this parameter is not specified, all types of CLB instances will be queried. Example: 1
LoadBalancerName	No	No	String	CLB instance name. Example: lb_internal
Domain	No	No	String	Domain name assigned by the cloud platform for the CLB instance. Example: lb-6m46rm32-sejl93o0q2g.clb.ap-region1.com
LoadBalancerVips	No	No	Array of String	VIP address of a CLB instance (there can be multiple addresses). Example: ["44.23.32.32"]
BackendPublicIps	No	No	Array of String	Public network IP of the real server bound to CLB. Example: ["102.23.32.43"]
BackendPrivateIps	No	No	Array of String	Private network IP of the real server bound to CLB. Example: ["172.16.76.87"]
Offset	No	No	Int64	Data offset. Default: 0. Example: 0
Limit	No	No	Int64	Number of CLB instances returned. Default: 20. Maximum: 100. Example: 20
OrderBy	No	No	String	Sorting parameter, supporting the following fields: LoadBalancerName, CreateTime, Domain, and LoadBalancerType. LoadBalancerName: LB instance name, CreateTime: creation time, Domain: domain name, LoadBalancerType: instance type. Example: LoadBalancerName
OrderType	No	No	Int64	1: Reverse; 0: Sequential. Default: reverse by creation time Example: 0
SearchKey	No	No	String	Search field, supporting fuzzy match by name, domain, and VIP. Example: lb-name
ProjectId	No	No	Int64	The ID of the project to which the CLB instance belongs. This parameter can be obtained through the DescribeProject API. Example: 1
WithRs	No	No	Int64	Whether CLB is bound to a real server. 0: No; 1: Yes; -1: Query all. Example: -1
VpcId	No	No	String	Unique ID of the VPC to which a CLB instance belongs, such as vpc-bhqkbhdx . Example: vpc-bhqkji93
SecurityGroup	No	No	String	Security group ID, such as sg-m1cc9123 Example: sg-m1ccji93
MasterZone	No	No	String	Primary AZ ID. For example, "100001". Example: 100001
Filters	No	No	Array of Filter	The maximum number of Filters per request is 10, and the maximum number of Filter.Values per request is 100. Detailed filter conditions are as follows: • internet-charge-type - String - required: no - (filter condition) filter by CLB network billing mode, including "BANDWIDTH_PREPAID","TRAFFIC_POSTPAID_BY_HOUR","BANDWIDTH_POSTPAID_BY_HOUR","BANDWIDTH_PACKAGE". • master-zone-id - String - required: no - (filter condition) filter by CLB primary AZ ID. For example, "100001". • tag-key - String - required: no - (filter condition) filter by CLB tag key. Example:View

3. Output Parameters

Parameter name	Type	Description
TotalCount	Uint64	Total number of CLB instances that meet the filter conditions. This value is unrelated to the Limit parameter in the input. Example: 0
LoadBalancerSet	Array of LoadBalancer	Returned CLB instance array. Example:View
RequestId	String	Unique request id, Each request will return. The RequestId of the request needs to be provided when locating the problem.

4. Error Code

The following lists only the error codes related to the interface business logic, For other error codes, see public ErrorCode.

Error Code	Description
InvalidParameter	Wrong parameter.
InvalidParameterValue	Incorrect parameter value
InternalError	Internal Error
UnauthorizedOperation	Unauthorized operation
FailedOperation	Failed to operate.
InvalidParameter.FormatError	Incorrect parameter format
InvalidParameterValue.Length	Invalid parameter length.
InvalidParameterValue.InvalidFilter	Filter parameter input error.

DescribeRewrite

1. Interface Description

Interface request domain name: clb.api3.convergecloud.com.

This API (DescribeRewrite) is used to query the redirection relationship between the forwarding rules of a CLB instance by instance ID. If no listener ID or forwarding rule ID is specified, all redirection relationships in the instance will be returned.

Default interface request rate limit: 20 times per second.

Interface update time: 2020-01-10 20:29:39.

The interface only verifies the signature but does not authenticate.

2. Input Parameters

The following list of request parameters only lists interface request parameters and some public parameters, For a complete list of public parameters see public request parameters.

Parameter name	Required	Allow NULL	Type	Description
Action	Yes	No	String	Public parameter, The value of this interface: DescribeRewrite
Version	Yes	No	String	Public parameter, The value of this interface: 2018-03-17
Region	No	No	String	Public parameter,Regional information A list of supported geographies is available through the DescribeRegions interface
LoadBalancerId	Yes	No	String	CLB instance ID Example: lb-qc2iji93
SourceListenerIds	No	No	Array of String	Array of CLB listener IDs Example: ["lbl-j36caqde"]
SourceLocationIds	No	No	Array of String	Array of CLB forwarding rule IDs Example: ["loc-5t7526km"]

3. Output Parameters

Parameter name	Type	Description
RewriteSet	Array of RuleOutput	Array of redirect forwarding rules. If there are no redirect rules, an empty array is returned Example:View
RequestId	String	Unique request id, Each request will return. The RequestId of the request needs to be provided when locating the problem.

4. Error Code

The following lists only the error codes related to the interface business logic, For other error codes, see public ErrorCode.

Error Code	Description
InvalidParameter	Wrong parameter.
InvalidParameterValue	Incorrect parameter value
InternalError	Internal Error
UnauthorizedOperation	Unauthorized operation
ResourceInsufficient	Insufficient resources
LimitExceeded	Quota limit reached
FailedOperation	Failed to operate.
InvalidParameter.FormatError	Incorrect parameter format
InvalidParameterValue.Length	Invalid parameter length.
InvalidParameter.LBIdNotFound	CLB instance ID error.

DescribeSubUinQuotas

1. Interface Description

Interface request domain name: clb.api3.convergecloud.com.

The main tenant queries the sub-account quotas. If there are no sub-accounts, the quota of the root account will be returned.

Default interface request rate limit: 20 times per second.

Interface update time: 2021-03-01 19:34:53.

The interface is both signature verification and authentication.

2. Input Parameters

The following list of request parameters only lists interface request parameters and some public parameters, For a complete list of public parameters see public request parameters.

Parameter name	Required	Allow NULL	Type	Description
Action	Yes	No	String	Public parameter, The value of this interface: DescribeSubUinQuotas
Version	Yes	No	String	Public parameter, The value of this interface: 2018-03-17
Region	No	No	String	Public parameter,Regional information A list of supported geographies is available through the DescribeRegions interface
SubUin	No	No	Array of String	Sub-account UIN list Example: ["110000004918"]

3. Output Parameters

Parameter name	Type	Description
QuotaData	Array of	List of types and sub-account quotas under the root account

Parameter name	Type	Description
	QuotaData	Example:View
RequestId	String	Unique request id, Each request will return. The RequestId of the request needs to be provided when locating the problem.

4. Error Code

The following lists only the error codes related to the interface business logic, For other error codes, see public ErrorCode.

Error Code	Description
FailedOperation	Failed to operate.
InvalidParameter	Wrong parameter.
InternalError	Internal Error

DescribeTargetHealth

1. Interface Description

Interface request domain name: clb.api3.convergecloud.com.

This API (DescribeTargetHealth) is used to query the health check result of a real server of a CLB instance. It is not supported by Classic CLB.

Default interface request rate limit: 20 times per second.

Interface update time: 2020-01-10 20:44:19.

The interface only verifies the signature but does not authenticate.

2. Input Parameters

The following list of request parameters only lists interface request parameters and some public parameters, For a complete list of public parameters see public request parameters.

Parameter name	Required	Allow NULL	Type	Description
Action	Yes	No	String	Public parameter, The value of this interface: DescribeTargetHealth
Version	Yes	No	String	Public parameter, The value of this interface: 2018-03-17
Region	No	No	String	Public parameter,Regional information A list of supported geographies is available through the DescribeRegions interface
LoadBalancerIds	Yes	No	Array of String	The CLB instance ID list to query Example: ["lb-qc2iq5yc"]

3. Output Parameters

Parameter name	Type	Description

Parameter name	Type	Description
LoadBalancers	Array of LoadBalancerHealth	CLB instance list Example:View
RequestId	String	Unique request id, Each request will return. The RequestId of the request needs to be provided when locating the problem.

4. Error Code

The following lists only the error codes related to the interface business logic, For other error codes, see public ErrorCode.

Error Code	Description
InvalidParameter	Wrong parameter.
InvalidParameterValue	Incorrect parameter value
InternalError	Internal Error
UnauthorizedOperation	Unauthorized operation
FailedOperation	Failed to operate.
InvalidParameter.LBIdNotFound	CLB instance ID error.
InvalidParameter.FormatError	Incorrect parameter format
InvalidParameterValue.Length	Invalid parameter length.

DescribeTargets

1. Interface Description

Interface request domain name: clb.api3.convergecloud.com.

This API (DescribeTargets) is used to query the list of real servers bound to some listeners of a CLB instance.

Default interface request rate limit: 20 times per second.

Interface update time: 2020-01-10 20:37:15.

The interface only verifies the signature but does not authenticate.

2. Input Parameters

The following list of request parameters only lists interface request parameters and some public parameters, For a complete list of public parameters see public request parameters.

Parameter name	Required	Allow NULL	Type	Description
Action	Yes	No	String	Public parameter, The value of this interface: DescribeTargets
Version	Yes	No	String	Public parameter, The value of this interface: 2018-03-17
Region	No	No	String	Public parameter,Regional information A list of supported geographies is available through the DescribeRegions interface
LoadBalancerId	Yes	No	String	CLB instance ID Example: lb-10iq9lou
ListenerIds	No	No	Array of String	List of listener IDs Example: ["lbl-4fo6k8na"]
Protocol	No	No	String	Listener protocol type, supporting TCP, UDP, HTTP, HTTPS, and TCP_SSL. Example: HTTP
Port	No	No	Int64	Listener port Example: 80

3. Output Parameters

Parameter name	Type	Description
Listeners	Array of ListenerBackend	Information of real servers bound to the listener Example:View

Parameter name	Type	Description
RequestId	String	Unique request id, Each request will return. The RequestId of the request needs to be provided when locating the problem.

4. Error Code

The following lists only the error codes related to the interface business logic, For other error codes, see public ErrorCode.

Error Code	Description
InvalidParameter	Wrong parameter.
InvalidParameterValue	Incorrect parameter value
InternalError	Internal Error
UnauthorizedOperation	Unauthorized operation
FailedOperation	Failed to operate.
InvalidParameter.FormatError	Incorrect parameter format
InvalidParameterValue.Length	Invalid parameter length.

DescribeTaskStatus

1. Interface Description

Interface request domain name: clb.api3.convergecloud.com.

This API is used to query the execution status of an async task. After non-query APIs (used to create/delete CLB instances, listeners, or rules or to bind/unbind real servers) are called successfully, this API needs to be used to query whether the task is successful.

Default interface request rate limit: 20 times per second.

Interface update time: 2020-01-10 19:42:49.

The interface is both signature verification and authentication.

2. Input Parameters

The following list of request parameters only lists interface request parameters and some public parameters, For a complete list of public parameters see public request parameters.

Parameter name	Required	Allow NULL	Type	Description
Action	Yes	No	String	Public parameter, The value of this interface: DescribeTaskStatus
Version	Yes	No	String	Public parameter, The value of this interface: 2018-03-17
Region	No	No	String	Public parameter,Regional information A list of supported geographies is available through the DescribeRegions interface
TaskId	No	No	String	Request ID, which is the RequestId parameter returned by the API Example: 55c85074-3e7f-4c6d-864f-673660d4f8de
DealName	No	No	String	Order ID Example: 20220828391256

3. Output Parameters

Parameter name	Type	Description
Status	Int64	Current task status. 0: Successful; 1: Failed; 2: In progress. Example: 0

Parameter name	Type	Description
RequestId	String	Unique request id, Each request will return. The RequestId of the request needs to be provided when locating the problem.

4. Error Code

The following lists only the error codes related to the interface business logic, For other error codes, see public ErrorCode.

Error Code	Description
InvalidParameter	Wrong parameter.
InternalError	Internal Error
FailedOperation	Failed to operate.

InquiryPriceModifyLoadBalancer

1. Interface Description

Interface request domain name: clb.api3.convergecloud.com.

This API (InquiryPriceModifyLoadBalancer) is used to query the price of adjusting the configuration of a CLB instance.

Default interface request rate limit: 20 times per second.

Interface update time: 2022-03-22 10:48:08.

The interface only verifies the signature but does not authenticate.

2. Input Parameters

The following list of request parameters only lists interface request parameters and some public parameters, For a complete list of public parameters see public request parameters.

Parameter name	Required	Allow NULL	Type	Description
Action	Yes	No	String	Public parameter, The value of this interface: InquiryPriceModifyLoadBalancer
Version	Yes	No	String	Public parameter, The value of this interface: 2018-03-17
Region	No	No	String	Public parameter,Regional information A list of supported geographies is available through the DescribeRegions interface
LoadBalancerId	Yes	No	String	CLB instance ID Example: lb-dr0mo6w4
InternetAccessible	Yes	No	InternetAccessible	Modified network bandwidth information Example:View
SlaType	No	No	String	Type of the LCU- supported instance. (This parameter is not supported on Private Cloud.) Example: clb.c3.small

3. Output Parameters

Parameter name	Type	Description
Price	Price	Describe price information Example:View
RequestId	String	Unique request id, Each request will return. The RequestId of the request needs to be provided when locating the problem.

