Elastic Load Balance

Best Practices

Issue	01
Date	2025-11-14

HUAWEI

HUAWEI CLOUD COMPUTING TECHNOLOGIES CO., LTD.

Trademarks and Permissions


and other Huawei trademarks are the property of Huawei Technologies Co., Ltd.
All other trademarks and trade names mentioned in this document are the property of their respective holders.

Notice

The purchased products, services and features are stipulated by the contract made between Huawei Cloud and the customer. All or part of the products, services and features described in this document may not be within the purchase scope or the usage scope. Unless otherwise specified in the contract, all statements, information, and recommendations in this document are provided "AS IS" without warranties, guarantees or representations of any kind, either express or implied.

The information in this document is subject to change without notice. Every effort has been made in the preparation of this document to ensure accuracy of the contents, but all statements, information, and recommendations in this document do not constitute a warranty of any kind, express or implied.

Contents

1 Load Balancer Selection and Service Planning

1.1 Selecting a Load Balancer That Fits Your Service Requirements

Scenarios

ELB provides dedicated and shared load balancers for you to choose based on your service size and type.

Dedicated load balancers provide fixed and elastic specifications. You can estimate the price based on your service size by referring to **Elastic Load Balance Price Calculator**.

- Elastic specifications work well for fluctuating traffic, and you will be billed for how many LCUs you use.
- Fixed specifications are suitable for stable traffic, and you will be billed for each specification you select.

This practice helps you select a load balancer that suits your service scenario.

Load Balancer Type Comparison

ELB provides dedicated and shared load balancers.

Dedicated load balancers work better than shared load balancers. Dedicated load balancers support a wider range of protocols and can route Layer 7 requests based on advanced forwarding policies you configure.

For details, see **Differences Between Dedicated and Shared Load Balancers**.

Table 1-1 Load balancer types

Item	Dedicated Load Balancer	Shared Load Balancer
Deployment mode	A dedicated load balancer gets dedicated resources. Its performance is not affected by the loads on other load balancers. In addition, there is a wide range of specifications available for you to choose from.	They are deployed in clusters and share resources with other instances. They support guaranteed performance.
Specifications	• Elastic specifications: You are billed for how long a load balancer is running and the number of LCUs you use. • Fixed specifications: Multiple specifications are available for you to select to best meet your needs.	N/A
Performance	A dedicated load balancer in an AZ can establish up to 20 million concurrent connections. If you select more than one AZ when creating a dedicated load balancer, the number of concurrent connections and new connections will be multiplied by the number of AZs. For details, see **Specifications of Dedicated Load Balancers**.	If guaranteed performance is enabled, each shared load balancer can handle up to 50,000 concurrent connections, 5,000 new connections per second, and 5,000 queries per second. The shared load balancer may not process extra requests if the guaranteed connection limit is exceeded.
AZ	You can select one or more AZs as needed.	N/A
Billed item	• Fixed specifications: billed by the LCUs based on the specifications you select. • Elastic specifications: billed by how many LCUs you use and how long a load balancer is retained in your account.	You are billed for how long you use a load balancer if guaranteed performance is enabled.

Key Feature Comparison

Compared with shared load balancers, dedicated load balancers support a wider range of protocols and can route Layer 7 requests based on advanced forwarding policies you configure.

For details, see **Feature Comparison Details**.

Table 1-2 Feature comparison

Item	Dedicated Load Balancer	Shared Load Balancer
Capabilities	Powerful capabilities to process Layer 4 and Layer 7 requests, advanced forwarding policies, and multiple protocols	Basic capabilities to process Layer 4 and Layer 7 requests
Application scenarios	Heavy-traffic and highly concurrent services, such as large websites, cloud-native applications, IoV, and multi-AZ disaster recovery applications	Services with low traffic
Frontend protocols	TCP, UDP, HTTP, HTTPS, QUIC, and TLS	TCP, UDP, HTTP, and HTTPS
Backend protocols	TCP, UDP, HTTP, HTTPS, QUIC, TLS, and gRPC	TCP, UDP, and HTTP
Forwarding capabilities	Provides powerful Layer 4 and Layer 7 processing capabilities to **forward requests** based on the following: • Forwarding rules: domain name, path, HTTP request method, HTTP header, query string, and CIDR block • Actions: forward to a backend server group, redirect to another listener, redirect to another URL, rewrite, and return a specific response body	Provides basic Layer 4 and Layer 7 processing capabilities to **forward requests** based on the following: • Forwarding rules: domain name and path • Actions: forward to a backend server group and redirect to another listener
Key features of backend server groups	• A backend server group can be associated with multiple load balancers and listeners. • Forwarding mode: load balancing and active/standby forwarding	• A backend server group can be associated with only one listener. • Forwarding mode: Load balancing

Item	Dedicated Load Balancer	Shared Load Balancer
Backend server	• ECSs and supplementary network interfaces in the same VPC as the load balancer • IP addresses of servers in different VPCs from the load balancer and on-premises data centers	Cloud servers in the same VPC as the load balancer

Differences Between Application and Network Load Balancing

You can select application load balancing, or network load balancing, or both as required.

- **Application load balancing (HTTP/HTTPS)**: supports HTTP, QUIC, and HTTPS. This option is a great fit for workloads that require high performance at Layer 7, such as real-time audio and video, interactive livestreaming, and game applications.
- **Networking load balancing (TCP/UDP/TLS)**: supports TCP, TLS, and UDP. This works well for heavy-traffic and high-concurrency workloads at Layer 4, such as file transfer, instant messaging, and online video applications.

Table 1-3 Protocols supported by ELB

Type	Protocol	Description	Application Scenario
Network listeners	TCP	• Source IP address–based sticky sessions • Fast data transfer	• Scenarios that require high reliability and data accuracy, such as file transfer, email, and remote login • Web applications that do not need to handle a large number of concurrent requests and do not require high performance
Network listeners	UDP	• Relatively low reliability • Fast data transfer	Scenarios that require quick response, such as video chat, gaming, and real-time financial news

Type	Protocol	Description	Application Scenario
Network listeners	TLS	• An extension of HTTP for encrypted data transmission that can prevent unauthorized access • Unidirectional/ Bidirectional authentication	Scenarios that require ultra-high performance and large-scale TLS offloading
Application listeners	HTTP	• Cookie-based sticky sessions • X-Forward-For request header	Applications that require content identification, for example, web applications and mobile games
Application listeners	HTTPS	• An extension of HTTP for encrypted data transmission that can prevent unauthorized access • Encryption and decryption performed on load balancers • Multiple versions of encryption protocols and cipher suites	Workloads that require encrypted transmission, such as e-commerce and financial services
Application listeners	QUIC	• UDP-based low-latency internet transport layer protocol • Multiplexing without head-of-line blocking • Improved congestion control	Applications with a poor network environment and whose users have to switch between networks

1.2 Planning Subnets for Dedicated Load Balancers

Scenarios

You can follow the subnet planning suggestions in this section to keep load balancers running smoothly and support future service growth.

Proper subnet planning ensures enough IP addresses for service expansion even when load balancers are using too many IP addresses.

Where Subnet IP Addresses Are Used

IP addresses in the frontend subnet will be assigned to dedicated load balancers to communicate with resources over the private network. IP addresses in backend subnets are assigned to forward requests to and perform health checks on backend servers.

Table 1-4 shows where subnet IP addresses are used and how many IP addresses are required in each AZ. You can plan a dedicated backend subnet for a load balancer to ensure enough IP addresses for service expansion even when load balancers are using too many IP addresses.

If you are using IPv4/IPv6 dual stack, you need twice as many IP addresses in a subnet compared to using only IPv4.

NOTE

TLS listeners forward Layer 7 requests and do Layer 7 health checks.

Table 1-4 Required IP addresses in an AZ

Use Case	Subnet	AZ-Dependent or Not	Required IP Addresses in Each AZ
Virtual IP address of the load balancer	Frontend subnet	No	1
Forwarding Layer 4 requests	Backend subnet	Increases linearly with the number of AZs.	• **IP as a Backend** not enabled: 0 • **IP as a Backend** enabled: 4
Layer 4 health checks	Backend subnet	Increases linearly with the number of AZs.	1

Use Case	Subnet	AZ-Dependent or Not	Required IP Addresses in Each AZ
Forwarding Layer 7 requests	Backend subnet	No	20 in general, but can range from 8 to 128. The number of IP addresses required may vary by region and service size. **NOTE** Load balancers sharing the same backend subnet can reuse the IP addresses in the subnet.
Layer 7 health checks	Backend subnet	No	IP addresses that are used to forward Layer 7 requests are reused.

Number of IP Addresses Required by a Load Balancer

If you are using IPv4/IPv6 dual stack, you need twice as many IP addresses in a subnet compared to using only IPv4.

If only IPv4 is used, an IPv4 address in frontend subnet is assigned to the load balancer to receive client requests. If IPv4/IPv6 dual stack is used, both an IPv4 and IPv6 address in the frontend subnet are assigned to the load balancer to receive client requests, regardless of the number of AZs.

The following table lists the number of required IP addresses in each backend subnet if only IPv4 is used.

NOTE

Generally, 20 IP addresses are required in each backend subnet for forwarding Layer 7 requests. However, this number can range from 8 to 128 depending on the region and service size. As shown in **Table 1-5**, 20 is used as an example.

Table 1-5 Required IP addresses in each backend subnet (IPv4 only)

AZs	Request Forwarding Scenario	Minimum Number of IP Addresses	Maximum Number of IP Addresses
One	Layer 4 only	1	5
	Layer 7 only	20	20
	Both Layer 4 and Layer 7	21	25
Two	Layer 4 only	2	10
	Layer 7 only	20	20

AZs	Request Forwarding Scenario	Minimum Number of IP Addresses	Maximum Number of IP Addresses
	Both Layer 4 and Layer 7	22	30
Three	Layer 4 only	3	15
	Layer 7 only	20	20
	Both Layer 4 and Layer 7	23	35

Planning Subnets for Load Balancers

A load balancer usually needs 10 to 20 IP addresses from the backend subnet to forward requests across backend servers. If you use the service subnet with a small CIDR block as your load balancer's backend subnet, you might run out of IP addresses for future service growth.

To address this issue, you can use a subnet that is different from the service subnet as the backend subnet of a load balancer. Also, plan a large CIDR block for the backend subnet to ensure enough IP addresses for service expansion. Then, you can select this subnet as the backend subnet of all load balancers to prevent load balancers from using the service subnet as their backend subnet. Remember that plan a proper subnet size based on service requirements.

If the VPC subnets are insufficient, you can add secondary CIDR blocks by referring to **Adding a Secondary IPv4 CIDR Block to a VPC**.

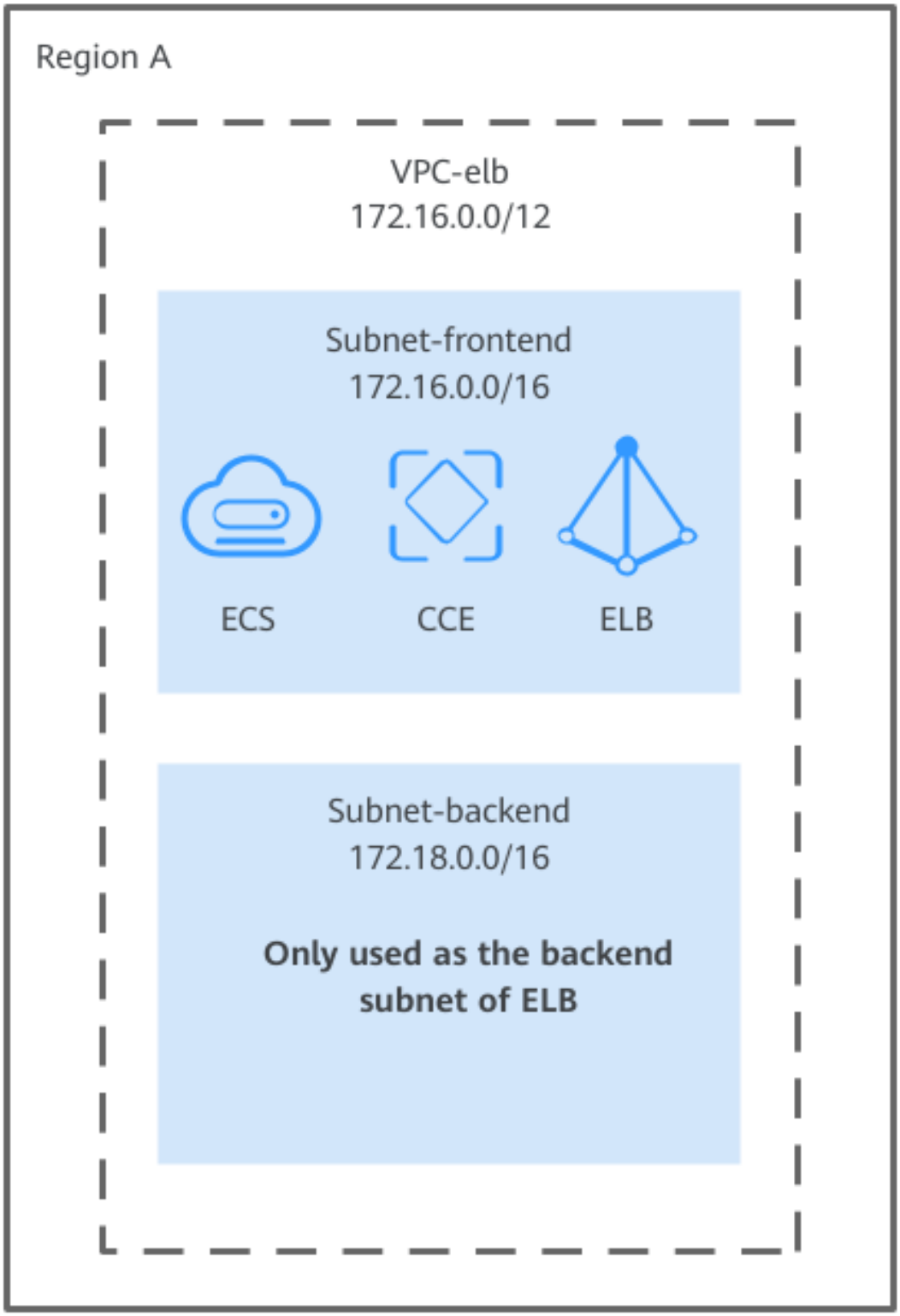
Table 1-6 Recommended subnet planning

Subnet	Scenario	Function
Service subnet	Deploying services	IP addresses are assigned to instances such as ECSs and network interfaces for running services.
Frontend subnet	IP addresses used by load balancers to receive client requests	Assigns IP addresses to load balancers to receive client requests. Service subnets can be used as frontend subnets.
Backend subnet	• Health checks • Forwarding service requests	Assigns IP addresses to load balancers to forward client requests across backend servers. Service subnets are not recommended to be used as backend subnets.

In **Figure 1-1**, subnet **Subnet-frontend** in region A is used as both the service subnet and the load balancer's frontend subnet. You are advised to deploy

backend servers in this subnet and use subnet **Subnet-backend** only as the load balancer's backend subnet.

Figure 1-1 One VPC with two subnets



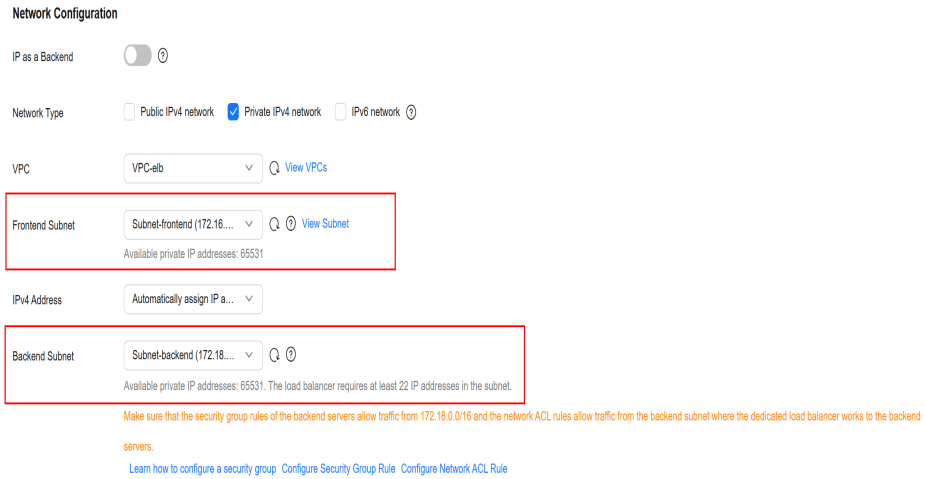
Key Procedure

Step 1 Create a VPC (**VPC-elb**) and two subnets (**Subnet-frontend** and **Subnet-backend**), as shown in **Figure 1-1**, by referring to **Creating a VPC with a Subnet**.

Step 2 Create a dedicated load balancer by referring to **Creating a Dedicated Load Balancer**.

Select **VPC-elb** created in **Step 1** for **VPC**, **Subnet-frontend** created in **Step 1** for **Frontend Subnet**, and **Subnet-backend** created in **Step 1** for **Backend Subnet**, as shown in **Figure 1-2**.

Figure 1-2 Configuring network parameters



Step 3 Add security group rules to allow traffic from the backend subnet where the load balancer works to the backend servers by referring to **Configuring Security Group Rules for Backend Servers**.

----End

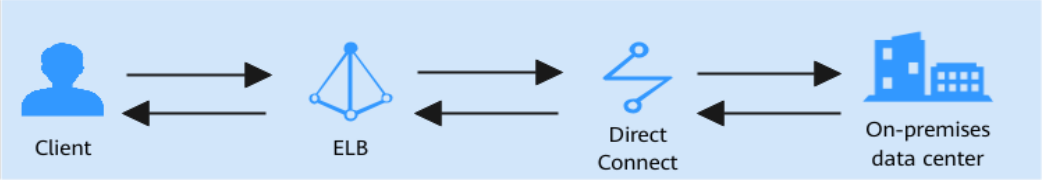
1.3 Adding the IP Address of an On-Premises Server as a Backend Server of a Dedicated Load Balancer

Scenarios

You can use Direct Connect or VPN to connect on-premises servers to Huawei Cloud, and then add the IP addresses of on-premises servers as backend servers of a dedicated load balancer. In this way, the load balancer can distribute traffic to these servers.

Solution Architecture

Figure 1-3 Distributing traffic to an on-premises server



In this practice, a Direct Connect connection is required to connect on-premises servers to a VPC and a dedicated load balancer is required to route requests to on-premises servers.

- Dedicated load balancer **ELB-Test** is running in a VPC named **VPC-Test-01** (172.16.0.0/12).
- An on-premises server (**IDC-IP-Test**) is deployed in the on-premises data center.
- The IP address of **IDC-IP-Test** is added to the backend server group of **ELB-Test**.

Resource Planning

In this practice, you need to create a VPC, dedicated load balancer, EIP, Direct Connect connection, ECS, and on-premises server. For details about the resource planning, see **Table 1-7**.

This practice uses public network access as an example. You can also use a load balancer to route client requests over a private network.

Table 1-7 Resource planning

Resource Type	Quantity	Description
VPC and subnet	One VPC and two subnets	• **VPC-Test-01**: The CIDR block is 172.16.0.0/12. • **subnet-frontend**: The CIDR block is 172.16.0.0/16. • **subnet-backend**: The CIDR block is 172.18.0.0/16.
Load balancer	1	The load balancer that is used to distribute client requests. • **VPC**: **VPC-Test-01** • Frontend subnet: **subnet-frontend** • Backend subnet: **subnet-backend**
Direct Connect connection	1	The connection that is used to connect the on-premises data center to a VPC.

Resource Type	Quantity	Description
ECS	1	**ECS-client**, which is used to access the on-premises server.
EIP	2	• One EIP will be bound to the load balancer to distribute client requests over the public network. • The other EIP will be bound to **ECS-client** to access the load balancer over the public network.
On-premises server	1	**IDC-IP-Test**, which is deployed in the on-premises data center.

Preparations

- Purchase an ECS (**ECS-client**) as the client and bind an EIP to it. For details about how to purchase an ECS, see **Purchasing and Using a Linux ECS (New Edition)**.
- Create a VPC (**VPC-Test-01**) with two subnets (**subnet-frontend** and **subnet-backend**). For details, see **Creating a VPC and Subnet**.
- Create a dedicated load balancer, enable **IP as a Backend**, and bind an EIP to the load balancer. For details, see **Buying a Dedicated Load Balancer** and **Binding an IPv4 EIP**.

Step 1: Connect an On-Premises Data Center to a VPC Using Direct Connect

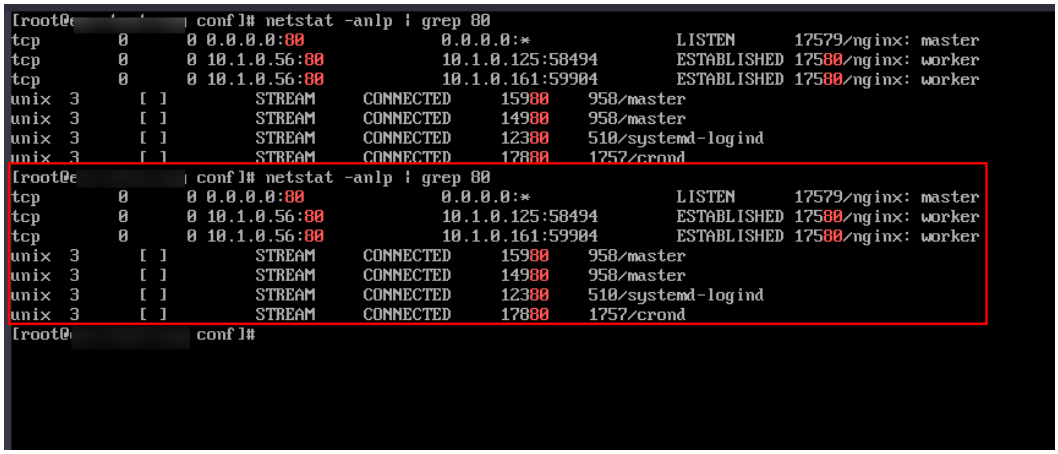
You can create a Direct Connect connection on the console to connect your on-premises data center or on-premises private network to a VPC.

For details, see **Using Direct Connect to Connect an On-Premises Data Center to the Cloud**.

Step 2: Deploy a Service on the On-Premises Server

Deploy Nginx on the on-premises server **IDC-IP-Test** to verify network connectivity.

Figure 1-4 Nginx deployed on the on-premises server



Step 3: Create a Backend Server Group and Add the IP Address of the On-Premises Server as a Backend Server

1. Go to the **backend server group list page**.
2. Click **Create Backend Server Group** in the upper right corner.
3. Configure the parameters based on **Table 1-8**. Retain the default values for other parameters.

Table 1-8 Parameters required for configuring a backend server group

Parameter	Example Value	Description
Name	server_group	Specifies the name of the backend server group.
Type	Dedicated	Specifies the type of the load balancer that can use the backend server group.
Load Balancer	Associate existing	Specifies whether to associate a load balancer now. Click **Associate existing** and select a load balancer you have created.
Backend Protocol	HTTP	Specifies the protocol that backend servers in the backend server group use to receive requests from the listeners. Select **HTTP**.

Parameter	Example Value	Description
Load Balancing Algorithm	Weighted round robin	Specifies the load balancing algorithm used by the load balancer to distribute traffic. **Weighted round robin**: Requests are routed to different servers based on their weights. Backend servers with higher weights receive proportionately more requests, whereas equal-weighted servers receive the same number of requests. For more information, see **Load Balancing Algorithms**.

4. Click **Next** to add backend servers and configure health check.
5. Switch to the **IP as Backend Servers** tab and click **Add IP as Backend Server**.
6. On the displayed page, add the IP address of the on-premises server as a backend server and set the parameters as follows:
   - **IP Address**: Set it to **10.1.0.56** in this example, which is the private IP address of the on-premises server **IDC-IP-Test**.
   - **Backend Port**: Set this parameter based on service requirements. In this practice, port 80 is used.
   - **Weight**: Retain the default value 1.
7. Click **OK**.
8. Enable health check and retain the default values for other health check parameters.
9. Click **Next**.
10. Confirm the configurations and click **Create Now**.

Step 4: Add a Listener and Select the Created Backend Server Group

1. Go to the **load balancer list page**.
2. Locate the target load balancer and click **Add Listener** in the **Operation** column.
3. On the **Add Listener** page, set **Frontend Protocol** to **HTTP** and **Listening Port** to **80**. Retain the default values for other parameters.
4. Click **Next: Configure Request Routing Policy** and select **Use existing** for **Backend Server Group**.

   Select the backend server group created in **Step 3** and click **Next: Confirm**.
5. Confirm the configurations and click **Submit**.

Step 5: Verify Request Routing to the On-Premises Server

1. Use the client to access the on-premises server to verify network connectivity.
   a. Remotely log in to **ECS_client**.

      Multiple methods are available for logging in to an ECS. For details, see **Logging In to an ECS**.

b. Run the **curl -v http://**<*IP-address*>**:**<*port*> command to test network connectivity.

In this practice, run the following command:

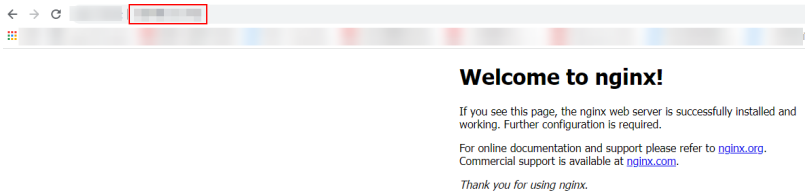
```
curl -v http://<EIP-of-the-load-balancer>:<Listening-port>
```

If the default Nginx welcome page is displayed, ELB successfully forwards the client request to the on-premises server.

2. Use a browser to verify the network connectivity.

Use a browser to access **http://**<*EIP-of-the-load-balancer*>. If the following page is displayed, the request is forwarded by the load balancer to the on-premises server.

Figure 1-5 The Nginx default welcome page when accessed using a browser



1.4 Using IP as a Backend to Route Traffic Across Backend Servers

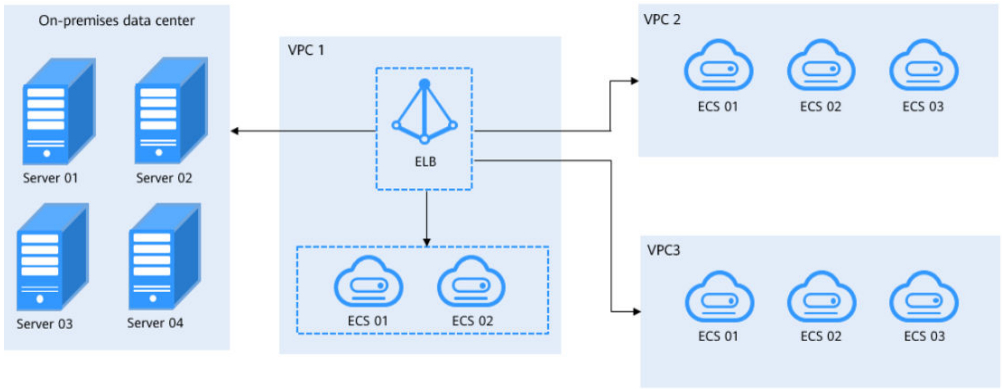
1.4.1 Overview

Scenarios

You have servers both in VPCs and your on-premises data center and want to use load balancers to distribute incoming traffic across these servers.

This section describes how you can use a dedicated load balancer to route incoming traffic across cloud and on-premises servers.

Figure 1-6 Routing traffic across cloud and on-premises servers



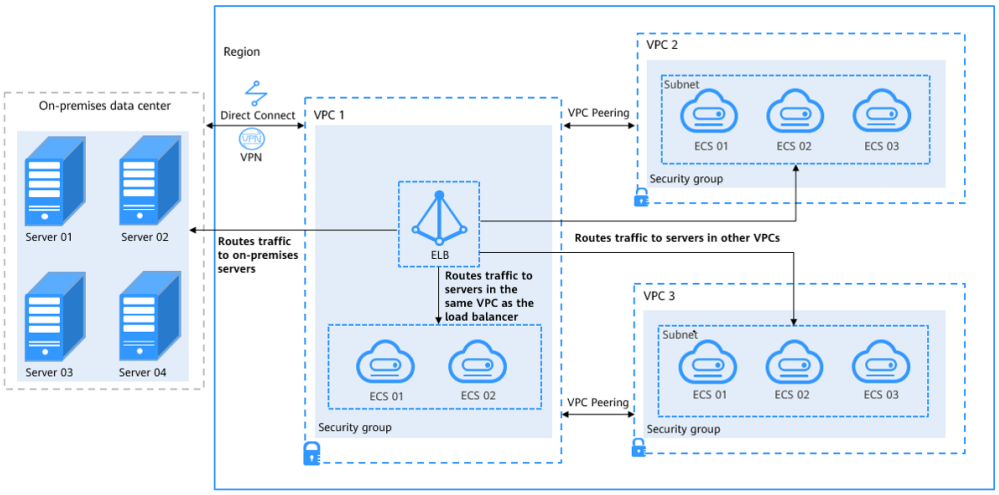
Solution

You can enable **IP as a Backend** when creating a dedicated load balancer and associate cloud and on-premises servers with this dedicated load balancer using their IP addresses.

As shown in **Figure 1-7**, ELB can realize hybrid load balancing.

- You can associate the servers in the same VPC as the load balancer no matter whether you enable **IP as a Backend**.
- If you enable **IP as a Backend**:
  - You can associate on-premises servers with the load balancer after the on-premises data center is connected to the cloud through Direct Connect or VPN.
  - You can also associate the servers in other VPCs different from the load balancer after the VPCs are connected to the VPC where the load balancer is running over VPC peering connections.
  - You can associate the servers in the same VPC as the load balancer.

Figure 1-7 Associating servers with the load balancer



Advantages

You can add servers in the VPC where the load balancer is created, in a different VPC, or in an on-premises data center, by using private IP addresses of the servers to the backend server group of the load balancer. In this way, incoming traffic can be flexibly distributed to cloud servers and on-premises servers for hybrid load balancing.

- You can add backend servers in the same VPC as the load balancer.
- You can add servers in other VPCs different from the load balancer by establishing a VPC peering connection between the two VPCs.
- You can add on-premises servers by connecting your on-premises data center to the cloud through Direct Connect or VPN.

Notes and Constraints

When you add IP as backend servers, note the following:

- Enable **IP as a Backend** first on the **Summary** page of the load balancer.
- IP as backend servers must use IPv4 addresses.
- To ensure requests are properly routed, IP as backend servers cannot use public IP addresses.
- To add IP as backend servers, the subnet where the load balancer works must have at least 16 available IP addresses. You can add more subnets for more IP addresses on the **Summary** page of the load balancer.
- To ensure normal health checks, security group rules configured for IP as backend servers must allow traffic from the backend subnet of the load balancer.
- **IP as a Backend** cannot be disabled after it is enabled.

1.4.2 Routing Traffic to Backend Servers in a Different VPC from the Load Balancer

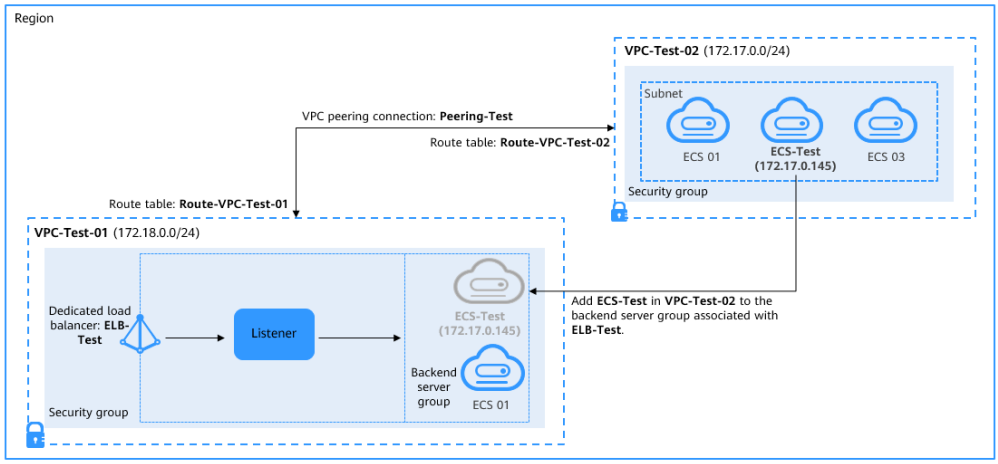
Scenarios

You can use ELB to route traffic to backend servers in a VPC that is different from where the load balancer works.

Solution

- Dedicated load balancer **ELB-Test** is running in a VPC named **VPC-Test-01** (172.18.0.0/24).
- An ECS named **ECS-Test** is running in **VPC-Test-02** (172.17.0.0/24).
- **IP as a Backend** is enabled for **ELB-Test**, and **ECS-Test** in **VPC-Test-02** (172.17.0.0/24) is added to the backend server group associated with **ELB-Test** in **VPC-Test-01**.

Figure 1-8 Topology



Advantages

You can enable **IP as a Backend** for a dedicated load balancer to route incoming traffic to servers in different VPCs from the load balancer.

Resource and Cost Planning

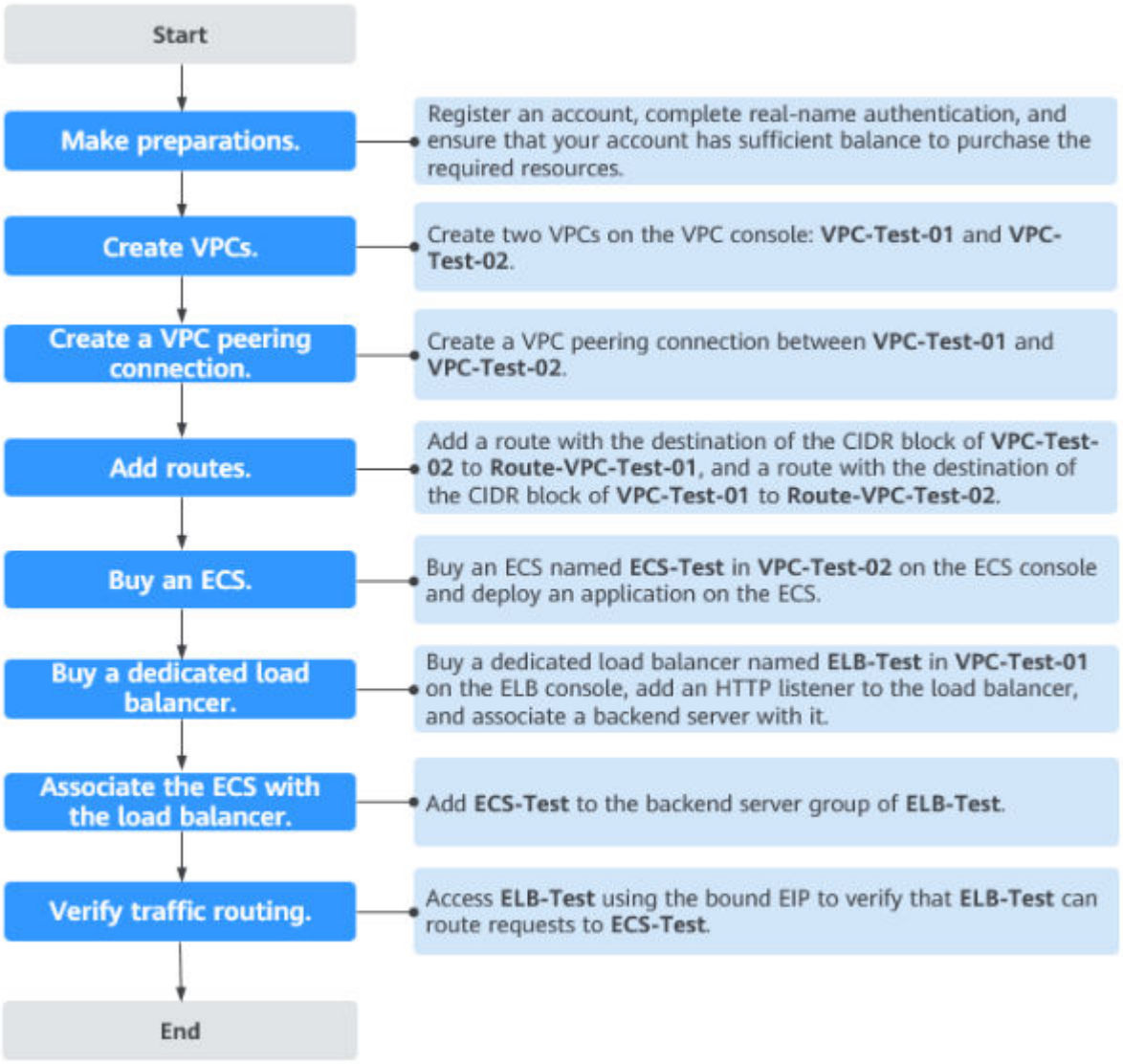
The actual cost shown on the Huawei Cloud console is used.

Table 1-9 Resource planning

Resource Type	Resource Name	Description	Quantity
VPC	VPC-Test-01	The VPC where **ELB-Test** is running: 172.18.0.0/24	1
	VPC-Test-02	The VPC where **ECS-Test** is running: 172.17.0.0/24	1
VPC peering connection	Peering-Test	The connection that connects the VPC where **ELB-Test** is running and the VPC where **ECS-Test** is running. **Local VPC**: **172.18.0.0/24** **Peer VPC**: **172.17.0.0/24**	1
Route table	Route-VPC-Test-01	The route table of **VPC-Test-01**. **Destination**: **172.17.0.0/24**	1
	Route-VPC-Test-02	The route table of **VPC-Test-02**. **Destination**: **172.18.0.0/24**	1
ELB	ELB-Test	The dedicated load balancer to distribute incoming traffic.	1
EIP	EIP-Test	The EIP bound to **ELB-Test**: 119.3.233.52	1
ECS	ECS-Test	The ECS that is running in **VPC-Test-02**. **Private IP address**: **172.17.0.145**	1

Operation Process

Figure 1-9 Process of associating servers in a VPC that is different from the dedicated load balancer

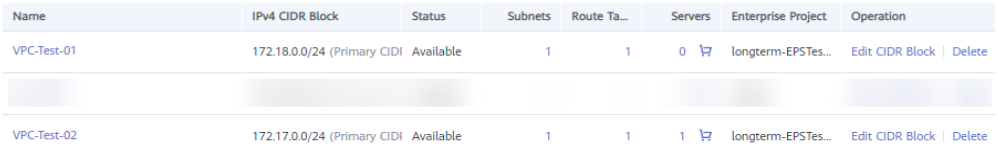


Step 1: Create VPCs

1. Log in to the **VPC console**.
2. Configure the parameters as described in **Table 1-9** and click **Create Now**. For details on how to create a VPC, see the ***Virtual Private Cloud User Guide***.
   - **Name**: **VPC-Test-01**
   - **IPv4 CIDR Block**: **172.18.0.0/24**
   - Configure other parameters as required.
3. Create the other VPC.
   - **Name**: **VPC-Test-02**
   - **IPv4 CIDR Block**: **172.17.0.0/24**

- Configure other parameters as required.

Figure 1-10 Viewing the two VPCs



Step 2: Create a VPC Peering Connection

1. In the navigation pane on the left, click **VPC Peering Connections**.
2. In the upper right corner, click **Create VPC Peering Connection**.
3. Configure the parameters as follows and click **Create Now**. For details on how to create a VPC peering connection, see the ***Virtual Private Cloud User Guide***.
   - **Name**: **Peering-Test**
   - **Local VPC**: **VPC-Test-01**
   - **Peer VPC**: **VPC-Test-02**
   - Configure other parameters as required.

Step 3: Add Routes for Peering-Test

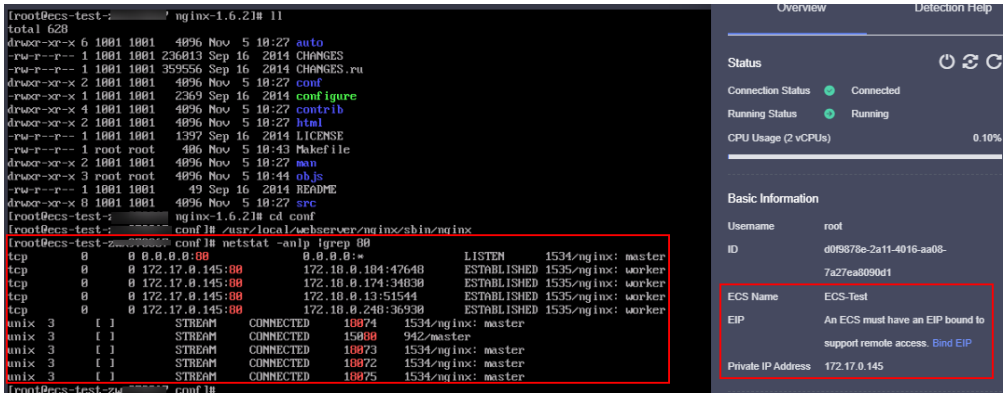
1. In the navigation pane on the left, click **Route Tables**.
2. In the upper right corner, click **Create Route Table**.
3. Configure the parameters as described in **Table 1-9** and click **OK**. For details on how to create a route table, see the ***Virtual Private Cloud User Guide***.
   - **Name**: **Route-VPC-Test-01**
   - **VPC**: **VPC-Test-01**
   - **Destination**: **172.17.0.0/24**
   - **Next Hop Type**: **VPC peering connection**
   - **Next Hop**: **Peering-Test**
4. Repeat the preceding steps to create the other route table.
   - **Name**: **Route-VPC-Test-02**
   - **VPC**: **VPC-Test-02**
   - **Destination**: **172.18.0.0/24**
   - **Next Hop Type**: **VPC peering connection**
   - **Next Hop**: **Peering-Test**

Step 4: Create an ECS

1. Under **Compute**, click **Elastic Cloud Server**.
2. In the upper right corner, click **Buy ECS**.
3. Select **VPC-Test-02** as the VPC and set **ECS Name** to **ECS-Test**. Configure other parameters as required. For details, see **Purchasing an ECS**.

4. Deploy Nginx on the **ECS-Test**.

Figure 1-11 Deploying Nginx on **ECS-Test**



Step 5: Create a Dedicated Load Balancer with an HTTP Listener and Associate a Backend Server Group

1. Go to the **Buy Elastic Load Balancer** page.
2. Configure the parameters as follows. For details, see **Elastic Load Balance User Guide**.
   - **Type**: **Dedicated load balancer**
   - **VPC**: **VPC-Test-01**
   - **Name**: **ELB-Test**
   - **IP as a Backend**: Enable it.
   - Configure other parameters as required.
3. Add an HTTP listener to **ELB-Test** and associate a backend server group with it.

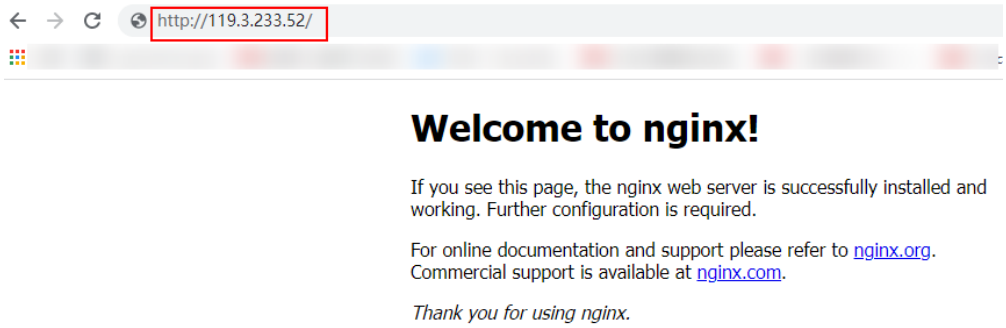
Step 6: Add ECS-Test to the Backend Server Group

1. Locate **ELB-Test** and click its name.
2. On the **Listeners** tab, locate the HTTP listener added to **ELB-Test** and click its name.
3. On the **Default Backend Server Group** area of the **Summary** tab, click **View/Add Backend Server** on the right.
4. The page for adding backend servers is displayed.
5. Click **IP as Backend Servers** on the lower part of the page. Click **Add** on the right, set parameters as required, and click **OK**. For details, see *Elastic Load Balance User Guide*.
   - **IP Address**: Set it to the private IP address of **ECS-Test** (172.17.0.145).
   - **Backend Port**: Set it as required.
   - **Weight**: Set it as required.
6. Click **OK**.

Step 7: Verify Traffic Routing

1. Locate **ELB-Test** and click **More** in the **Operation** column.
2. Select **Bind IPv4 EIP** to bind an EIP (**EIP-Test**: 119.3.233.52) to **ELB-Test**.
3. Enter **http://119.3.233.52/** in the address box of your browser to access **ELB-Test**. If the following page is displayed, **ELB-Test** routes the request to **ECS-Test**, which processes the request and returns the requested page.

Figure 1-12 Verifying that the request is routed to **ECS-Test**



1.4.3 Routing Traffic to Backend Servers in the Same VPC as the Load Balancer

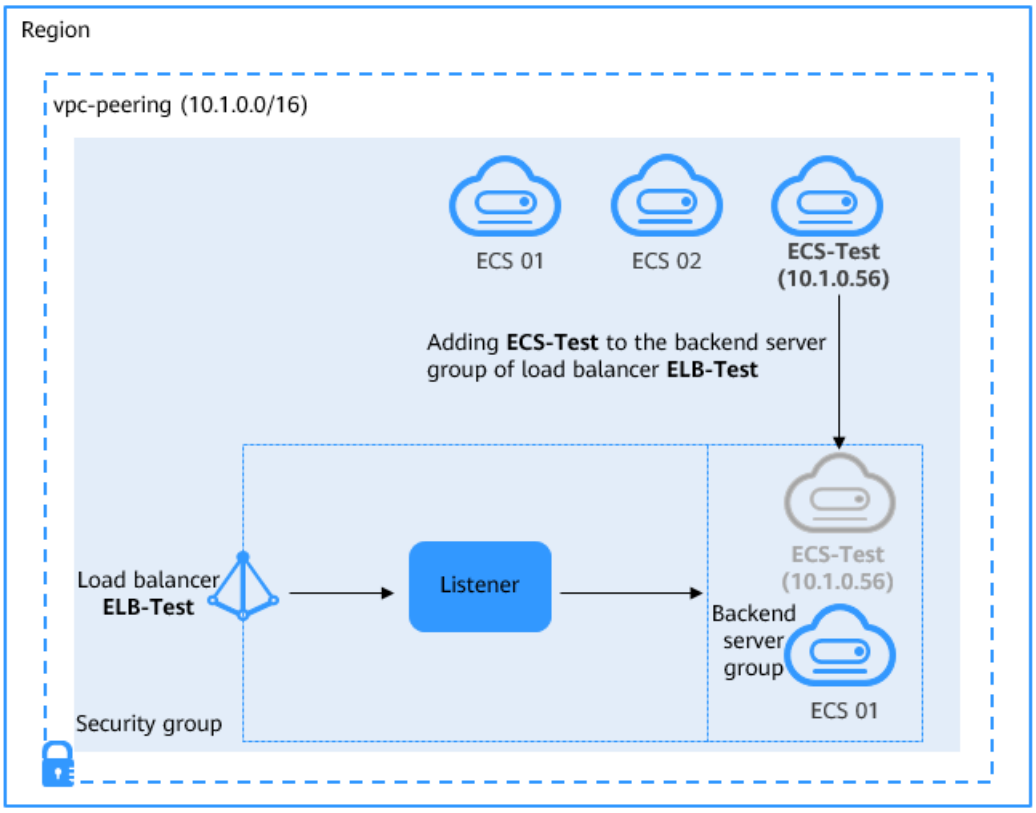
Scenarios

You can use ELB to route traffic to backend servers in the same VPC as the load balancer.

Solution

- Dedicated load balancer **ELB-Test** is running in a VPC named **vpc-peering** (10.1.0.0/16).
- An ECS named **ECS-Test** is also running in **vpc-peering** (10.1.0.0/16).
- **IP as a Backend** is enabled for **ELB-Test**, and **ECS-Test** in **vpc-peering** is added to the backend server group associated with **ELB-Test** in **vpc-peering**.

Figure 1-13 Topology



Advantages

You can enable **IP as a Backend** for a dedicated load balancer to route traffic to backend servers in the same VPC as the load balancer.

Resource and Cost Planning

The actual cost shown on the Huawei Cloud console is used.

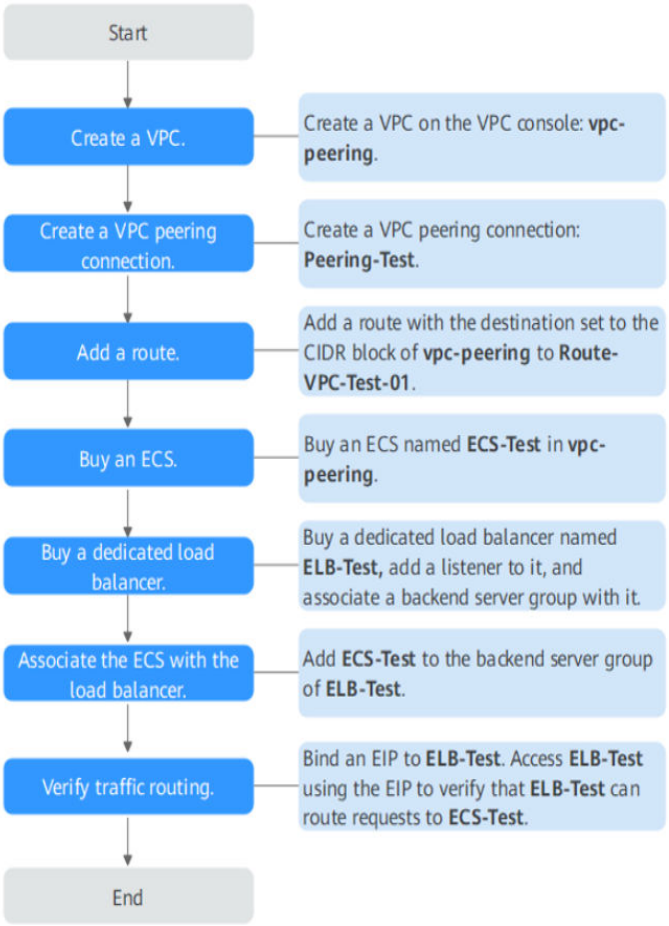
Table 1-10 Resource planning

Resource Type	Resource Name	Description	Quantity
VPC	vpc-peering	The VPC where **ELB-Test** and **ECS-Test** are running: 10.1.0.0/16	1
VPC peering connection	Peering-Test	The connection that connects the VPC where **ELB-Test** is running and other VPCs. **Local VPC**: **10.1.0.0/16** **Peer VPC**: any VPC	1

Resource Type	Resource Name	Description	Quantity
Route table	Route-VPC-Test-01	The route table of **vpc-peering**. **Destination**: **10.1.0.0/16**	1
ELB	ELB-Test	The dedicated load balancer to distribute incoming traffic. **Private IP address**: **10.1.0.9**	1
EIP	EIP-Test	The EIP bound to **ELB-Test**: 120.46.131.153	1
ECS	ECS-Test	The ECS that is running in **vpc-peering**. **Private IP address**: **10.1.0.56**	1

Operation Process

Figure 1-14 Process for adding backend servers in the same VPC as the load balancer



Step 1: Create a VPC

1. Log in to the **VPC console**.
2. Configure the parameters as follows and click **Create Now**. For details on how to create a VPC, see the ***Virtual Private Cloud User Guide***.
   - **Name**: **vpc-peering**
   - **IPv4 CIDR Block**: **10.1.0.0/16**
   - Configure other parameters as required.

Step 2: Create a VPC Peering Connection

1. In the navigation pane on the left, click **VPC Peering Connections**.
2. In the upper right corner, click **Create VPC Peering Connection**.
3. Configure the parameters as follows and click **Create Now**. For details on how to create a VPC peering connection, see the ***Virtual Private Cloud User Guide***.
   - **Name**: **Peering-Test**
   - **Local VPC**: **vpc-peering**
   - **Peer VPC**: any VPC
   - Configure other parameters as required.

Step 3: Add Routes for Peering-Test

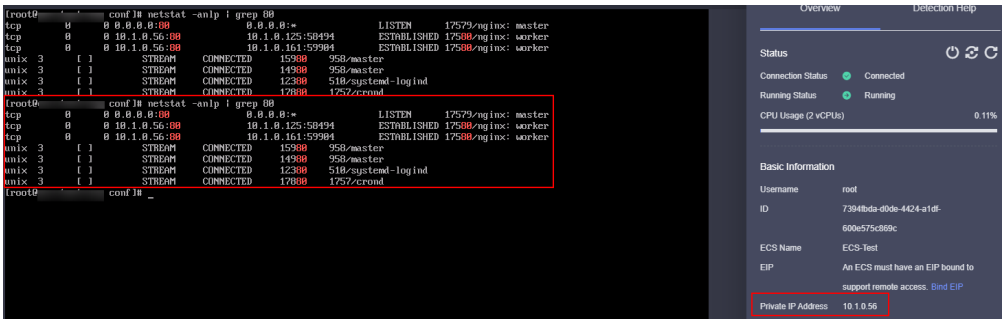
1. In the navigation pane on the left, click **Route Tables**.
2. In the upper right corner, click **Create Route Table**.
3. Configure the parameters as follows and click **OK**. For details on how to create a route table, see the ***Virtual Private Cloud User Guide***.
   - **Name**: **Route-VPC-Test-01**
   - **VPC**: **vpc-peering**
   - **Destination**: 10.1.0.0/16
   - **Next Hop Type**: **VPC peering connection**
   - **Next Hop**: **Peering-Test**

Step 4: Create an ECS

1. Go to the **Buy ECS** page.
2. Configure the parameters as required. For details, see the **Elastic Cloud Server User Guide**.

   Select **vpc-peering** as the VPC and set **Name** to **ECS-Test**.
3. Deploy Nginx on the **ECS-Test**.

Figure 1-15 Deploying Nginx on **ECS-Test**



Step 5: Create a Dedicated Load Balancer with an HTTP Listener and Associate a Backend Server Group

1. Go to the **Buy Elastic Load Balancer** page.
2. Configure the parameters as follows. For details, see **Elastic Load Balance User Guide**.
   - **Type**: **Dedicated**
   - **VPC**: **vpc-peering**
   - **Name**: **ELB-Test**
   - **IP as a Backend**: Enable it.
   - Configure other parameters as required.
3. Add an HTTP listener to **ELB-Test** and associate a backend server group with it.

Step 6: Add ECS-Test to the Backend Server Group

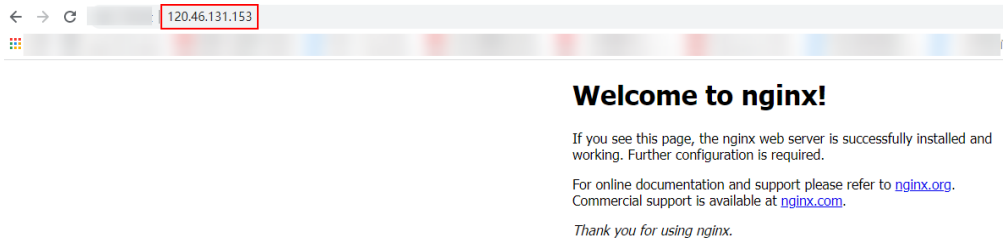
Locate **ELB-Test** and click its name.

1. On the **Default Backend Server Group** area of the **Summary** tab, click **View/Add Backend Server** on the right.
2. The page for adding backend servers is displayed.
3. Click **IP as Backend Servers** on the lower part of the page. Click **Add** on the right, set parameters as required, and click **OK**. For details, see *Elastic Load Balance User Guide*.
   - **IP Address**: Set it to the private IP address of **ECS-Test** (10.1.0.56).
   - **Backend Port**: Set it as required.
   - **Weight**: Set it as required.

Step 7: Verify Traffic Routing

1. Locate **ELB-Test** and click **More** in the **Operation** column.
2. Select **Bind IPv4 EIP** to bind an EIP (120.46.131.153) to **ELB-Test**.
3. Enter **http://120.46.131.153/** in the address box of your browser to access **ELB-Test**. If the following page is displayed, **ELB-Test** routes the request to **ECS-Test**, which processes the request and returns the requested page.

Figure 1-16 Verifying that the request is routed to **ECS-Test**



1.5 Adding a Load Balancer to an AS Group to Automatically Add or Remove Backend Servers

After you add a load balancer to an AS group, Auto Scaling adds servers when there are sudden traffic spikes, and then removes them when traffic returns to normal, according to your configured policies. This saves resources and costs while maintains smooth service operation.

Scenarios

Increased website traffic raises cloud server CPU usage, which falls when traffic declines. In this case, you can configure two CPU usage alarm policies, one for adding a server when the CPU usage goes above 70% and the other for removing a server when the CPU usage drops below 30%. This ensures your website has the right number of servers as traffic changes.

Procedure

This practice provides an example for setting up a scalable Discuz! forum website.

Table 1-11 Setting up a Discuz! forum

Step	Category	Substep	Description
Setting up a website	Applying for resources	Creating a VPC	Create a VPC, for example, **vpc-DISCUZ**, that provides network services for the ECS on which the website is deployed.
		Purchasing an EIP	Purchase an EIP that allows ECSs to access the Internet.
		Creating a security group and adding security group rules	To ensure network security, create a security group, for example, **sg-DISCUZ**, to control network access.

Step	Category	Substep	Description
		Purchasing ECSs	Buy two ECSs, for example, discuz01 for deploying the forum database and discuz02 for deploying the forum. Bind the purchased EIP to the ECS discuz01 when purchasing discuz01.
	Configuring the ECSs	Installing the database on discuz01	Install an RDS for MySQL DB instance on discuz01, start the database, and configure it to automatically start upon ECS startup.
		Deploying the website code on discuz02	Unbind the EIP from discuz01, bind it to discuz02, and deploy a web environment and website code on discuz02.
	Configuring features	Unbinding the EIP	To save EIP resources, release the EIP bound to discuz02 before using the load balancing service.
		Configuring ELB	Purchase a dedicated load balancer, for example, **elb-DISCUZ**, to balance the website traffic in an AS group.
		Creating an image	To ensure that the ECSs to be added to the AS group automatically deploy the web environment and website code, create an image, for example, **discuz_centos6.5 (40 GB)** based on discuz02. The image is used as a private one for creating an AS configuration.
Configuring AS	-	Creating an AS configuration	An AS configuration is an ECS template in the AS group, specifying specifications of the ECS to be added. Create an AS configuration, for example, **as-config-discuz**.
		Creating an AS group	An AS group is the basis for performing scaling actions. Create an AS group, for example, **as-group-discuz**.
		Creating an AS policy	An AS policy triggers scaling actions. Configure two CPU usage alarm policies for increasing or reducing the number of ECSs when website traffic rises or drops.
		Manually adding an ECS to an AS group	To ensure that discuz02 and the ECSs to be added to the AS group carry forum services together, manually add discuz02 to the AS group.

Step	Category	Substep	Description
		Configuring **Min. Instances**	**Min. Instances** defines the minimum number of ECSs in an AS group. When the parameter is set to **1**, there is at least one ECS in the AS group. The ECS discuz02 is manually added and has the lowest priority to be removed when **Instance Removal Policy** is configured. Therefore, configuring **Min. Instances** prevents discuz02 from being removed.
Visiting the forum website	Verifying the configuration	Checking whether the forum website can be accessed	Obtain the EIP bound to the load balancer and visit http://*EIP*/forum.php. If the forum website is accessible, the configurations have taken effect.

Preparations

- **Set up a Discuz! forum**.
- **Create a dedicated load balancer (elb-DISCUZ)**, **bind an EIP to the load balancer**, and ensure that the load balancer and the AS group are in the same VPC.
- Create a backend server group and associate it with dedicated load balancer elb-DISCUZ.

Step 1: Create an AS Configuration

An AS configuration specifies the specifications of servers to be added. To enable the servers to automatically run web services, use the image discuz_centos6.5 (40 GB) and ensure the parameter settings in the AS configuration are the same as those of discuz02.

1. Log in to the management console. Under **Compute**, click **Auto Scaling**.
2. On the **Instance Scaling** page, click **Create AS Configuration**.

   Configure parameters listed in **Table 1-12**. Retain default settings for other parameters.

   Table 1-12 Parameters required for creating an AS configuration

Parameter	Description	Example Value
Configuration Template	Select **Create new template** and configure the parameters, such as the ECS type, vCPUs, memory, image, and disks, to create an AS configuration.	Create new template

Parameter	Description	Example Value
Specifications	You can select multiple flavors to minimize the probability of capacity expansion failures due to insufficient resources of a flavor. Set **Flavor selection policy** to **Sequenced** or **Cost-centric** as required.	s3.medium.2 s3.large.2
Image	Specifies the software and system configuration template for the instances in an AS group.	discuz_centos6.5 (40 GB)
Disk	Stores data and manages the stored data for the instances in an AS group.	System disk: high I/O, 40 GB Data disk: high I/O, 100 GB
Security Group	Controls ECS access within or between security groups by defining access rules. Select security group **sg-DISCUZ**.	sg-DISCUZ
EIP	Not required if you have configured a load balancer for an AS group. The system automatically associates instances in the AS group to the load balancer. These instances will provide services via the EIP bound to the load balancer.	Do not use

3. After setting the parameters, click **Create Now**.

Step 2: Create an AS Group and Associate It with the Load Balancer

1. Click **Create AS Group**.

   Configure parameters listed in **Table 1-13**. Retain default settings for other parameters.

   Table 1-13 Parameters required for creating an AS group

Parameter	Description	Example Value
Max. Instances	Specifies the maximum number of instances in an AS group.	50

Parameter	Description	Example Value
Expected Instances	Specifies the expected number of instances in an AS group. The ECSs in this practice are manually added to the AS group. To prevent scaling actions before the manually adding, set **Expected Instances** to **0**.	0
Min. Instances	Specifies the minimum number of instances in an AS group.	0
VPC	Provides the network used by instances in an AS group. Ensure that the parameter value is the same as the VPC in which discuz02 is deployed.	VPC-DISCUZ
Subnet	Manages networks in the VPC. Select the subnet created when you apply for the VPC.	vpc-test
Load Balancing	Evenly distributes traffic to instances in an AS group. Select the dedicated load balancer **elb-DISCUZ**. A backend port is a service port on which a backend ECS listens for traffic. For example, set **Backend Port** to **80** and **Weight** to **1**.	Dedicated load balancer
Health Check Method	**ELB health check** is recommended, in which heartbeat messages are sent to backend ECSs for check.	ELB health check

2. After setting the parameters, click **Create Now**.
3. Back to the AS group list. The AS group is successfully created if its status changes to **Enabled**.

Step 3: Create an AS Policy

To automatically scale cloud servers, configure two alarm policies to monitor CPU usage, one (as-policy-discuz01) for adding servers when the website traffic rises, and the other (as-policy-discuz02) for removing servers when the website traffic drops.

1. Locate the row containing the created AS group **as-group-discuz** and click **View AS Policy** in the **Operation** column.
2. On the displayed page, click **Add AS Policy**.

Configure parameters listed in **Table 1-14** for **as-policy-discuz01**. When the system detects that the CPU usage exceeds 70% for three consecutive times, **as-policy-discuz01** is triggered and an ECS is added to the AS group.

Table 1-14 Key parameters for creating AS policy **as-policy-discuz01**

Parameter	Description	Example Value
Policy Name	Specifies the name of the AS policy to be created.	as-policy-discuz01
Policy Type	Select **Alarm**.	Alarm
Alarm Rule	Specifies whether a new alarm rule is to be created (**Create**) or an existing alarm rule will be used (**Use existing**).	Create
Rule Name	Specifies the name of the alarm rule.	as-alarm-cpu-01
Monitoring Type	Specifies the type of monitoring metrics, which can be **System monitoring** or **Custom monitoring**. Select **System monitoring**.	System monitoring
Trigger Condition	Select monitoring metrics supported by AS and set alarm conditions for the metrics.	CPU Usage Max. >70%
Monitoring Interval	Specifies the interval at which the alarm status is updated based on the alarm rule.	5 minutes
Consecutive Occurrences	Specifies the number of sampling points when an alarm is triggered.	3

Parameter	Description	Example Value
Scaling Action	Specifies an action and the number or percentage of instances. The following scaling action options are available: • Add Adds instances to an AS group when the scaling action is performed. • Reduce Removes instances from an AS group when the scaling action is performed. • Set to Sets the expected number of instances in an AS group to a specified value.	Add 1 instance
Cooldown Period	To prevent an alarm-based policy from being triggered repeatedly by the same event, configure a cooldown period.	900

3. Click **OK**.
4. Click **Add AS Policy** again and create AS policy **as-policy-discuz02**. When the system detects that the CPU usage is lower than 30% for three consecutive times, **as-policy-discuz02** is triggered and an ECS is removed from the AS group.
5. Click **OK**.
6. Back to the AS policy list. The AS policies are successfully created if their statuses change to **Enabled**.