4. Error Code

The following lists only the error codes related to the interface business logic, For other error codes, see public ErrorCode.

Error Code	Description

Error Code	Description
InternalError	Internal Error
InvalidParameter.FormatError	Incorrect parameter format
FailedOperation	Failed to operate.
InvalidParameter	Wrong parameter.
InvalidParameter.LBIdNotFound	CLB instance ID error.
UnauthorizedOperation	Unauthorized operation
InvalidParameterValue	Incorrect parameter value

ManualRewrite

1. Interface Description

Interface request domain name: clb.api3.convergecloud.com.

After the original access address and the redirection address are configured manually, the system will automatically redirect requests made to the original access address to the target address of the corresponding path. Multiple paths can be configured as a redirection policy under one domain name to achieve automatic redirection between HTTP and HTTPS. A redirection policy should meet the following rules: if A has already been redirected to B, then it cannot be redirected to C (unless the original redirection relationship is deleted and a new one is created), and B cannot be redirected to any other addresses.
This is an async API. After it returns a successful result, you need to call the DescribeTaskStatus API with the returned RequestID as an input parameter to check whether this task is successful.

Default interface request rate limit: 20 times per second.

Interface update time: 2020-01-10 20:30:03.

The interface only verifies the signature but does not authenticate.

2. Input Parameters

The following list of request parameters only lists interface request parameters and some public parameters, For a complete list of public parameters see public request parameters.

Parameter name	Required	Allow NULL	Type	Description
Action	Yes	No	String	Public parameter, The value of this interface: ManualRewrite
Version	Yes	No	String	Public parameter, The value of this interface: 2018-03-17
Region	No	No	String	Public parameter,Regional information A list of supported geographies is available through the DescribeRegions interface
LoadBalancerId	Yes	No	String	CLB instance ID Example: lb-51jjji93
SourceListenerId	Yes	No	String	Source listener ID Example: lbl-0nonji93
TargetListenerId	Yes	No	String	Target listener ID Example: lbl-r4iaji93
RewriteInfos	Yes	No	Array of RewriteLocationMap	Redirection relationship between forwarding rules Example:View

3. Output Parameters

Parameter name	Type	Description
RequestId	String	Unique request id, Each request will return. The RequestId of the request needs to be provided when locating the problem.

4. Error Code

The following lists only the error codes related to the interface business logic, For other error codes, see public ErrorCode.

Error Code	Description
InvalidParameter	Wrong parameter.
InvalidParameterValue	Incorrect parameter value
InternalError	Internal Error
UnauthorizedOperation	Unauthorized operation
ResourceInsufficient	Insufficient resources
LimitExceeded	Quota limit reached
FailedOperation	Failed to operate.
InvalidParameter.LBIdNotFound	CLB instance ID error.
InvalidParameter.SomeRewriteNotFound	Some redirection rules do not exist.
InvalidParameter.ListenerIdNotFound	Listener ID error.
InvalidParameter.LocationNotFound	Unable to find eligible forwarding rules.
InvalidParameter.RewriteAlreadyExist	The forwarding rule has been bound to a redirection relationship.
InvalidParameter.ProtocolCheckFailed	Listener protocol check failed because, for example, the protocol used is incompatible with the corresponding operation.
InvalidParameter.FormatError	Incorrect parameter format
InvalidParameterValue.Length	Invalid parameter length.

ModifyCertAlias

1. Interface Description

Interface request domain name: clb.api3.convergecloud.com.

Modify tenant-side certificate remarks

Default interface request rate limit: 20 times per second.

Interface update time: 2021-08-18 18:32:08.

The interface is both signature verification and authentication.

2. Input Parameters

The following list of request parameters only lists interface request parameters and some public parameters, For a complete list of public parameters see public request parameters.

Parameter name	Required	Allow NULL	Type	Description
Action	Yes	No	String	Public parameter, The value of this interface: ModifyCertAlias
Version	Yes	No	String	Public parameter, The value of this interface: 2018-03-17
Region	No	No	String	Public parameter,Regional information A list of supported geographies is available through the DescribeRegions interface
CertId	Yes	No	String	Certificate ID Example: GpW7fKkH
Alias	Yes	No	String	New remarks of the certificate Example: cert-2

3. Output Parameters

Parameter name	Type	Description

Parameter name	Type	Description
RequestId	String	Unique request id, Each request will return. The RequestId of the request needs to be provided when locating the problem.

4. Error Code

The following lists only the error codes related to the interface business logic, For other error codes, see public ErrorCode.

Error Code	Description
UnauthorizedOperation	Unauthorized operation
FailedOperation	Failed to operate.
InternalError	Internal Error
InvalidParameter	Wrong parameter.
InvalidParameterValue.Length	Invalid parameter length.

ModifyDomainAttributes

1. Interface Description

Interface request domain name: clb.api3.convergecloud.com.

This API (ModifyDomainAttributes) is used to modify the domain name-level attributes of a layer-7 listener's forwarding rule, such as changing the domain name, changing the DefaultServer, enabling/disabling HTTP/2, and changing certificates.
This is an async API. After it returns a successful result, you need to call the DescribeTaskStatus API with the returned RequestID as an input parameter to check whether this task is successful.

Default interface request rate limit: 20 times per second.

Interface update time: 2021-04-06 21:40:04.

The interface only verifies the signature but does not authenticate.

2. Input Parameters

The following list of request parameters only lists interface request parameters and some public parameters, For a complete list of public parameters see public request parameters.

Parameter name	Required	Allow NULL	Type	Description
Action	Yes	No	String	Public parameter, The value of this interface: ModifyDomainAttributes
Version	Yes	No	String	Public parameter, The value of this interface: 2018-03-17
Region	No	No	String	Public parameter,Regional information A list of supported geographies is available through the DescribeRegions interface
LoadBalancerId	Yes	No	String	CLB instance ID Example: lb-1wvlji93
ListenerId	Yes	No	String	CLB listener ID Example: lbl-n8mbji93
Domain	Yes	No	String	Domain name (must be a domain name under an already created forwarding rule) Example: foo.net
NewDomain	No	No	String	The new domain name for modification Example: www.new.example.com
Certificate	No	No	CertificateInput	Domain name-related certificate information, applicable only to listeners with SNI enabled. Example:View
Http2	No	No	Bool	Whether to enable HTTP/2. 1: yes, 0: no. Note: HTTP/2 can be enabled only for HTTPS domain names. Example: true
DefaultServer	No	No	Bool	Whether to set as the default domain name. true: Set as the default domain name. 0: Do not set as the default domain name. Note: Only one default domain name can be set under a listener. Example: true
NewDefaultServerDomain	No	No	String	A default domain name must be configured under the listener. To disable the original default domain name, another domain name must be simultaneously designated as the new default domain name. Example: www.yuncloud.example.com

3. Output Parameters

Parameter name	Type	Description
RequestId	String	Unique request id, Each request will return. The RequestId of the request needs to be provided when locating the problem.

4. Error Code

The following lists only the error codes related to the interface business logic, For other error codes, see public ErrorCode.

Error Code	Description
InvalidParameter	Wrong parameter.
InvalidParameter.FormatError	Incorrect parameter format
InvalidParameterValue	Incorrect parameter value
UnauthorizedOperation	Unauthorized operation
InvalidParameterValue.Length	Invalid parameter length.
MissingParameter	Missing parameter error.
FailedOperation	Failed to operate.
InternalError	Internal Error

ModifyListener

1. Interface Description

Interface request domain name: clb.api3.convergecloud.com.

This API (ModifyListener) is used to modify the attributes of a CLB listener, such as listener name, health check parameter, certificate information, and forwarding policy. This API is not supported by Classic CLB.
This is an async API. After it returns a successful result, you need to call the DescribeTaskStatus API with the returned RequestID as an input parameter to check whether this task is successful.

Default interface request rate limit: 20 times per second.

Interface update time: 2019-10-22 12:17:13.

The interface only verifies the signature but does not authenticate.

2. Input Parameters

The following list of request parameters only lists interface request parameters and some public parameters, For a complete list of public parameters see public request parameters.

Parameter name	Required	Allow NULL	Type	Description
Action	Yes	No	String	Public parameter, The value of this interface: ModifyListener
Version	Yes	No	String	Public parameter, The value of this interface: 2018-03-17
Region	No	No	String	Public parameter,Regional information A list of supported geographies is available through the DescribeRegions interface
LoadBalancerId	Yes	No	String	CLB instance ID Example: lb-cuxwjj93
ListenerId	Yes	No	String	CLB listener ID Example: lbl-d1ubji93
ListenerName	No	No	String	New listener name Example: newlis
SessionExpireTime	No	No	Int64	Session persistence time. Unit: sec. Value range: 30-3,600. Default: 0, indicating disabled. This parameter is applicable only to TCP/UDP listeners. Example: 120
HealthCheck	No	No	HealthCheck	A health check parameter, applicable only to TCP/UDP/TCP_SSL listeners Example:View
Certificate	No	No	CertificateInput	Certificate information, applicable only to HTTPS/TCP_SSL listeners Example:View
Scheduler	No	No	String	Listener forwarding mode. Valid values: WRR (weighted round robin) and LEAST_CONN (least connections) Default: WRR. Example: LEAST_CONN
SniSwitch	No	No	Int64	Whether to enable the SNI feature. 1: yes, 0: no. This parameter is applicable only to HTTPS listeners. Note: SNI can be enabled for listeners that do not have it enabled. SNI cannot be disabled for listeners that have it enabled. Example: 0
TargetType	No	No	String	Target real server type. NODE: Bind a general node. Example: NODE
DefaultSerSwitch	No	No	Bool	Whether the listener supports setting the default domain name. 0: no, 1: yes. The default value is 1. Example: false
NewDefaultServerDomain	No	No	String	When the Default Domain Name switch of the listener is enabled, designate any domain name under the listener as the default domain name. This parameter is required and takes effect simultaneously when DefaultSerSwitch is set to true and the listener has existing domain names. Example: foo.net

3. Output Parameters

Parameter name	Type	Description
RequestId	String	Unique request id, Each request will return. The RequestId of the request needs to be provided when locating the problem.

4. Error Code

The following lists only the error codes related to the interface business logic, For other error codes, see public ErrorCode.

Error Code	Description
InvalidParameter	Wrong parameter.
InvalidParameterValue	Incorrect parameter value
InternalError	Internal Error
UnauthorizedOperation	Unauthorized operation
FailedOperation	Failed to operate.
InvalidParameter.FormatError	Incorrect parameter format
InvalidParameterValue.Length	Invalid parameter length.
MissingParameter	Missing parameter error.

ModifyLoadBalancerAttributes

1. Interface Description

Interface request domain name: clb.api3.convergecloud.com.

This API is used to modify the attributes of a CLB instance, such as name and cross-region attributes.

Default interface request rate limit: 20 times per second.

Interface update time: 2020-09-04 14:41:39.

The interface only verifies the signature but does not authenticate.

2. Input Parameters

The following list of request parameters only lists interface request parameters and some public parameters, For a complete list of public parameters see public request parameters.

Parameter name	Required	Allow NULL	Type	Description
Action	Yes	No	String	Public parameter, The value of this interface: ModifyLoadBalancerAttributes
Version	Yes	No	String	Public parameter, The value of this interface: 2018-03-17
Region	No	No	String	Public parameter,Regional information A list of supported geographies is available through the DescribeRegions interface
LoadBalancerId	Yes	No	String	Unique ID of CLB Example: lb-6efswuxa
LoadBalancerName	No	No	String	CLB Instance Name Example: newlbname
TargetRegionInfo	No	No	TargetRegionInfo	Geographical information of the real server bound to CLB Example:View
InternetChargeInfo	No	No	InternetAccessible	A network billing parameter Example:View
LoadBalancerPassToTarget	No	No	Bool	Whether to allow CLB traffic to the target. True: Allow CLB traffic to the target and verify security groups only on CLB. False: Deny CLB traffic to the target and verify security groups on both CLB and backend instances. Example: true
ChargeType	No	No	String	Billing type of the CLB instance, Postpaid: POSTPAID_BY_HOUR; Prepaid: PREPAID. Example: POSTPAID_BY_HOUR
SwitchFlag	No	No	Uint64	Switch between different billing modes. 0: No switch; 1: Switch between prepaid and postpaid options; 2: Switch between postpaid options. Default: 0 Example: 0
PrepaidInfo	No	No	LBChargePrepaid	Prepaid billing attributes of a CLB instance Example:View
ExclusiveCluster	No	No	ExclusiveCluster	Layer-7 cluster list Example:View
SnatPro	No	No	Bool	Whether to enable the feature of cross-region binding 2.0. (This parameter is not supported on Private Cloud.) Example: false

3. Output Parameters

Parameter name	Type	Description
DealName	String	When switching the billing mode for CLB, you can use this parameter to check if the switch task is successful. Example: null
RequestId	String	Unique request id, Each request will return. The RequestId of the request needs to be provided when locating the problem.

4. Error Code

The following lists only the error codes related to the interface business logic, For other error codes, see public ErrorCode.

Error Code	Description
LimitExceeded	Quota limit reached
MissingParameter	Missing parameter error.
FailedOperation	Failed to operate.
InvalidParameter	Wrong parameter.
InvalidParameter.FormatError	Incorrect parameter format
InvalidParameter.LBIdNotFound	CLB instance ID error.
InvalidParameterValue.Duplicate	Duplicate parameter values.
ResourceInsufficient	Insufficient resources
UnauthorizedOperation	Unauthorized operation
InternalError	Internal Error
InvalidParameterValue	Incorrect parameter value
InvalidParameterValue.Length	Invalid parameter length.
InvalidParameter.RegionNotFound	Invalid region.

ModifyRule

1. Interface Description

Interface request domain name: clb.api3.convergecloud.com.

This API (ModifyRule) is used to modify the attributes of a forwarding rule under a layer-7 CLB listener, such as forwarding path, health check attribute, and forwarding policy.
This is an async API. After it returns a successful result, you need to call the DescribeTaskStatus API with the returned RequestID as an input parameter to check whether this task is successful.

Default interface request rate limit: 20 times per second.

Interface update time: 2019-10-22 20:16:49.

The interface only verifies the signature but does not authenticate.

2. Input Parameters

The following list of request parameters only lists interface request parameters and some public parameters, For a complete list of public parameters see public request parameters.

Parameter name	Required	Allow NULL	Type	Description
Action	Yes	No	String	Public parameter, The value of this interface: ModifyRule
Version	Yes	No	String	Public parameter, The value of this interface: 2018-03-17
Region	No	No	String	Public parameter,Regional information A list of supported geographies is available through the DescribeRegions interface
LoadBalancerId	Yes	No	String	CLB instance ID Example: lb-cuxw2rm0
ListenerId	Yes	No	String	CLB listener ID Example: lbl-4fbxq45k
LocationId	Yes	No	String	The ID of the forwarding rule to be modified. Example: loc-9dr7bsl3
Url	No	No	String	New forwarding path of the forwarding rule. This parameter is not required if the URL does not need to be modified. Example: /bar
HealthCheck	No	No	HealthCheck	Health check information Example:View
Scheduler	No	No	String	Request forwarding method of the rule. Valid values: WRR (weighted round robin), LEAST_CONN (least connections), and IP_HASH (IP hash) Default: WRR Example: LEAST_CONN
SessionExpireTime	No	No	Int64	Session persistence Duration Example: 75
ForwardType	No	No	String	Forwarding protocol between the CLB instance and the real server. Default: HTTP. Valid values: HTTP and HTTPS Example: HTTP
TargetType	No	No	String	Target real server type. NODE: Bind a general node. Example: NODE

3. Output Parameters

Parameter name	Type	Description
RequestId	String	Unique request id, Each request will return. The RequestId of the request needs to be provided when locating the problem.

4. Error Code

The following lists only the error codes related to the interface business logic, For other error codes, see public ErrorCode.

Error Code	Description
InvalidParameter	Wrong parameter.
InvalidParameterValue	Incorrect parameter value
InternalError	Internal Error
UnauthorizedOperation	Unauthorized operation
FailedOperation	Failed to operate.
InvalidParameter.FormatError	Incorrect parameter format
InvalidParameterValue.Length	Invalid parameter length.
MissingParameter	Missing parameter error.

ModifyTargetPort

1. Interface Description

Interface request domain name: clb.api3.convergecloud.com.

This API (ModifyTargetPort) is used to modify the port of a real server bound to a listener.
This is an async API. After it returns a successful result, you need to call the DescribeTaskStatus API with the returned RequestID as an input parameter to check whether this task is successful.

Default interface request rate limit: 20 times per second.

Interface update time: 2020-01-10 20:37:58.

The interface only verifies the signature but does not authenticate.

2. Input Parameters

The following list of request parameters only lists interface request parameters and some public parameters, For a complete list of public parameters see public request parameters.

Parameter name	Required	Allow NULL	Type	Description
Action	Yes	No	String	Public parameter, The value of this interface: ModifyTargetPort
Version	Yes	No	String	Public parameter, The value of this interface: 2018-03-17
Region	No	No	String	Public parameter,Regional information A list of supported geographies is available through the DescribeRegions interface
LoadBalancerId	Yes	No	String	CLB instance ID Example: lb-cuxwji93
ListenerId	Yes	No	String	CLB listener ID Example: lbl-d1ubji93
LocationId	No	No	String	Forwarding rule ID. When binding a real server to a layer-7 forwarding rule, you must provide either this parameter or Domain+Url. Example: loc-fi8gji93
Domain	No	No	String	Domain name of the target rule. This parameter will not take effect when the LocationId parameter is provided. Example: www.yuncloud.example.com
Url	No	No	String	The URL of the target rule. This parameter will not take effect when the LocationId parameter is provided. Example: /index
Targets	Yes	No	Array of Target	List of real servers for which the ports are to be modified Example:View
NewPort	Yes	No	Int64	The new port through which the real server is bound to the listener or forwarding rule Example: 334

3. Output Parameters

Parameter name	Type	Description
RequestId	String	Unique request id, Each request will return. The RequestId of the request needs to be provided when locating the problem.

4. Error Code

The following lists only the error codes related to the interface business logic, For other error codes, see public ErrorCode.

Error Code	Description
InvalidParameter	Wrong parameter.
InvalidParameterValue	Incorrect parameter value
InternalError	Internal Error
UnauthorizedOperation	Unauthorized operation
FailedOperation	Failed to operate.
InvalidParameter.FormatError	Incorrect parameter format
InvalidParameterValue.Length	Invalid parameter length.

ModifyTargetWeight

1. Interface Description

Interface request domain name: clb.api3.convergecloud.com.

This API (ModifyTargetWeight) is used to modify the forwarding weight of a real server bound to a CLB instance.
This is an async API. After it returns a successful result, you need to call the DescribeTaskStatus API with the returned RequestID as an input parameter to check whether this task is successful.

Default interface request rate limit: 20 times per second.

Interface update time: 2021-04-27 18:07:01.

The interface only verifies the signature but does not authenticate.

2. Input Parameters

The following list of request parameters only lists interface request parameters and some public parameters, For a complete list of public parameters see public request parameters.

Parameter name	Required	Allow NULL	Type	Description
Action	Yes	No	String	Public parameter, The value of this interface: ModifyTargetWeight
Version	Yes	No	String	Public parameter, The value of this interface: 2018-03-17
Region	No	No	String	Public parameter,Regional information A list of supported geographies is available through the DescribeRegions interface
LoadBalancerId	Yes	No	String	CLB instance ID Example: lb-cuxwji93
ListenerId	Yes	No	String	CLB listener ID Example: lbl-d1ubji93
LocationId	No	No	String	Forwarding rule ID. When binding a real server to a layer-7 forwarding rule, you must provide either this parameter or Domain+Url. Example: loc-fi8gji93
Domain	No	No	String	Domain name of the target rule. This parameter will not take effect when the LocationId parameter is provided. Example: www.new.example.com
Url	No	No	String	The URL of the target rule. This parameter will not take effect when the LocationId parameter is provided. Example: /index
Targets	Yes	No	Array of Target	The list of real servers for which the weights are to be modified Example:View
Weight	No	No	Int64	New forwarding weight of a real server. Value range: 0-100. Default: 10. This parameter will not take effect when the Targets.Weight parameter is provided. Example: 8

3. Output Parameters

Parameter name	Type	Description
RequestId	String	Unique request id, Each request will return. The RequestId of the request needs to be provided when locating the problem.

4. Error Code

The following lists only the error codes related to the interface business logic, For other error codes, see public ErrorCode.

Error Code	Description
InvalidParameter	Wrong parameter.
InvalidParameterValue	Incorrect parameter value
InternalError	Internal Error
UnauthorizedOperation	Unauthorized operation
FailedOperation	Failed to operate.
InvalidParameter.FormatError	Incorrect parameter format
InvalidParameterValue.Length	Invalid parameter length.

RegisterTargets

1. Interface Description

Interface request domain name: clb.api3.convergecloud.com.

This API (RegisterTargets) is used to bind one or more real servers to a CLB listener (or layer-7 forwarding rule). Before using this API, you need to create a relevant layer-4 listener or layer-7 forwarding rule. For the former (TCP/UDP), only the listener ID needs to be specified, while for the latter (HTTP/HTTPS), the forwarding rule also needs to be specified through LocationId or Domain+Url.
This is an async API. After it returns a successful result, you need to call the DescribeTaskStatus API with the returned RequestID as an input parameter to check whether this task is successful.

Default interface request rate limit: 20 times per second.

Interface update time: 2020-01-10 20:37:04.

The interface only verifies the signature but does not authenticate.

2. Input Parameters

The following list of request parameters only lists interface request parameters and some public parameters, For a complete list of public parameters see public request parameters.

Parameter name	Required	Allow NULL	Type	Description
Action	Yes	No	String	Public parameter, The value of this interface: RegisterTargets
Version	Yes	No	String	Public parameter, The value of this interface: 2018-03-17
Region	No	No	String	Public parameter,Regional information A list of supported geographies is available through the DescribeRegions interface
LoadBalancerId	Yes	No	String	CLB instance ID Example: lb-cuxwji93
ListenerId	Yes	No	String	CLB listener ID Example: lbl-d1ubji93
LocationId	No	No	String	Forwarding rule ID. When binding a real server to a layer-7 forwarding rule, you must provide either this parameter or Domain+Url. Example: loc-fi8gji93
Domain	No	No	String	Domain name of the target forwarding rule. This parameter will not take effect when the LocationId parameter is provided. Example: www.yuncloud.example.com
Url	No	No	String	The URL of the target forwarding rule. This parameter will not take effect when the LocationId parameter is provided. Example: /index
Targets	Yes	No	Array of Target	List of real servers to be bound. Array length limit: 20 Example:View

3. Output Parameters

Parameter name	Type	Description
RequestId	String	Unique request id, Each request will return. The RequestId of the request needs to be provided when locating the problem.

4. Error Code

The following lists only the error codes related to the interface business logic, For other error codes, see public ErrorCode.

Error Code	Description
InvalidParameter	Wrong parameter.
InvalidParameterValue	Incorrect parameter value
InternalError	Internal Error
UnauthorizedOperation	Unauthorized operation
ResourceInsufficient	Insufficient resources
LimitExceeded	Quota limit reached
FailedOperation	Failed to operate.
InvalidParameter.FormatError	Incorrect parameter format

Error Code	Description
InvalidParameterValue.Length	Invalid parameter length.
MissingParameter	Missing parameter error.

ReplaceCert

1. Interface Description

Interface request domain name: clb.api3.convergecloud.com.

Replace an existing certificate with a new one (Note: If the existing certificate is used in multiple regions, rules bound to it will be replaced at the same time.)

Default interface request rate limit: 20 times per second.

Interface update time: 2021-08-23 21:36:44.

The interface is both signature verification and authentication.

2. Input Parameters

The following list of request parameters only lists interface request parameters and some public parameters, For a complete list of public parameters see public request parameters.

Parameter name	Required	Allow NULL	Type	Description
Action	Yes	No	String	Public parameter, The value of this interface: ReplaceCert
Version	Yes	No	String	Public parameter, The value of this interface: 2018-03-17
Region	No	No	String	Public parameter,Regional information A list of supported geographies is available through the DescribeRegions interface
OldCertId	Yes	No	String	Original certificate ID Example: GpW7fKkH
NewCertId	No	No	String	New certificate ID. Note: This parameter cannot be specified at the same time with new certificate convention, remarks, or private key. Example: GpW7fKkH
NewCertContent	No	No	String	Public key of the new certificate. Note: New certificate ID and this parameter cannot be specified at the same time. Example: -----BEGIN CERTIFICATE----- \r\nMIIEjTCCA3WgAwIBAgIQA4ZHUAVOv4yrszI3lLfo3TANBgkqhkiG9w0BAQsFADBy\r\nzVT4IsaIn5LKkKxw=\r\n----- END CERTIFICATE-----\r\n
NewCertAlias	No	No	String	New certificate remark. Note: The new certificate ID and this parameter cannot be specified at the same time. Example: cert-new
NewCertKey	No	No	String	Private key of the new certificate. Note: New certificate ID and this parameter cannot be specified at the same time. Example: -----BEGIN RSA PRIVATE KEY-----\r\nxz7PDqBylUnAkbCF0HAa\r\n-----END RSA PRIVATE KEY-----\r\n
NewSignCert	No	No	String	Public key of new SM certificate signature. Note: New certificate ID and this parameter cannot be specified at the same time. Example: -----BEGIN CERTIFICATE-----\r\nMIIEjTCCAYOuRvokzVT4IsaIn5LKkKxw=\r\n-----END CERTIFICATE-----\r\n
NewSignCertPrivateKey	No	No	String	Private key of new SM certificate signature. Note: New certificate ID and this parameter cannot be specified at the same time. Example: -----BEGIN RSA PRIVATE KEY-----\r\nHyaM23sHkbCF0HAa\r\n-----END RSA PRIVATE KEY-----\r\n

3. Output Parameters

Parameter name	Type	Description
RequestId	String	Unique request id, Each request will return. The RequestId of the request needs to be provided when locating the problem.