Step 4: Add the Server to the AS Group

Manually add ECS discuz02 to the AS group.

1. Click the name of the AS group **as-group-discuz** to switch to the page providing details about the AS group.
2. Click the **Instances** tab and manually add **discuz02** to the AS group.

Step 5: Modify the Minimum Number of Instances

Modify the minimum number of instances to ensure discuz02 is not removed from the AS group.

1. Click the name of the AS group **as-group-discuz** to switch to the page providing details about the AS group.

2. Click **Modify AS Group** in the upper right corner of the page. Set **Min. Instances** to **1**.
3. Click **OK**.

Step 6: Verify Traffic Routing

Check whether the forum website can be used. If the CPU usage of ECSs in the AS group remains higher than 70%, as shown on the **Monitoring** tab of the page providing details about the AS group, an ECS will be automatically added to the AS group (shown on the **Scaling Actions** tab). If the CPU usage remains lower than 30% and the AS group contains at least two ECSs, an ECS will be automatically removed from the AS group. If not, contact technical support to locate the fault.

Helpful Links

- To deploy new applications on cloud servers, use AS lifecycle hooks to add and remove cloud servers to and from an AS group and manage the lifecycle of these cloud servers in the AS group. For details, see **Managing Lifecycle Hooks**.
- To modify the specifications of servers in an AS group, create a new AS configuration first. For details, see **Creating an AS Configuration from a New Specifications Template**. Then, replace the AS configuration used by the AS group with the one you created. For details, see **Changing the AS Configuration for an AS Group**.

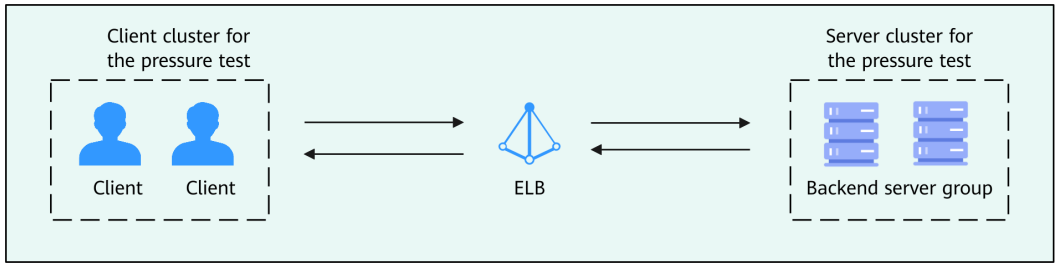
1.6 Load Balancer Pressure Test Solution

A load balancer can distribute high-volume service traffic. But its performance and stability directly impact service reliability. Before using a load balancer to distribute real service traffic, you can perform pressure tests on ELB and simulate extreme service scenarios to ensure that the load balancer is highly available under heavy load conditions.

Pressure Test Solution Architecture

Figure 1-17 shows the reference architecture of the pressure test solution.

Figure 1-17 Pressure test solution architecture



Pressure Test Solution

- **Metrics for pressure testing on TCP or TLS listeners**

You can refer to the three key metrics in **Table 1-15** when performing pressure tests on TCP or TLS listeners. For details about the monitoring metrics, see **ELB Monitoring Metrics**.

Table 1-15 Metrics for pressure testing on TCP or TLS listeners

Metric	Pressure Test Suggestions
New Connections	Use short connections to test how well the load balancer and backend servers can process new connections.
Concurrent Connections	Use persistent connections to test how well the load balancer and backend servers can process concurrent connections.
Bandwidth	Use large packets to test the load balancer's ability to handle high bandwidth. This ensures it can manage heavy traffic efficiently and reliably in actual scenarios.

- **Metrics for pressure testing on HTTP or HTTPS listeners**

  You can refer to the four key metrics in **Table 1-16** when performing pressure tests on HTTP or HTTPS listeners. For details about the monitoring metrics, see **ELB Monitoring Metrics**.

  Table 1-16 Metrics for pressure testing on HTTP or HTTPS listeners

Metric	Pressure Test Suggestions
New Connections	Use short connections to test how well the load balancer and backend servers can process new connections.
Concurrent Connections	Use persistent connections to test how well the load balancer and backend servers can process concurrent connections.
Bandwidth	Use large packets to test the load balancer's ability to handle high bandwidth. This ensures it can manage heavy traffic efficiently and reliably in actual scenarios.
Queries per second (QPS)	Configure a high request rate to test how well the load balancer and backend servers can process requests.

- Backend server group configuration
  - Cloud servers are recommended to be used as the backend servers for the pressure test. There are some restrictions on using IP as backend servers. For details, see .
  - You are not advised to use source IP hash as the load balancing algorithm or enable sticky session. If you do, the requests from a client will be distributed to a specific backend server, causing uneven loads across backend servers.

Pressure Test Tool Suggestions

You are advised to use CodeArts PerfTest to perform pressure tests. CodeArts PerfTest simulates high user traffic during peak times. It allows you to define the contents and time sequences of packets and supports complex combinations of multiple transactions. After tests are complete, CodeArts PerfTest provides professional test reports to evaluate your service quality.

Pressure Test Example

This practice uses HTTP listeners as an example. Before pressure testing an HTTP listener, you need to:

Create an HTTP backend server group, add two ECSs to the backend server group, and set the backend port to 80 for each ECS.

Create a dedicated load balancer that supports both network and application load balancing, add an HTTP listener, and configure the backend server group to which traffic will be directed.

Configure the two ECSs as described in **Table 1-17**.

Table 1-17 Basic configuration

vCPUs	4 vCPUs
Memory	8 GiB
OS	Huawei Cloud EulerOS 2.0 Standard 64-bit (10 GiB)

Procedure

1. Remotely log in to each ECS and install the HTTP service.
   a. Install Nginx:
      ```
      yum install -y nginx
      ```
   b. Initialize the default page:
      ```
      echo "performance test" > /usr/share/nginx/html/index.html
      ```
   c. Modify **/etc/nginx/nginx.conf** to listen on more ports based on the pressure test requirements as Nginx listens on TCP port 80 by default.
   d. Start the HTTP service:
      ```
      systemctl start nginx
      ```
   e. Run either of the following commands to check whether the HTTP service can be accessed (port 80 is used by default):
      ```
      curl -X GET http://localhost or curl -X GET http://127.0.0.1:80
      ```
2. Use CodeArts PerfTest to perform pressure tests by referring to .

Causes of Poor Pressure Test Results

- Poor CPU performance of the backend server

  Poor CPU performance of the backend server causes bad pressure test results.

  Solution: Check the CPU usage of all backend servers and upgrade the specifications of the backend servers.

- Poor performance of antecedent services

  The service running on the backend server may rely on other services (such as the database and DNS) to function properly. The poor performance of antecedent services can cause poor pressure test results.

  Solution: Check all the antecedent services and improve their performance.

- Insufficient ports

  Insufficient ports on the client or server may reduce the number of concurrent connections.

  Check the following possible causes and take the corresponding measures:

  – The TCP connection that is actively closed will enter the TIME_WAIT state. If there are a large number of connections in the TIME_WAIT state, ports will be exhausted.

    Suggestion:

    i. Use persistent connections instead of short connections to reduce the need for repeatedly opening and closing connections.

    ii. Set **sysctl -w net.ipv4.tcp_tw_reuse** to **1** to allow the connections in the TIME_WAIT state to be reused.

    iii. Set **sysctl -w net.ipv4.ip_local_port_range** to **"1024 65535"** on the client to increase available ports and prevent port exhaustion.

  – The numbers of clients and backend servers used for the pressure test are limited, causing insufficient ports.

    Suggestion: Increase the number of clients and backend servers.

2 Security

2.1 ELB Security Best Practices

1. **Use identity credentials properly to improve account security.**

   No matter whether you access ELB resources through the console or APIs, you are required to provide identity credentials for validity verification. In addition, login and login authentication policies are provided to enhance identity authentication security. With Identity and Access Management (IAM), ELB supports four identity authentication methods: username and password, access key, temporary access key, and access code. In addition, **Login Protection** and **Login Authentication Policy** are provided.

   a. Use a temporary AK/SK.

      When you use ELB APIs or SDKs to query resources like metrics and alarms, identity authentication is required to ensure the confidentiality, integrity, and correctness of requests. You are advised to configure an IAM agency to obtain a temporary access key, or directly configure temporary AK/SKs for your applications or cloud services. Temporary AK/SKs will expire after a short period, which reduces data leakage risks. For details, see **Temporary Access Keys** and **Obtaining Temporary Access Keys and Security Tokens of an Agency**.

   b. Periodically change permanent access keys.

   c. Regularly change your username and password and avoid weak passwords.

      Regularly resetting passwords is one important measure to enhance system and application security. This practice not only lowers the chances of password exposure but also helps you meet compliance requirements, mitigate internal risks, and boost your security awareness. Also, complex passwords are recommended to reduce risks. For details, see **Password Policy**.

2. **Assign different API management permissions to different users.**

   For details about ELB API permission management, see **ELB Permissions** and **Permissions and Supported Actions**.

3. **Disable private_key_echo through the API.**

Before using the HTTPS or TLS listeners of ELB, you need to upload your certificates to ELB. Users can call APIs to query the certificates and private keys. Improperly assigned certificate API permissions can expose private keys, which may leave attackers the chances to intercept and forge data. To avoid these risks, you can:

a. Disable **private_key_echo**.

   You can **call this API** to disable this option.

b. Restrict API permissions.

   The certificate owner disables **private_key_echo** through the API first.

   Then create a custom policy as below on the IAM console:

```
{
    "Version": "1.1",
    "Statement": [
        {
            "Effect": "Deny",
            "Action": [
                "elb:certificates:setPrivateKeyEcho"
            ]
        }
    ]
}
```

4. **Enable access control for listeners.**

   You can add IP addresses to a whitelist or blacklist to control access to a listener.

   A whitelist allows the specified IP addresses to access the listener.

   A blacklist denies access from specified IP addresses.

   For details, see **What Is Access Control?**

5. **Redirect HTTP requests to HTTPS listeners to improve service security.**

   You can use ELB to **redirect HTTP requests to an HTTPS listener** to improve your service security.

6. **Use HTTPS mutual authentication and custom TLS security policies for services requiring high security.**

   a. Mutual authentication

      In common HTTPS service scenarios, only the server certificate is required for authentication. For some mission-critical services, mutual authentication between the server and the client is required.

      In this case, you need to deploy both the server certificate and client certificate. For details, see **Configuring Mutual Authentication When Adding an HTTPS Listener**.

   b. Custom TLS security policies

      HTTPS encryption is commonly used for applications that require encrypted data transmission. ELB allows you to use common TLS security policies to secure data transmission.

      When you add an HTTPS listener, you can select the default security policies or **create a custom policy** to improve service security. A security policy is a combination of TLS protocols of different versions and supported cipher suites. For details, see **TLS Security Policy**.

7. **Enable CTS to record operations on ELB for auditing.**

Cloud Trace Service (CTS) is a log audit service for Huawei Cloud security. It allows you to collect, store, and query cloud resource operation records. You can use these records to perform security analysis, audit compliance, track resource changes, and locate faults.

After CTS is enabled, it can record ELB operations.

a. If you want to enable and configure CTS, refer to **CTS Getting Started**.

b. For details about ELB operations recorded by CTS, see **Key Operations Recorded by CTS**.

c. If you want to view traces, refer to **Viewing Traces**.

8. **Create alarm rules to monitor key metrics of ELB to avoid server overloading.**

   Cloud Eye can monitor resources, resource groups, and websites, and timely report alarms to help you keep track of your resource usages and service status on the cloud.

   With Cloud Eye, you can dynamically analyze potential risks by viewing the network traffic and error logs of ELB during selected period of time.

   For details about the monitoring metrics supported by ELB and how to create alarm rules, see **Monitoring ELB Resources**.

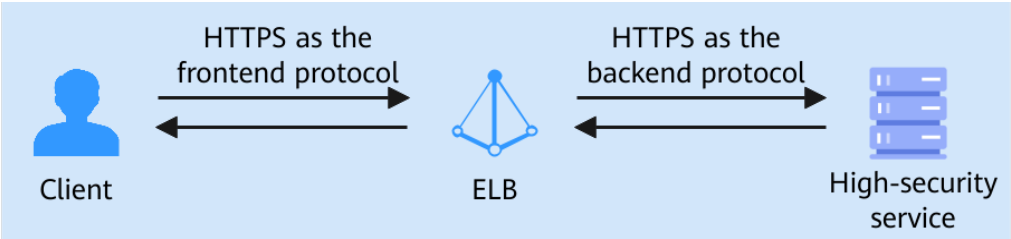
   For details about how to configure alarms for ELB resources, see **Using Cloud Eye to Monitor ELB Resources**.

2.2 Configuring HTTPS at the Frontend and Backend for Access Encryption

Scenarios

If your sensitive service data (such as finance and government service data) requires high secure data transmission on the cloud, and you want secure communications between the client and the load balancer, and between the load balancer and backend servers, you can use HTTPS as the frontend and backend protocols. This allows you to use ELB to route traffic securely from clients to backend servers, ensuring high performance and O&M efficiency.

Figure 2-1 HTTPS at both the frontend and backend



Prerequisites

- There is a dedicated load balancer with an EIP bound to it. If there is no such resource, you can **buy one** and **bind an IPv4 EIP to the load balancer**.

- You have either purchased a certificate or uploaded a third-party certificate to the SSL Certificate Manager (SCM) console, and configured a domain name for the certificate. It is recommended to **purchase an SSL certificate** on the Cloud Certificate & Manager (CCM) console.

  You have **uploaded the certificate to ELB**.
- You have purchased an ECS (ECS01) and deployed an application on it. For details about how to deploy a test service, see **Deploy the Application**.

Procedure

Figure 2-2 Procedure for configuring HTTPS at the frontend and backend

Step 1: Create an HTTPS Backend Server Group

1. Go to the **backend server group list page**.
2. Click **Create Backend Server Group** in the upper right corner.
3. Configure the parameters based on **Table 2-1**. Retain the default values for other parameters.

Table 2-1 Parameters required for configuring a routing policy

Parameter	Example Value	Description
Backend Server Group Name	server_group	Specifies the name of the backend server group.
Type	Dedicated	Specifies the type of load balancer that can use the backend server group.
Load Balancer	Associate existing	Specifies whether to associate a load balancer. Click **Associate existing** and select a load balancer you have created.
Backend Protocol	HTTPS	Specifies the protocol that backend servers in the backend server group use to receive requests from the listeners.

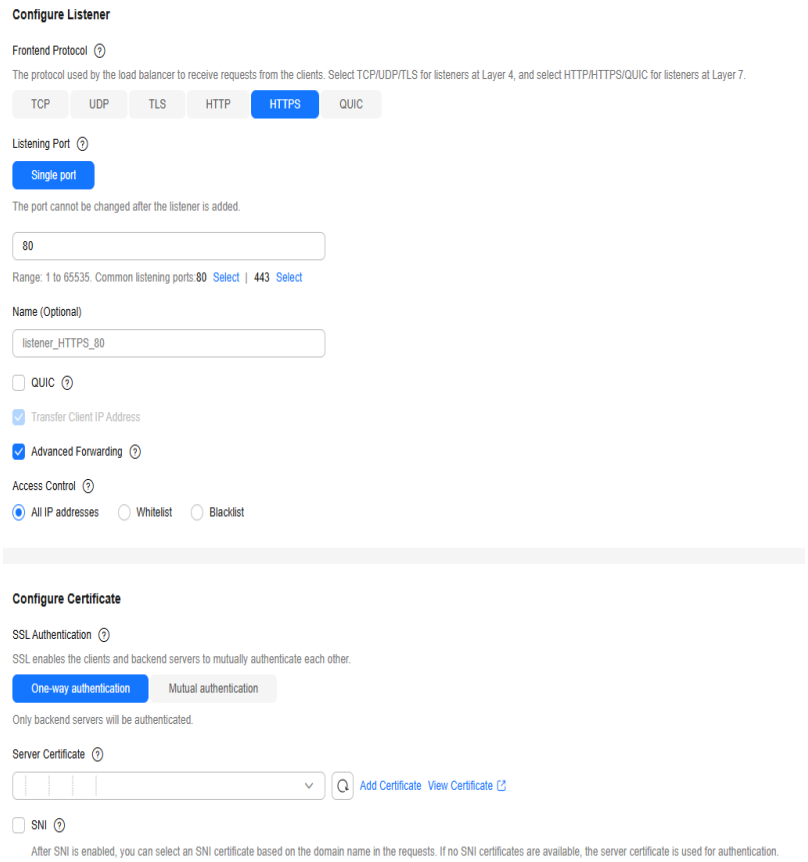
Parameter	Example Value	Description
Load Balancing Algorithm	Weighted round robin	Specifies the load balancing algorithm used by the load balancer to distribute traffic. **Weighted round robin**: Requests are routed to different servers based on their weights. Backend servers with higher weights receive proportionately more requests, whereas equal-weighted servers receive the same number of requests. For more information, see **Load Balancing Algorithms**.

4. Click **Next** to add backend servers and configure health check.
5. Click **Add Cloud Server**, select **ECS01**, set the service port to 443, and retain the default values for other parameters.
6. Enable health check and retain the default values for other health check parameters.
7. Click **Next**.
8. Confirm the configuration and click **Create Now**.

Step 2: Add an HTTPS Listener

1. Go to the **load balancer list page**.
2. Locate the target load balancer and click **Add Listener** in the **Operation** column.
3. On the **Add Listener** page, set **Frontend Protocol** to **HTTPS**.

Figure 2-3 Configuring one-way authentication



4. Click **Next: Configure Request Routing Policy** and select **Use existing** for **Backend Server Group**. Select an existing backend server group and click **Next: Confirm**.
5. Confirm the configurations and click **Submit**.

Step 3: Configure Domain Name Resolution

You can add an A record set to resolve the domain name to the public IP address of the load balancer so that clients can access the load balancer using the public domain name.

The following provides an example for resolving a website domain name to an IPv4 address. For details about how to configure an A record set, see **Routing Internet Traffic to a Website**.

1. Go to the **DNS console**.
2. In the navigation pane on the left, choose **Public Zones**.

   The zone list is displayed.
3. Locate the public zone and click **Manage Record Sets** in the **Operation** column.
4. Click **Add Record Set**.
5. Configure the parameters based on **Table 2-2**.

Table 2-2 Parameters for adding an A record set

Parameter	Example Value	Description
Type	A – Map domains to IPv4 addresses	Type of the record set. In this example, set it to **A - Map domains to IPv4 addresses**.
Name	www	Prefix of the domain name to be resolved.
Line	Default	Resolution line. The DNS server will return the IP address of the specified line, depending on where end users come from. The default value is **Default**. **Default**: returns the default resolution result irrespective of where the visitors come from.
TTL (s)	300	Cache duration of the record set on a local DNS server, in seconds. In this example, the default value 300 is used.
Value	192.168.12.2	IPv4 addresses mapped to the domain name. In this example, set this parameter to the EIPs bound to the load balancer.
Advanced Settings (Optional)	-	Click ⌄ to expand the advanced settings, set the alias and weight of the record set, and add a description and tags. In this example, the default settings are used.

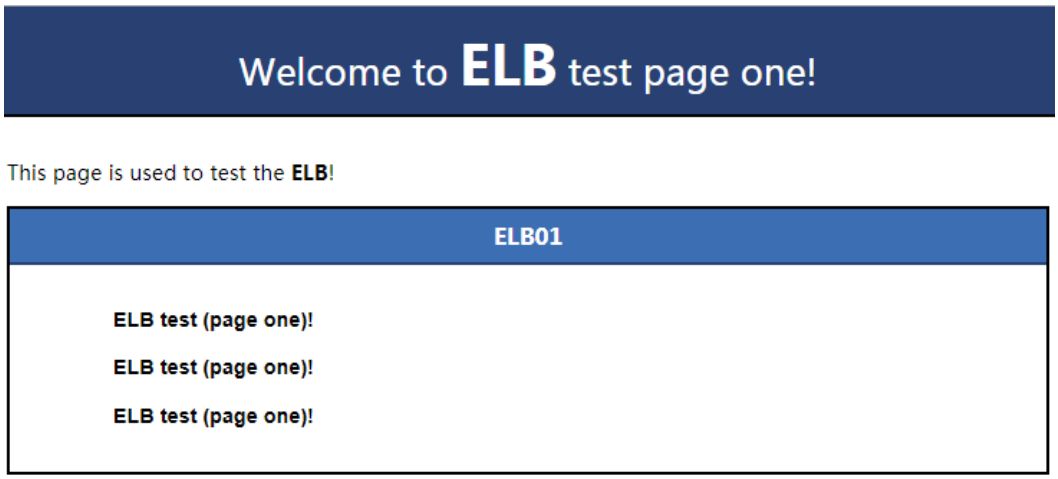
6. Click **OK**.
7. Switch back to the **Record Sets** tab.
   The added record set is in the **Normal** state.

Step 4: Verify Load Balancing

Deploy an application on ECS01, so that a page with message "Welcome to ELB test page one!" is returned when ECS01 is accessed. For details, see **Deploy the Application**.

Use a browser to access the domain name (**https://***load-balancer-domain-name*) of the load balancer. If the following page is displayed, the load balancer forwards the access request to ECS01, and HTTPS is successfully configured as both the frontend and backend protocols.

Figure 2-4 Accessing ECS01



2.3 Integrating WAF with ELB to Protect Your Websites

Scenarios

If your service servers are deployed on Huawei Cloud, you can purchase dedicated WAF instances to protect important domain names or websites that only use IP addresses to provide services.

In this way, ELB routes HTTP or HTTPS requests first to dedicated WAF instances for filtering out malicious traffic and the latter then directs normal traffic to backend servers.

This document describes how you can add dedicated WAF instances to the backend server group of your load balancer to protect your websites.

Constraints

- The security group rules configured for backend servers must allow traffic from the backend subnet where the load balancer resides to the backend servers over the backend port. For details, see **Configuring Security Group Rules for Backend Servers**.
- The security group rules configured for WAF instances must allow traffic over the specified port. For details, see **Adding a Security Group Rule**.

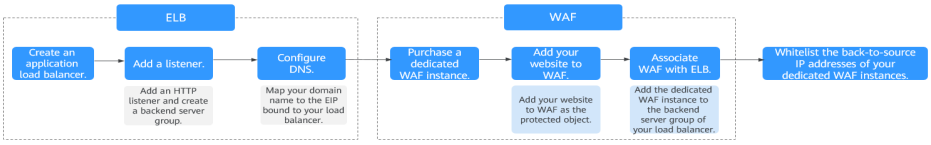
Traffic Path

After WAF is integrated with ELB, the traffic flow is as illustrated in **Figure 2-5**.

Figure 2-5 Traffic path

Procedure

Figure 2-6 Process for associating a dedicated WAF instance with an application load balancer

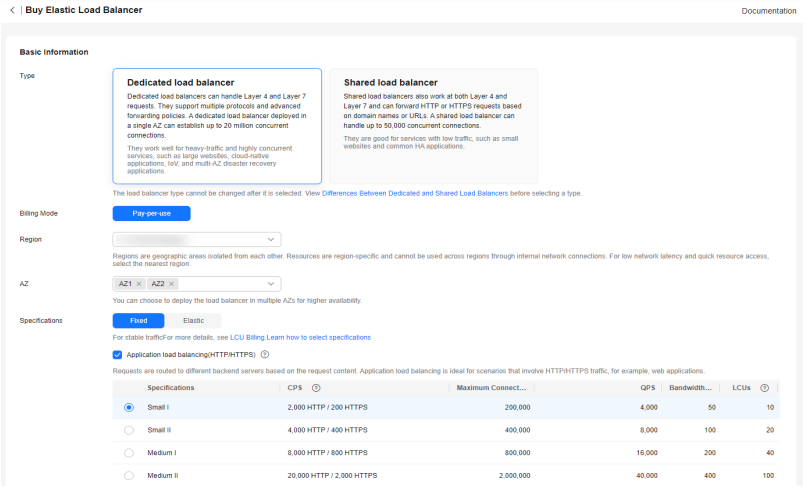


Step 1: Create an Application Load Balancer

1. Go to the **load balancer list page**.
2. On the **Load Balancers** page, click **Buy Elastic Load Balancer**. For details, see **Creating a Dedicated Load Balancer**.

   Complete the basic configuration of the load balancer as prompted. For example, select **Application load balancing (HTTP/HTTPS)** for **Specifications**.

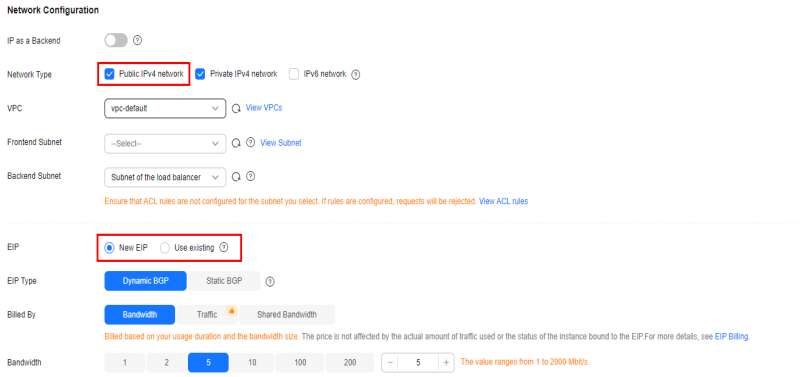
   Figure 2-7 Creating an application load balancer (Dedicated)



3. Configure the network as prompted.

   Choose **Public IPv4 network** for **Network Type** and select an existing EIP for or assign a new EIP to the load balancer to receive requests from public networks.

Figure 2-8 Selecting an EIP for the load balancer



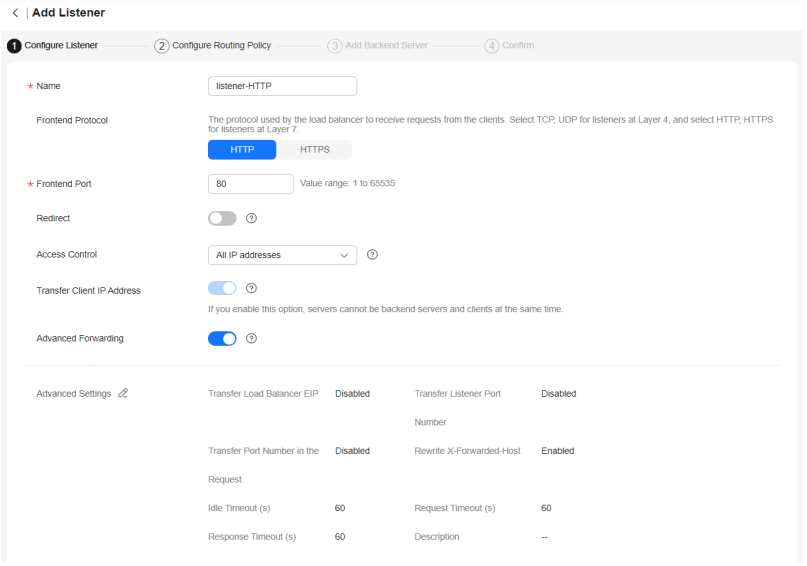
4. Confirm the information, click **Buy Now**.

Step 2: Add an HTTP Listener and Associate It with a Backend Server Group

1. Go to the **load balancer list page**.
2. On the **Load Balancers** page, locate the created load balancer and click its name.
3. Under **Listeners**, click **Add Listener**. Add an HTTP listener and specify a frontend port for it.

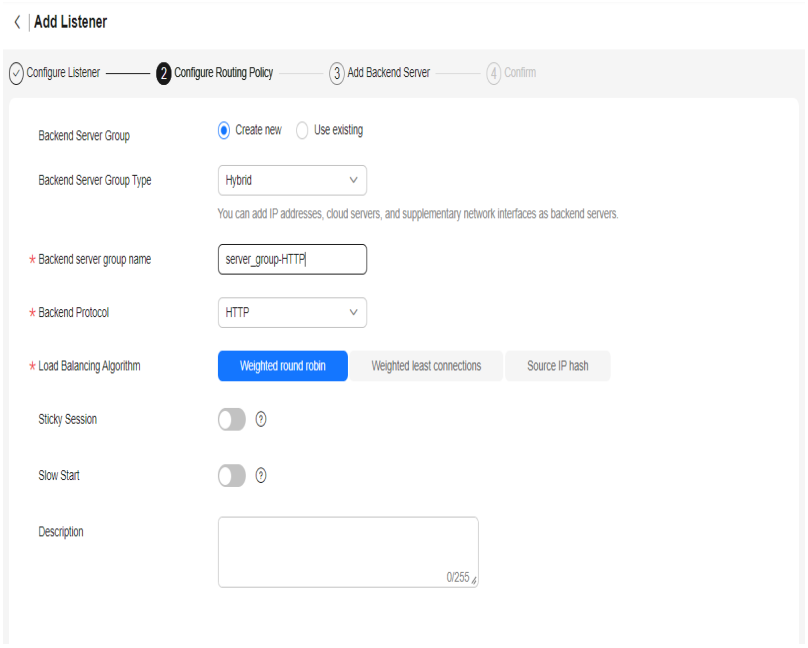
   For details, see **Adding an HTTP Listener**.

Figure 2-9 Adding an HTTP listener



4. Click **Next: Configure Request Routing Policy**. On the **Configure Routing Policy** page, select **Create new** for **Backend Server Group**.

Figure 2-10 Creating a backend server group



5. Click **Next: Add Backend Server**. Add backend servers and configure a health check for the backend server group.
6. Click **Next: Confirm**, confirm the settings, and click **Submit**.

Step 3: Translate Domain Names into the EIP of the Load Balancer

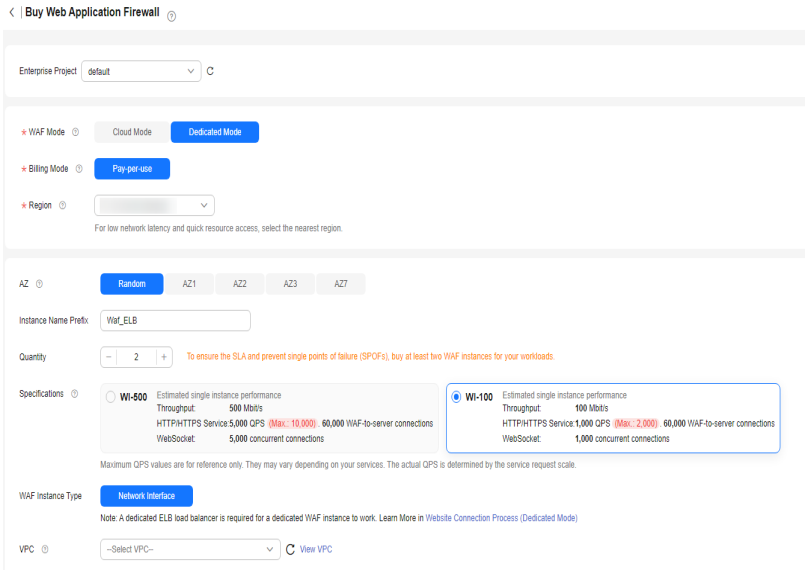
Use Huawei Cloud DNS to translate your domain name, such as www.example.com, into the EIP bound to your load balancer.

For details about how to configure DNS, see **Routing Internet Traffic to a Website**.

Step 4: Create a Dedicated WAF Instance

1. Log in to the management console.
2. In the upper left corner of the page, click  and select the desired region and project.
3. Click  in the upper left corner and choose **Web Application Firewall** under **Security & Compliance**.
4. In the upper right corner of the page, click **Buy WAF**. Configure the parameters as prompted. Select **Dedicated Mode** for **WAF Mode**.

Figure 2-11 Buying a dedicated WAF instance



NOTE

Dedicated WAF instances cannot be purchased. Only existing dedicated WAF instances can be used.