4. Error Code

The following lists only the error codes related to the interface business logic, For other error codes, see public ErrorCode.

Error Code	Description
UnauthorizedOperation	Unauthorized operation
FailedOperation	Failed to operate.
InternalError	Internal Error
InvalidParameterValue	Incorrect parameter value
InvalidParameter	Wrong parameter.
MissingParameter	Missing parameter error.
InvalidParameterValue.Length	Invalid parameter length.

SetCustomizedConfigForLoadBalancer

1. Interface Description

Interface request domain name: clb.api3.convergecloud.com.

This API is used to create, delete, modify, bind, and unbind custom CLB configurations.

Default interface request rate limit: 20 times per second.

Interface update time: 2020-02-24 17:42:43.

The interface is both signature verification and authentication.

2. Input Parameters

The following list of request parameters only lists interface request parameters and some public parameters, For a complete list of public parameters see public request parameters.

Parameter name	Required	Allow NULL	Type	Description
Action	Yes	No	String	Public parameter, The value of this interface: SetCustomizedConfigForLoadBalancer
Version	Yes	No	String	Public parameter, The value of this interface: 2018-03-17
Region	No	No	String	Public parameter,Regional information A list of supported geographies is available through the DescribeRegions interface
OperationType	Yes	No	String	Operation type, supporting ADD, DELETE, UPDATE, BIND, and UNBIND. Example: ADD
UconfigId	No	No	String	This field is required to create configurations other than custom ones. For example, pz-j9x7gx3g. Example: pz-n651fsue
ConfigContent	No	No	String	This field is required when you create or modify the content of a custom configuration. Example: client_max_body_size 222M;
ConfigName	No	No	String	This field is required when you create or modify the name of a custom configuration. Example: config_1
LoadBalancerIds	No	No	Array of String	This field is required for binding/unbinding. Example: [lb-55rh44i6]

3. Output Parameters

Parameter name	Type	Description
ConfigId	String	Custom configuration ID. For example, pz-j9x7gx3g. Example: pz-j9x7gx3g
RequestId	String	Unique request id, Each request will return. The RequestId of the request needs to be provided when locating the problem.

4. Error Code

The following lists only the error codes related to the interface business logic, For other error codes, see public ErrorCode.

Error Code	Description
InvalidParameter	Wrong parameter.
InvalidParameterValue	Incorrect parameter value
InternalError	Internal Error
UnauthorizedOperation	Unauthorized operation
LimitExceeded	Quota limit reached
FailedOperation	Failed to operate.
InvalidParameter.LBIdNotFound	CLB instance ID error.
InvalidParameter.FormatError	Incorrect parameter format
MissingParameter	Missing parameter error.

SetSubUinQuotas

1. Interface Description

Interface request domain name: clb.api3.convergecloud.com.

Set sub-account CLB instance quotas through the root tenant account

Default interface request rate limit: 20 times per second.

Interface update time: 2021-02-22 14:29:33.

The interface is both signature verification and authentication.

2. Input Parameters

The following list of request parameters only lists interface request parameters and some public parameters, For a complete list of public parameters see public request parameters.

Parameter name	Required	Allow NULL	Type	Description
Action	Yes	No	String	Public parameter, The value of this interface: SetSubUinQuotas
Version	Yes	No	String	Public parameter, The value of this interface: 2018-03-17
Region	No	No	String	Public parameter,Regional information A list of supported geographies is available through the DescribeRegions interface
SubQuota	Yes	No	Array of SubQuota	Sub-account quota list Example:View

3. Output Parameters

Parameter name	Type	Description
RequestId	String	Unique request id, Each request will return. The RequestId of the request needs to be provided when locating the problem.

4. Error Code

The following lists only the error codes related to the interface business logic, For other error codes, see public ErrorCode.

Error Code	Description
FailedOperation	Failed to operate.
InvalidParameter	Wrong parameter.
InternalError	Internal Error

UpLoadCert

1. Interface Description

Interface request domain name: clb.api3.convergecloud.com.

Upload an SSL certificate from the tenant console

Default interface request rate limit: 20 times per second.

Interface update time: 2021-08-18 18:37:39.

The interface is both signature verification and authentication.

2. Input Parameters

The following list of request parameters only lists interface request parameters and some public parameters, For a complete list of public parameters see public request parameters.

Parameter name	Required	Allow NULL	Type	Description
Action	Yes	No	String	Public parameter, The value of this interface: UpLoadCert
Version	Yes	No	String	Public parameter, The value of this interface: 2018-03-17
Region	No	No	String	Public parameter,Regional information A list of supported geographies is available through the DescribeRegions interface
CertContent	Yes	No	String	Certificate Content Example: -----BEGIN CERTIFICATE----- \r\nMIIEjTCCA3WgAwIBAgIQA4ZHUAqdNY9c\r\nYOuRvokzVT4lsaln5LKkKxw=\r\n----- END CERTIFICATE-----\r\n
CertType	Yes	No	String	Certificate type (CA: Client certificate; SVR: Server certificate) Example: SVR
CertKey	No	No	String	Private key of the certificate Example: -----BEGIN RSA PRIVATE KEY----- \r\nMIIEoAIBAAKCAQEAl9wOHyaM23sHkbCF0HAa\r\n-----END RSA PRIVATE KEY----- \r\n
CertAlias	Yes	No	String	Certificate remarks Example: cert-2
SignCert	No	No	String	Content of SM signature certificate Example: -----BEGIN CERTIFICATE----- \r\nMIIEjTCCA3WgAwIBAgIQA4ZHr\nYOuRvokzVT4lsaln5LKkKxw=\r\n----- END CERTIFICATE-----\r\n
SignCertPrivateKey	No	No	String	Private key of SM signature certificate Example: -----BEGIN RSA PRIVATE KEY----- \r\nMIIEoAIBAAKCAQEHyaM23sHkbCF0HAa\r\n-----END RSA PRIVATE KEY-----\r\n

3. Output Parameters

Parameter name	Type	Description
CertID	String	Certificate ID

Parameter name	Type	Description
		Example: GpW7fKkH
RequestId	String	Unique request id, Each request will return. The RequestId of the request needs to be provided when locating the problem.

4. Error Code

The following lists only the error codes related to the interface business logic, For other error codes, see public ErrorCode.

Error Code	Description
InternalError	Internal Error
InvalidParameter	Wrong parameter.
InvalidParameter.FormatError	Incorrect parameter format
UnauthorizedOperation	Unauthorized operation
FailedOperation	Failed to operate.

Other APIs
DescribeLoadBalancerListByCertId

1. Interface Description

Interface request domain name: clb.api3.convergecloud.com.

This API is used to query the list of CLB instances associated with a certificate in a region by certificate ID.

Default interface request rate limit: 20 times per second.

Interface update time: 2020-01-10 20:20:28.

The interface is both signature verification and authentication.

2. Input Parameters

The following list of request parameters only lists interface request parameters and some public parameters, For a complete list of public parameters see public request parameters.

Parameter name	Required	Allow NULL	Type	Description
Action	Yes	No	String	Public parameter, The value of this interface: DescribeLoadBalancerListByCertId
Version	Yes	No	String	Public parameter, The value of this interface: 2018-03-17
Region	No	No	String	Public parameter,Regional information A list of supported geographies is available through the DescribeRegions interface
CertIds	Yes	No	Array of String	Server or client certificate ID Example: ["RwFAfr8Y"]

3. Output Parameters

Parameter name	Type	Description
CertSet	Array of CertIdRelatedWithLoadBalancers	Certificate ID and the list of CLB instances associated with the certificate ID Example:View
RequestId	String	Unique request id, Each request will return. The RequestId of the request needs to be provided when locating the problem.

4. Error Code

The following lists only the error codes related to the interface business logic, For other error codes, see public ErrorCode.

Error Code	Description
InvalidParameter	Wrong parameter.
InvalidParameterValue	Incorrect parameter value
InternalError	Internal Error
UnauthorizedOperation	Unauthorized operation
FailedOperation	Failed to operate.
InvalidParameter.FormatError	Incorrect parameter format
InvalidParameterValue.Length	Invalid parameter length.

SetLoadBalancerSecurityGroups

1. Interface Description

Interface request domain name: clb.api3.convergecloud.com.

This API is used to support setting (binding/unbinding) security groups for a public network CLB instance. You can use the DescribeLoadBalancers API to query the currently bound security groups of a CLB instance. This API is a semantic API.
When binding is performed, input parameters should include all security groups (existing and newly bound ones) that should be bound to CLB instances.
When unbinding is performed, input parameters should include all security groups that remain bound after unbinding. If you want to unbind all security groups, this parameter can be left unspecified, or an empty array can be passed in.

Default interface request rate limit: 20 times per second.

Interface update time: 2020-01-10 20:30:50.

The interface is both signature verification and authentication.

2. Input Parameters

The following list of request parameters only lists interface request parameters and some public parameters, For a complete list of public parameters see public request parameters.

Parameter name	Required	Allow NULL	Type	Description
Action	Yes	No	String	Public parameter, The value of this interface: SetLoadBalancerSecurityGroups
Version	Yes	No	String	Public parameter, The value of this interface: 2018-03-17
Region	No	No	String	Public parameter,Regional information A list of supported geographies is available through the DescribeRegions interface
LoadBalancerId	Yes	No	String	CLB instance ID Example: lb-cuxw2r00
SecurityGroups	No	No	Array of String	Array of security group IDs. One CLB instance can bind up to 50 security groups. To unbind all security groups, you can either skip this parameter or specify an empty array. Example: ["sg-0936o7sd","sg-dfsv32sx"]

3. Output Parameters

Parameter name	Type	Description
RequestId	String	Unique request id, Each request will return. The RequestId of the request needs to be provided when locating the problem.

4. Error Code

The following lists only the error codes related to the interface business logic, For other error codes, see public ErrorCode.

Error Code	Description
InvalidParameterValue	Incorrect parameter value
InternalError	Internal Error
UnauthorizedOperation	Unauthorized operation
FailedOperation	Failed to operate.
InvalidParameter.LBIdNotFound	CLB instance ID error.
InvalidParameter.FormatError	Incorrect parameter format
InvalidParameterValue.Length	Invalid parameter length.

SetSecurityGroupForLoadbalancers

1. Interface Description

Interface request domain name: clb.api3.convergecloud.com.

This API is used to bind or unbind a security group to or from multiple public network CLB instances.

Default interface request rate limit: 20 times per second.

Interface update time: 2020-01-10 20:30:27.

The interface only verifies the signature but does not authenticate.

2. Input Parameters

The following list of request parameters only lists interface request parameters and some public parameters, For a complete list of public parameters see public request parameters.

Parameter name	Required	Allow NULL	Type	Description
Action	Yes	No	String	Public parameter, The value of this interface: SetSecurityGroupForLoadbalancers
Version	Yes	No	String	Public parameter, The value of this interface: 2018-03-17
Region	No	No	String	Public parameter,Regional information A list of supported geographies is available through the DescribeRegions interface
SecurityGroup	Yes	No	String	Security group ID. For example, sg-j9x73ko1. Example: sg-j9x73ko1
OperationType	Yes	No	String	ADD: Bind a security group. DEL: Unbind a security group. Example: ADD
LoadBalancerIds	Yes	No	Array of String	CLB instance ID array Example: ["lb-0936o712","lb-tttt5555"]

3. Output Parameters

Parameter name	Type	Description
RequestId	String	Unique request id, Each request will return. The RequestId of the request needs to be provided when locating the problem.

4. Error Code

The following lists only the error codes related to the interface business logic, For other error codes, see public ErrorCode.

Error Code	Description
InvalidParameter	Wrong parameter.
InvalidParameterValue	Incorrect parameter value
InternalError	Internal Error
UnauthorizedOperation	Unauthorized operation
FailedOperation	Failed to operate.
InvalidParameter.LBIdNotFound	CLB instance ID error.
InvalidParameter.FormatError	Incorrect parameter format
InvalidParameterValue.Length	Invalid parameter length.

Data structure

MasterZoneInfo

Primary AZ information

Referenced by the following interfaces: DescribeMasterZones

name	required	NULL allowed	type	description
ZoneId	yes	yes	Uint64	Unique ID of the AZ in numeric form, such as 100001. Example: 50010001
Zone	yes	yes	String	Unique ID of the AZ in string format. For example: ap-region1-1. Example: 50010001
ZoneName	yes	yes	String	AZ name. For example, AZ 1. Example: 50010001
BackupZoneSet	yes	yes	Array of ZoneInfo	List of secondary AZs Example:View
LBType	no	yes	String	Returned availability zone type. Valid values: open, internal. Example: open
AvailableIsps	no	yes	Array of AvailableIspInfo	Returned availability status of each ISP. Example:View

RuleOutput

HTTP/HTTPS listener forwarding rule (output)

Referenced by the following interfaces: DescribeListeners, DescribeRewrite

name	required	NULL allowed	type	description
LocationId	no	no	String	Forwarding rule ID Example: loc-jkdsiw03
Domain	yes	yes	String	Domain name of the forwarding rule Example: www.realdomain.com
Url	yes	yes	String	Path of the forwarding rule. Example: /url1
SessionExpireTime	no	no	Int64	Session persistence Duration Example: 0
HealthCheck	no	yes	HealthCheck	Health check information Example:View
Certificate	no	yes	CertificateOutput	Certification information Example:View
Scheduler	no	no	String	Request forwarding method of the rule Example: WRR
ListenerId	yes	no	String	ID of the listener to which the forwarding rule belongs Example: lbl-8cnlji93
RewriteTarget	yes	yes	RewriteTarget	Redirection target information of the forwarding rule Example:View
HttpGzip	yes	no	Bool	Whether to enable gzip Example: true
BeAutoCreated	yes	no	Bool	Whether the forwarding rule is automatically created Example: false
DefaultServer	yes	no	Bool	Whether to set as the default domain name Example: true
Http2	yes	no	Bool	Whether to enable Http2 Example: false
ForwardType	yes	no	String	Forwarding protocol between CLB and the real server Example: HTTP
CreateTime	yes	no	String	Forwarding rule creation time Example: 2024-01-22 14:16:12
TargetType	yes	no	String	Real server type Example: NODE
TargetGroup	yes	yes	BasicTargetGroupInfo	Basic information of a bound target group. This field will be returned if a target group is bound to a rule. Example:View
WafDomainId	yes	yes	String	WAF instance ID Example: ""
QuicStatus	no	yes	String	QUIC status Example: QUIC_INACTIVE
TrpcCallee	no	yes	String	TRPC callee server route, which takes effect when ForwardType is TRPC Example: ""
TrpcFunc	no	yes	String	TRPC calling service API, which takes effect when ForwardType is TRPC Example: ""

TargetCountForLoadBalancer

CLB instance and the total number of real servers bound to it

Referenced by the following interfaces: DescribeTargetCountForLoadBalancers

name	required	NULL allowed	type	description
LoadBalancerId	yes	no	String	CLB ID Example: lb-di3lor002
TargetCount	yes	no	Uint64	Total number of real servers bound to the CLB instance Example: 2

SnatIp

SnatIp information structure

Referenced by the following interfaces: DescribeLoadBalancerListByCertId, DescribeLoadBalancers, DescribeLoadBalancersForVpc

name	required	NULL allowed	type	description
SubnetId	yes	no	String	Unique VPC subnet ID. For example, subnet-j9x73ao1. Example: 432
Ip	no	no	String	IP address, such as 192.168.0.1 Example: 1.1.1.1

ExclusiveCluster

ExCluster

Referenced by the following interfaces: CreateLoadBalancer, DescribeLoadBalancerListByCertId, DescribeLoadBalancers, DescribeLoadBalancersForVpc, ModifyLoadBalancerAttributes

name	required	NULL allowed	type	description
L4Clusters	no	yes	Array of ClusterItem	Layer-4 exclusive cluster list Example:View
L7Clusters	no	yes	Array of ClusterItem	Layer-7 exclusive cluster list Example:View
ClassicalCluster	no	yes	ClusterItem	VPCGW cluster Example:View

AvailableIspInfo

Availability status of ISPs for the availability zone.

Referenced by the following interfaces: DescribeLoadBalancerListByCertId, DescribeLoadBalancers, DescribeLoadBalancersForVpc, DescribeMasterZones

name	required	NULL allowed	type	description
Isp	yes	no	String	Operator Example: Private Network
IspAlias	no	no	String	ISP alias Example: Private Network
EnableShared	yes	no	Bool	Whether there are shared cluster resources. Example: true
Enable14SetLabels	no	no	Array of String	Tag of the available Layer-4 cluster. Example: ["14-label1"]
AvailableVpcids	no	no	Array of String	VPC available for the IPv6-BGP mode. Example: ["vpc-sh39ay45"]

FunctionInfo

Serverless Cloud Function (SCF) information

Referenced by the following interfaces: DescribeTargetHealth, DescribeTargets

name	required	NULL allowed	type	description
FunctionNamespace	no	yes	String	Function namespace Example: default
FunctionName	no	yes	String	Function Name Example: function-1
FunctionQualifier	no	yes	String	Function version name or alias Example: ydbs3bzs
FunctionQualifierType	no	yes	String	FunctionQualifier identifier type. Valid values: VERSION and ALIAS Example: VERSION

HealthCheck

Health check information.
Note: This is a custom check parameter. Currently, it is supported only in a few regions (You need to submit a ticket if you need to use this API).

Referenced by the following interfaces: CreateListener, CreateRule, DescribeListeners, DescribeRewrite, ModifyListener, ModifyRule

name	required	NULL allowed	type	description
HealthSwitch	no	no	Int64	Whether to enable health check. 1: Enable; 0: Disable. Example: 0
TimeOut	no	yes	Int64	Response timeout duration for health checks (applicable only to layer-4 listeners). Value range: 2-60. Default: 2. Unit: sec. Response timeout duration should be less than the check interval. Example: 2
IntervalTime	no	yes	Int64	Health check interval. Default: 5. Value range: 5-300. Unit: sec. Example: 5
HealthNum	no	yes	Int64	Healthy threshold. Default: 3, indicating that if a forwarding is found healthy three consecutive times, it is considered to be normal. Value range: 2-10. Unit: times. Example: 3
UnHealthNum	no	yes	Int64	Unhealthy threshold. Default: 3, indicating that if a forwarding is found unhealthy three consecutive times, it is considered to be exceptional. Value range: 2-10. Unit: times. Example: 3
HttpCode	no	yes	Int64	Health check status code (applicable only to HTTP/HTTPS forwarding rules and HTTP health checks of TCP listeners). Value range: 1-31. Default: 31. 1: Return code 1xx for healthy status. 2: Return code 2xx for healthy status. 4: Return code 3xx for healthy status. 8: Return code 4xx for healthy status. 16: Return code 5xx for healthy status. If you want multiple return codes to represent healthy status, sum up the corresponding values. Note: The HTTP health checks of TCP listeners only support specifying one health check status code. Example: 8
HttpCheckPath	no	yes	String	Health check path (applicable only to HTTP/HTTPS forwarding rules and HTTP health checks of TCP listeners). Example: /url1
HttpCheckDomain	no	yes	String	Health check domain name (applicable only to HTTP/HTTPS forwarding rules and HTTP health checks of TCP listeners). Example: www.healthcheck.com
HttpCheckMethod	no	yes	String	Health check method (applicable only to HTTP/HTTPS forwarding rules and HTTP health checks of TCP listeners). Default: HEAD. Valid values: HEAD and GET. Example: HEAD
CheckPort	no	yes	Int64	A custom check parameter, representing the health check port, which is the port of the real server by default. Unless you want to specify a port, it is recommended to leave it empty. (Applicable only to TCP/UDP listeners) Example: 8000
ContextType	no	yes	String	Custom detection parameters. When the value of the health check protocol CheckType is set to CUSTOM, this field is required and represents the input format for the health check. Valid values: HEX or TEXT. HEX indicates binary data, while TEXT indicates text data, which only supports visible ASCII characters. When the value is HEX, the characters of SendContext and RecvContext can only be selected from 0123456789ABCDEF, and the length should be an even number. (Applicable only t UDP listeners.) Example: TEXT
SendContext	no	yes	String	A custom check parameter, which is required if the value of CheckType (health check protocol) is CUSTOM. This parameter represents the content of the request sent by the health check. Only ASCII visible characters are allowed, and the maximum length is 500. (Applicable only to TCP/UDP listeners.) Example: request-info

name	required	NULL allowed	type	description
RecvContext	no	yes	String	A custom check parameter, which is required if the value of CheckType (health check protocol) is CUSTOM. This parameter represents the result returned by the health check. Only ASCII visible characters are allowed, and the maximum length is 500. (Applicable only to TCP/UDP listeners.) Example: response-info
CheckType	no	yes	String	Custom detection parameters. Protocol used for health check: TCP
HttpVersion	no	yes	String	A custom check parameter, representing the HTTP version of the real server, which is required if the value of CheckType is HTTP. Valid values: HTTP/1.0 and HTTP/1.1. (Applicable only to TCP listeners) Example: HTTP/1.1

ZoneInfo

AZ information

Referenced by the following interfaces: DescribeLoadBalancerListByCertId, DescribeLoadBalancers, DescribeLoadBalancersForVpc, DescribeMasterZones

name	required	NULL allowed	type	description
ZoneId	yes	yes	Uint64	Unique ID of the AZ in numeric form, such as 100001 Example: 100001
Zone	yes	yes	String	Unique ID of the AZ in string format. For example: ap-region1-1. Example: ap-region1-1
ZoneName	yes	yes	String	AZ name. For example, AZ 1. Example: AZ 1
ZoneRegion	no	yes	String	Region of the AZ Example: ap-region1
LocalZone	no	yes	Bool	Whether the AZ is a LocalZone Example: false
EnableShared	no	yes	Bool	Whether the availability zone has available shared clusters. Example: true
Enable14SetLabels	no	yes	Array of String	Tag of the available Layer-4 cluster for the availability zone. Example: ["14-label"]
AvailableIsps	no	no	Array of AvailableIspInfo	Information set of available ISPs for the availability zone. Example:View

Listener

Listener information

Referenced by the following interfaces: DescribeListeners

name	required	NULL allowed	type	description
ListenerId	yes	no	String	CLB listener ID Example: lbl-cdt45q4g
Protocol	yes	no	String	Listener protocol Example: HTTP
ProtocolFake	no	no	String	Listener protocol (Fake). Example: TCP
Port	yes	no	Int64	Listener port Example: 80
Certificate	yes	yes	CertificateOutput	Information of certificate bound to the listener Example:View
HealthCheck	yes	yes	HealthCheck	Health check information of the listener Example:View
Scheduler	yes	yes	String	Requested scheduling method Example: WRR
SessionExpireTime	yes	yes	Int64	Session persistence Duration Example: 0
SniSwitch	yes	yes	Int64	Whether to enable the SNI feature. (This parameter is applicable only to HTTPS listeners.) Example: 0
Rules	yes	yes	Array of RuleOutput	All forwarding rules under a listener (This parameter is applicable only to HTTP/HTTPS Listeners.) Example:View
ListenerName	yes	yes	String	Listener name Example: Name
CreateTime	yes	yes	String	Listener creation time Example: 2021-03-01 15:27:04
EndPort	yes	yes	Int64	End port of a port range Example: 0
TargetType	yes	yes	String	Real server type Example: NODE
TargetGroup	yes	yes	BasicTargetGroupInfo	Basic information of a bound target group. This field will be returned if a target group is bound to a listener. Example:View
DefaultSerSwitch	yes	yes	Uint64	Whether the default domain identifier is enabled for the listener. 0: Not enabled; 1: Enabled Example: 0
DeregisterTargetRst	yes	yes	Bool	Whether to send the RST packet to the client when a real server is unbound. This parameter is applicable only to TCP listeners. Example: false
Toa	yes	yes	Bool	Supports Nat64 CLB TCP listeners only Example: false
SessionType	no	yes	String	Session persistence type. NORMAL: Default session persistence type; QUIC_CID: Session persistence by Quic Connection ID. Example: NORMAL
KeepaliveEnable	no	yes	Int64	Whether to enable a persistent connection. 1: Enable; 0: Disable. (This parameter is applicable only to HTTP/HTTPS listeners.) Example: 0

BindItem

Configuration binding relationship

Referenced by the following interfaces: AssociateCustomizedConfig

name	required	NULL allowed	type	description
LoadBalancerId	yes	no	String	ID of the CLB instance bound to the configuration Example: lb-4xZyjt93
ListenerId	yes	yes	String	ID of the listener bound to the configuration Example: lbl-dsZyjt93
Domain	yes	yes	String	Domain name bound to the configuration Example: foo.net
LocationId	no	yes	String	Rule bound to the configuration Example: loc-jdhsjt93

RuleInput

HTTP/HTTPS forwarding rule (input)

Referenced by the following interfaces: CreateRule

name	required	NULL allowed	type	description
Domain	yes	no	String	Domain name of the forwarding rule. Length limit: 1-80. Example: www.realdomain.com
Url	yes	no	String	Path of the forwarding rule. Length limit: 1-200. Example: /local
SessionExpireTime	no	no	Int64	Session persistence time. 0: Disable session persistence. Value range: 30-3600. Unit: sec. Example: 30
HealthCheck	no	no	HealthCheck	Health check information Example:View
Certificate	no	no	CertificateInput	Certification information Example:View
Scheduler	no	no	String	Request forwarding method of the rule. Valid values: WRR (weighted round robin), LEAST_CONN (least connections), and IP_HASH (IP hash) Default: WRR Example: WRR
ForwardType	no	no	String	Forwarding protocol between CLB and the real server. Valid values: HTTP, HTTPS, and TRPC Example: HTTP
DefaultServer	no	no	Bool	Whether to set this domain name as the default domain name. Note: Only one default domain name can be set under a listener. Example: true
Http2	no	no	Bool	Whether to enable Http2. Note that Http2 can be enabled only for HTTPS domains. Example: true
TargetType	no	no	String	Target real server type. NODE: Binding a general node; TARGETGROUP: Binding a target group. Example: NODE
TrpcCallee	no	no	String	TRPC callee server route, which is required when ForwardType is TRPC Example: trpcall
TrpcFunc	no	no	String	TRPC calling service API, which is required when ForwardType is TRPC Example: trpcFunc