5. Confirm the settings and go through the subsequent steps to complete the purchase.

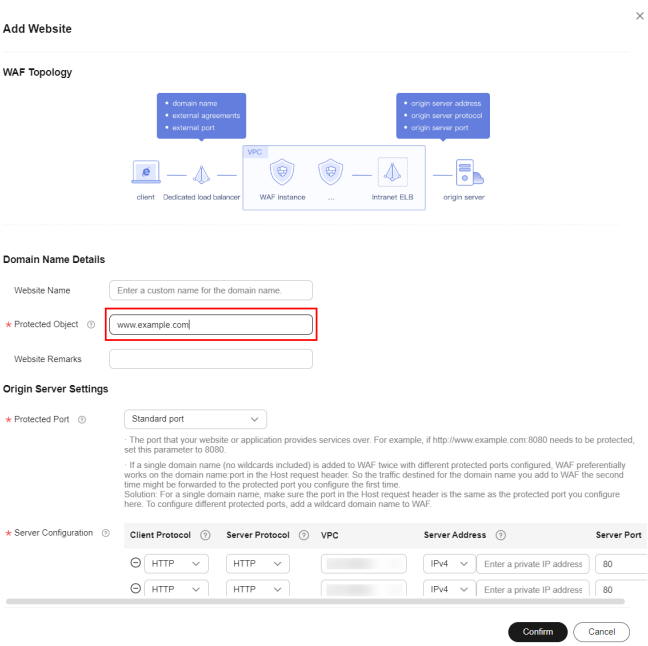
Step 5: Add Your Website to WAF

You can add your website (domain name: **www.example.com**) to WAF by following the below steps. For more details, see **Adding a Website to WAF (Dedicated Mode)**.

1. Log in to the management console.
2. In the upper left corner of the page, click and select the desired region and project.
3. Click in the upper left corner and choose **Web Application Firewall** under **Security & Compliance**.
4. In the navigation pane on the left, choose **Website Settings**.
5. In the upper left corner of the website list, click **Add Website**.

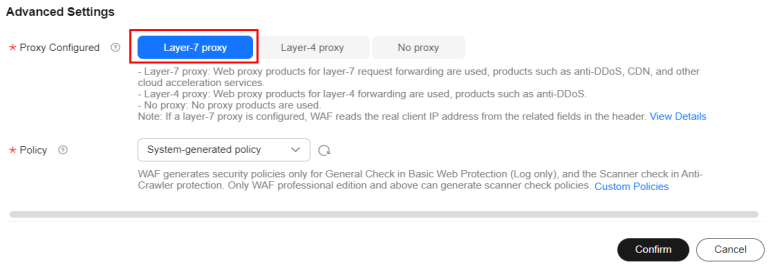
   In the displayed dialog box, select **Dedicated mode** and click **OK**.

Figure 2-12 Adding a domain name to WAF



6. Confirm the advanced settings. Set **Proxy Configured** to **Layer-7 proxy**.

Figure 2-13 Confirming advanced settings



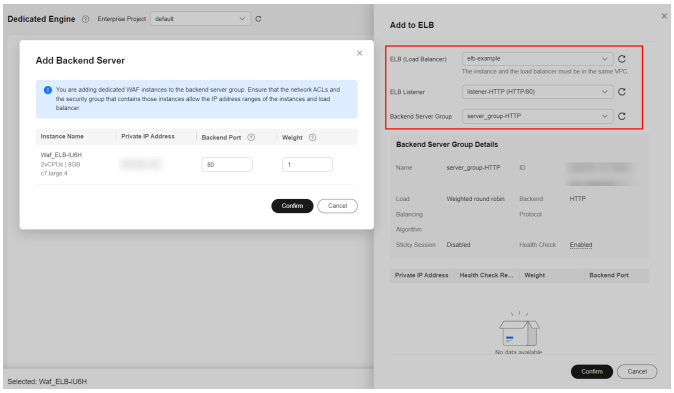
Step 6: Associate WAF with Your Load Balancer

You can add the dedicated WAF instance to the backend server group. Ensure that the network ACL and the security group that contains the WAF instance allow traffic from IP address ranges where the WAF instance and load balancer are deployed.

1. Log in to the management console.
2. In the upper left corner of the page, click and select the desired region and project.
3. Click in the upper left corner and choose **Web Application Firewall** under **Security & Compliance**.
4. In the navigation pane on the left, choose **Instance Management** > **Dedicated Engine** to go to the dedicated WAF instance page.
5. Locate the WAF instance created in **Step 4: Create a Dedicated WAF Instance** and choose **More** > **Add to ELB** in the **Operation** column.

6. In the **Add to ELB** dialog box, specify **ELB (Load Balancer)**, **ELB Listener**, and **Backend Server Group**.

   Figure 2-14 Adding the WAF instance to a load balancer



7. Click **Confirm**. Set **Backend Port** to the one configured for **Protected Port** in **Step 5: Add Your Website to WAF**.
8. Click **Confirm**.

Step 7: Whitelist Back-to-Source IP Addresses of Dedicated WAF Instances

In dedicated mode, website traffic is directed to the load balancer and then to dedicated WAF instances. The latter filters out malicious traffic and routes only normal traffic to the origin server.

In this way, the origin server only communicates with WAF back-to-source IP addresses. By doing so, WAF protects the origin server from being attacked. In dedicated mode, the WAF back-to-source IP addresses are the subnet IP addresses of the dedicated WAF instances.

The security software on the origin server may most likely regard WAF back-to-source IP addresses as malicious and block them. Once they are blocked, the origin server will deny all WAF requests. As a result, your website may become unavailable or respond very slowly. Therefore, ACL rules must be configured on the origin server to trust only the subnet IP addresses of your dedicated WAF instances.

For details, see **Pointing Traffic to a Load Balancer**.

2.4 Using ELB and CNAD Advanced to Enhance the Defense Against DDoS Attacks

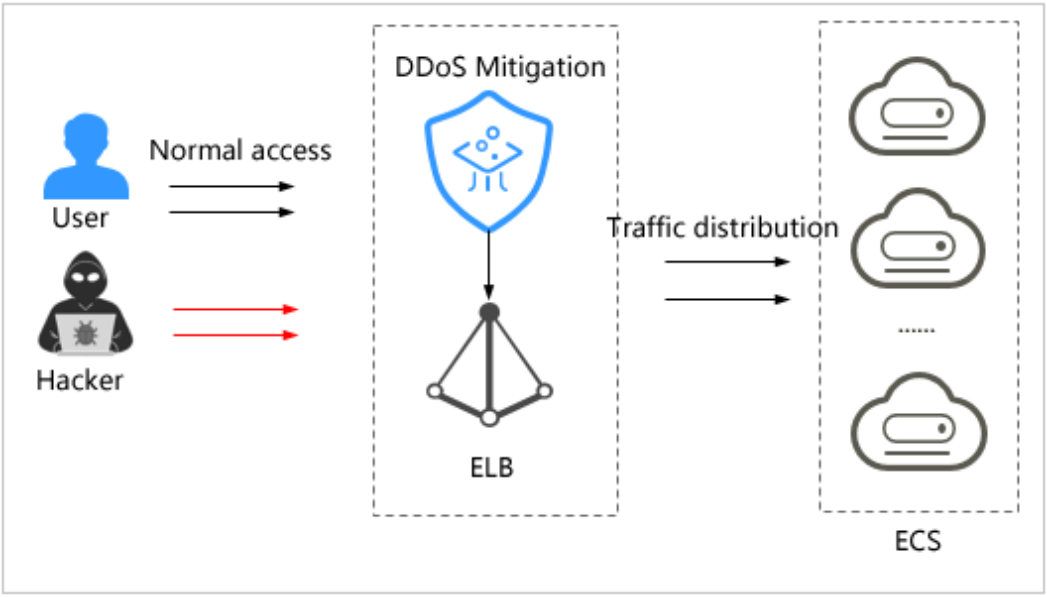
Application Scenarios

Cloud Native Anti-DDoS Advanced (CNAD Advanced) can improve the anti-DDoS capability of cloud services and ensure service security. You can deploy a load balancer and add its EIP to a CNAD instance to significantly enhance the defense against various types of DDoS attacks.

Solution Architecture

If your website is deployed on an ECS, you can deploy a load balancer on the origin server of the ECS, and add the EIP of the load balancer to a CNAD advanced instance to protect your website against DDoS attacks.

Figure 2-15 Using CNAD Advanced together with ELB



Advantages

Compared to enabling CNAD Advanced for ECSs, combining CNAD Advanced and Elastic Load Balance (ELB) allows for the discarding of traffic from unlistened protocols and ports. This enhances defense against various DDoS attacks (including reflection attacks like SSDP, NTP, and Memcached, as well as UDP flood and SYN flood attacks), significantly improving the DDoS protection capability of ECSs and ensuring the security and reliability of user services.

Resource and Cost Planning

Resource	Description	Quantity	Cost
Load balancer	Distributes access traffic across ECSs to eliminate single point of failures (SPOFs) caused by DDoS attacks.	1	For details, see **Billing Overview**.
CNAD advanced instance	Protects the EIP of the load balancer against DDoS attacks.	1	For details about CNAD Advanced billing modes and standards, see **Billing Overview**.

Procedure

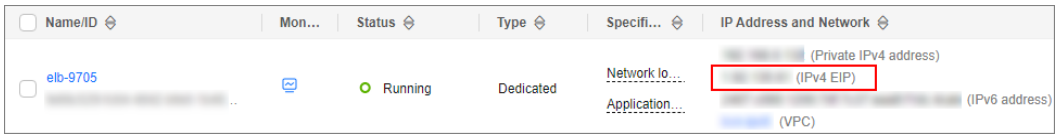
Step 1 Create a load balancer. For details, see **Creating a Load Balancer**.

Table 2-3 Parameter description

Parameter	Description
Region	Select the region where the ECS is located.
EIP	Select **Auto assign**.
EIP Type	Select **Dynamic BGP**.

Step 2 Obtain the public IP address of the created load balancer, as shown in **Figure 2-16**.

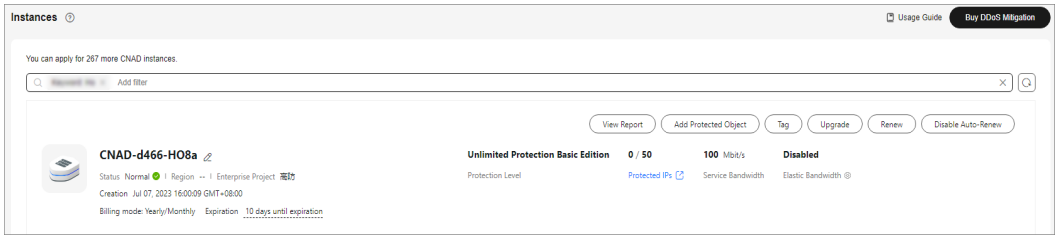
Figure 2-16 Public IP address of the ELB instance



Step 3 **Buy a CNAD Advanced instance** in the same region as the ECS.

Step 4 In the navigation pane on the left, choose **Cloud Native Anti-DDoS Advanced** > **Instances**. The **Instances** page is displayed.

Figure 2-17 Instance list



Step 5 In the upper right corner of the target instance box, click **Add Protected Object**.

Step 6 In the **Add Protected Object** dialog box that is displayed, select the elastic IP address of the load balancer obtained in **Step 2** and click **OK**.

After adding protected objects, you can configure protection policies for them. Cloud Native Anti-DDoS Advanced provides unlimited protection against DDoS attacks for ECSs. When a DDoS attack occurs, traffic scrubbing is automatically triggered.

For details about how to configure a protection policy, see **Adding a Protection Policy**.

----End

2.5 Using Cloud Eye to Monitor ELB Resources

Scenarios

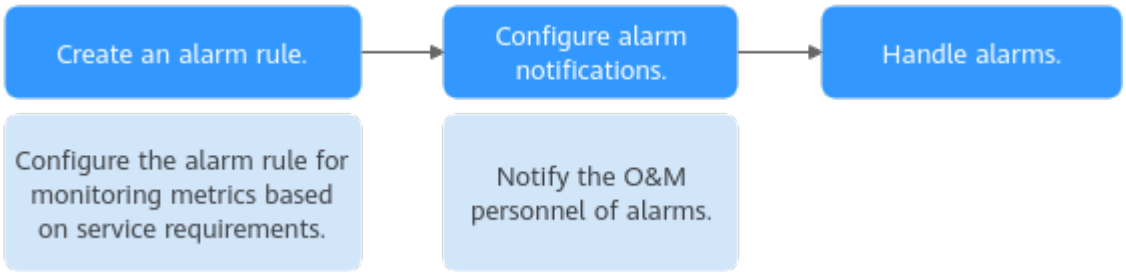
Cloud Eye is a multi-dimensional resource monitoring service. You can use Cloud Eye to monitor ELB resources in real time, set alarm rules, identify resource exceptions, and quickly respond to resource changes.

You can set alarm rules on the Cloud Eye console to send you notifications in case of exceptions.

This section describes how you can use monitoring metrics to monitor ELB resources in basic service scenarios. You can adjust the monitoring thresholds as required, but keep them below the recommended values.

Procedure

Figure 2-18 Process for setting alarm rules for monitoring ELB resources



Creating an Alarm Rule and Configuring Alarm Notifications

After an alarm rule is created, if a metric reaches the specified threshold or there is an event, Cloud Eye immediately informs you of the exception through SMN. The following describes how you can create an alarm rule and set notification rules.

1. Log in to the **Cloud Eye console**.
2. In the navigation pane on the left, choose **Alarm Management** > **Alarm Rules**.
3. Click **Create Alarm Rule** in the upper right corner.
4. On the **Create Alarm Rule** page, configure parameters as needed.
   a. Configure the basic information for the alarm rule.

Table 2-4 Parameters for configuring the basic information of the alarm rule

Parameter	Description	Example Value
Name	Name of the alarm rule. The system generates a name randomly, and you can change it as you want.	alarm-elb
Description	(Optional) Supplementary information about the alarm rule.	-

b. Select an object to be monitored and set alarm parameters.

Table 2-5 Alarm parameters

Parameter	Description	Example Value
Alarm Type	The alarm type that the alarm rule applies to.	Metric
Cloud Product	The cloud product you want to monitor. This parameter is only available if you select **Metric** for **Alarm Type**. For details about the metrics supported by ELB, see **ELB Monitoring Metrics**.	Elastic Load Balance - Elastic Load Balancers
Resource Level	The resource level of the alarm rule. This parameter is only available if you select **Metric** for **Alarm Type**. You can select **Cloud product** (recommended) or **Specific dimension**. For ELB, the dimension can be **Elastic Load Balancers**, **Elastic Load Balancers - Listeners**, **Elastic Load Balancers - Backend Server Groups**, or **Elastic Load Balancers - Availability Zones**.	Cloud product

Parameter	Description	Example Value
Monitoring Scope	The resource scope that the alarm rule will apply to. This parameter is only available if you select **Metric** for **Alarm Type**. You can select **All resources**, **Resource groups**, or **Specific resources**. **NOTE** • **All resources**: An alarm will be triggered if any resource of the current cloud product meets the alarm policy. To exclude resources that do not need to be monitored, click **Select Resources to Exclude**. • **Resource groups**: An alarm will be triggered if any resource in the resource group meets the alarm policy. To exclude resources that do not need to be monitored, click **Select Resources to Exclude**. • **Specific resources**: Click **Select Specific Resources** to select resources.	All resources
Method	• **Configure manually**: Specify the metric name and alarm policy. An alarm will be triggered when any policy is met. • **Associate template**: If the associated template is modified, the alarm policies in this alarm rule will be modified as well.	Configure manually
Alarm Policy	If a metric exceeds its threshold repeatedly within a specified period, you will be notified. For details about how to configure monitoring metrics, see **Recommended Monitoring Metrics for Basic Service Scenarios (Dedicated Load Balancers)** or **Recommended Monitoring Metrics for Basic Service Scenarios (Shared Load Balancers)**. **NOTE** A maximum of 50 alarm policies can be added to an alarm rule. If any of these alarm policies is met, an alarm will be triggered.	-
Alarm Severity	Alarm severity, which can be **Critical**, **Major**, **Minor**, or **Warning**.	-

c. Configure parameters for alarm notifications.

Figure 2-19 Configuring an alarm notification

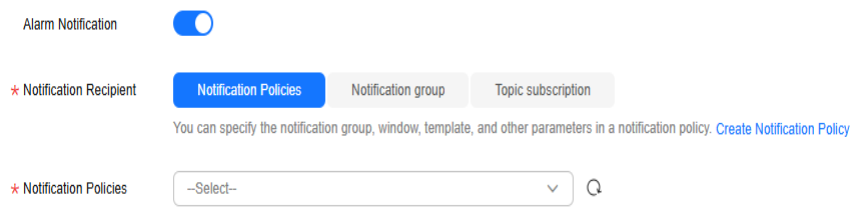


Table 2-6 Parameters for configuring an alarm notification

Parameter	Description
Alarm Notifications	Whether to send notifications to users via SMS, email, voice notification, HTTP, HTTPS, FunctionGraph (function), FunctionGraph (workflow), WeCom chatbot, DingTalk chatbot, Lark chatbot, or WeLink chatbot.
Notified By	The following options are available: • **Notification groups**: Configure notification templates on Cloud Eye. • **Topic subscriptions**: Configure notification templates on SMN.
Notification Policies	This parameter is only available if you select **Notification policies** for **Notified By**. Select one or more notification policies. You can specify the notification group, window, template, and other parameters in a notification policy.
Notification Group	This parameter is only available if you select **Notification groups** for **Notified By**. Select the notification groups to which alarm notifications will be sent.
Recipient	This parameter is only available if you select **Topic subscriptions** for **Notified By**. You can select the account contact or a topic name as the object to which alarm notifications will be sent. • **Account contact** is the mobile number and email address of the registered account. • Topic is used to publish messages and subscribe to notifications. If the required topic is unavailable, create one first and add subscriptions to it. For details, see **Creating a Topic** and **Adding Subscriptions**.
Notification Template	This parameter is only available if you select **Notification groups** or **Topic subscriptions** for **Notified By**. You can select an existing template or create a new one.

Parameter	Description
Notification Window	This parameter is only available if you select **Notification groups** or **Topic subscriptions** for **Notified By**. Cloud Eye sends notifications only within the notification window you specified. If **Notification Window** is set to **08:00-20:00**, Cloud Eye sends notifications only within this window.
Trigger Condition	This parameter is only available if you select **Notification groups** or **Topic subscriptions** for **Notified By**. You can select either **Generated alarm** or **Cleared alarm**, or both.

d. Configure **Enterprise Project** and **Tags**.

Table 2-7 Parameter descriptions

Parameter	Description
Enterprise Project	Enterprise project to which the alarm rule belongs. Only users with the enterprise project permissions can manage the alarm rule.
Tags	Key-value pairs that you can use to easily categorize and search for cloud resources. • A key can contain up to 128 characters, and a value can contain up to 225 characters. • You can add up to 20 tags.

e. Click **Create**.

After the alarm rule is created, if a metric reaches the specified threshold or there is an event, Cloud Eye immediately informs you of the exception through SMN.

Recommended Monitoring Metrics for Basic Service Scenarios (Dedicated Load Balancers)

You can use the following metrics to monitor the performance, Layer 7 service operations, and the health status of backend servers of dedicated load balancers.

For details about the metrics supported by dedicated load balancers, see **ELB Monitoring Metrics**.

Performance Monitoring

You can use the metrics in the below table to quickly identify whether the service traffic exceeds the threshold.

You can **modify the load balancer specifications and add additional AZs** to handle the alarms.

Table 2-8 Recommended performance monitoring metrics

Metric			Alarm Policy				
ID	Metric Name	Monitored Object	Statistic	Consecutive Triggering Times	Operator	Threshold	Frequency
l4_ncps_usage	Layer 4 New Connection Usage	• Load balancer • AZ	Raw data	3	>	Critical: 80%	Every 1 hour
l4_con_usage	Layer 4 Concurrent Connection Usage	• Load balancer • AZ	Raw data	3	>	Critical: 80%	Every 1 hour
l4_in_bps_usage	Layer 4 Inbound Bandwidth Usage	• Load balancer • AZ	Raw data	3	>	Major: 80%	Every 1 hour
l4_out_bps_usage	Layer 4 Outbound Bandwidth Usage	• Load balancer • AZ	Raw data	3	>	Major: 80%	Every 1 hour
l7_ncps_usage	Layer 7 New Connection Usage	• Load balancer • AZ	Raw data	3	>	Critical: 80%	Every 1 hour
l7_con_usage	Layer 7 Concurrent Connection Usage	• Load balancer • AZ	Raw data	3	>	Critical: 80%	Every 1 hour
l7_qps_usage	Layer 7 QPS Usage	• Load balancer • AZ	Raw data	3	>	Critical: 80%	Every 1 hour
l7_in_bps_usage	Layer 7 Inbound Bandwidth Usage	• Load balancer • AZ	Raw data	3	>	Major: 80%	Every 1 hour

Metric			Alarm Policy				
l7_out_bps_usage	Layer 7 Outbound Bandwidth Usage	• Load balancer • AZ	Raw data	3	>	Major: 80%	Every 1 hour
dropped_connections	Dropped Connections	Load balancer	Raw data	1	>	Critical: 0	Every 1 hour
dropped_packets	Dropped Packets	Load balancer	Raw data	1	>	Critical: 0	Every 1 hour
dropped_traffic	Bandwidth for Dropping Packets	Load balancer	Raw data	1	>	Critical: 0	Every 1 hour

Layer 7 Service Monitoring Metrics

You can use the Layer 7 status codes to quickly identify whether service requests are correctly processed.

You can view the **access logs** to check service status and handle alarms.

Table 2-9 Recommended Layer 7 service monitoring metrics

Metric			Alarm Policy				
ID	Metric Name	Monitored Object	Statistic	Consecutive Triggering Times	Operator	Threshold	Frequency
mb_l7_qps	Layer 7 Query Rate	Listener	Raw data	1	Increase or decrease compared with last period	Major: 20%	Every 1 hour

Metric			Alarm Policy				
mc_l7_http_2xx	2xx Status Codes (Total)	Listener	Raw data	1	Increase or decrease compared with last period	Major: 20%	Every 1 hour
me_l7_http_4xx	4xx Status Codes (Total)	Listener	Raw data	1	Increase or decrease compared with last period	Major: 20%	Every 1 hour
mf_l7_http_5xx	5xx Status Codes (Total)	Listener	Raw data	1	Increase or decrease compared with last period	Major: 20%	Every 1 hour
m14_l7_rt	Average Layer 7 Response Time	Listener	Raw data	1	Increase or decrease compared with last period	Major: 20%	Every 1 hour

Metric			Alarm Policy				
m15_l7_upstream_4xx	4xx Status Codes (Backend Servers)	Listener	Raw data	1	Increase or decrease compared with last period	Major: 20%	Every 1 hour
m16_l7_upstream_5xx	5xx Status Codes (Backend Servers)	Listener	Raw data	1	Increase or decrease compared with last period	Major: 20%	Every 1 hour

Backend Server Health

You can view the **Unhealthy Servers** metric to learn about the health status of backend servers in a timely manner.

You can **troubleshoot an unhealthy backend server** and handle the alarms.

Table 2-10 Recommended metrics for monitoring backend server health status

Metric			Alarm Policy				
ID	Metric Name	Monitored Object	Statistic	Consecutive Triggering Times	Operator	Threshold	Frequency

Metric			Alarm Policy				
m9_abnormal_servers	Unhealthy Servers	• Load balancer • Listener • Backend server group	Raw data	1	>	Critical: 0	Every 1 hour

Recommended Monitoring Metrics for Basic Service Scenarios (Shared Load Balancers)

You can use the following metrics to monitor the performance, Layer 7 service operations, and the health status of backend servers of shared load balancers.

For details about the metrics supported by shared load balancers, see **ELB Monitoring Metrics**.

Performance Monitoring

You can use the metrics in the below table to quickly identify whether the service traffic exceeds the threshold.

Table 2-11 Recommended performance monitoring metrics

Metric			Alarm Policy				
ID	Metric Name	Monitored Object	Statistic	Consecutive Triggering Times	Operator	Threshold	Frequency
m1_cps	Concurrent Connections	Load balancer	Raw data	3	>	Critical: 40000	Every 1 hour
m4_ncps	New Connections	Load balancer	Raw data	3	>	Critical: 4000	Every 1 hour

Layer 7 Service Monitoring Metrics

You can use the Layer 7 status codes to quickly identify whether service requests are correctly processed.

You can view the **access logs** to check service status and handle alarms.

Table 2-12 Recommended Layer 7 service monitoring metrics

Metric			Alarm Policy				
ID	Metric Name	Monitored Object	Statistic	Consecutive Triggering Times	Operator	Threshold	Frequency
mb_l7_qps	Layer 7 Query Rate	Listener	Raw data	1	Increase or decrease compared with last period	Major: 20%	Every 1 hour
m14_l7_rt	Average Layer 7 Response Time	Listener	Raw data	1	Increase or decrease compared with last period	Major: 20%	Every 1 hour
mc_l7_http_2xx	2xx Status Codes (Total)	Listener	Raw data	1	Increase or decrease compared with last period	Major: 20%	Every 1 hour

Metric			Alarm Policy				
me_l7_http_4xx	4xx Status Codes (Total)	Listener	Raw data	1	Increase or decrease compared with last period	Major: 20%	Every 1 hour
mf_l7_http_5xx	5xx Status Codes (Total)	Listener	Raw data	1	Increase or decrease compared with last period	Major: 20%	Every 1 hour
m15_l7_upstream_4xx	4xx Status Codes (Backend Servers)	Listener	Raw data	1	Increase or decrease compared with last period	Major: 20%	Every 1 hour
m16_l7_upstream_5xx	5xx Status Codes (Backend Servers)	Listener	Raw data	1	Increase or decrease compared with last period	Major: 20%	Every 1 hour

Backend Server Health

You can view the **Unhealthy Servers** metric to learn about the health status of backend servers in a timely manner.

You can **troubleshoot an unhealthy backend server** and handle the alarms.

Table 2-13 Recommended metrics for monitoring backend server health status

Metric			Alarm Policy				
ID	Metric Name	Monitored Object	Statistic	Consecutive Triggering Times	Operator	Threshold	Frequency
m9_abnormal_servers	Unhealthy Servers	Load balancer	Raw data	1	>	Critical: 0	Every 1 hour

Reference

- **Viewing Monitoring Metrics**
- **ELB Event Monitoring**

2.6 Querying ELB Operation Records on the CTS Console

Scenarios

Cloud Trace Service (CTS) records operations performed on cloud service resources. A record contains information such as the user who performed the operation, IP address, operation content, and returned response message. These operation records let you do security audit, trace issues, and locate resources quickly. They also help you plan and use resources, and identify high-risk or non-compliant operations.

After CTS is enabled, it can record ELB operations. This section describes the fields in CTS records.

What Is a Trace?

A trace is an operation log for a cloud service resource, tracked and stored by CTS. Traces record operations such as adding, modifying, or deleting cloud service resources. You can view them to identify who performed operations and when for detailed tracking.

What Is a Management Tracker and Data Tracker?

A management tracker identifies and associates with all your cloud services, recording all user operations. It records management traces, which are operations

performed by users on cloud service resources, such as their creation, modification, and deletion.

A data tracker records details of user operations on data in OBS buckets. It records data traces reported by OBS, detailing user operations on data in OBS buckets, including uploads and downloads.

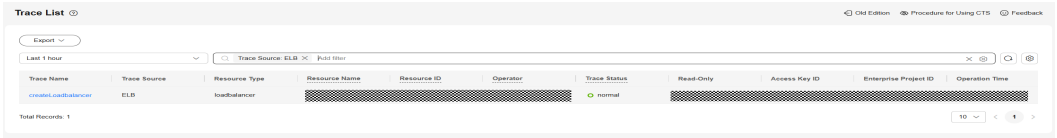
Viewing the Trace for Creating a Load Balancer in the Trace List of the New Edition

1. **Create a dedicated load balancer**.
2. Log in to the **CTS console**.
3. In the navigation pane, choose **Trace List**.
4. In the time range drop-down list above the trace list, select a desired query time range: **Last 1 hour**, **Last 1 day**, or **Last 1 week**. You can also select **Custom** to specify a time range within the last seven days.
5. The search box above the trace list supports advanced queries. Combine one or more filters to refine your search. In this practice, select **Cloud service: ELB**.

Table 2-14 Trace filtering parameters

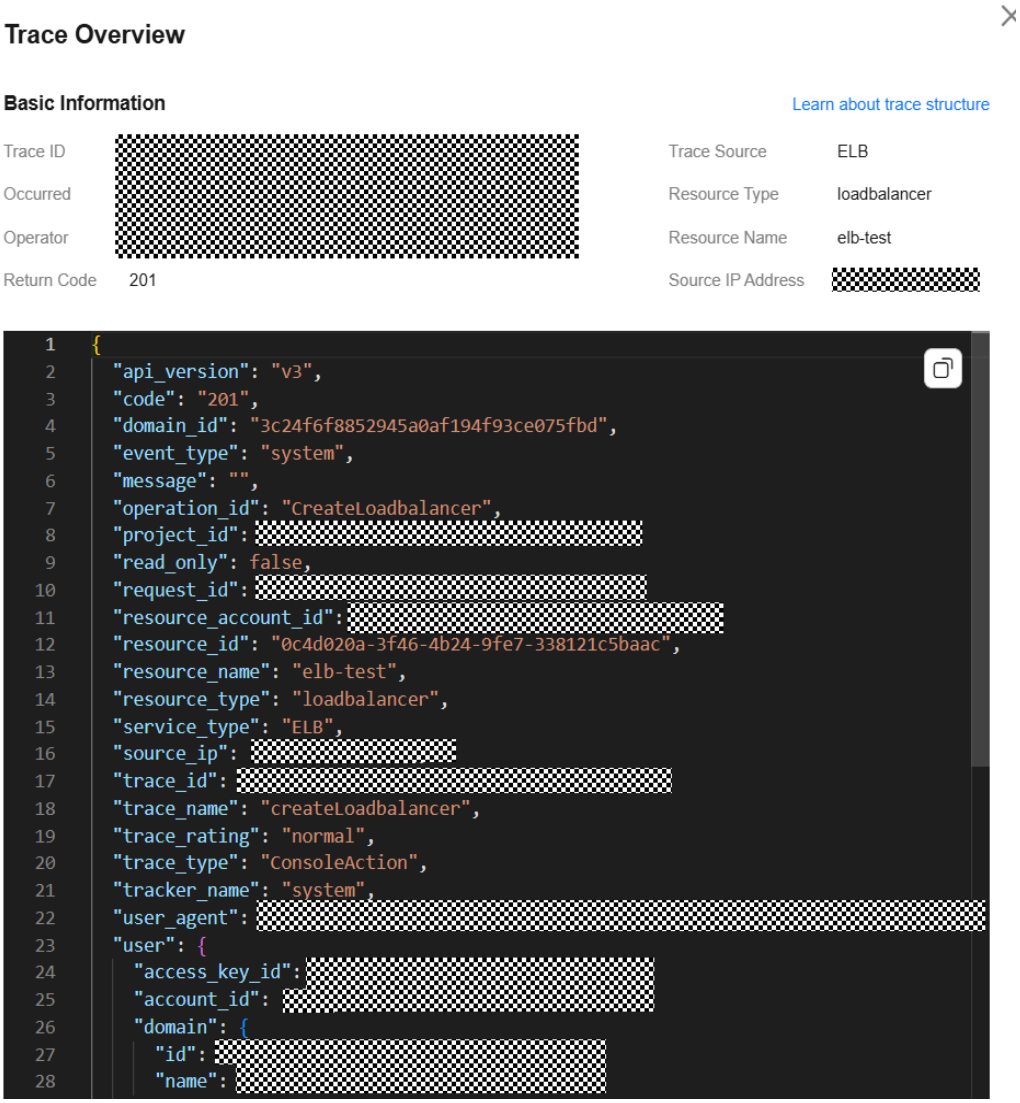
Parameter	Description
Cloud service	Abbreviation of the cloud service name. In this practice, select **Cloud service: ELB**. The entered value is case-sensitive and requires an exact match. Fuzzy matching is not supported.

Figure 2-20 Filtering ELB traces



6. On the trace list page, you can view the traces of the load balancer created in step **1**.
7. Click the trace name to view its details.

Figure 2-21 ELB trace details



8. On the **Trace List** page, export operation records, refresh the trace list, and specify columns to display.
   a. Enter any keyword in the search box and press **Enter** to filter desired traces.
   b. Click **Export** to export all traces in the query result as an .xlsx file. The file can contain up to 5,000 records.
   c. Click to view the latest information about traces.
   d. Click to define the information to be displayed in the trace list. If **Auto wrapping** is enabled ( ), excess text will be moved down to the next line; otherwise, the text will be truncated. By default, this function is disabled.
9. (Optional) On the **Trace List** page of the new edition, click **Old Edition** in the upper right corner to switch to the **Trace List** page of the old edition.

CTS Record Fields

Table 2-15 CTS record fields

Field	Description
request	Request body content. In this practice, this field indicates the request body for calling the API to create a load balancer.
trace_id	ID of the trace recorded by CTS.
code	HTTP status code returned by the associated API.
trace_name	Trace name.
resource_type	Type of the resource. In this practice, the resource type is **loadbalancer**.
trace_rating	Trace severity. If the load balancer is created successfully, **trace_rating** is **normal**. Value options: • **normal**: The operation succeeded. • **warning**: The operation failed. • **incident**: The operation caused a serious consequence, for example, a node failure or service interruption.
message	Remarks added by ELB to a trace.
source_ip	Source IP address of the request.
domain_id	Domain ID.
trace_type	Trace source. The value can be **ApiCall**, **ConsoleAction**, or **SystemAction**.
service_type	Type of the cloud service whose traces are to be queried. **service_type** of ELB is **ELB**.
event_type	Event type.
project_id	Project ID.
read_only	Whether the request is read-only. **read_only** is **false** for adding, deleting, or modifying resources, and **true** for querying resources.
resource_id	ID of the resource whose traces are to be queried.
tracker_name	Name of the tracker. • If **trace_type** is **system**, this field defaults to **system**. • If **trace_type** is **data**, this field indicates the name of the data tracker.
operation_id	Operation ID of the trace.
resource_account_id	ID of the account to which the resource belongs.

Field	Description
time	Time when the trace was generated.
resource_name	Resource name.
user	Information about the user who performs operations on the resource.
record_time	Time when a trace was recorded by CTS.
user_agent	ID of the client agent that sends the request.
api_version	API version.
response	Response body.
request_id	Request ID of the current request, which can be used to locate faults for ELB.

Helpful Links

- Key Operations Supported by CTS
- Viewing CTS Traces in the Trace List

2.7 Configuring Client Retry to Improve Service Availability

ELB High Availability

To improve the high availability of ELB, you can purchase load balancers in multiple AZs and enable the health check function for backend servers.

- **System HA deployment**: Load balancers can be deployed in multiple AZs in active-active mode. If the load balancer in an AZ goes down, the load balancer in another AZ will take over to distribute traffic. In addition, session persistence within an AZ is implemented, eliminating the impact of single points of failure (SPOFs) of servers in a single AZ of the ELB cluster and ensuring system stability.
- **Health check**: ELB checks the health of backend servers based on how you configure the health check. This ensures that new requests can be forwarded to healthy backend servers.

Client Retry Application Scenarios

Generally, ELB's high availability system can handle disaster recovery for essential services. However, in extreme scenarios, issues like connection resets or timeouts can disrupt services and compromise user experience. To address these extreme issues, you can configure the client retry logic to start new connections when there are connection resets or timeouts, to improve the fault tolerance and stability of the system.

The client retry logic is recommended in the following scenarios:

1. **Backend server health check failures**: If the health check on a backend server fails, traffic can still be routed over the existing Layer 4 connections to the unhealthy backend server, within the sticky session or deregistration delay duration.
2. **Cross-AZ switchover for high availability**: In extreme scenarios, if the AZ where the ELB cluster is located is faulty, the load balancer deployed in another AZ will divert the traffic from the faulty AZ to the healthy AZ. In this case, the persistent connection that is transmitting data in the faulty AZ cannot be restored, so the client needs to start a connection again.

Importance of Retry

Both the client and server may encounter temporary faults, such as transient network or disk jitter, service unavailability, or invoking timeout, due to infrastructure or running environment reasons. As a result, service access may time out.

You can design automated client retry systems to reduce the impacts of such faults on services and ensure successful execution.

Backend Service Unavailable Scenarios

Table 2-16 Recommended retry scenarios

Scenario	Description
Backend server unavailable	The health check on a backend server (ECS or container) fails due to faults, such as service process suspension, service process faults, hardware faults, virtualization migration failures, or network disruptions.
Complex network environment	Due to the complex network environment among the clients, load balancers, and backend servers, network jitter, packet loss, and data retransmission may occur occasionally. In this case, client requests may temporarily fail.
Complex hardware issues	Client requests may temporarily fail due to occasional hardware faults, such as VM HA and disk latency jitter.

Recommended Client Retry Rules

Table 2-17 Client retry rules

Retry Rule	Description
Remember the conditions that trigger retries.	Abnormal scenarios such as connection timeouts and resets.

Retry Rule	Description
Retry only idempotent operations.	It is recommended that client only retry idempotent operations. A retried operation may be repeatedly executed. Therefore, not all operations are suitable to be retried.
Configure proper retry times and interval.	Configure the retry times and interval based on service requirements in actual scenarios to prevent the following problems: • If the number of retries is insufficient or the interval is too long, the application may fail to complete operations. • If the number of retries is too large or the interval is too short, the application may overload the system with excessive resource usage and the server may be blocked due to too many requests. Common retry interval policies include immediate retry, fixed-interval retry, exponential backoff retry, and random backoff retry.
Avoid retry nesting.	Retry nesting may cause the retry interval to be exponentially amplified.
Record retry exceptions and print failure reports.	During retry, you can print retry error logs at the WARN level.
Reuse the retry system of a mature open-source ecosystem library.	Mature open-source middleware software has a rich client library. Based on the keepalive and detection mechanism of the connection pool, set a proper retry interval, retry times, and backoff policy. For details about how to design a retry system, see the keepalive and detection mechanism of the open-source ecosystem connection pool.

Helpful Links

- **Configuring Redis Client Retry**
- **phpredis Retry**

3 Basic Functions

3.1 Using ELB to Redirect HTTP Requests to an HTTPS Listener for Higher Service Security

Scenarios

HTTPS is an extension of HTTP. HTTPS encrypts data between a web server and a browser. You can use ELB to redirect HTTP requests to an HTTPS listener to improve your service security.

CAUTION

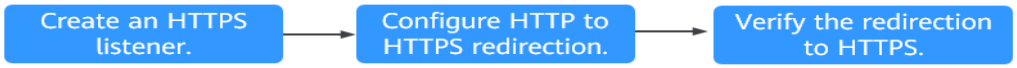
- If the listener protocol is HTTP, only the GET or HEAD method can be used for redirection. If you create a redirect for an HTTP listener, the client browser will change POST or other methods to GET. If you want to use other methods rather than GET and HEAD, add an HTTPS listener.
- HTTP requests are forwarded to the HTTPS listener as HTTPS requests, which are then routed to backend servers over HTTP.
- If HTTP requests are redirected to an HTTPS listener, no certificate can be deployed on the backend servers associated with the HTTPS listener. If certificates are deployed, HTTPS requests will not take effect.

Prerequisites

- You have created a dedicated load balancer. For details, see **Creating a Dedicated Load Balancer**.
- You have created two ECSs (ECS_client and ECS_server) that are running in the same VPC as the dedicated load balancer. ECS_client sends HTTPS requests, while ECS_server processes requests. For details, see **Purchasing an ECS**.
- You have gotten a server certificate ready for adding an HTTPS listener. For details, see **Adding a Server Certificate**.

Procedure

Figure 3-1 Procedure for redirecting HTTP requests to an HTTPS listener



Step 1: Add an HTTPS Listener

1. Go to the **load balancer list page**.
2. On the displayed page, locate the target load balancer and click its name.
3. On the **Listeners** tab, click **Add Listener**. Configure the parameters based on **Table 3-1**.

Figure 3-2 Adding an HTTPS listener

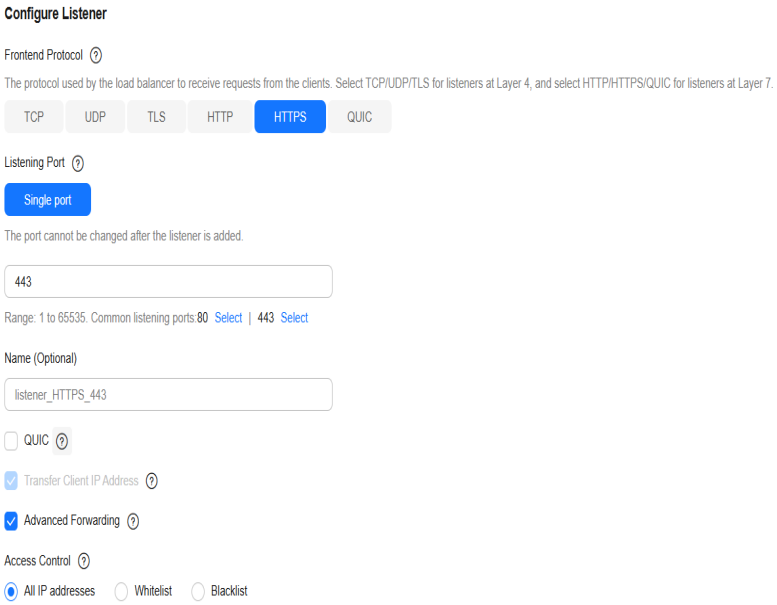


Table 3-1 Parameters for configuring an HTTPS listener

Parameter	Example Value	Description
Frontend Protocol	HTTPS	Specifies the protocol that will be used by the load balancer to receive requests from clients.
Listening Port	443	Specifies the port that will be used by the load balancer to receive requests from clients.
Name (Optional)	listener-HTTPS	Specifies the listener name.

Parameter	Example Value	Description
Transfer Client IP Address	Enabled by default	Specifies whether to transmit IP addresses of the clients to backend servers.
Advanced Forwarding	Enabled	Specifies whether to enable advanced forwarding. You can configure advanced forwarding policies to forward requests to different backend server groups based on a wide range of forwarding rules and actions.
Access Control	All IP addresses	Specifies how access to the listener is controlled. Access from specific IP addresses can be controlled using a whitelist or blacklist.
SSL Authentication	One-way authentication	Specifies how you want the clients and backend servers to be authenticated. In this practice, **One-way authentication** is selected.
Server Certificate	The existing server certificate	Specifies the certificate that will be used by the backend server for SSL handshake negotiation to authenticate clients and ensure encrypted transmission.
SNI	Not enabled	Specifies whether to enable SNI when HTTPS is used as the frontend protocol. SNI can be used when a server uses multiple domain names and certificates.

4. Retain the default values for parameters under **More (Optional)** and click **Next: Configure Request Routing Policy**.
5. Select **Create new** for **Backend Server Group**, retain the default values for other parameters, and click **Next: Add Backend Server**.
6. Add **ECS_server** to the backend server group you have created, enable **Health Check**, and retain the default values for the health check.
7. Click **Next: Confirm** and then click **Submit**.

Step 2: Add an HTTP Listener and Enable Redirect to Another Listener

You can enable redirection when adding an HTTP listener and select an HTTPS listener to which requests are redirected. Alternatively, you can add a forwarding policy for an HTTP listener to redirect requests to an HTTPS listener.

1. Go to the **load balancer list page**.
2. On the **Load Balancers** page, locate the target load balancer and click its name.
3. On the **Listeners** tab, click **Add Listener**. Configure the parameters based on **Table 3-2**.

Figure 3-3 Adding an HTTP listener

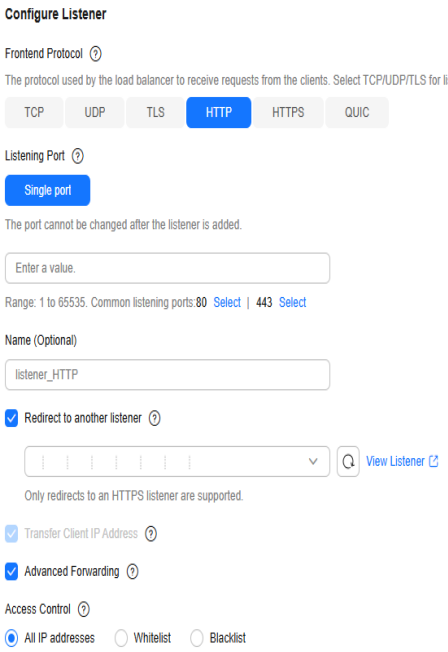


Table 3-2 Parameters for configuring an HTTP listener

Parameter	Example Value	Description
Frontend Protocol	HTTP	Specifies the protocol that will be used by the load balancer to receive requests from clients.
Listening Port	80	Specifies the port that will be used by the load balancer to receive requests from clients.
Name (Optional)	listener-HTTP	Specifies the listener name.
Redirect to another listener	Select it and choose the HTTPS listener created in **Step 1: Add an HTTPS Listener**.	Specifies whether to enable redirection. You can use this function to redirect the requests from an HTTP listener to an HTTPS listener to ensure security.
Transfer Client IP Address	Enabled by default	Specifies whether to transmit IP addresses of the clients to backend servers.
Advanced Forwarding	Enabled	Specifies whether to enable advanced forwarding. You can configure advanced forwarding policies to forward requests to different backend server groups.

Parameter	Example Value	Description
Access Control	All IP addresses	Specifies how access to the listener is controlled. Access from specific IP addresses can be controlled using a whitelist or blacklist.

4. Retain the default values for parameters under **More (Optional)** and click **Next: Confirm**.
5. Click **Submit**.

NOTE

- After the redirection is added, the configurations for the HTTP listener will not be applied, but access control configured for that listener will still be applied.
- After the redirection is added for an HTTP listener, the backend server will return 301 Moved Permanently to the clients.

Step 3: Verify the Redirection to HTTPS

Remotely log in to **ECS_client** and run **curl -H "Accept-Language: zh-CN,zh" "http://*ELB-private-IP-address*:80** to check whether HTTP requests are redirected.

If 301 Moved Permanently is returned, as shown in the below figure, HTTP requests are directed to an HTTPS listener.

Figure 3-4 Verifying redirection to an HTTPS listener

```
]#curl -H "Accepte-Language: zh-CN,zh" 'http://          :80'
<html>
<head><title>301 Moved Permanently</title></head>
<body>
<center><h1>301 Moved Permanently</h1></center>
</body>
</html>
```

3.2 Configuring One-way Authentication When Adding an HTTPS Listener

Scenarios

If only server authentication is required, you can configure one-way authentication when adding an HTTPS listener to a load balancer.

This section uses certificates purchased on the **Cloud Certificate & Manager (CCM)** console.

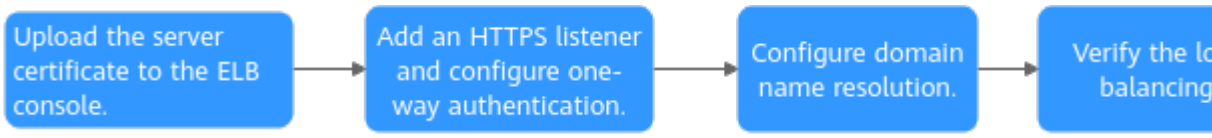
Prerequisites

- There is a dedicated load balancer with an EIP bound to it. If there is no such resource, you can **buy one** and **bind an IPv4 EIP to the load balancer**.

- There is an HTTPS backend server group with an ECS (ECS01) running in it. The ECS hosts an application.
- You have either purchased a certificate or uploaded a third-party certificate to SSL Certificate Manager (SCM), and configured a public domain name for the certificate. It is recommended that you **purchase an SSL certificate** on the CCM console.

Procedure

Figure 3-5 Procedure for configuring one-way authentication



Step 1: Upload the Server Certificate to ELB

Before adding an HTTPS listener to a load balancer, you need to upload your certificate to the ELB console.

1. Go to the **load balancer list page**.
2. In the navigation pane on the left, choose **Certificates**.
3. Click **Add Certificate** in the top right corner and set parameters by referring to **Table 3-3**.

Table 3-3 Server certificate parameters

Parameter	Description
Certificate Type	Specifies the certificate type. Select **Server certificate**.
Source	Specifies the source of a certificate. There are two options: **SSL Certificate Manager** and **Your certificate**. **SSL Certificate Manager** is used in this example, so that you can select the SSL certificates you have purchased on the CCM console.
Certificate	Specifies the certificate that you want to upload to the ELB console.
Enterprise Project	Specifies an enterprise project by which cloud resources and members are centrally managed.
SNI Domain Name (Optional)	All domain names of the SSL certificate will be automatically selected. If the certificate is intended for SNI, you can select an SNI certificate based on the domain name in the HTTPS requests.

Parameter	Description
Description (Optional)	Provides supplementary information about the certificate.

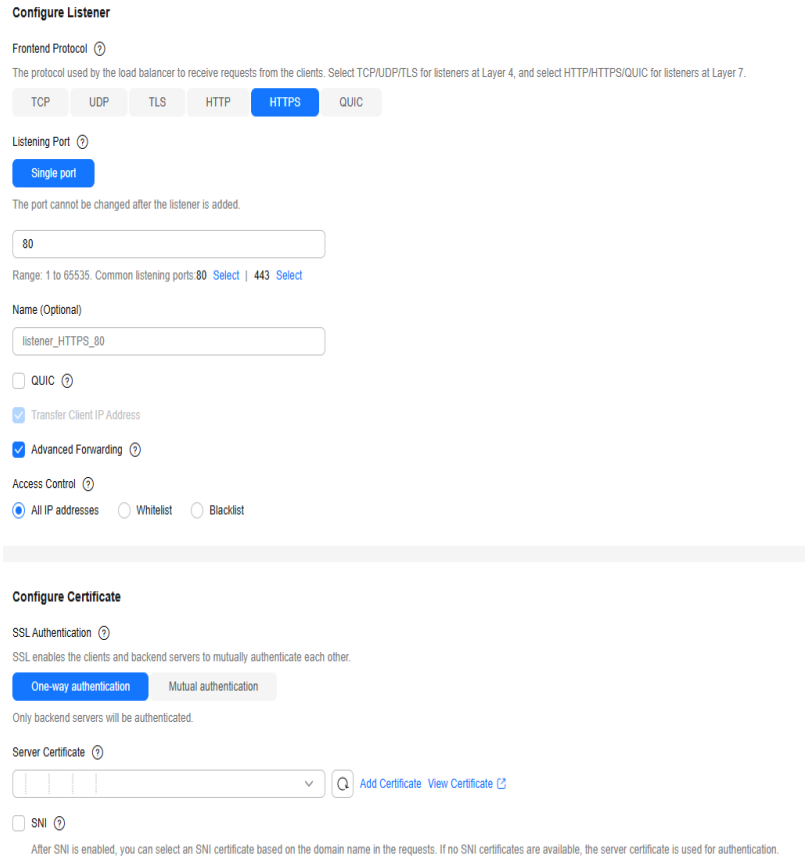
4. Click **OK**.

Step 2: Add an HTTPS Listener and Configure One-Way Authentication

1. Go to the **load balancer list page**.
2. Locate the target load balancer and click **Add Listener** in the **Operation** column.
3. On the **Add Listener** page, select **HTTPS** for **Frontend Protocol** and **One-way authentication** for **SSL Authentication**.

   Select the server certificate uploaded to the ELB console in **Step 1**.

   Figure 3-6 Configuring one-way authentication



4. Click **Next: Configure Request Routing Policy** and select **Use existing** for **Backend Server Group**. Select an existing backend server group and click **Next: Confirm**.
5. Confirm the configurations and click **Submit**.

Step 3: Configure Domain Name Resolution

You can add an A record set to resolve the domain name to the public IP address of the load balancer so that clients can access the load balancer using the public domain name.

The following provides an example for resolving a website domain name to an IPv4 address. For details about how to configure an A record set, see **Routing Internet Traffic to a Website**.

1. Go to the **DNS console**.
2. In the navigation pane on the left, choose **Public Zones**.

   The zone list is displayed.
3. Locate the public zone and click **Manage Record Sets** in the **Operation** column.
4. Click **Add Record Set**.
5. Configure the parameters based on **Table 3-4**.

   Table 3-4 Parameters for adding an A record set

Parameter	Example Value	Description
Type	A – Map domains to IPv4 addresses	Type of the record set. In this example, set it to **A - Map domains to IPv4 addresses**.
Name	www	Prefix of the domain name to be resolved.
Line	Default	Resolution line. The DNS server will return the IP address of the specified line, depending on where end users come from. The default value is **Default**. **Default**: returns the default resolution result irrespective of where the visitors come from.
TTL (s)	300	Cache duration of the record set on a local DNS server, in seconds. In this example, the default value 300 is used.
Value	192.168.12.2	IPv4 addresses mapped to the domain name. In this example, set this parameter to the EIPs bound to the load balancer.
Advanced Settings (Optional)	-	Click ⌄ to expand the advanced settings, set the alias and weight of the record set, and add a description and tags. In this example, the default settings are used.

6. Click **OK**.

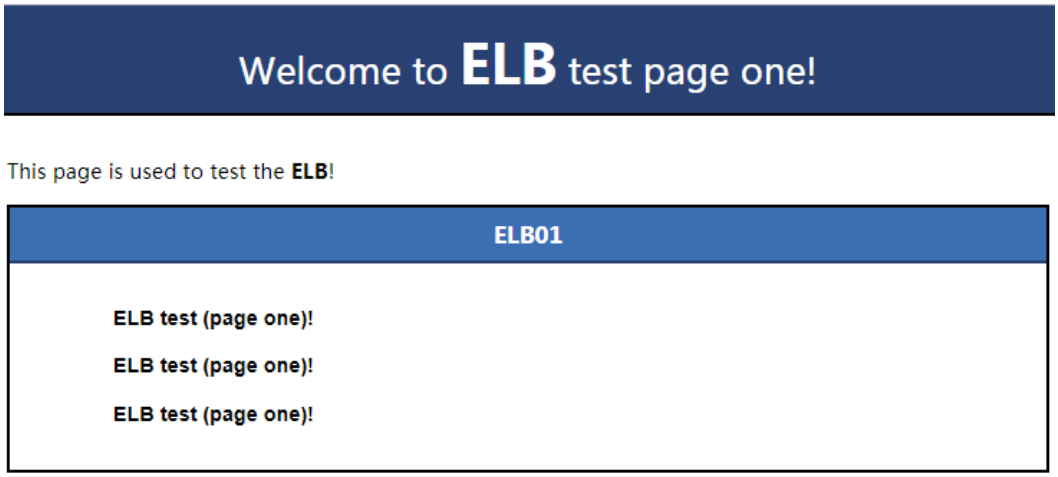
7. Switch back to the **Record Sets** tab.
   The added record set is in the **Normal** state.

Step 4: Verify Load Balancing

Deploy an application on ECS01, so that a page with message "Welcome to ELB test page one!" is returned when ECS01 is accessed. For details, see **Deploy the Application**.

Use a browser to access the domain name (**https://***load-balancer-domain-name*) of the load balancer. If the following page is displayed, the load balancer forwards the access request to ECS01, and the HTTPS one-way authentication is successfully configured.

Figure 3-7 Accessing ECS01



3.3 Configuring Mutual Authentication When Adding an HTTPS Listener

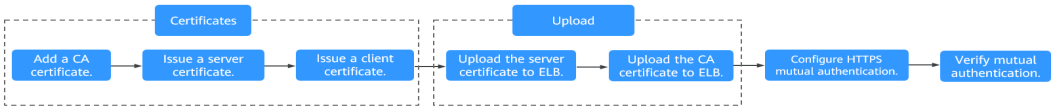
Scenarios

In common HTTPS service scenarios, only the server certificate is required for authentication. For some mission-critical services, you need to deploy both the server certificate and the client certificate for mutual authentication.

Self-signed certificates are used as an example to describe how to configure mutual authentication. Self-signed certificates do not provide all the security properties provided by certificates signed by a CA. It is recommended that you purchase certificates from **Cloud Certificate & Manager (CCM)** or CAs.

Procedure

Figure 3-8 Procedure for configuring mutual authentication



Step 1: Add a CA Certificate Using OpenSSL

1. Log in to a Linux server with OpenSSL installed.
2. Create the **server** directory and switch to the directory:

   mkdir ca

   cd ca
3. Create the certificate configuration file **ca_cert.conf**. The file content is as follows:

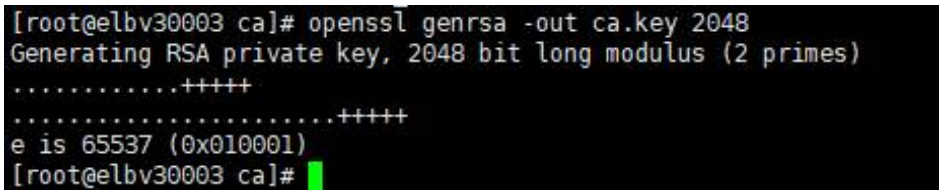
```
[ req ]
distinguished_name     = req_distinguished_name
prompt                 = no

[ req_distinguished_name ]
 O                     = ELB
```

4. Create the CA certificate private key **ca.key**.

   openssl genrsa -out ca.key 2048

   Figure 3-9 Private key of the CA certificate

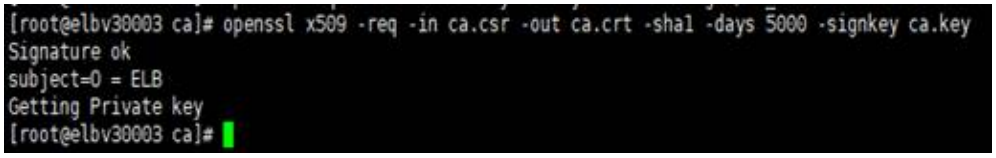


5. Create the certificate signing request (CSR) file **ca.csr** for the CA certificate.

   openssl req -out ca.csr -key ca.key -new -config ./ca_cert.conf
6. Create the self-signed CA certificate **ca.crt**.

   openssl x509 -req -in ca.csr -out ca.crt -sha1 -days 5000 -signkey ca.key

   Figure 3-10 Creating a self-signed CA certificate



Step 2: Issue a Server Certificate Using the CA Certificate

The server certificate can be a CA signed certificate or a self-signed one. In the following steps, a self-signed certificate is used as an example to describe how to create a server certificate.

1. Log in to the server where the CA certificate is generated.
2. Create a directory at the same level as the directory of the CA certificate and switch to the directory.

   mkdir server

   cd server
3. Create the certificate configuration file **server_cert.conf**. The file content is as follows:

```
[ req ]
distinguished_name     = req_distinguished_name
```



```
prompt              = no

[ req_distinguished_name ]
 O                  = ELB
 CN                 = www.test.com
```

NOTE

Set the **CN** field to the domain name or IP address of the Linux server.

4. Create the server certificate private key **server.key**.

   openssl genrsa -out server.key 2048

5. Create the CSR file **server.csr** for the server certificate.

   openssl req -out server.csr -key server.key -new -config ./server_cert.conf

6. Use the CA certificate to issue the server certificate **server.crt**.

   openssl x509 -req -in server.csr -out server.crt -sha1 -CAcreateserial -days 5000 -CA ../ca/ca.crt -CAkey ../ca/ca.key

Figure 3-11 Issuing a server certificate

```
[root@elbv30003 server]# openssl x509 -req -in server.csr -out server.crt -sha1 -CAcreateserial -days 5000 -CA ../ca/ca.crt -CAkey ../ca/ca.key
Signature ok
subject=O = ELB, CN = www.test.com
Getting CA Private Key
[root@elbv30003 server]#
```

Step 3: Issue a Client Certificate Using the CA Certificate

1. Log in to the server where the CA certificate is generated.
2. Create a directory at the same level as the directory of the CA certificate and switch to the directory.

   mkdir client

   cd client

3. Create the certificate configuration file **client_cert.conf**. The file content is as follows:

```
[ req ]
distinguished_name     = req_distinguished_name
prompt              = no

[ req_distinguished_name ]
 O                  = ELB
 CN                 = www.test.com
```

NOTE

Set the **CN** field to the domain name or IP address of the Linux server.

4. Create the client certificate private key **client.key**.

   openssl genrsa -out client.key 2048

Figure 3-12 Creating a client certificate private key

```
[root@elbv30003 client]# openssl genrsa -out client.key 2048
Generating RSA private key, 2048 bit long modulus (2 primes)
.....................................................................+++++
.....+++++
e is 65537 (0x010001)
[root@elbv30003 client]#
```


5. Create the CSR file **client.csr** for the client certificate.

   openssl req -out client.csr -key client.key -new -config ./client_cert.conf

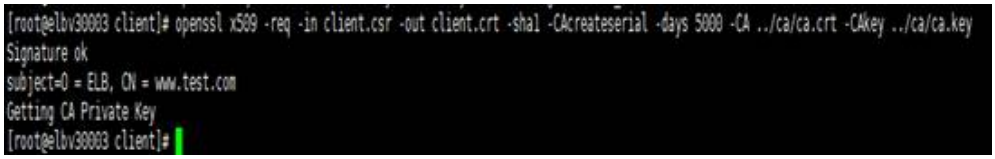
   Figure 3-13 Creating a client certificate CSR file

6. Use the CA certificate to issue the client certificate **client.crt**.

   openssl x509 -req -in client.csr -out client.crt -sha1 -CAcreateserial -days 5000 -CA ../ca/ca.crt -CAkey ../ca/ca.key

   Figure 3-14 Issuing a client certificate

7. Convert the client certificate to a **.p12** file that can be identified by the browser.

   openssl pkcs12 -export -clcerts -in client.crt -inkey client.key -out client.p12

   NOTE

   A password is required during command execution. Save this password, which will be required when you import the certificate using the browser.

Step 4: Upload the Server Certificate to ELB

1. Log in to the load balancer management console.
2. In the navigation pane on the left, choose **Certificates**.
3. In the navigation pane on the left, choose **Certificates**. On the displayed page, click **Add Certificate**. In the **Add Certificate** dialog box, select **Server certificate**, copy the content of server certificate **server.crt** to the **Certificate Content** area and the content of private key file **server.key** to the **Private Key** area, and click **OK**.

   NOTE

   Delete the last newline character before you copy the content.

   NOTE

   The certificate and private key must be PEM-encoded.

Step 5: Upload the CA Certificate to ELB

Step 1 Log in to the load balancer management console.

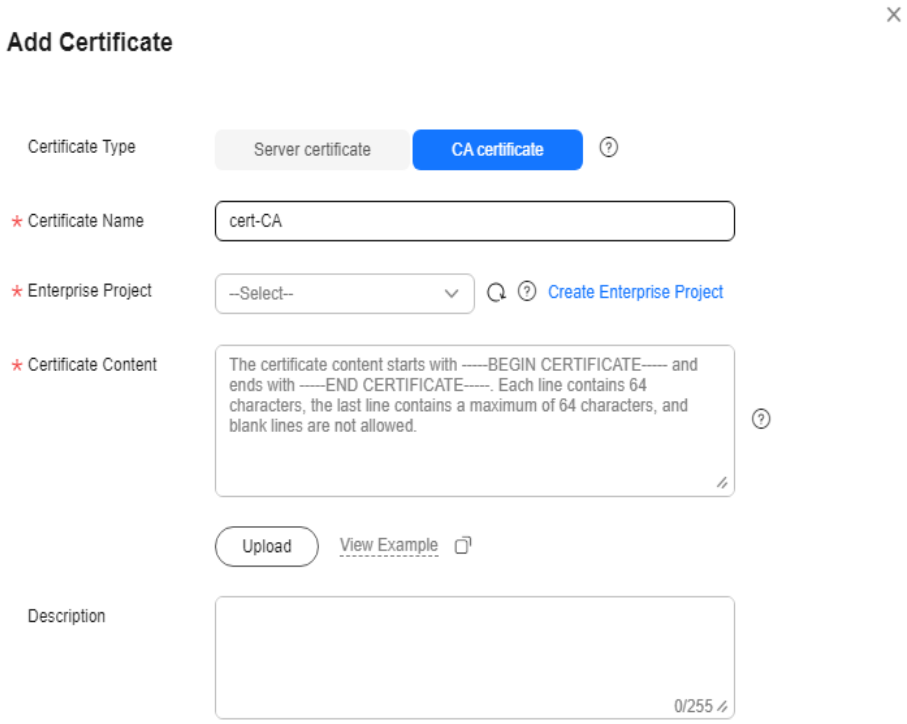
Step 2 In the navigation pane on the left, choose **Certificates**.

Step 3 Click **Add Certificate**. In the **Add Certificate** dialog box, select **CA certificate**, copy the content of CA certificate **ca.crt** created in **Step 1: Add a CA Certificate Using OpenSSL** to the **Certificate Content** area, and click **OK**.

NOTE

Delete the last newline character before you copy the content.