ClusterItem

Exclusive cluster information

Referenced by the following interfaces: CreateLoadBalancer, DescribeLoadBalancerListByCertId, DescribeLoadBalancers, DescribeLoadBalancersForVpc, ModifyLoadBalancerAttributes

name	required	NULL allowed	type	description
ClusterId	yes	no	String	Unique ID of the cluster Example: cluste-jd9e3bxs
ClusterName	no	yes	String	Cluster Name Example: cluster-1
Zone	no	yes	String	Cluster AZ. For example, ap-region1-1. Example: ap-region1

TagInfo

CLB tag information

Referenced by the following interfaces: CreateLoadBalancer, DescribeLoadBalancerListByCertId, DescribeLoadBalancers, DescribeLoadBalancersForVpc

name	required	NULL allowed	type	description
TagKey	yes	no	String	Tag key Example: key-djx983xs
TagValue	yes	no	String	Tag value Example: value-djx983xs

IspSet

ISP information description

Referenced by the following interfaces: DescribeIspInfo

name	required	NULL allowed	type	description
Id	yes	no	Uint64	ISP ID Example: 2
Type	yes	no	String	ISP type Example: CMCC
Name	yes	no	String	ISP name Example: Move
IspStatus	yes	no	Uint64	ISP IPv4 status Example: 1
IspStatusIPv6	yes	no	Uint64	ISP IPv6 status Example: 1
V4alias	no	no	String	ISP v4 alias Example: Move
V6alias	no	no	String	ISP v6 alias Example: Move

Target

Forwarding target, namely, a real server bound to the CLB instance

Referenced by the following interfaces: BatchModifyTargetWeight, DeregisterTargets, ModifyTargetPort, ModifyTargetWeight, RegisterTargets

name	required	NULL allowed	type	description
Type	no	yes	String	Real server type. Valid values: CVM (Cloud Virtual Machine), ENI (Elastic Network Interface), and BMS (Bare Metal Server). Currently, this parameter does not take effect as an input parameter. Example: CVM
InstanceId	no	yes	String	Unique ID of a CVM instance, which is required when you bind a CVM instance. This parameter can be obtained from the InstanceId field in the response of the DescribeInstances API. Note: One and only one of the InstanceId, EniIp, and InstanceBmsId parameters must be provided. Example: ins-di3kx902
Port	yes	yes	Int64	Real server listening port Example: 80
Weight	no	no	Int64	Forwarding weight of a real server. Value range: [0, 100]. Default: 10. Example: 10
EniIp	no	yes	String	ENI IP, which is required when an ENI is bound. The ENI must first be bound to CVM before it can be bound to a CLB instance. Note: One and only one of the InstanceId, EniIp, and InstanceBmsId parameters must be provided. Note: To bind an ENI, you need to submit a ticket to add it to the allowlist. Example: x.x.x.x
InstanceBmsId	no	yes	String	Unique ID of a BMS instance, which is required when you bind a BMS instance. This parameter can be obtained from the bmsId field in the response of the DescribeInstances API of the BMS instance. Note: One and only one of the InstanceId, EniIp, and InstanceBmsId parameters must be provided. Binding bms and cvm or eni simultaneously is not supported. Example: bms-di3kx902x

Filter

Filter conditions

Referenced by the following interfaces: DescribeListeners, DescribeLoadBalancers

name	required	NULL allowed	type	description
Name	yes	no	String	Filter name Example: loadbalancerids
Values	yes	no	Array of String	Filter value array Example: ["lb-di3kx902"]

Price

Indicates the price of CLB

Referenced by the following interfaces: InquiryPriceCreateLoadBalancer, InquiryPriceModifyLoadBalancer

name	required	NULL allowed	type	description
InstancePrice	yes	yes	ItemPrice	Describes the instance pricing. Example:View
BandwidthPrice	yes	yes	ItemPrice	Describes the network pricing. Example:View

RewriteTarget

Redirection target information

Referenced by the following interfaces: DescribeListeners, DescribeRewrite

name	required	NULL allowed	type	description
TargetListenerId	yes	yes	String	Listener ID of the redirection target Note: This field may return null, indicating no redirection. Example: lbl-di3kx902
TargetLocationId	yes	yes	String	Forwarding rule ID of the redirection target Note: This field may return null, indicating no redirection. Example: loc-di3kx902
RewriteCode	yes	yes	Int64	Redirect status code Example: 302
TakeUrl	yes	yes	Bool	Whether the matched URL is carried in redirection Example: true
RewriteType	yes	yes	String	Redirection type. Manual: Manual redirection; Auto: Automatic redirection Example: Auto

TargetRegionInfo

Information of the real server bound to a CLB instance, including region and network to which it belongs.

Referenced by the following interfaces: DescribeLoadBalancerListByCertId, DescribeLoadBalancers, DescribeLoadBalancersForVpc, ModifyLoadBalancerAttributes

name	required	NULL allowed	type	description
Region	yes	no	String	Target region. For example, ap-region1. Example: ap-region1
VpcId	yes	no	String	Network of the target. For example, vpc-gx3gl9x7 (for VPCs) and 0 (for classic networks). Example: vpc-6lxqgj93

CertificateInput

Certification information

Referenced by the following interfaces: CreateListener, CreateRule, ModifyDomainAttributes, ModifyListener

name	required	NULL allowed	type	description
SSLMode	no	no	String	Authentication type. UNIDIRECTIONAL: One-way authentication; MUTUAL: Mutual authentication Example: MUTUAL
CertId	no	no	String	ID of the server RSA certificate. If this parameter is not specified, you must upload the certificate, including CertContent, CertKey, and CertName. Example: jSsI3sx2
CertCaId	no	no	String	ID of the client CA certificate. When the listener adopts mutual authentication, that is, SSLMode=MUTUAL, this field is required if the client certificate is not uploaded, including CertCaContent, CertCaName. Example: jSsI3sx2
CertName	no	no	String	Uploads the name of the server RSA certificate. This parameter is required if there is no CertId. Example: cert-1
CertKey	no	no	String	Uploads the key of the server RSA certificate. This parameter is required if there is no CertId. Example: -----BEGIN RSA PRIVATE KEY-----\r\nj93cx\r\n-----END RSA PRIVATE KEY-----\r\n
CertContent	no	no	String	Uploads the content of the server RSA certificate. This parameter is required if there is no CertId. Example: -----BEGIN CERTIFICATE-----\r\nMIIEjTCCA3WSLkKKxw==\r\n-----END CERTIFICATE-----\r\n
CertCaName	no	no	String	Uploads the name of the client CA certificate. This parameter is required if SSLMode=mutual and there is no CertCaId. Example: certca1
CertCaContent	no	no	String	Uploads the content of the client certificate. This parameter is required if SSLMode=mutual and there is no CertCaId. Example: -----BEGIN CERTIFICATE-----\r\nMIIEjTCCA3WSgAwIBAgIcVJeYOuRvokxVT4IoaIn5LKkKxw==\r\n-----END CERTIFICATE-----\r\n
EccCertId	no	no	String	ID of the server ECC certificate. If this parameter is not specified, you must upload the certificate, including CertContent, CertKey, and CertName. Example: x9j3jghx
EccCertName	no	no	String	Uploads the name of the server ECC certificate. This parameter is required if there is no CertEccId. Example: cert1
EccCertContent	no	no	String	Uploads the content of the server ECC certificate. This parameter is required if there is no CertEccId. Example: -----BEGIN CERTIFICATE-----\r\nMIIEjTCCA3WSgAwIBAgIMYbcYyYOuRvokxVT4IoaInSLKkKxw==\r\n-----END CERTIFICATE-----\r\n
EccCertKey	no	no	String	Uploads the key of the server ECC certificate. This parameter is required if there is no CertEccId. Example: -----BEGIN RSA PRIVATE KEY-----\r\nj93fdf\r\n-----END RSA PRIVATE KEY-----\r\n
SmCertId	no	no	String	ID of the server SM certificate. If this parameter is not specified, you must upload the certificate, including CertSMContent, CertSmKey, CertSMName, CertSMSignContent, and CertSMSignKey. Example: x9j3jghx
SmCertName	no	no	String	Uploads the name of the server SM certificate. This parameter is required if there is no CertSMId. Example: cert1
SmCertSignContent	no	no	String	Uploads the signature convention of the server SM certificate. This parameter is required if there is no CertSMId. Example: -----BEGIN CERTIFICATE-----\r\nj93f\r\n-----END CERTIFICATE-----\r\n
SmCertSignKey	no	no	String	Uploads the signature key of the server SM certificate. This parameter is required if there is no CertSMId. Example: -----BEGIN RSA PRIVATE KEY-----\r\nj93dfs\r\n-----END RSA PRIVATE KEY-----\r\n
SmCertContent	no	no	String	Uploads the certificate convention of the server SM certificate. This parameter is required if there is no CertSMId. Example: -----BEGIN CERTIFICATE-----\r\nj93f\r\n-----END CERTIFICATE-----\r\n
SmCertKey	no	no	String	Uploads the certificate key of the server SM certificate. This parameter is required if there is no CertSMId. Example: -----BEGIN RSA PRIVATE KEY-----\r\nj93fdf\r\n-----END RSA PRIVATE KEY-----\r\n

ItemPrice

Describe pricing information of an item

Referenced by the following interfaces: InquiryPriceCreateLoadBalancer, InquiryPriceModifyLoadBalancer

name	required	NULL allowed	type	description
UnitPrice	yes	yes	Float	Postpaid unit price. Example: 10
ChargeUnit	yes	yes	String	Subsequent billing unit. Value range: HOUR: Billing by hour. This billing unit is involved in scenarios with the billing type set to POSTPAID_BY_HOUR or BANDWIDTH_POSTPAID_BY_HOUR. GB: Billing by traffic in GB. This billing unit is involved in scenarios with the billing type set to TRAFFIC_POSTPAID_BY_HOUR. Example: HOUR
OriginalPrice	yes	yes	Float	Original prepaid price. Example: 50
DiscountPrice	yes	yes	Float	Discounted prepaid price. Example: 50
UnitPriceDiscount	yes	yes	Float	Discounted postpaid unit price. Example: 10
Discount	yes	yes	Float	Discount, such as 20.0 representing 80% off. Example: 100

BasicTargetGroupInfo

Basic information of a target group bound to a listener or forwarding rule

Referenced by the following interfaces: DescribeListeners, DescribeRewrite

name	required	NULL allowed	type	description
TargetGroupId	yes	no	String	Target group ID Example: tg-di3kx902
TargetGroupName	yes	no	String	Target Group Name Example: tg-1

ConfigListItem

Configuration

Referenced by the following interfaces: DescribeCustomizedConfigList

name	required	NULL allowed	type	description
UconfigId	yes	no	String	Configuration ID Example: pz-di3kx902
ConfigType	yes	no	String	Configuration Type Example: CLB
ConfigName	yes	yes	String	Configuration name Example: cm-1
ConfigContent	yes	no	String	Configuration Example: add_header aaa bbb;\r\n
CreateTimestamp	yes	no	String	Configuration creation time Example: 1702903612
UpdateTimestamp	yes	no	String	Configuration modification time Example: 1702903612

CertificateOutput

Certificate information

Referenced by the following interfaces: DescribeListeners, DescribeRewrite

name	required	NULL allowed	type	description
SSLMode	yes	no	String	Authentication type. UNIDIRECTIONAL: One-way authentication; MUTUAL: Mutual authentication Example: UNIDIRECTIONAL

name	required	NULL allowed	type	description
CertId	yes	no	String	Server certificate ID. (Compatible with earlier versions. Default: RSA certificate) Example: ALZj493p
CertCaId	yes	yes	String	Client certificate ID. Example: 49y8316
SmCertId	yes	yes	String	Server SM certificate ID. Example: fds2dgsx
EccCertId	yes	yes	String	ID of the server ECC certificate. Example: fds2dgsx

ListenerHealth

Health check information of the listener

Referenced by the following interfaces: DescribeTargetHealth

name	required	NULL allowed	type	description
ListenerId	yes	no	String	Listener ID Example: lbl-dl3kx902
ListenerName	yes	yes	String	Listener name Example: lbl-1
Protocol	yes	no	String	Listener protocol Example: TCP
Port	yes	no	Int64	Listener port Example: 80
Rules	yes	yes	Array of RuleHealth	List of forwarding rules of the listener Example:View

TargetHealth

Describes the health information of a target

Referenced by the following interfaces: DescribeTargetHealth

name	required	NULL allowed	type	description
IP	yes	no	String	Private network IP of target Example: 10.0.0.1
Port	yes	no	Int64	Port bound to the target Example: 80
HealthStatus	yes	no	Bool	Current health status. true: Healthy; false: Unhealthy (Check not started/Checking/Unknown status). CLB forwards traffic to a target only when the target is healthy (and with a weight greater than 0). Example: true
TargetId	yes	no	String	Target instance ID. For example, ins-j9x73ko1. Example: 'ins-j9x73ko1'
HealthStatusDetial	yes	no	String	Detailed information about the current health status. Alive: Healthy; Dead: Exceptional; Unknown: Check not started/Checking/Unknown status. Example: 'Alive'

ProjectInfo

Certificate item

Referenced by the following interfaces: DescribeCerts

name	required	NULL allowed	type	description
ProjectId	yes	no	String	Project ID Example: kdhoz3o9
OwnerUin	yes	no	String	Project owner Example: 110000004918
Name	yes	no	String	Project Name Example: pj1
CreatorUin	yes	no	String	Creator Example: 110000004918
CreateTime	yes	no	String	Creation Time Example: 2014-08-03 12:00:00
Info	yes	no	String	Project Information Example: info-1

SubQuota

Sub-account quota description

Referenced by the following interfaces: SetSubLimQuotas

name	required	NULL allowed	type	description
SubUin	yes	no	String	Sub-account UIN Example: 110000004918
Type	yes	no	Uint64	Quota type (4: Public network instance quota; 7: Private network instance quota) Example: 4
Quota	yes	no	Uint64	Quota value Example: 10

CertIdRelatedWithListener

Certificate ID and the list of CLB listeners bound to the certificate.