Figure 3-15 Adding a CA certificate



NOTE

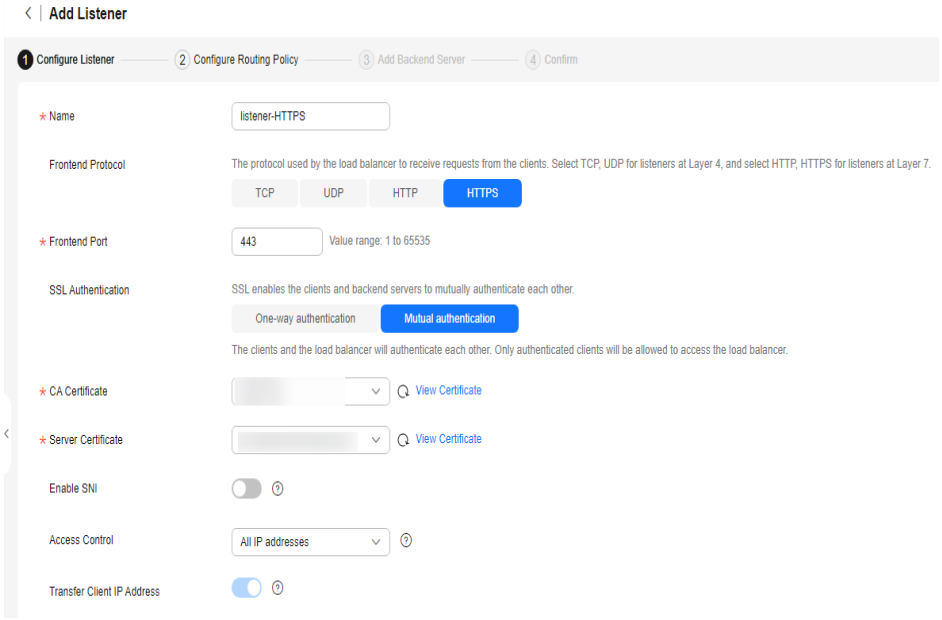
The certificate must be PEM-encoded.

----End

Step 6: Configure HTTPS Mutual Authentication

1. Log in to the load balancer management console.
2. Locate the target load balancer and click its name. Under **Listeners**, click **Add Listener**. Select **HTTPS** for **Frontend Protocol** and **Mutual authentication** for **SSL Authentication**, and select the CA certificate and server certificate you have added.

Figure 3-16 Configuring mutual authentication



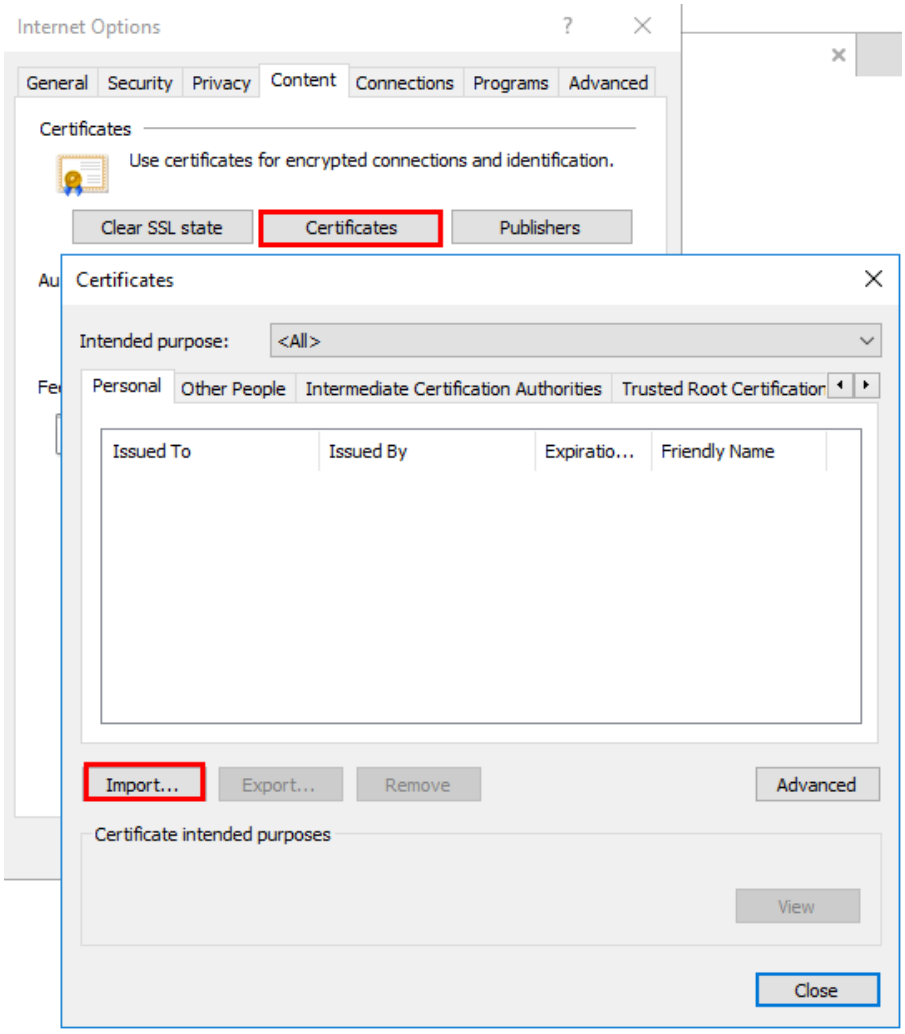
3. Click **Next: Configure Request Routing Policy** and select **Use existing** for **Backend Server Group**. Select an existing backend server group and click **Next: Confirm**.
4. Confirm the configurations and click **Submit**.

Step 7: Import the Client Certificate and Verify Mutual Authentication

Method 1: Using a browser

1. Import the client certificate using a browser (Internet Explorer 11 is used as an example).
   a. Export **client.p12** from the Linux server.
   b. Open the browser, choose **Settings** > **Internet Options** and click **Content**.
   c. Click **Certificates** and then **Import** to import the **client.p12** certificate.

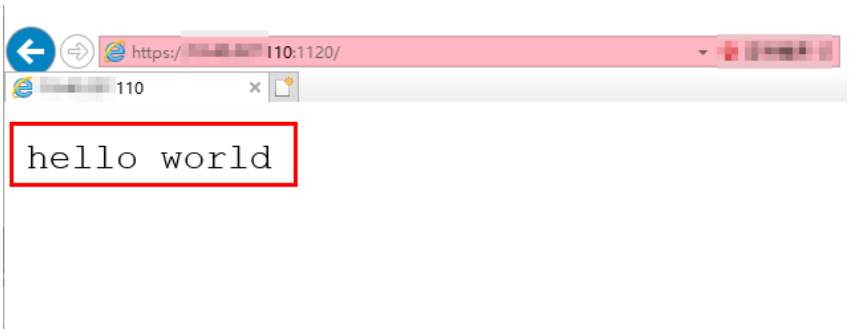
Figure 3-17 Importing the **client.p12** certificate



2. Verify the import.

   Enter the access address in the address box of your browser. A window is displayed asking you to select the certificate. Select the client certificate and click **OK**. If the website can be accessed, the certificate is successfully imported.

Figure 3-18 Accessing the website



Method 2: Using cURL

1. Import the client certificate.

   Copy client certificate **client.crt** and private key **client.key** to a new directory, for example, **/home/client_cert**.

2. Verify the import.

   On the Shell screen, run the following command:

   ```
   curl -k --cert /home/client_cert/client.crt --key /home/client_cert/client.key https://XXX.XXX.XXX.XXX:XXX/ -I
   ```

   Ensure that the certificate address, private key address, IP address and listening port of the load balancer are correct. Replace **https://XXX.XXX.XXX.XXX:XXX** with the actual IP address and port number. If the expected response code is returned, the certificate is successfully imported.

   Figure 3-19 Example of a correct response code

   ```
   [192.168.10.216  test]#curl -k --cert client.crt --key client.key https://192.168.10.16:4500 -I
   HTTP/1.1 200 OK
   Date: Fri, 25 Sep 2020 10:11:17 GMT
   Content-Type: application/octet-stream
   Connection: keep-alive
   Set-Cookie: name=d92f80b6-55e9-4b61-9c37-932ccd7b02f2; path=/; Expires=Sat, 26-Sep-20 10:11:19 GMT
   Server: elb
   ```

3.4 Using a Dedicated Load Balancer for TLS Offloading (One-Way Authentication)

Scenarios

If your Layer 4 services have high security requirements, you can use SSL encryption to improve service security. However, configuring SSL encryption on backend servers may lower their performance. To address this issue, you can add a TLS listener to a dedicated load balancer to forward requests and deploy certificates on the listener. The load balancer decrypts incoming requests and forwards them as plaintext to your backend servers. You do not need to configure certificates on backend servers.

TLS offloading enhances network security, improves backend server performance, simplifies backend server configuration and O&M, and helps efficiently and securely forward Layer 4 service traffic.

Prerequisites

- There is a dedicated load balancer with an EIP bound to it. If there is no such resource, you can **buy one** and **bind an IPv4 EIP to the load balancer**.
- You have either purchased a certificate or uploaded a third-party certificate to SSL Certificate Manager (SCM), and configured a public domain name for the certificate. It is recommended to **purchase an SSL certificate** on the Cloud Certificate & Manager (CCM) console.
- There is a TLS backend server group with two ECSs (ECS01 and ECS02) running in it. Each ECS hosts an application.

  Deployment Commands of ECS01

  - yum install -y nginx
  - systemctl start nginx.service

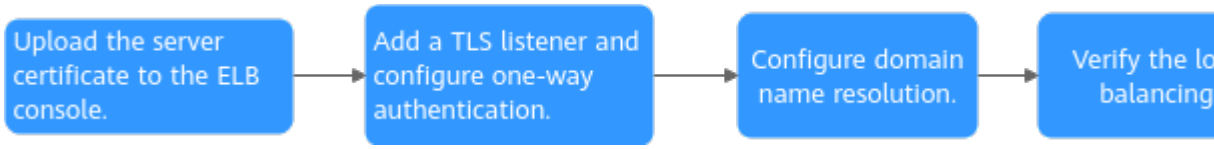
- cd /usr/share/nginx/html/
- echo "Hello World! This is ECS01 for the ELB test." > index.html

Deployment Commands of ECS02

- yum install -y nginx
- systemctl start nginx.service
- cd /usr/share/nginx/html/
- echo "Hello World! This is ECS02 for the ELB test." > index.html

Procedure

Figure 3-20 Procedure for configuring one-way authentication for TLS offloading



Step 1: Upload the Server Certificate to the ELB Console

Before adding a TLS listener to a load balancer, you need to upload your server certificate to the ELB console.

1. Go to the **load balancer list page**.
2. In the navigation pane on the left, choose **Certificates**.
3. Click **Add Certificate** in the top right corner and set parameters by referring to **Table 3-5**.

Table 3-5 Server certificate parameters

Parameter	Description
Certificate Type	Specifies the certificate type. Select **Server certificate**.
Source	Specifies the source of a certificate. There are two options: **SSL Certificate Manager** and **Your certificate**. **SSL Certificate Manager** is used in this example, so that you can select the SSL certificates you have purchased on the CCM console.
Certificate	Specifies the certificate that you want to upload to the ELB console.
Enterprise Project	Specifies an enterprise project by which cloud resources and members are centrally managed.

Parameter	Description
SNI Domain Name (Optional)	All domain names of the SSL certificate will be automatically selected. If the certificate is intended for SNI, you can select an SNI certificate based on the domain name in the HTTPS requests.
Description (Optional)	Provides supplementary information about the certificate.

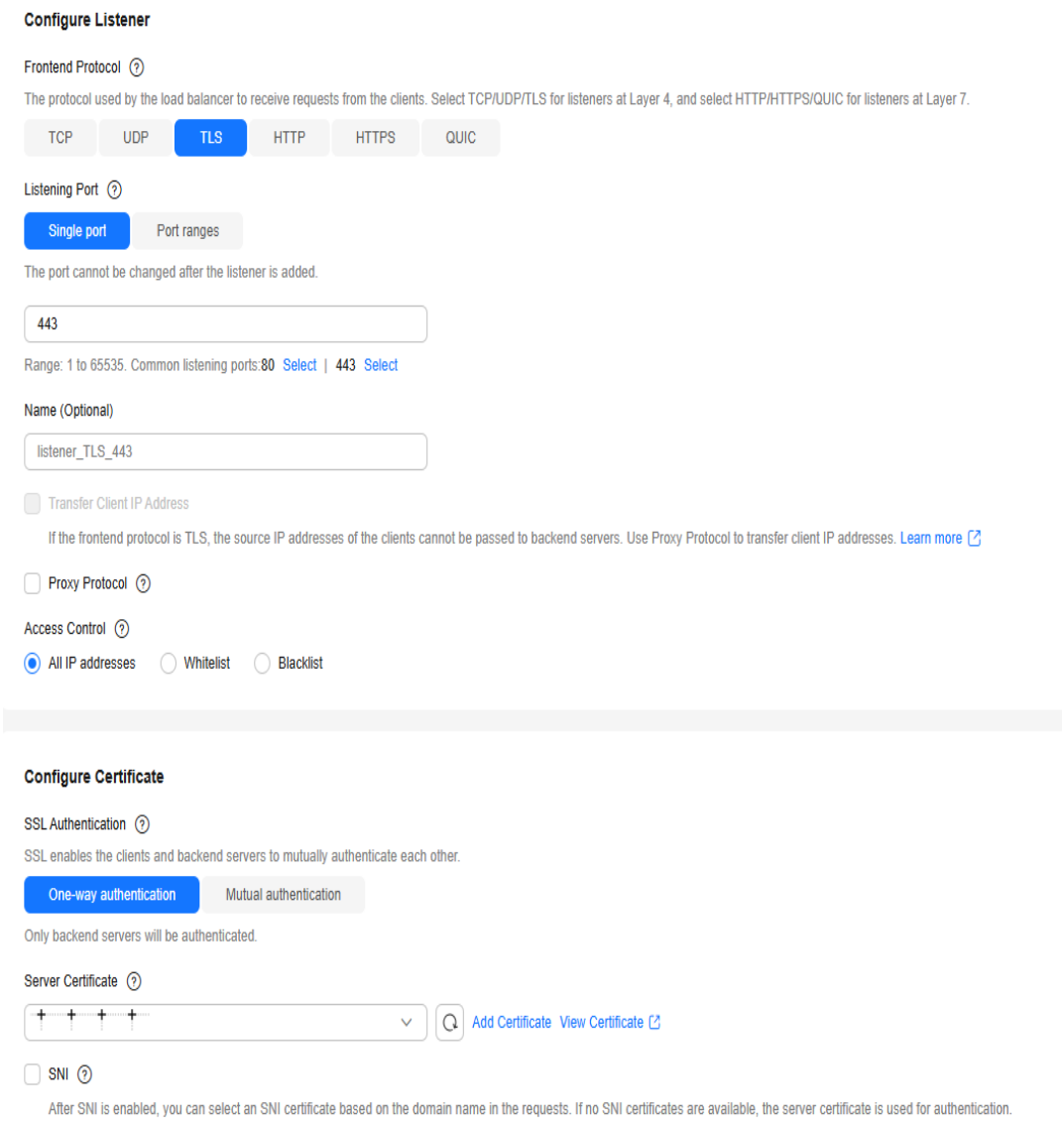
4. Click **OK**.

Step 2: Add a TLS Listener and Configure One-Way Authentication

1. Go to the **load balancer list page**.
2. Locate the target load balancer and click **Add Listener** in the **Operation** column.
3. On the **Add Listener** page, select **TLS** for **Frontend Protocol** and **One-way authentication** for **SSL Authentication**.

   Select the server certificate uploaded to the ELB console in **Step 1** for **Server Certificate**, and retain the default values for other parameters or change them as needed.

Figure 3-21 Configuring one-way authentication



4. Click **Next: Configure Request Routing Policy** and select **Use existing** for **Backend Server Group**. Select an existing backend server group and click **Next: Confirm**.
5. Confirm the configurations and click **Submit**.

Step 3: Configure Domain Name Resolution

You can add an A record set to resolve the domain name to the public IP address of the load balancer so that clients can access the load balancer using the public domain name.

The following provides an example for resolving a website domain name to an IPv4 address. For details about how to configure an A record set, see **Routing Internet Traffic to a Website**.

1. Go to the **DNS console**.

2. In the navigation pane on the left, choose **Public Zones**.
   The zone list is displayed.
3. Locate the public zone and click **Manage Record Sets** in the **Operation** column.
4. Click **Add Record Set**.
5. Configure the parameters based on **Table 3-6**.

   Table 3-6 Parameters for adding an A record set

Parameter	Example Value	Description
Type	A – Map domains to IPv4 addresses	Type of the record set. In this example, set it to **A - Map domains to IPv4 addresses**.
Name	www	Prefix of the domain name to be resolved.
Line	Default	Resolution line. The DNS server will return the IP address of the specified line, depending on where end users come from. The default value is **Default**. **Default**: returns the default resolution result irrespective of where the visitors come from.
TTL (s)	300	Cache duration of the record set on a local DNS server, in seconds. In this example, the default value 300 is used.
Value	192.168.12.2	IPv4 addresses mapped to the domain name. In this example, set this parameter to the EIPs bound to the load balancer.
Advanced Settings (Optional)	-	Click ⌄ to expand the advanced settings, set the alias and weight of the record set, and add a description and tags. In this example, the default settings are used.

6. Click **OK**.
7. Switch back to the **Record Sets** tab.
   The added record set is in the **Normal** state.

Step 4: Verify Load Balancing

Enter the domain name of the load balancer in the address box of the browser, for example, **https://www.elbtest.com**. Browser cache can cause clients to reuse existing TLS sessions. For accurate testing, open the website in incognito mode. Refresh the page multiple times and you will see that requests are distributed across the two ECSs. Using a self-signed certificate may trigger a browser warning about insecure connections. This does not affect load balancing verification, but reduces the browser's trust in the connections.

Figure 3-22 Requests forwarded to ECS01

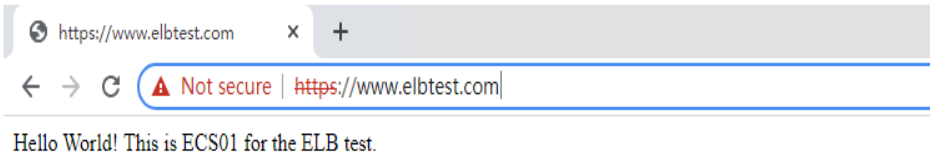
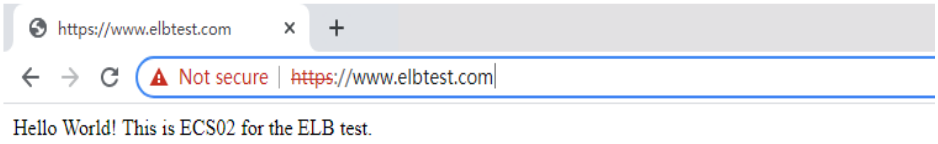


Figure 3-23 Requests forwarded to ECS02



Reference

- For details about how to add a TLS listener, see **Adding a TLS Listener**.
- Related APIs
  - **Creating a Listener**
  - **Creating a Certificate**

3.5 Using a Dedicated Load Balancer for TLS Offloading (Mutual Authentication)

Scenarios

If your Layer 4 services have strict security requirements, you can configure mutual authentication for TLS listeners to allow clients and servers to authenticate each other to improve service security.

Prerequisites

- There is a dedicated load balancer with an EIP bound to it. If there is no such resource, you can **buy one** and **bind an IPv4 EIP to the load balancer**.
- You have either purchased a certificate or uploaded a third-party certificate to SSL Certificate Manager (SCM), and configured a public domain name for the

certificate. It is recommended to **purchase an SSL certificate** on the Cloud Certificate & Manager (CCM) console.

- You have purchased a CA certificate and exported the CA certificate to the local PC, or you have a self-signed CA certificate. If you do not have such certificates, you can **purchase a private CA** from Huawei Cloud CCM and **export a private CA certificate**.
- You have issued a private certificate using the private CA and install the certificate on the client by referring to **Applying for a Private Certificate** and **Installing a Private Certificate on the Client**.
- There is a TLS backend server group with two ECSs (ECS01 and ECS02) running in it. Each ECS hosts an application.

  Deployment Commands of ECS01

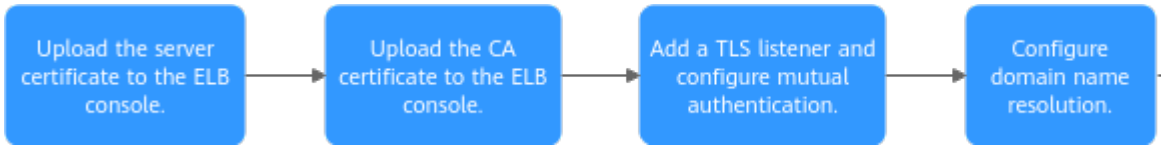
  - yum install -y nginx
  - systemctl start nginx.service
  - cd /usr/share/nginx/html/
  - echo "Hello World! This is ECS01 for the ELB test." > index.html

  Deployment Commands of ECS02

  - yum install -y nginx
  - systemctl start nginx.service
  - cd /usr/share/nginx/html/
  - echo "Hello World! This is ECS02 for the ELB test." > index.html

Procedure

Figure 3-24 Procedure for configuring mutual authentication for TLS offloading



Step 1: Upload the Server Certificate to the ELB Console

Before adding a TLS listener to a load balancer, you need to upload your server certificate to the ELB console.

1. Go to the **load balancer list page**.
2. In the navigation pane on the left, choose **Certificates**.
3. Click **Add Certificate** in the top right corner and set parameters by referring to **Table 3-7**.

Table 3-7 Server certificate parameters

Parameter	Description
Certificate Type	Specifies the certificate type. Select **Server certificate**.

Parameter	Description
Source	Specifies the source of a certificate. There are two options: **SSL Certificate Manager** and **Your certificate**. **SSL Certificate Manager** is used in this example, so that you can select the SSL certificates you have purchased on the CCM console.
Certificate	Specifies the certificate that you want to upload to the ELB console.
Enterprise Project	Specifies an enterprise project by which cloud resources and members are centrally managed.
SNI Domain Name (Optional)	All domain names of the SSL certificate will be automatically selected. If the certificate is intended for SNI, you can select an SNI certificate based on the domain name in the HTTPS requests.
Description (Optional)	Provides supplementary information about the certificate.

4. Click **OK**.

Step 2: Upload the CA Certificate to the ELB Console

Before adding a TLS listener to a load balancer, you need to upload your CA certificate to the ELB console.

1. Go to the **load balancer list page**.
2. In the navigation pane on the left, choose **Certificates**.
3. Click **Add Certificate** in the top right corner and set parameters by referring to **Table 3-8**.

Table 3-8 CA certificate parameters

Parameter	Description
Certificate Type	Specifies the certificate type. Select **CA certificate**.
Certificate Name	Specifies the name of the CA certificate.
Enterprise Project	Specifies an enterprise project by which cloud resources and members are centrally managed.

Parameter	Description
Certificate Content	Specifies the content of the CA certificate in PEM format. Click **Upload** and select the CA certificate to be uploaded. Ensure that your browser is the latest version. The format of the certificate body is as follows: -----BEGIN CERTIFICATE----- Base64-encoded certificate -----END CERTIFICATE-----
Description (Optional)	Provides supplementary information about the certificate.

4. Click **OK**.

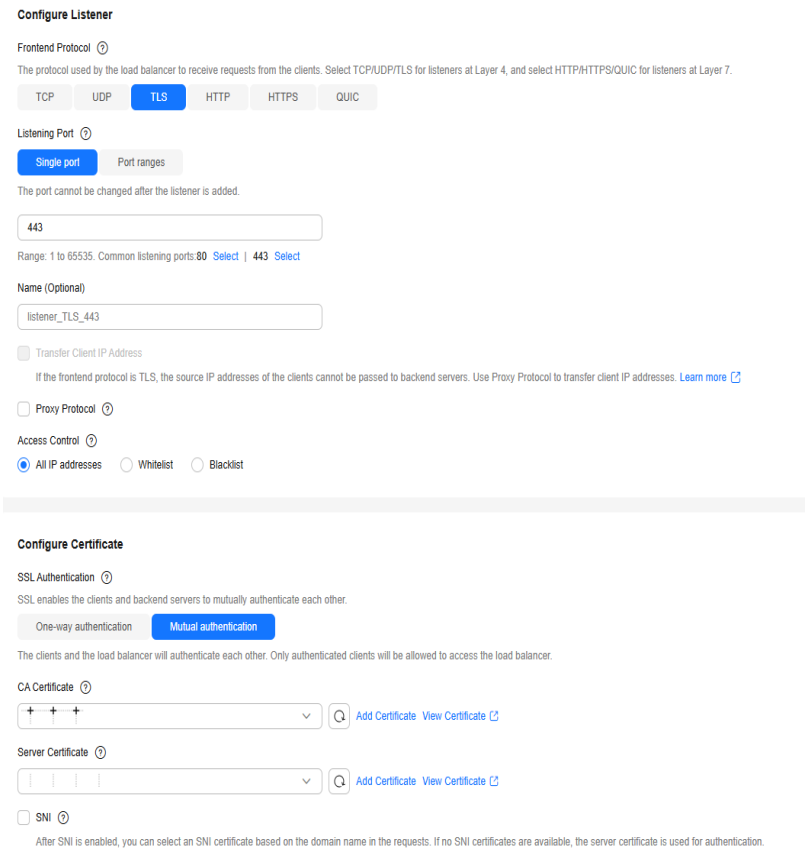
Step 3: Add a TLS Listener and Configure Mutual Authentication

1. Go to the **load balancer list page**.
2. Locate the target load balancer and click **Add Listener** in the **Operation** column.
3. On the **Add Listener** page, select **TLS** for **Frontend Protocol** and **Mutual authentication** for **SSL Authentication**.

   Select the server certificate uploaded to the ELB console in **Step 1**.

   Select the CA certificate uploaded to the ELB console in **Step 2**.

Figure 3-25 Configuring mutual authentication



4. Click **Next: Configure Request Routing Policy** and select **Use existing** for **Backend Server Group**. Select an existing backend server group and click **Next: Confirm**.
5. Confirm the configurations and click **Submit**.

Step 4: Configure Domain Name Resolution

You can add an A record set to resolve the domain name to the public IP address of the load balancer so that clients can access the load balancer using the public domain name.

For details about how to configure A record sets, see **Routing Internet Traffic to a Website**.

1. Go to the **DNS console**.
2. In the navigation pane on the left, choose **Public Zones**.

   The zone list is displayed.
3. Locate the public zone and click **Manage Record Sets** in the **Operation** column.
4. Click **Add Record Set**.
5. Configure the parameters based on **Table 3-9**.

Table 3-9 Parameters for adding an A record set

Parameter	Example Value	Description
Type	A – Map domains to IPv4 addresses	Type of the record set. In this example, set it to **A - Map domains to IPv4 addresses**.
Name	www	Prefix of the domain name to be resolved.
Line	Default	Resolution line. The DNS server will return the IP address of the specified line, depending on where end users come from. The default value is **Default**. **Default**: returns the default resolution result irrespective of where the visitors come from.
TTL (s)	300	Cache duration of the record set on a local DNS server, in seconds. In this example, the default value 300 is used.
Value	192.168.12.2 192.168.12.3	IPv4 addresses mapped to the domain name. In this example, set this parameter to the EIPs bound to the load balancer.
Advanced Settings (Optional)	-	Click ⌄ to expand the advanced settings, set the alias and weight of the record set, and add a description and tags. In this example, the default settings are used.

6. Click **OK**.
7. Switch back to the **Record Sets** tab.
   The added record set is in the **Normal** state.

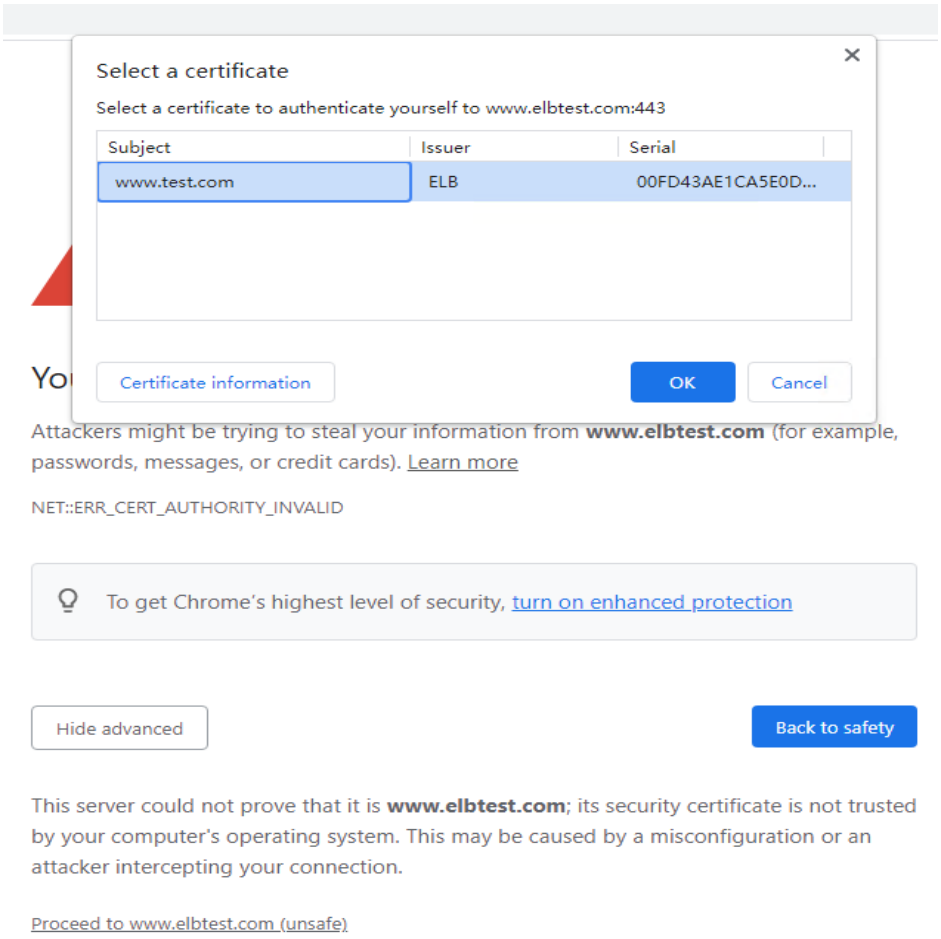
Step 5: Verify TLS Mutual Authentication

The following describes how you can verify TLS mutual authentication.

Verifying Mutual Authentication Between a Windows Client and Backend Servers

1. Enter the domain name of the load balancer in the address box of the browser, for example, **https://www.elbtest.com**. In the displayed dialog box, select a certificate to authenticate yourself and click **OK**.

Figure 3-26 Selecting a certificate to authenticate yourself



2. Open the website in the incognito mode for accurate testing, because browser cache can cause clients to reuse existing TLS sessions. Refresh the page multiple times and you will see that requests are distributed across the two ECSs.

Figure 3-27 Requests forwarded to ECS01

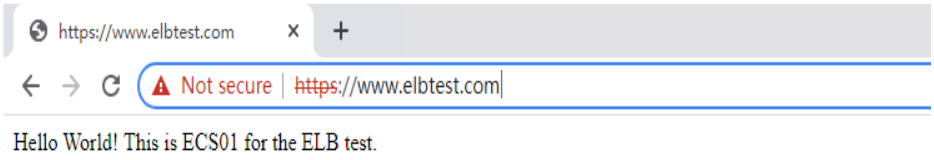
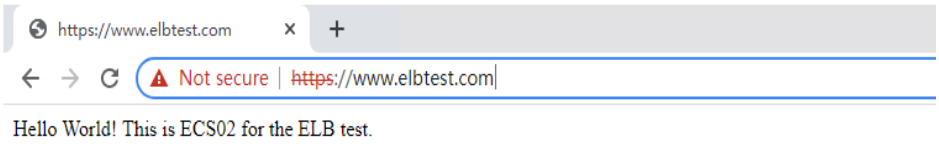


Figure 3-28 Requests forwarded to ECS02



Verifying Mutual Authentication Between a Linux Client and Backend Servers

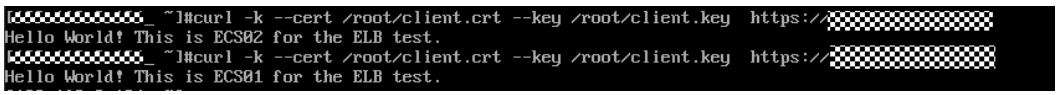
Log in to the Linux client and run the following command to verify the mutual authentication:

```
curl -k --cert /root/client.crt --key /root/client.key https://www.elbtest.com
```

--cert /root/client.crt defines where the client certificate file is stored, and **--key /root/client.key** indicates where the private key of the client certificate is stored.

If the following information is displayed, the client and servers have authenticated each other, allowing requests to reach the two ECSs.

Figure 3-29 Verifying mutual authentication (Linux)



Reference

- For details about how to add a TLS listener, see **Adding a TLS Listener**.
- Related APIs
  - **Creating a Listener**
  - **Creating a Certificate**

4 Advanced Functions

4.1 Using QUIC to Accelerate Access to an Application

You can add a QUIC listener to forward requests. The Quick UDP Internet Connection (QUIC) is a UDP-based protocol at the transport layer. It improves congestion control and does not depend on kernel protocols.

QUIC features low latency and avoids head-of-line blocking. It makes video and page loading faster, improving network performance and data security.

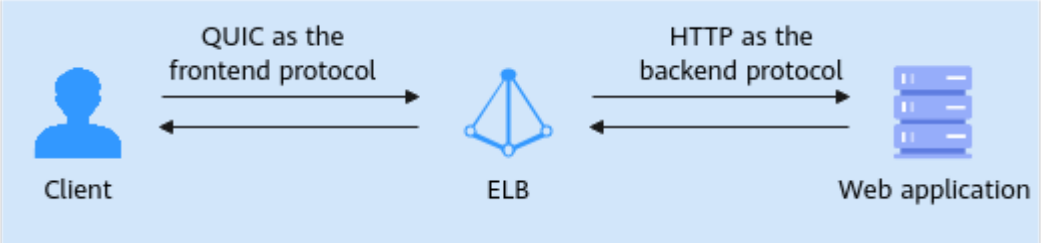
QUIC Overview

QUIC is built on top of UDP. Unlike TCP, QUIC does not require a three-way handshake for connection establishment, and it also avoids head-of-line blocking. QUIC's stream multiplexing allows multiple independent streams in a single connection. This means that if one stream experiences a loss or delay, it does not block the progress of other streams. Compared with TCP, QUIC is more flexible.

QUIC and HTTP/3

- HTTP/3, based on QUIC, is the third major version of the Hypertext Transfer Protocol (HTTP). When a browser makes its first request to a server, the connection is typically established using either HTTP/1.1 or HTTP/2. The server then informs the browser about its support for QUIC. The browser will then attempt a QUIC connection and a TCP connection in parallel for the second request. If the QUIC connection is established faster, the browser will use QUIC over HTTP/3 to communicate with the server.
- Clients can use QUIC only over HTTP/3. If no HTTP/3 connection can be established, HTTP/1.1 or HTTP/2 will be used.

Figure 4-1 Accessing an application through ELB using QUIC



Preparations

- There is a dedicated load balancer with an EIP bound to it. If there is no such a load balancer, you can **buy one** and **bind an IPv4 EIP to the load balancer**.
- You have **created two ECSs** running CentOS in the same VPC as the load balancer. The first ECS (ECS_client) is used as the client to send HTTP requests, and the second ECS (ECS_server) is used as the backend server for deploying the web application.
- You have purchased a certificate or uploaded a third-party certificate to SSL Certificate Manager (SCM). You are advised to purchase a server certificate on Cloud Certificate & Manager (CCM). For details, see **Purchasing an SSL Certificate**.
- There is an **HTTP** backend server group with an ECS (ECS_server) added to it. The backend port in this practice is 442.

Step 1: Configure the Client to Support HTTP/3

For web applications, the client that supports QUIC connections must support HTTP/3. For more information, see **HTTP/3 and QUIC Support**.

1. Remotely log in to ECS_client.

   Multiple methods are available for logging in to an ECS. For details, see **Logging In to an ECS**.

2. Install the basic dependencies.

   a. Install the epel-release package to enable the Extra Packages for Enterprise Linux (EPEL) repository and update the cache.

   ```
   sudo yum install -y epel-release
   sudo yum makecache
   ```

   b. Install the required software packages.

   ```
   sudo yum install -y git perl-IPC-cmd autoconf automake libtool libpsl-devel
   ```

3. Build the curl tool that supports QUIC and HTTP/3.

   Installation Reference

   – Install OpenSSL to provide TLS support for QUIC encrypted communication. The OpenSSL version must be 3.5 or later.

   ```
   % git clone --quiet --depth=1 -b openssl-$OPENSSL_VERSION https://github.com/openssl/openssl
   % cd openssl
   % ./config --prefix=<somewhere1> --libdir=lib
   % make
   % make install
   ```

   – Install nghttp3 to implement HTTP/3 and QPACK.

   ```
   % cd ..
   % git clone -b $NGHTTP3_VERSION https://github.com/ngtcp2/nghttp3
   ```



```
% cd nghttp3
% git submodule update --init
% autoreconf -fi
% ./configure --prefix=<somewhere2> --enable-lib-only
% make
% make install
```

- Install ngtcp2, the core library that implements QUIC.

```
% cd ..
% git clone -b $NGTCP2_VERSION https://github.com/ngtcp2/ngtcp2
% cd ngtcp2
% autoreconf -fi
% ./configure PKG_CONFIG_PATH=<somewhere1>/lib/pkgconfig:<somewhere2>/lib/pkgconfig
LDFLAGS="-Wl,-rpath,<somewhere1>/lib" --prefix=<somewhere3> --enable-lib-only --with-
openssl
% make
% make install
```

- Install curl to initiate HTTP/3 requests.

```
% cd ..
% git clone https://github.com/curl/curl
% cd curl
% autoreconf -fi
% LDFLAGS="-Wl,-rpath,<somewhere1>/lib" ./configure --with-openssl=<somewhere1> --with-
nghttp3=<somewhere2> --with-ngtcp2=<somewhere3>
% make
% make install
```

Step 2: Deploy the Web Application on the Backend Server

1. Remotely log in to the backend server ECS_server.
2. Create the **quic_testweb** directory on the backend server.

   ```
   mkdir quic_testweb
   ```

3. Create an **index.html** file in the **quic_testweb** directory.
   a. Create an **index.html** file.

      ```
      vi index.html
      ```

   b. Press **i** to enter the editing mode. The following shows an index.html script:

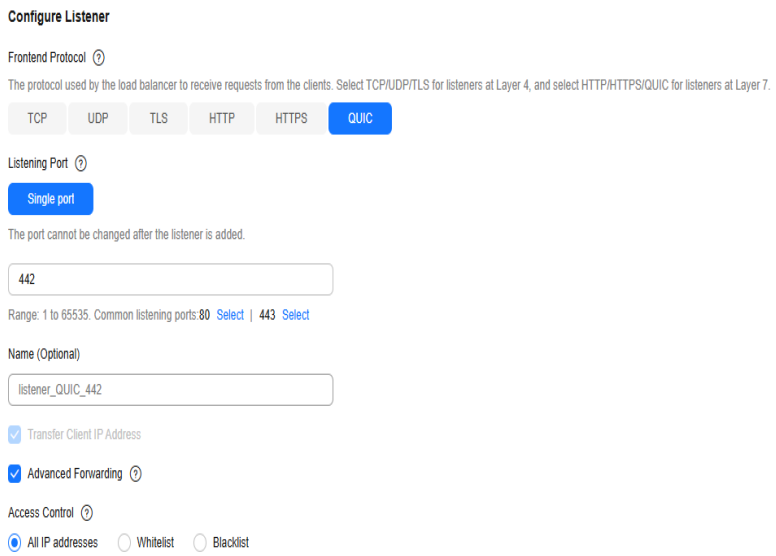
      ```
      hello, this is quic!
      ```

4. Press **Esc** to exit the editing mode. Then, enter **:wq** to save the **index.html** file.

Step 3: Add a QUIC Listener

1. Go to the **load balancer list page**.
2. Locate the target load balancer and click **Add Listener** in the **Operation** column.
3. On the **Add Listener** page, set the protocol to **QUIC** and port to **442**.

Figure 4-2 Adding a QUIC listener

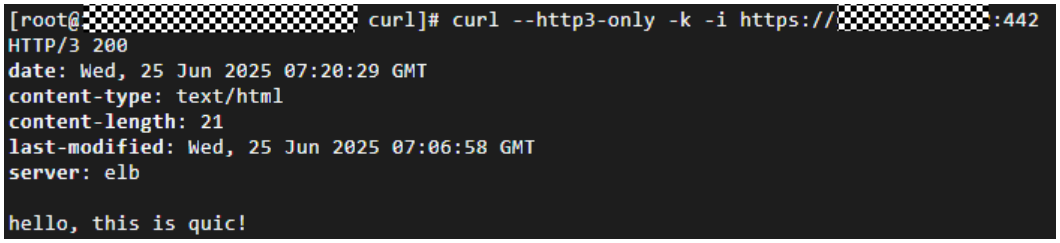


4. Click **Next: Configure Request Routing Policy** and select **Use existing** for **Backend Server Group**. Select the **HTTP** backend server group you have created and click **Next: Confirm**.
5. Confirm the configurations and click **Submit**.

Step 4: Verify QUIC Connections

1. Run the following command to access the EIP and listening port of the load balancer:
   ```
   curl --http3-only -k -i https://<EIP>:<PORT>
   ```
2. If the page shown in **Figure 4-3** is displayed, the client and backend server communicate with each other using HTTP/3 through QUIC negotiation.

Figure 4-3 Verifying communication using HTTP/3 through QUIC negotiation



4.2 Using ELB to Distribute gRPC Requests to Improve Concurrency Efficiency

Scenarios

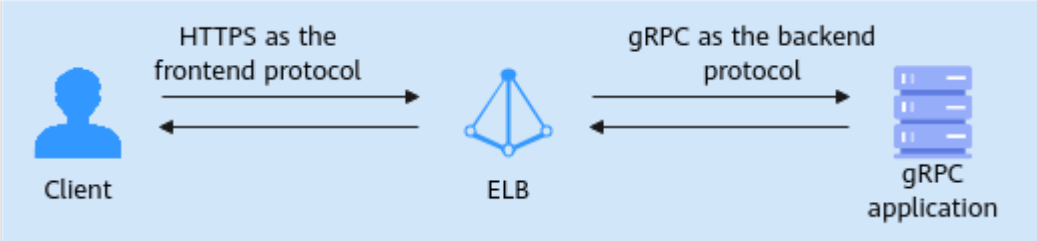
gRPC is an open-source remote procedure calls (RPC) framework for building high-performance, low-latency, language-agnostic APIs for distributed systems, such as microservice communication and mobile applications. You can use an HTTPS listener of a load balancer to distribute gRPC requests across backend

servers in a gRPC backend server group. This helps you use gRPC and HTTP/2 multiplexing to improve the throughput efficiency.

Solution Architecture

If your company deploys a gRPC application on an ECS in a region and creates a load balancer in the VPC where the ECS is running, you can add an HTTPS listener to the load balancer, create a gRPC backend server group, and add the ECS to this backend server group. Then the load balancer can route gRPC requests to this ECS, which processes these requests and returns responses back to clients.

Figure 4-4 Accessing the gRPC application through ELB

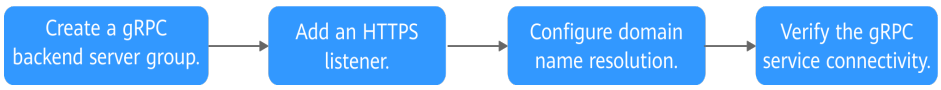


Preparations

- There is a dedicated load balancer with an EIP bound to it. If there is no such resource, you can **buy one** and **bind an IPv4 EIP to the load balancer**.
- You have either purchased a certificate or uploaded a third-party certificate to SSL Certificate Manager (SCM), and configured a public domain name for the certificate. It is recommended that you **purchase an SSL certificate** on the Cloud Certificate & Manager (CCM) console.
- You have purchased an ECS (ECS01) and deployed a gRPC application on it. For more information about gRPC, see **What Is gRPC?**

Procedure

Figure 4-5 Procedure for forwarding gRPC requests



Step 1: Create a gRPC Backend Server Group

1. Go to the **backend server group list page**.
2. Click **Create Backend Server Group** in the upper right corner.
3. Configure the parameters based on **Table 4-1** and retain the default values for other parameters.

Table 4-1 Parameters required for configuring a routing policy

Parameter	Example Value	Description
Backend Server Group Name	server_group_gRPC	Specifies the name of the backend server group.
Type	Dedicated	Specifies the type of the load balancer that can use the backend server group.
Load Balancer	Associate existing	Specifies whether to associate a load balancer now. Click **Associate existing** and select a load balancer you have created.
Backend Protocol	GRPC	Specifies the protocol that backend servers in the backend server group use to receive requests from the listeners. Select **GRPC**.
Load Balancing Algorithm	Weighted round robin	Specifies the load balancing algorithm used by the load balancer to distribute traffic. **Weighted round robin**: Requests are routed to different servers based on their weights. Backend servers with higher weights receive proportionately more requests, whereas equal-weighted servers receive the same number of requests. For more information, see **Load Balancing Algorithms**.

4. Click **Next** to add backend servers and configure health check.
5. Click **Add Cloud Server**, select **ECS01**, set a service port, and retain the default values for other parameters.

> ⚠ CAUTION
>
> The service port must be the same as the port used by the gRPC service. The security group of the backend server must allow traffic over this port.

6. Enable health check and retain the default values for other health check parameters.
7. Click **Next**.
8. Confirm the configurations and click **Create Now**.

Step 2: Add an HTTPS Listener

1. Go to the **load balancer list page**.

2. Locate the target load balancer and click **Add Listener** in the **Operation** column.
3. On the **Add Listener** page, set **Frontend Protocol** to **HTTPS** and enable **HTTP/2** in **More (Optional)**.

Figure 4-6 Configuring one-way authentication

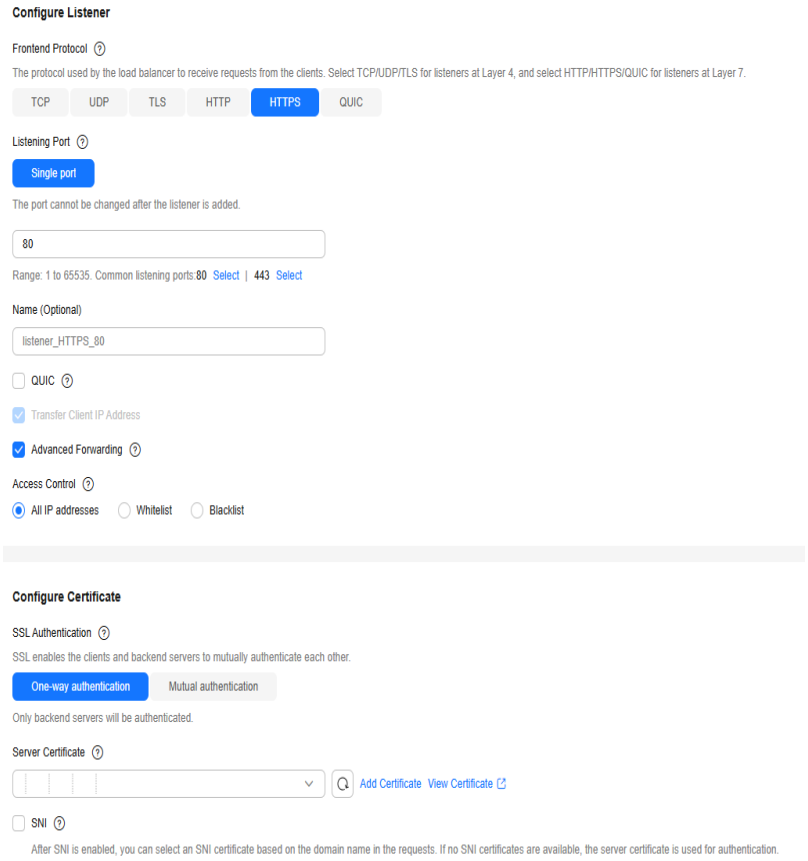
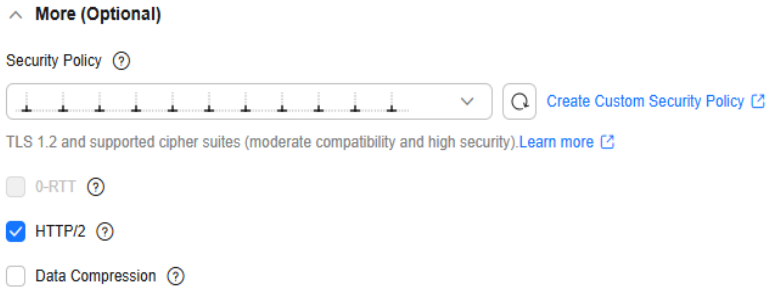


Figure 4-7 Enabling HTTP/2



4. Click **Next: Configure Request Routing Policy** and select **Use existing** for **Backend Server Group**. Select the gRPC backend server group created in **Step 1** and click **Next: Confirm**.
5. Confirm the configurations and click **Submit**.

Step 3: Configure Domain Name Resolution

You can add an A record set to resolve the domain name to the public IP address of the load balancer so that clients can access the load balancer using the public domain name.

The following provides an example for resolving a website domain name to an IPv4 address. For details about how to configure an A record set, see **Routing Internet Traffic to a Website**.

1. Go to the **DNS console**.
2. In the navigation pane on the left, choose **Public Zones**.

   The zone list is displayed.
3. Locate the public zone and click **Manage Record Sets** in the **Operation** column.
4. Click **Add Record Set**.
5. Configure the parameters based on **Table 4-2**.

   Table 4-2 Parameters for adding an A record set

Parameter	Example Value	Description
Type	A – Map domains to IPv4 addresses	Type of the record set. In this example, set it to **A - Map domains to IPv4 addresses**.
Name	www	Prefix of the domain name to be resolved.
Line	Default	Resolution line. The DNS server will return the IP address of the specified line, depending on where end users come from. The default value is **Default**. **Default**: returns the default resolution result irrespective of where the visitors come from.
TTL (s)	300	Cache duration of the record set on a local DNS server, in seconds. In this example, the default value 300 is used.
Value	192.168.12.2	IPv4 addresses mapped to the domain name. In this example, set this parameter to the EIPs bound to the load balancer.
Advanced Settings (Optional)	-	Click ⌄ to expand the advanced settings, set the alias and weight of the record set, and add a description and tags. In this example, the default settings are used.

6. Click **OK**.

7. Switch back to the **Record Sets** tab.
   The added record set is in the **Normal** state.

Step 4: Verify the gRPC Service Connectivity

After the preceding operations are done, the client can access the gRPC service deployed on the backend server through the load balancer. The browser cannot natively support the gRPC protocol's data frame format. You can perform the following steps to test the connectivity between the client and the gRPC service:

1. Remotely log in to the service ECS from the client ECS.
   Multiple methods are available for logging in to an ECS. For details, see **Logging In to an ECS**.
2. Install **grpcurl**, a command-line tool for interacting with gRPC service, on the client.
   a. Install the **grpcurl** software package using the following ways:
      - Run the yum command to install the **grpcurl** software package.
        ```
        yum install grpcurl
        ```
      - Manually install the grpcurl software package of a specific version.
        ```
        rpm -ivh grpcurl_1.9.3_linux_386.rpm
        ```
   b. Verify the installation.
      ```
      grpcurl --version
      ```

      Figure 4-8 Verifying the installation

      ```
      [root@elb- bin]# grpcurl --version
      grpcurl v1.9.3
      ```
3. Run the following command on the client to access the gRPC service on the backend server:
   grpcurl -insecure -proto <*.proto-file*> <*domain-name*>:<*listening-port*> <*gRPC-service-name*>/<*method*>
   If information similar to the following is displayed, the client can access the backend server where the gRPC service is deployed through ELB.

   Figure 4-9 Verifying the access to the gRPC service

   ```
   [root@elb-e2a 1 elb-test]# grpcurl -insecure -v -proto hello.proto .com:1082 hello.Hello/Say

   Resolved method descriptor:
   // Sends a greeting.
   rpc Say ( .hello.Request ) returns ( .hello.Response );

   Request metadata to send:
   (empty)

   Response headers received:
   content-type: application/grpc
   date: Sat, 17 May 2025 07:32:32 GMT
   server: elb

   Response contents:
   {
     "message": "Hello  I'm good"
   }

   Response trailers received:
   (empty)
   Sent 0 requests and received 1 response
   ```

4.3 Using WebSocket for Real-time Messaging

WebSocket is a network protocol that enables full-duplex communication over a single TCP connection. With WebSocket, servers can proactively send data to clients. This makes the connection between the client and server more persistent,

data exchange simpler, and communication more efficient. Application load balancers support WebSocket by default, helping you balance real-time communication traffic.

WebSocket Overview

- **What Is WebSocket?**

  As web applications are serving increasingly purposes, they demand better communication technologies. For example, social networking, collaborative office, and online customer service require real-time data push. However, the traditional round robin algorithm needs a client to send requests to a server at a fixed interval to obtain the latest data, which is not efficient for these needs. Clients need to frequently send requests with large HTTP headers but less useful information. These requests not only increase the load on the server but also waste substantial bandwidths.

  To address this issue, HTML5 introduces WebSocket. WebSocket defines a persistent two-way communication between servers and clients, meaning that both parties can exchange message data at the same time. This two-way communication cuts unnecessary messages, enhances real-time experience, and saves server and bandwidth costs.

  For more information about WebSocket, see **The WebSocket Protocol**.

  HTTP and HTTPS listeners support WebSocket by default.

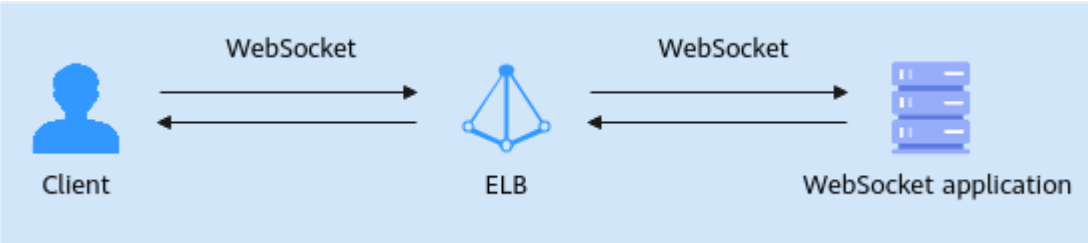
- **Websocket Application Scenarios**

  WebSocket enables **full-duplex real-time communication** and is widely used in service scenarios that require **high-frequency interaction** and **low latency**.

  - **Real-time social networking and interaction**: online chat rooms, live comments, and multiplayer online board & card gaming that require simultaneous interaction
  - **Collaborative office** and **online education**: multiuser applications with simultaneous editing and online classroom interaction
  - **Map navigation**: real-time push of passenger location changes and traffic congestion information
  - **Customer service and notification**: instant information exchange between users and customer service personnel

Solution Architecture

Figure 4-10 Accessing a WebSocket application through ELB



This practice provides a simple example of real-time messaging. A client sends a message to a server, which responds immediately. After getting the response, the

client sends another message, and the server responds once more. This is a real-time interaction process.

Preparations

- **Create a dedicated load balancer** (application load balancing) and **bind an IPv4 EIP to it**.
- **Create two ECSs** running CentOS in the same VPC as the load balancer. The first ECS (ECS_client) is used as the client to send HTTP requests, and the second ECS (ECS_server) is used as the backend server for deploying the WebSocket application.

Step 1: Deploy the WebSocket Application on the Backend Server

1. Remotely log in to the backend server ECS_server.

   Multiple methods are available for logging in to an ECS. For details, see **Logging In to an ECS**.

2. Run the following command to ensure that the Python version on the server is 3.7 or later:
   ```
   yum install python39
   ```
3. Install the latest websockets library, which provides a simple API to make it easy to establish WebSocket connections in Python.
   a. Install the Python package management tool pip.
      ```
      sudo yum install python3-pip
      ```
   b. Install WebSocket.
      ```
      pip install websockets
      ```
4. Create the websocket directory on the backend server.
   ```
   mkdir websocket
   ```
5. In the websocket directory, create a **websocket_server.py** file and deploy and test WebSocket application in the file.
   a. Create the **websocket_server.py** file.
      ```
      vi websocket_server.py
      ```
   b. Press **i** to enter editing mode.

      Example Code

      - The following shows the **websocket_server.py** script and the default port is 8081.
        ```
        import asyncio
        import websockets

        async def echo(websocket):
            print(websocket.request.path)
            try:
                async for message in websocket:
                    print(f"Received: {message}")
                    await websocket.send("Hello client,this is server!")
                    print(f"Sent: Hello client,this is server!")
            except websockets.exceptions.ConnectionClosed as e:
                print(f"Connection closed with error: {e}")
            except Exception as e:
                print(f"Unexpected error: {e}")

        async def main():
            print("Starting server...")
            start_server = await websockets.serve(echo, "0.0.0.0", 8081)
            print("Server started")
        ```



```
    while True:
        await asyncio.sleep(1)

asyncio.run(main())
```

6. Press **Esc** and enter **:wq** to save the **websocket_server.py** file.
7. Run the **websocket_server.py** file.
   ```
   python3 websocket_server.py
   ```
8. If the command output shown in **Figure 4-11** is displayed, the WebSocket application is deployed.

Figure 4-11 WebSocket application deployed

Step 2: Deploy the WebSocket Application on the Client

1. Remotely log in to ECS_client.

   Multiple methods are available for logging in to an ECS. For details, see **Logging In to an ECS**.
2. Run the following command to ensure that the Python version on the client is 3.7 or later:
   ```
   yum install python39
   ```
3. Run the following command to install the latest websockets library on the client:
   a. Install the Python package management tool pip.
      ```
      sudo yum install python3-pip
      ```
   b. Install WebSocket.
      ```
      pip install websockets
      ```
4. Create the **websocket_client** directory on the client server.
   ```
   mkdir websocket_client
   ```
5. In the websocket directory, create a **websocket_client.py** file and deploy the WebSocket application in the file.
   a. Create the **websocket_client.py** file.
      ```
      vi websocket_client.py
      ```
   b. Press **i** to enter editing mode.

      Example Code

      - The following shows the **websocket_client.py** script. It allows the client to access the EIP and port (8081) of the load balancer.
        ```
        import asyncio
        import websockets

        async def hello():
            uri = "ws://EIP_ELB:8081"
            try:
                async with websockets.connect(uri) as websocket:
                    while True:
                        await websocket.send("Hello server,this is client!")
                        print("Sent: Hello server,this is client!")
                        response = await websocket.recv()
                        print(f"Received: {response}")
                        await asyncio.sleep(1)
            except websockets.exceptions.ConnectionClosedError as e:
                print(f"Connection closed with error: {e}")
        ```



```
asyncio.get_event_loop().run_until_complete(hello())
```

6. Press **Esc** and enter **:wq** to save the **websocket_client.py** file.

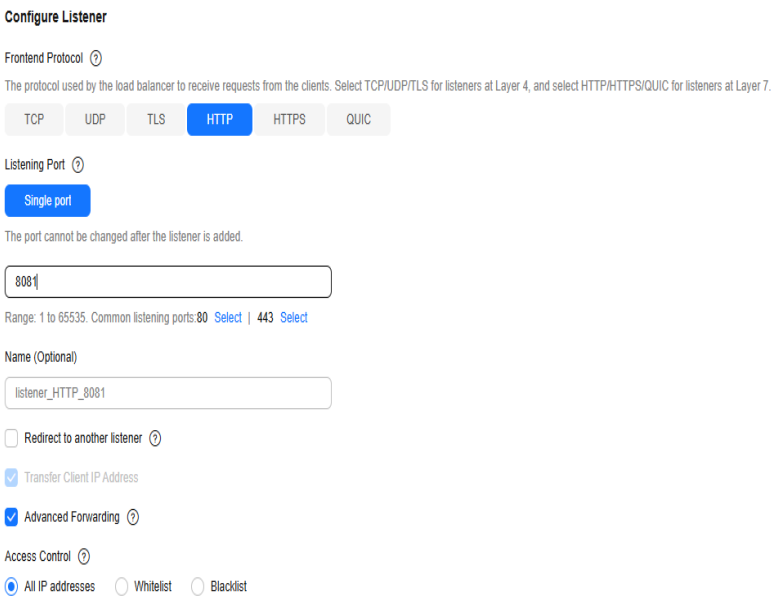
Step 3: Create an HTTP Backend Server Group and Add a Backend Server

1. Go to the **backend server group list page**.
2. Click **Create Backend Server Group** in the upper right corner.
3. On the **Configure Routing Policy** page, associate the backend server group **Server_Group** with your load balancer and set the backend protocol to HTTP.
4. Retain the default values for other parameters, and click **Next**.
5. On the **Add Backend Server** page, click **Add Cloud Server** on the **Cloud Servers** tab.
6. In the **Add Backend Server** dialog box, select **ECS_server** and set the service port to 8081.
7. Click **Next** and confirm the configuration.
8. Click **Create Now**.

Step 4: Add an HTTP Listener and Associate it with the Backend Server Group

1. Go to the **load balancer list page**.
2. Locate the target load balancer and click **Add Listener** in the **Operation** column.
3. On the **Add Listener** page, set the protocol to **HTTP** and port to **8081**.

Figure 4-12 Adding an HTTP listener



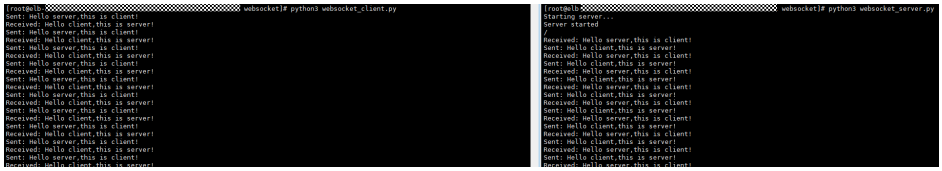
4. Click **Next: Configure Request Routing Policy** and select **Use existing** for **Backend Server Group**. Select the backend server group Server_Group created in **Step 3** and click **Next: Confirm**.

5. Confirm the configurations and click **Submit**.

Step 5: Verify the Real-time Messaging

1. Remotely log in to ECS_client.
2. Run the following command:
   ```
   python3 websocket_client.py
   ```
3. Check whether the client and backend server print "Sent: Hello server, this is client!" and "Received: Hello client, this is server!" respectively. If they do, the load balancer enables real-time messaging through WebSocket.

   Figure 4-13 Real-time messaging through the load balancer over WebSocket



4.4 Using a Dedicated Load Balancer to Forward Traffic by Port Ranges

Scenarios

If your service has dynamic ports or you need to listen to multiple ports, instead of configuring a fixed port for each listener, you can use a dedicated load balancer and enable **Forward by Port Ranges** to route traffic across backend servers over multiple ports or port ranges. This simplifies listener configurations and makes O&M easier.

Prerequisites

- There is a dedicated network load balancer with an EIP bound to it. If there is no such resource, you can **buy one** and **bind an IPv4 EIP to the load balancer**.
- There are two ECSs (ECS01 and ECS02), each hosting an application. The security group rules of ECS01 and ECS02 allow access over ports 30000 to 30005.

  Deployment Commands of ECS01

  – Log in to ECS01 and run the following command to create a script file:
    ```
    vi ECS01_install.sh
    ```
  – Enter the editing mode and copy the following content to the file:
    ```
    #!/bin/bash

    # Install Nginx and automatically configure multiple ports.
    yum install -y nginx

    for PORT in {30000..30005}; do
      # Generate an HTML file.
      echo "Hello World! This is ECS01, server port is $PORT." > /usr/share/nginx/html/index_
    $PORT.html
      # Generate the Nginx configuration.
      printf "server { listen $PORT; location / { root /usr/share/nginx/html; index index_
    ```



```
$PORT.html; } }\n" > /etc/nginx/conf.d/app_$PORT.conf
done

# Start Nginx and test it.
nginx -t && systemctl restart nginx
curl --parallel-max 11 $(for PORT in {30000..30005}; do echo "http://localhost:$PORT "; done)
```

- Enter **:wq!** to save the file.
- Run the following command to run the script file:

```
sudo sh ECS01_install.sh
```

- If the following information is displayed, the two ECSs can be accessed over ports 30000 to 30005:

```
Hello World! This is ECS01, server port is 30000.
Hello World! This is ECS01, server port is 30001.
Hello World! This is ECS01, server port is 30002.
Hello World! This is ECS01, server port is 30003.
Hello World! This is ECS01, server port is 30004.
Hello World! This is ECS01, server port is 30005.
```

Step 1: Create a Backend Server Group and Enable Forward to Same Port

In this practice, a backend server group that supports **Forward to Same Port** is used. If this option is enabled, you do not need to specify a backend port when you add a backend server. The listener routes requests to the backend server over the same port as the frontend port.

1. Go to the **backend server group list page**.
2. Click **Create Backend Server Group** in the upper right corner.
3. Configure the parameters based on **Table 4-3** and retain the default values for other parameters.

Table 4-3 Parameters required for configuring a routing policy

Parameter	Example Value	Description
Backend Server Group Name	server_group	Specifies the name of the backend server group.
Type	Dedicated	Specifies the type of load balancer that can use the backend server group.
Load Balancer	Associate existing	Specifies whether to associate a load balancer. Click **Associate existing** and select a load balancer you have created.
Backend Protocol	TCP	Specifies the protocol that backend servers in the backend server group use to receive requests from the listeners. Select **TCP**.