Referenced by the following interfaces: DescribeListenerListByCertId

name	required	NULL allowed	type	description
CertId	yes	no	String	Certificate ID Example: jfs8jk23x
Listeners	yes	no	Array of CertListener	List of listeners bound to the certificate. Example:View

CertIdRelatedWithLoadBalancers

Certificate ID and the list of CLB instances associated with the certificate ID

Referenced by the following interfaces: DescribeLoadBalancerListByCertId

name	required	NULL allowed	type	description
CertId	yes	no	String	Certificate ID Example: jfs13sc2
LoadBalancers	yes	yes	Array of LoadBalancer	List of CLB instances associated with the certificate Example:View

RuleHealth

Health check status of a forwarding rule

Referenced by the following interfaces: DescribeTargetHealth

name	required	NULL allowed	type	description
LocationId	yes	no	String	Forwarding rule ID Example: loc-dl3kx902
Domain	yes	yes	String	Domain name of the forwarding rule Example: foo.net
Url	yes	yes	String	URL of the forwarding rule Example: /url1
Targets	yes	yes	Array of TargetHealth	Health check status of the real server bound to this rule Example:View
FunctionTargets	no	yes	Array of FunctionTarget	Backend cloud function information. (This parameter is not supported on Private Cloud.) Example:View

BindDetailItem

Binding relationship, including listener name, protocol, URL, and vport.

Referenced by the following interfaces: DescribeCustomizedConfigAssociateList

name	required	NULL allowed	type	description
LoadBalancerId	yes	no	String	ID of the CLB instance bound to the configuration Example: lb-4d2yjl93
ListenerId	yes	yes	String	ID of the listener bound to the configuration Example: lbl-4apyjl93
Domain	yes	yes	String	Domain name bound to the configuration Example: foo.net
LocationId	yes	yes	String	Rule bound to the configuration Example: loc-jdhsjl93
ListenerName	yes	yes	String	Listener name Example: lbl-1
Protocol	yes	yes	String	Listener protocol Example: HTTP
Vport	yes	yes	Int64	Listener port Example: 80
Url	yes	yes	String	Location URL Example: /loc1
UconfigId	yes	yes	String	Configuration ID Example: pz-fd8dd0da

RsWeightRule

Modify the data type of a node weight

Referenced by the following interfaces: BatchModifyTargetWeight

name	required	NULL allowed	type	description
ListenerId	yes	no	String	CLB listener ID Example: lb-curxejl93
LocationId	no	no	String	Forwarding rule ID, required for layer-7 rules and not required for layer-4 rules Example: loc-o8cnyl93
Domain	no	no	String	Domain name of the target rule. This parameter will not take effect when the LocationId parameter is provided. Example: www.demo.example.com
Url	no	no	String	The URL of the target rule. This parameter will not take effect when the LocationId parameter is provided. Example: /index
Targets	yes	no	Array of Target	The list of real servers for which the weights are to be modified Example:View
Weight	no	no	Int64	New forwarding weight of a real server. Value range: 0-100. Example: 10

SubQuotaRsp

Description of results for sub-account quotas query

Referenced by the following interfaces: DescribeSubLimQuotas

name	required	NULL allowed	type	description
SubUin	yes	no	String	Sub-account UIN Example: 110000004918
UsedQuota	yes	no	Int64	Quota used by the sub-account Example: 1
CurQuota	yes	no	Int64	Current quota for the sub-account Example: 4

name	required	NULL allowed	type	description
LastSetTime	yes	no	String	Sub-account modification time Example: 2014-08-03 12:00:00

FunctionTarget

Serverless Cloud Function (SCF) as a real server

Referenced by the following interfaces: DescribeTargetHealth, DescribeTargets

name	required	NULL allowed	type	description
Function	no	yes	FunctionInfo	SCF information Example:View
Weight	no	yes	Uint64	Weight Example: 100

BatchTarget

Batch binding type

Referenced by the following interfaces: BatchDeregisterTargets, BatchRegisterTargets

name	required	NULL allowed	type	description
ListenerId	yes	no	String	Listener ID Example: lbl-2qaqj593
InstanceId	no	no	String	Node ID Example: ins-1v6jil93
EniIp	no	no	String	ENI IP Example: 10.0.X.X
Port	yes	no	Int64	Bound port Example: 80
Weight	no	no	Int64	Node weight. Value range: [0, 100]. Defaults to 10 if not present during binding. Example: 10
LocationId	no	no	String	Layer-7 rule ID Example: lbl-4qepjl93

ExtraInfo

Reserved. Generally unnecessary for users to concern.

Referenced by the following interfaces: DescribeLoadBalancerListByCertId, DescribeLoadBalancers, DescribeLoadBalancersForVpc

name	required	NULL allowed	type	description
ZhiTong	yes	yes	Bool	Whether to enable VIP Direct Connect Example: 0
TgwGroupName	yes	yes	String	TgwGroup name Example: tg-1

InternetAccessible

Network billing mode, maximum outbound bandwidth

Referenced by the following interfaces: CreateLoadBalancer, DescribeLoadBalancerListByCertId, DescribeLoadBalancers, DescribeLoadBalancersForVpc, InquiryPriceCreateLoadBalancer, InquiryPriceModifyLoadBalancer, ModifyLoadBalancerAttributes

name	required	NULL allowed	type	description
InternetChargeType	no	yes	String	TRAFFIC_POSTPAID_BY_HOUR: Postpaid by traffic on an hourly basis; BANDWIDTH_POSTPAID_BY_HOUR: Postpaid by bandwidth on an hourly basis; BANDWIDTH_PACKAGE: Billed by bandwidth package (Currently, this billing mode is supported only when a ISP is specified.) Example: BANDWIDTH_POSTPAID_BY_HOUR
InternetMaxBandwidthOut	no	yes	Int64	Maximum outbound bandwidth. Unit: Mbps. Value range: 0-65535. This parameter takes effect only on public network CLB. Default: 10 Example: 1
BandwidthpkgSubType	no	yes	String	Bandwidth package type. For example, SINGLEISP. This parameter is not supported on Private Cloud. Example: SINGLEISP

Backend

Details of a real server bound to a listener

Referenced by the following interfaces: DescribeTargets

name	required	NULL allowed	type	description
Type	yes	no	String	Real server type. Valid values: CVM, BMS, and ENI Example: CVM
InstanceId	yes	no	String	Unique backend service ID. For example, ins-gx3qj9x7. Example: ins-gx3qj9x7
Port	yes	no	Int64	Real server listening port Example: 80
Weight	yes	no	Int64	Forwarding weight of a real server. Value range: [0, 100]. Default: 10. Example: 10
PublicIpAddresses	yes	yes	Array of String	Public network IP of the real server Example: 1.1.1.1
PrivateIpAddresses	yes	yes	Array of String	Private network IP of the real server Example: 10.0.0.1
InstanceName	yes	yes	String	Instance name of the real server Example: cvm-1
RegisteredTime	yes	yes	String	Real server binding time Example: 2021-12-29 14:38:50
EniId	yes	yes	String	Unique ENI ID. For example, eni-j9x7gx3g. Example: eni-j9x7gx3g

OwnerLabel

User and tag correspondence

Referenced by the following interfaces: DescribeAppIdLabel

name	required	NULL allowed	type	description
Owner	yes	no	Uint64	User AppId Example: 127000001
Label	yes	no	String	Cluster Tag Example: label1
SetType	yes	no	String	Cluster tag type. Valid values: L4_LAN_CLB, L4_WAN_CLB, L4_CLB, L7_LAN_CLB, L7_WAN_CLB, and L7_CLB Example: L4_LAN_CLB

CertList

Tenant certificate list

Referenced by the following interfaces: DescribeCerts

name	required	NULL allowed	type	description
OwnerUin	yes	no	String	Owner Example: 110000004918
ProjectId	yes	no	String	Project ID Example: 0
From	yes	no	String	Certificate Source Example: upload
CertType	yes	no	String	Product Type Example: SVR
Product2thName	yes	no	String	Product Name Example: zhangsan@gmail.com
Domain	yes	no	String	Project domain name Example: zhangsan@gmail.com
Alias	yes	no	String	Certificate remark or name Example: zhangsan1
Status	yes	no	Int64	Certificate Status Example: 1
VulnerAbilityStatus	yes	no	String	Vulnerability status Example: INACTIVE
CertBeginTime	yes	no	String	Start Time Example: 2024-07-27 16:31:29
CertEndTime	yes	no	String	Expiration Time Example: 2025-07-27 16:31:29
ValidityPeriod	yes	no	String	Policy validity period Example: 12
InsertTime	yes	no	String	Storage time Example: 2024-10-22 10:21:38
CertID	yes	no	String	Certificate ID Example: b4eSkEn
StatusName	yes	no	String	Status name Example: Approved
ProjectInfo	yes	no	ProjectInfo	Project Information Example:View
Cert	no	no	String	Certificate Content Example: -----BEGIN CERTIFICATE-----\r\nMIIEjTCCA3WgAwIBAgIQA4ZHUAVO9cYuYOuRvokzVT4IsaInSLKkKxw=\r\n-----END CERTIFICATE-----\r\n
SignCert	no	yes	String	Content of SM certificate signature Example: -----BEGIN CERTIFICATE-----\r\nMIIEjTCCA3WgYuYOuRvokzVT4IsaInSLKkKxw=\r\n-----END CERTIFICATE-----\r\n
SubjectAltName	yes	yes	Array of String	List of secondary domain names Example: ["domain2.com"]
CertAlgorithm	yes	yes	String	Value type. Valid values: rsa, ecc, and sm2. Default: rsa Example: rsa

ListenerBackend

Information of the real server bound to a listener

Referenced by the following interfaces: DescribeTargets

name	required	NULL allowed	type	description
ListenerId	yes	no	String	Listener ID Example: lbl-di3lx902
Protocol	yes	no	String	Listener protocol Example: TCP
Port	yes	no	Int64	Listener port Example: 80
Rules	yes	yes	Array of RuleTargets	Information of rules under a listener (applicable only to HTTP/HTTPS listeners) Example:View
Targets	yes	yes	Array of Backend	List of real servers bound to the listener (applicable only to TCP/UDP/TCP_SSL listener) Example:View
EndPort	no	yes	Int64	End port in a port range if port range is supported, or 0 if port range is not supported Example: 80

QuotaData

Description of the root account and sub-account quotas

Referenced by the following interfaces: DescribeSubLimQuotas

name	required	NULL allowed	type	description
Type	yes	no	Int64	Quota type (4: Public network instance quota; 7: Private network instance quota) Example: 4

name	required	NULL allowed	type	description
UsedQuota	yes	no	Int64	Used quota Example: 1
CurQuota	yes	no	Int64	Root account quota Example: 3
SubQuota	yes	yes	Array of SubQuotaRsp	Quota list for this type of sub-account Example:View
SubUsedTotalQuota	yes	yes	Int64	Total quota used by the sub-account Example: 5

LBChargePrepaid

Attributes for monthly subscription configuration of a CLB instance

Referenced by the following interfaces: DescribeLoadBalancerListByCertId, DescribeLoadBalancers, DescribeLoadBalancersForVpc, InquiryPriceCreateLoadBalancer, ModifyLoadBalancerAttributes

name	required	NULL allowed	type	description
RenewFlag	no	yes	String	Renewal type. AUTO_RENEW: Automatic renewal; MANUAL_RENEW: Manual renewal Example: MANUAL_RENEW
Period	no	yes	Int64	Purchase period. Unit: month Example: 100

LoadBalancerHealth

CLB instance health check status

Referenced by the following interfaces: DescribeTargetHealth

name	required	NULL allowed	type	description
LoadBalancerId	yes	no	String	CLB instance ID Example: lb-d1i9x902
LoadBalancerName	yes	yes	String	CLB Instance Name Example: lb-1
Listeners	yes	yes	Array of ListenerHealth	Listener list Example:View

RuleTargets

Information of the real server bound to a forwarding rule under an HTTP/HTTPS listener

Referenced by the following interfaces: DescribeTargets

name	required	NULL allowed	type	description
LocationId	yes	no	String	Forwarding rule ID Example: loc-iil2kj93
Domain	yes	no	String	Domain name of the forwarding rule Example: wagnsha.com
Url	yes	no	String	Path of the forwarding rule. Example: /url1
Targets	yes	yes	Array of Backend	Real server information Example:View
FunctionTargets	no	yes	Array of FunctionTarget	Backend cloud function information. (This parameter is not supported on Private Cloud.) Example:View

RewriteLocationMap

Redirection relationship between forwarding rules

Referenced by the following interfaces: DeleteRewrite, ManualRewrite

name	required	NULL allowed	type	description
SourceLocationId	yes	no	String	Source forwarding rule ID Example: loc-iil2kj93
TargetLocationId	yes	no	String	Target forwarding rule ID for redirection Example: loc-lavxjl93
RewriteCode	no	no	Uint64	Redirection status code. Valid values: 301, 302, and 307. Example: 302
TakeUrl	no	no	Bool	Whether the matched URL is carried in redirection. This parameter is required when you configure RewriteCode. Example: true
SourceDomain	no	no	String	Source domain name of redirection, which must be the domain name corresponding to SourceLocationId. This parameter is required when you configure RewriteCode. Example: source.com

CertListener

Attribute of the CLB listener bound to the certificate.

Referenced by the following interfaces: DescribeListenerListByCertId

name	required	NULL allowed	type	description
ListenerId	no	no	String	CLB listener ID Example: lbl-jd8k32x3
ListenerName	no	no	String	Listener name Example: Listener---Portal
LBId	no	no	String	CLB ID Example: lb-jc8k3d0b
Protocol	no	no	String	Listener protocol Example: TCP
Port	no	no	Int64	Listener port Example: 80
SSLMode	no	no	String	Authentication type. UNIDIRECTIONAL: One-way authentication; MUTUAL: Mutual authentication Example: UNIDIRECTIONAL
CertId	no	no	String	Certificate ID. Example: jlkjk23x
CertCaId	no	no	String	Client certificate ID. Example: jlkjk23x
SniSwitch	no	no	Int64	Whether to enable the SNI feature. (This parameter is applicable only to HTTPS listeners.) Example: 0
AddTimestamp	no	no	String	Listener creation time Example: 2024-02-21 12:00:00

LoadBalancer

CLB instance information

Referenced by the following interfaces: DescribeLoadBalancerListByCertId, DescribeLoadBalancers, DescribeLoadBalancersForVpc

name	required	NULL allowed	type	description
LoadBalancerId	no	no	String	CLB instance ID. Example: lb-fxzz9i3x
LoadBalancerName	no	no	String	CLB instance name. Example: Name
LoadBalancerType	no	no	String	CLB instance network type: OPEN: Public network; INTERNAL: Private network. Example: OPEN
Forward	no	no	Uint64	Load balancer type identifier. 1: CLB; 0: Classic CLB. Example: 1
Domain	no	yes	String	Domain name of the CLB instance, provided only for public network Classic CLB instances Example: www.xaa.com
LoadBalancerVips	no	yes	Array of String	List of VIPs of a CLB instance. Example: [43.23.46.32]
Status	no	yes	Uint64	CLB instance status. Valid values: 0: Creating; 1: Normal running. Example: 1
CreateTime	no	yes	String	CLB instance creation time Example: 2020-12-08 08:00:00
StatusTime	no	yes	String	Last status transition time of the CLB instance. Example: 2020-12-08 08:00:00
ProjectId	no	no	Uint64	The ID of the project to which the CLB instance belongs. 0: Default project. Example: 0
VpcId	no	yes	String	VPC ID Example: vpc-mj65svx2
OpenBgp	no	yes	Uint64	Anti-DDoS Pro identifier for CLB. 1: CLB with Anti-DDoS Pro; 0: CLB without Anti-DDoS Pro. Example: 0
Snat	no	yes	Bool	Private network Classic CLB instances before December 2016 had SNAT enabled. Example: false
Isolation	no	yes	Uint64	0: Not isolated; 1: Isolated. Example: 0
Log	no	yes	String	User-enabled log information. Only public network CLB instances that have an HTTP or HTTPS listener can generate logs. Example: null
SubnetId	no	yes	String	Subnet of the CLB instance (applicable only to VPC-type CLB instances on private networks) Example: subnet-lyj3cj09
Tags	no	yes	Array of TagInfo	CLB instance tag information Example:View
SecureGroups	no	yes	Array of String	Security group of the CLB instance Example: sg-hx6djv9x
TargetRegionInfo	no	yes	TargetRegionInfo	Basic information of real servers bound to the CLB instance Example:View
AnycastZone	no	yes	String	Anycast CLB release domain. For non-anycast CLB, this field returns an empty string. Example: null
AddressIPVersion	no	yes	String	IP version. Valid values: ipv4 and ipv6 Example: ipv4
NumericalVpcId	no	yes	Uint64	VPC ID in numerical format Example: 77599
Vipisp	no	yes	String	ISP of the CLB IP address Example: BGP
MasterZone	no	yes	ZoneInfo	Primary Availability Zone Example:View
BackupZoneSet	no	yes	Array of ZoneInfo	Secondary Availability Zone Example:View
IsolatedTime	no	yes	String	CLB instance isolation time Example: 2022-02-12 12:15:08
ExpireTime	no	yes	String	CLB instance expiration time, applicable only to prepaid CLB Example: 2021-02-12 12:15:08
ChargeType	no	yes	String	CLB instance billing type Example: POSTPAID_BY_HOUR
NetworkAttributes	no	yes	InternetAccessible	CLB instance network attributes Example:View
PrepaidAttributes	no	yes	LBChargePrepaid	Prepaid billing attributes of a CLB instance Example:View
LogSetId	no	yes	String	Log set ID of Cloud Log Service (CLS) for CLB Example: c12fc818-e984-414a-b8f5-39858b4c5
LogTopicId	no	yes	String	Log topic ID of Cloud Log Service (CLS) for CLB Example: c12fc818-e984-414a-b8f5-39858b4c5
AddressIPv6	no	yes	String	IPv6 address of the CLB instance Example: 2402:4e00:1000:4e00:0:920b:5519:d96d
ExtraInfo	no	yes	ExtraInfo	Reserved. Generally unnecessary for users to concern. Example:View

name	required	NULL allowed	type	description
IsDDos	no	yes	Bool	Whether an Anti-DDoS Pro instance can be bound Example: false
ConfigId	no	yes	String	Custom configuration IDs of CLB instances Example: pp-pbohak30s
LoadBalancerPassToTarget	no	yes	Bool	Whether the real server allows traffic from CLB Example: false
ExclusiveCluster	no	yes	ExclusiveCluster	Exclusive cluster on the private network Example:View
IPv6Mode	no	yes	String	This field is meaningful when the IP address version is ipv6. Valid values: IPv6Nat64 and IPv6FullChain Example: IPv6FullChain
SnatPro	no	yes	Bool	Whether to enable SnatPro Example: true
SnatIps	no	yes	Array of SnatIp	SnatIp list after SnatPro CLB is enabled Example:View
SlaType	no	yes	String	Performance guarantee specification Example: null
IsBlock	no	yes	Bool	Whether VIP is blocked Example: false
VipIspId	no	yes	Int64	ID of the ISP of the CLB IP address Example: 5
VipIspName	no	yes	String	ISP of the CLB IP address Example: BGP
AttributeFlags	no	yes	Array of String	Attribute tag list Example: DeleteProtect
TgwSetLabels	no	yes	Array of String	Parameter specific to Private Cloud. Layer-4 cluster tag bound to the CLB instance. Example: ["l4-label"]
StgwSetLabels	no	yes	Array of String	Parameter specific to Private Cloud. Layer-7 cluster tag bound to the CLB instance. Example: ["l7-label"]
Zones	no	yes	Array of String	VPC-based private network CLB. AZ where the rule is in proximity access mode Example: null
NfvInfo	no	yes	String	Whether CLB is NFV. Empty: No; l7nfv: NFV for layer 7. Example: null
IsBlockTime	no	yes	String	Blocking or unblocking time Example: 2021-02-12 12:15:08
HealthLogSetId	no	yes	String	Health check log set ID of Cloud Log Service (CLS) for CLB Example: c12fc818-e984-414e-b8f5-398554c51111
HealthLogTopicId	no	yes	String	Health check log topic ID of Cloud Log Service (CLS) for CLB Example: c12fc818-e984-414e-b8f5-398554c52222
LocalBgp	no	yes	Bool	Whether the IP type is Local BGP Example: false
MixIpTarget	no	yes	Bool	If the layer-7 listener of an IPv6FullChain CLB instance is enabled, the CLB instance can be bound with IPv4 and IPv6 CVM instances simultaneously. Example: false
ClusterTag	no	yes	String	Layer-7 exclusive tag. Example: FOR_L7
ClusterIds	no	yes	Array of String	Layer-7 exclusive cluster ID. Example: ["cluster-skd1bk30"]
VipIspAlias	no	yes	String	Alias of the ISP of the CLB IP address Example: BGP

ErrorCode

Feature Description

If there is an Error field in the response, it means that the API call failed. For example:

```
{
    "Response": {
        "Error": {
            "Code": "AuthFailure.SignatureFailure",
            "Message": "The provided credentials could not be validated. Please check your signature is cor
rect."
        },
        "RequestId": "ed93f3cb-f35e-473f-b9f3-0d451b8b79c6"
    }
}
```

Code in Error indicates the error code, and Message indicates the specific information of the error.

Error Code List

Common Error Codes

Error codes	Description
AuthFailure.InvalidSecretId	Invalid key (not a Cloud API key type).
AuthFailure.MFAFailure	MFA failed.
AuthFailure.SecretIdNotFound	Key does not exist. Check if the key has been deleted or disabled in the console, and if not, check if the key is correctly entered. Note that whitespaces should not exist before or after the key.
AuthFailure.SignatureExpire	Signature expired. Timestamp and server time cannot differ by more than five minutes. Please ensure your current local time matches the standard time.
AuthFailure.SignatureFailure	Invalid signature. Signature calculation error. Please ensure you've followed the signature calculation process described in the Signature API documentation.
AuthFailure.TokenFailure	Token error.

Error codes	Description
AuthFailure.UnauthorizedOperation	The request does not have CAM authorization.
DryRunOperation	DryRun Operation. It means that the request would have succeeded, but the DryRun parameter was used.
FailedOperation	Operation failed.
InternalError	Internal error.
InvalidAction	The API does not exist.
InvalidParameter	Incorrect parameter.
InvalidParameterValue	Invalid parameter value.
LimitExceeded	Quota limit exceeded.
MissingParameter	A parameter is missing.
NoSuchVersion	The API version does not exist.
RequestLimitExceeded	The number of requests exceeds the frequency limit.
ResourceInUse	Resource is in use.
ResourceInsufficient	Insufficient resource.
ResourceNotFound	The resource does not exist.
ResourceUnavailable	Resource is unavailable.
UnauthorizedOperation	Unauthorized operation.
UnknownParameter	Unknown parameter.
UnsupportedOperation	Unsupported operation.
UnsupportedProtocol	HTTPS request method error. Only GET and POST requests are supported.
UnsupportedRegion	API does not support the requested region.

Service Error Codes

Error codes	Description
InvalidParameter.RewriteAlreadyExist	The forwarding rule has been bound to a redirection relationship.

Error codes	Description
InvalidParameterValue.Duplicate	Duplicate parameter values.
InvalidParameter.ProtocolCheckFailed	Listener protocol check failed because, for example, the protocol used is incompatible with the corresponding operation.
UnauthorizedOperation	Unauthorized operation
InvalidParameterValue.Range	Invalid parameter value range.
InvalidParameterValue.Length	Invalid parameter length.
LimitExceeded	Quota limit reached
InternalError	Internal Error
InvalidParameter	Wrong parameter.
InvalidParameterValue.InvalidFilter	Filter parameter input error.
InvalidParameter.LBIdNotFound	CLB instance ID error.
ResourceInsufficient	Insufficient resources
InvalidParameter.RegionNotFound	Invalid region.
InvalidParameter.SomeRewriteNotFound	Some redirection rules do not exist.
InvalidParameter.LocationNotFound	Unable to find eligible forwarding rules.
InvalidParameter.ListenerIdNotFound	Listener ID error.
MissingParameter	Missing parameter error.
InvalidParameterValue	Incorrect parameter value
InvalidParameter.PortCheckFailed	Listener port check due to port conflicts or other reasons.
InvalidParameter.FormatError	Incorrect parameter format
FailedOperation	Failed to operate.