Parameter	Example Value	Description
Forward to Same Port	Enable it.	Specifies whether to enable the forward to same port option. After you enable it, you do not need to specify a backend port when you add a backend server. The listener routes requests to the backend server over the same port as the frontend port. This option cannot be disabled after being enabled.
Load Balancing Algorithm	Weighted round robin	Specifies the load balancing algorithm used by the load balancer to distribute traffic. **Weighted round robin**: Requests are routed to different servers based on their weights. Backend servers with higher weights receive proportionately more requests, whereas equal-weighted servers receive the same number of requests. For more information, see **Load Balancing Algorithms**.

4. Click **Next** to add backend servers and configure health check.
5. Click **Add Cloud Server**, select **ECS01** and **ECS02**, and retain the default values for other parameters.
6. Enable **Health Check**. The ECSs do not have default backend ports because **Forward to Same Port** is enabled for the backend server group. You only need to configure a health check port.

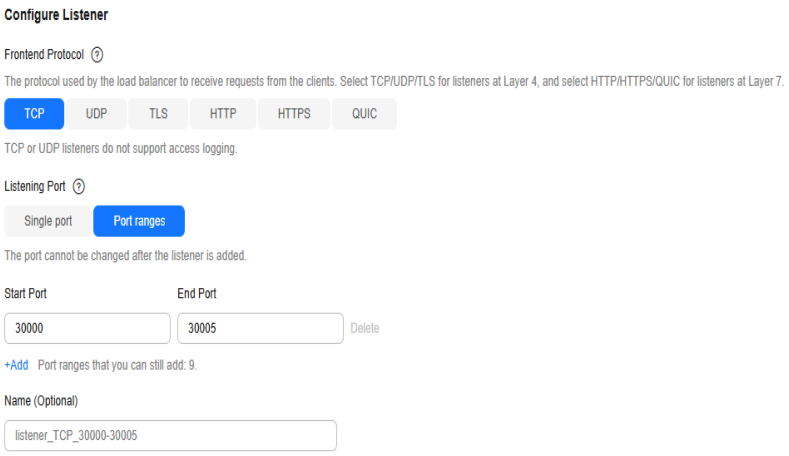
   In this practice, set the health check port to 80 and retain the default values for other health check parameters.
7. Click **Next**.
8. Confirm the configuration and click **Create Now**.

Step 2: Add a TCP Listener and Enable Forwarding by Port Ranges

This practice uses a TCP listener as an example to distribute traffic by port ranges.

1. Go to the **load balancer list page**.
2. Locate the target load balancer and click **Add Listener** in the **Operation** column.
3. On the displayed page, set **Frontend Protocol** to **TCP**, enable **Forwarding by Port Ranges**, and set the port range to 30000-30005.

Figure 4-14 Adding a TCP listener and selecting port ranges



4. Click **Next: Configure Request Routing Policy** and configure the backend server group.

   Click **Use Existing** and select the backend server group created in **Step 1: Create a Backend Server Group and Enable Forward to Same Port**.

5. Click **Next: Confirm**, confirm the configurations, and submit your request.

Step 3: Configure Domain Name Resolution

You can add an A record set to resolve the domain name to the public IP address of the load balancer so that clients can access the load balancer using the public domain name.

For details about how to configure A record sets, see **Routing Internet Traffic to a Website**.

1. Go to the **DNS console**.
2. In the navigation pane on the left, choose **Public Zones**.

   The zone list is displayed.

3. Locate the zone and click **Manage Record Sets** in the **Operation** column.
4. Click **Add Record Set**.
5. Configure the parameters based on **Table 4-4**.

Table 4-4 Parameters for adding an A record set

Parameter	Example Value	Description
Type	A – Map domains to IPv4 addresses	Type of the record set. In this example, set it to **A - Map domains to IPv4 addresses**.
Name	www	Prefix of the domain name to be resolved.

Parameter	Example Value	Description
Line	Default	Resolution line. The DNS server will return the IP address of the specified line, depending on where visitors come from. The default value is **Default**. **Default**: returns the default resolution result irrespective of where the visitors come from.
TTL (s)	300	Cache duration of the record set on a local DNS server, in seconds. In this example, the default value 300 is used.
Value	192.168.12.2	IPv4 addresses mapped to the domain name. In this example, set this parameter to the EIPs bound to the load balancer.
Advanced Settings (Optional)	-	Click ⌄ to expand the advanced settings, set the alias and weight of the record set, and add a description and tags. In this example, the default settings are used.

6. Click **OK**.
7. Switch back to the **Record Sets** tab.

   Check the record you just added in the record set list. If its status is **Normal**, the record set is added.

Step 4: Verifying Forwarding Traffic by Port Ranges

- Testing Load Balancer Availability

  a. Use any Linux client that can access the public network as an example. Run **curl** *<domain-name> <any-port-between-30000-and-30005>* multiple times. If information similar to "**Hello World! This is ECS01, server port is** *specific-port-number*" is displayed, the load balancer can forward requests to the backend server.

  Figure 4-15 Linux client requests distributed to ECS01

  ```
  Hello World! This is ECS01, server port is 30000.
  ```

  b. Access the domain name and any port number between 30000 and 30005 using a browser, for example, **http://***domain-name***:30000**. If information similar to the following figure is displayed, the client can access the application.

  Figure 4-16 Browser requests distributed to ECS01

  Hello World! This is ECS01, server port is 30000.

- Simulating a Service Fault
  a. Run **systemctl stop nginx.service** to disable the application running on ECS01.

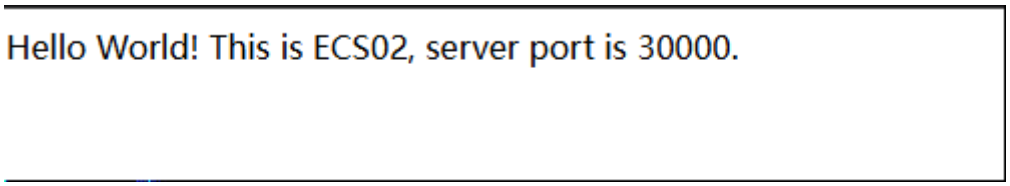
     Wait for several minutes and run **curl** *<domain-name> <any-port-between-30000-and-30005>* on the client again. The information similar to the following figure is still displayed.

     Figure 4-17 Client requests distributed to ECS02

  b. Access the domain name and any port number between 30000 and 30005 using a browser, for example, **http://***domain-name***:30000**. If information similar to the following figure is displayed, the client can access the application.

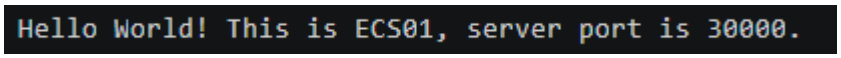
     Figure 4-18 Browser requests distributed to ECS02

  c. Run **systemctl start nginx.service** to enable the application on ECS01 and **systemctl stop nginx.service** to disable the application on ECS02.

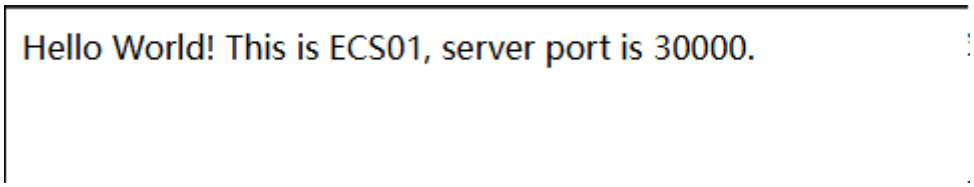
     Wait for several minutes and run **telnet** *<domain-name> <any-port-between-30000-and-30005>* on the client again. The information similar to the following figure is still displayed.

     Figure 4-19 Linux client requests distributed to ECS01

  d. Access the domain name and any port number between 30000 and 30005 using a browser, for example, **http://***domain-name***:30000**. If information similar to the following figure is displayed, the client can access the application.

     Figure 4-20 Browser requests distributed to ECS01

  e. The preceding test result shows that ELB can route requests to the healthy backend server over any port between 30000 and 30005, if a single backend server fails.

4.5 Enabling IPv4/IPv6 Translation to Enable IPv6 Clients to Access IPv4 Services

Scenarios

If your Layer 4 service is running in an IPv4 network but needs to be accessed by IPv6 clients, you can use IPv4/IPv6 translation to enable IPv6 clients to access your IPv4 service, without having to deploy your service on an IPv6 network.

NOTE

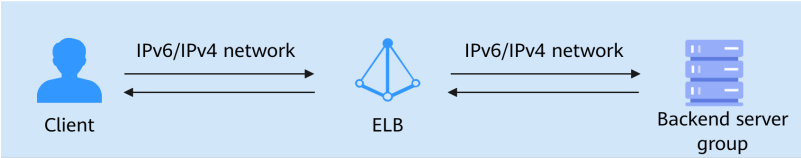
IPv4/IPv6 Translation is available in certain regions. You can see which regions support this option on the console. If you want to use this feature, **submit a service ticket**.

What Is IPv4/IPv6 Translation?

ELB supports **IPv4/IPv6 translation**, which works with both **NAT64 and NAT46**. Clients accessing either the IPv4 or IPv6 address of a load balancer can communicate with IPv4 or IPv6 backend servers.

- **NAT64**: converts IPv6 traffic into IPv4 traffic. After IPv4/IPv6 translation is enabled, IPv6 clients can access IPv4 servers through the load balancer.
- **NAT46**: converts IPv4 traffic into IPv6 traffic. After IPv4/IPv6 translation is enabled, IPv4 clients can access IPv6 servers through the load balancer.

Figure 4-21 Service architecture with IPv4/IPv6 translation enabled for TCP and UDP listeners

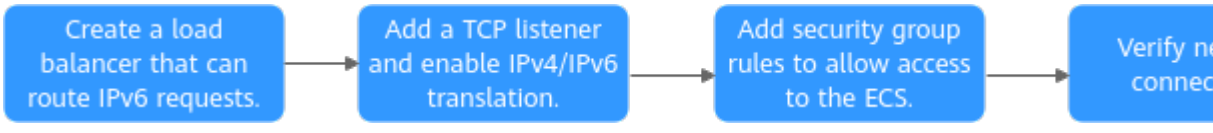


Prerequisites

- You have purchased an ECS (ECS01). The primary network interface of the ECS uses an IPv4 address. You have deployed an application on ECS01. For details, see **Deploy the Application**.
- There is a TCP or UDP backend server group. This practice uses a TCP backend server group as an example, with ECS01 in it.

Procedure

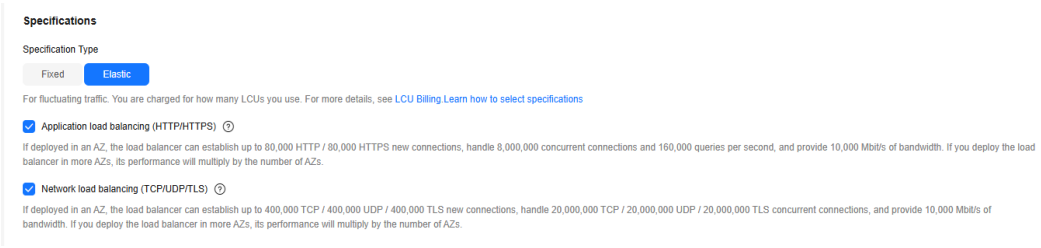
Figure 4-22 Procedure for enabling IPv6 clients to access IPv4 services



Step 1: Create a Load Balancer That Can Route IPv6 Requests

1. Go to the **Buy Elastic Load Balancer** page.
2. Complete the basic configuration of the load balancer as prompted. For example, select **Network load balancing (TCP/UDP/TLS)** for **Specifications**.

   Figure 4-23 Creating a network load balancer (Dedicated)

3. Configure the network as prompted.

   Select **IPv6 network** for **Network Type**. The VPC subnet must support IPv6.

   Figure 4-24 Selecting IPv6 network

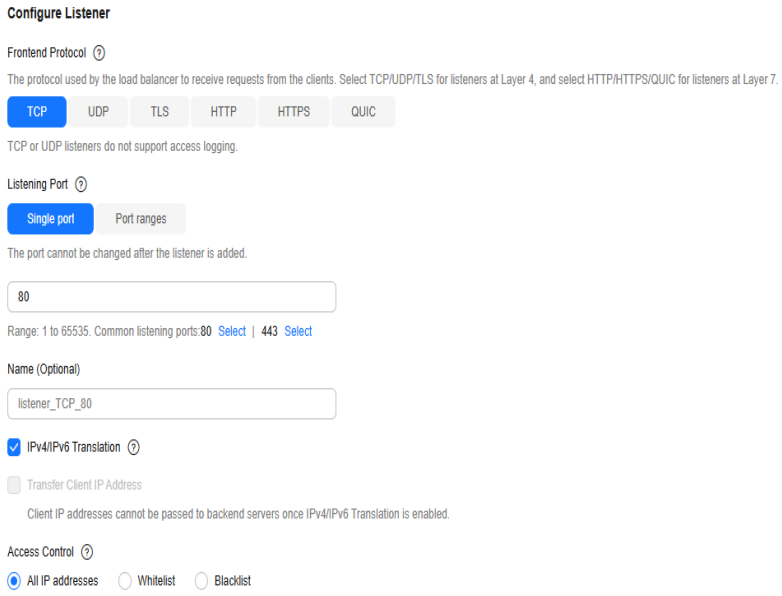
   

4. Confirm the information, click **Buy Now**.

Step 2: Add a TCP Listener and Enable IPv4/IPv6 Translation

1. Go to the **load balancer list page**.
2. Locate the target load balancer and click **Add Listener** in the **Operation** column.
3. On the **Add Listener** page, set **Frontend Protocol** to **TCP** and enable **IPv4/IPv6 Translation**.

Figure 4-25 Adding a TCP listener and enabling IPv4/IPv6 translation



4. Click **Next: Configure Request Routing Policy** and select **Use existing** for **Backend Server Group**. Select an existing backend server group and click **Next: Confirm**.
5. Confirm the configurations and click **Submit**.

Step 3: Add Security Group Rules to Allow Access

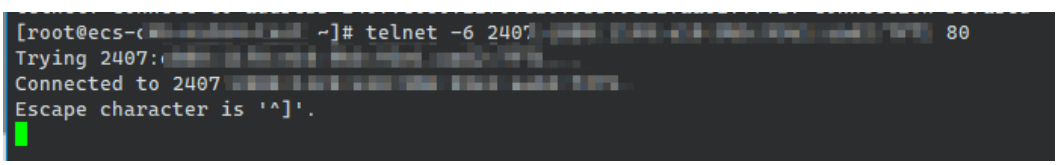
IPv6 clients use IP addresses in the backend subnet of the load balancer to access ECS01. So the security group of ECS01 must have rules to allow access from the backend subnet of the load balancer. For details, see **Configuring Security Group Rules for Backend Servers**.

Step 4: Verify the Connectivity Between the IPv6 Client and the IPv4 Backend Server

- Run the following command on the IPv6 client to access the IPv6 address of the load balancer:
  telnet -6 ${*IPv6-address*} ${*port-number*}

  If information similar to the following is displayed, the client can access the IPv6 address.

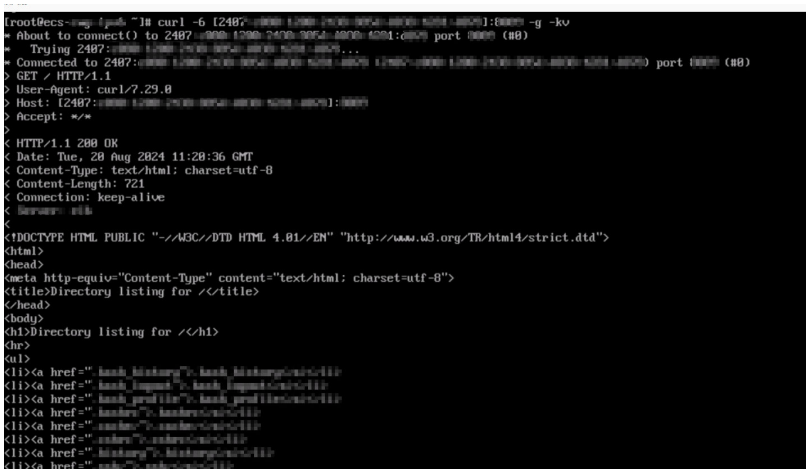
  Figure 4-26 Successful IPv6 client access to the IPv6 address of the load balancer

- Run the curl command to verify the connectivity of the web application.
  curl -6 [${IPv6-address-of-the-load-balancer}]:${listening-port-of-the load-balancer} -g -v

  If information similar to the following is displayed, the web application is reachable.

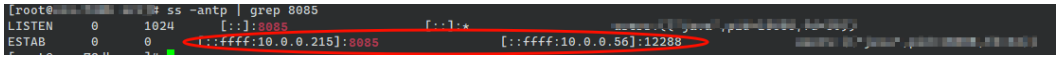
Figure 4-27 Verifying the connectivity of the web application



- Run the following command to determine whether the IPv6 client can access the backend server:
  ss -antp |grep ${*backend-server-port*}

  If information similar to the following is displayed, the IPv6 client can access the backend server. This means that the load balancer translates the source IPv6 address into an IPv4 address in its backend subnet and routes the request to ECS01.

Figure 4-28 Successful IPv6 client access to the IPv4 backend server



Reference

After IPv4/IPv6 translation is enabled, **Transfer Client IP Address** does not work. If TCP listeners are used, you can obtain client IP addresses by referring to **Using ProxyProtocol to Transfer Client IP Addresses**.

4.6 Using a Dedicated Load Balancer at Layer 4 to Transfer Client IP Addresses

Scenarios

When you are using ELB to distribute traffic, you may need to obtain the real IP addresses of clients for further analysis, especially in typical service scenarios such as security, data analysis, user behavior analysis, and troubleshooting.

When forwarding Layer 4 requests, load balancers communicate with backend servers using the client IP addresses by default. However, in certain cases, such as when a load balancer communicates with IP as backend servers, when IPv4/IPv6 translation is enabled for TCP and UDP listeners, or when TLS listeners are used for forwarding traffic, client IP addresses are translated by the load balancer. You can refer to this section to obtain client IP addresses.

Constraints

- If you are using a NAT gateway, you cannot obtain the IP addresses of the clients.
- If the client is a container, you can obtain only the IP address of the node where the container is located, but cannot obtain the IP address of the container.
- **Transfer Client IP Address** is enabled by default for TCP and UDP listeners. A cloud server cannot be used as a backend server and a client at the same time.

  If this happens, the backend server will think the packet from the client is sent by itself and will not return a response packet to the load balancer.

Transferring Client IP Addresses

Transfer Client IP Address is enabled by default for TCP and UDP listeners of dedicated load balancers. Load balancers communicate with backend servers using client IP addresses.

In some special cases, **Transfer Client IP Address** does not work. You can obtain client IP addresses by referring to **Table 4-5**.

Table 4-5 Transferring client IP addresses

Listener Protocol	Transfer Client IP Address	Transferring Client IP Addresses in Special Cases
TCP	**Transferring Client IP Addresses When TCP or UDP Listeners Are Used**	**Transfer Client IP Address** does not work in the following scenarios. You can **configure the TOA plug-in** or **use ProxyProtocol** to obtain the client IP addresses. • TCP listeners communicate with IP as backend servers. • IPv4/IPv6 translation is enabled for TCP listeners. In this case, client IP addresses are translated.
UDP	**Transferring Client IP Addresses When TCP or UDP Listeners Are Used**	Client IP addresses cannot be obtained in the following scenarios: • Load balancers communicate with IP as backend servers. • IPv4/IPv6 translation is enabled for UDP listeners.
TLS	Not supported	**Using ProxyProtocol to Transfer Client IP Addresses**

Transferring Client IP Addresses When TCP or UDP Listeners Are Used

When TCP and UDP listeners are used to forward traffic, load balancers communicate with backend servers using the client IP addresses by default.

Without any additional operations, you can check the backend server logs to determine whether the client IP address is obtained.

For a Nginx server, perform the following steps:

1. Run the following command to modify the HTTP configuration block and configure access logs for the Nginx server:

```
http {
    log_format main '$remote_addr- $remote_user [$time_local] "$request" '
                '$status $body_bytes_sent "http_referer" '
                '"$http_user_agent" "$http_x_forwarded_for"';
    }
```

Figure 4-29 Configuring the access log

2. Check the Nginx access logs to obtain the client IP address.

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

In the log, the first IP address is the real IP address of the client.

Figure 4-30 Viewing access logs of a Nginx server

Configuring the TOA Plug-in to Transfer Client IP Addresses

You can install the **TCP Option Address (TOA)** kernel on the backend servers of a load balancer to extract client IPv4 addresses.

Using ProxyProtocol to Transfer Client IP Addresses

You can enable **ProxyProtocol** on listeners and ensure that the **backend servers can parse ProxyProtocol** to transfer client IP addresses.

- **Transfer Client IP Address** does not work for **TCP listeners** in the following scenarios:
  - TCP listeners communicate with IP as backend servers.
  - IPv4/IPv6 translation is enabled for TCP listeners. In this case, client IP addresses are translated.
- You can use ProxyProtocol to transfer client IP addresses when you are using **TLS listeners** to forward requests.

WARNING

Ensure that the backend server can translate the proxy protocol. If they do not, services may be interrupted.

For details, see the following procedure.

Step 1: Enable ProxyProtocol On the Listener

1. Go to the **load balancer list page**.
2. On the displayed page, locate the load balancer and click its name.
3. On the listener list page, click the name of the target listener.
4. On the **Summary** tab, click **Edit** on the top right.
5. On the **Edit** page, enable **ProxyProtocol**.
6. Click **OK**.

Step 2: Configure Backend Servers to Support Proxy Protocol

The following uses a backend server that runs CentOS 7.5 as an example to describe how to install Nginx on the server.

1. Run the following commands to install **http_realip_module**:

```
yum -y install gcc pcre pcre-devel zlib zlib-devel openssl openssl-devel
wget http://nginx.org/download/nginx-1.17.0.tar.gz
tar zxvf nginx-1.17.0.tar.gz
cd nginx-1.17.0
./configure --prefix=/path/server/nginx --with-http_stub_status_module --without-http-cache --with-http_ssl_module --with-http_realip_module
make
make install
```

2. Open the Nginx configuration file **nginx.conf**.

```
vi /path/server/nginx/conf/nginx.conf
```

3. Modify the server configuration block.

   a. Enable Proxy Protocol listening.

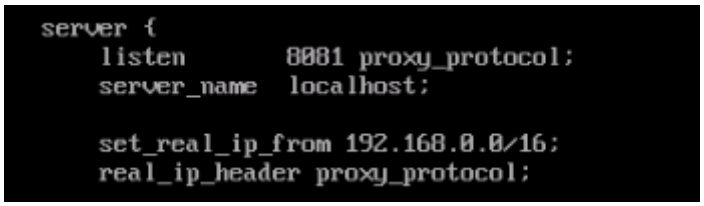
   b. Configure real IP address extraction.

```
server {
  listen 8081 proxy_protocol;
  server_name localhost;

  set_real_ip_from 192.168.0.0/16;
  real_ip_header proxy_protocol;
 }
```

   Enter the CIDR block of the proxy server as the value of **set_real_ip_from**. For dedicated load balancers, enter the CIDR block of its backend subnet.

   Figure 4-31 Modifying the server configuration block in the Nginx configuration file



4. Modify the http configuration block and configure access logs.

```
http {
   log_format main '$remote_addr- $remote_user [$time_local] "$request"'
             '$status $body_bytes_sent "$http_referer" '
             '"$http_user_agent" "$proxy_protocol_addr" ';

}
```


Figure 4-32 Modifying the http configuration block in the Nginx configuration file

5. Start Nginx.
```
/path/server/nginx/sbin/nginx
```

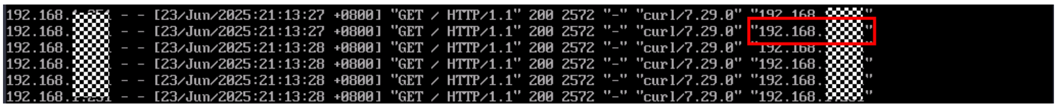
Step 3: Verify Transferring Client IP Addresses to the Backend Server

Run the following command to check the client IP addresses in the access logs:

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

In the logs, the IP addresses in the **$proxy_protocol_addr** header are the client IP addresses.

Figure 4-33 Transferring the Client IP address using ProxyProtocol



4.7 Using a Dedicated Load Balancer at Layer 7 to Transfer Client IP Addresses

Scenarios

When you are using ELB to distribute traffic, you may need to obtain the real IP addresses of clients for further analysis, especially in typical service scenarios such as security, data analysis, user behavior analysis, and troubleshooting. However, when you are using HTTP, HTTPS, or QUIC listeners of a load balancer to forward traffic, client IP addresses are translated when passing through the load balancer. This section describes how to obtain client IP addresses.

Constraints

- If you are using a NAT gateway, you cannot obtain the IP addresses of the clients.
- If the client is a container, you can obtain only the IP address of the node where the container is located, but cannot obtain the IP address of the container.

Transferring Client IP Addresses

- **Transfer Client IP Address** is enabled by default for HTTP, HTTPS, and QUIC listeners of dedicated load balancers, which means that client IP addresses can be placed in the X-Forwarded-For header and transferred to backend servers.
- You can configure the backend servers to ensure that they can correctly parse the X-Forwarded-For header to obtain client IP addresses.

  The X-Forwarded-For header is in the following format:

```
X-Forwarded-For: <client-IP-address>, <proxy-server-1-IP-address>, <proxy-server-2-IP-address>, ...
```

The first IP address included in the X-Forwarded-For header is the client IP address.

Preparations

- There is a dedicated load balancer with an EIP bound to it. If there is no such resource, you can **buy one** and **bind an IPv4 EIP to the load balancer**.
- There is an HTTP backend server group with an ECS (ECS01) running in it. The ECS hosts an application.

Step 1: Enable Transfer Client IP Address for a Layer 7 Listener

Transfer Client IP Address is enabled by default for HTTP, HTTPS, and QUIC listeners of dedicated load balancers, which means that client IP addresses can be placed in the X-Forwarded-For header and transferred to the backend servers.

Step 2: Configure Backend Servers

Configure the servers based on the backend server type to ensure that they can parse the X-Forwarded-For header.

Configuring Nginx Servers

For example, if CentOS 7.5 is used as the OS, run the following command to install the software:

1. Run the following commands to download the **Nginx source code package** and install **http_realip_module**:
   ```
   yum -y install gcc pcre pcre-devel zlib zlib-devel openssl openssl-devel
   wget http://nginx.org/download/nginx-1.17.0.tar.gz
   tar zxvf nginx-1.17.0.tar.gz
   cd nginx-1.17.0
   ./configure --prefix=/path/server/nginx --with-http_stub_status_module --without-http-cache --with-http_ssl_module --with-http_realip_module
   make
   make install
   ```
2. Run the following command to open the **nginx.conf** file:
   ```
   vi /path/server/nginx/conf/nginx.conf
   ```
3. Press **i** to enter the editing mode.
4. Modify the server configuration block as follows:
   ```
   set_real_ip_from 192.168.0.0/16;
   real_ip_header X-Forwarded-For;
   ```

Figure 4-34 Adding information

```
server {
    listen       80;
    server_name  localhost;

    set_real_ip_from 192.168.0.0/16;
    real_ip_header X-Forwarded-For;
```


NOTE

You need to enter the CIDR block of the proxy server as the value of **set_real_ip_from**. For dedicated load balancers, enter the CIDR block of its backend subnet.

5. Run the following command to modify the HTTP configuration block and, configure access logs for Nginx servers, and transfer the client request information in the X-Forwarded-For header to the access log through **http_x_forwarded_for**:

```
http {
    log_format main '$remote_addr- $remote_user [$time_local] "$request" '
                '$status $body_bytes_sent "http_referer" '
                '"$http_user_agent" "$http_x_forwarded_for"';
    }
```

Figure 4-35 Configuring the access log

```
http {
    log_format  main  '$remote_addr - $remote_user [$time_local] "$request" '
                      '$status $body_bytes_sent "$http_referer" '
                      '"$http_user_agent" "$http_x_forwarded_for"';
```

6. Start Nginx.

```
/path/server/nginx/sbin/nginx
```

Configuring Tomcat Servers

In the following operations, the Tomcat installation path is **/usr/tomcat/tomcat8/**.

1. Log in to a server on which Tomcat is installed.
2. Check whether Tomcat is running properly.

```
ps -ef|grep tomcat
netstat -anpt|grep java
```

Figure 4-36 Tomcat running properly

```
[root@lilian apache-tomcat-9.0.10]# ps -ef |grep tomcat
root      1009   995  0 15:01 pts/0    00:00:00 grep --color=auto tomcat
root     32223     1  0 14:37 pts/0    00:00:12 /usr/java/jdk-10.0.1/bin/java -Djava.util.logging.config.file=/usr/local/tomc
t-9.0.10/conf/logging.properties -Djava.util.logging.manager=org.apache.juli.ClassLoaderLogManager -Djdk.tls.ephemeralDHKeySi
tocol.handler.pkgs=org.apache.catalina.webresources -Dorg.apache.catalina.security.SecurityListener.UMASK=0027 -Dignore.endor
th /usr/local/tomcat-01/apache-tomcat-9.0.10/bin/bootstrap.jar:/usr/local/tomcat-01/apache-tomcat-9.0.10/bin/tomcat-juli.jar
sr/local/tomcat-01/apache-tomcat-9.0.10 -Dcatalina.home=/usr/local/tomcat-01/apache-tomcat-9.0.10 -Djava.io.tmpdir=/usr/local/
tomcat-9.0.10/temp org.apache.catalina.startup.Bootstrap start
[root@lilian apache-tomcat-9.0.10]# netstat -anpt|grep java
tcp        0      0 127.0.0.1:32001         0.0.0.0:*               LISTEN      882/java
tcp6       0      0 :::8020                 :::*                    LISTEN      32223/java
tcp6       0      0 :::8888                 :::*                    LISTEN      32223/java
tcp6       0      0 127.0.0.1:8006          :::*                    LISTEN      32223/java
tcp6       0      0 10.0.0.20:8888          100.125.134.52:38390    ESTABLISHED 32223/java
tcp6       0      0 127.0.0.1:31001         127.0.0.1:32001         ESTABLISHED 882/java
tcp6       0      0 10.0.0.20:8888          100.125.134.53:57771    ESTABLISHED 32223/java
tcp6       0      0 10.0.0.20:8888          100.125.134.46:62833    ESTABLISHED 32223/java
tcp6       0      0 10.0.0.20:8888          100.125.19.50:58124     ESTABLISHED 32223/java
tcp6       0      0 10.0.0.20:8888          100.125.19.47:49597     ESTABLISHED 32223/java
tcp6       1      0 10.0.0.20:50648         100.125.15.62:80        CLOSE_WAIT  882/java
tcp6       0      0 10.0.0.20:8888          100.125.19.53:27108     ESTABLISHED 32223/java
```

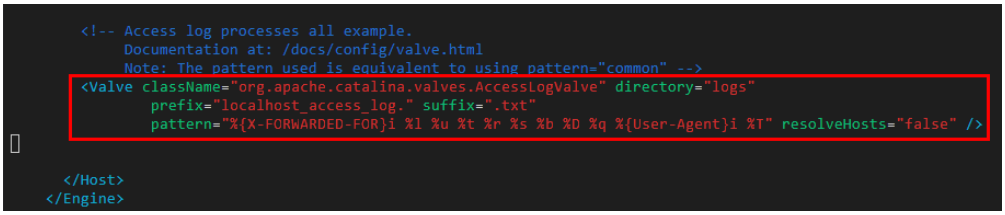
3. Modify **className="org.apache.catalina.valves.AccessLogValve"** in the **server.xml** file as follows:
   a. Run the following command to open the **server.xml** file:

```
vim /usr/tomcat/tomcat8/conf/server.xml
```

   b. Modify the file as follows:

```
<Valve className="org.apache.catalina.valves.AccessLogValve" directory="logs"
prefix="localhost_access_log." suffix=".txt"
pattern="%{X-FORWARDED-FOR}i %l %u %t %r %s %b %D %q %{User-Agent}i %T"
resolveHosts="false" />
```


Figure 4-37 Example configuration



4. Restart the Tomcat service.

```
cd /usr/tomcat/tomcat8/bin && sh shutdown.sh && sh startup.sh
```

/usr/tomcat/tomcat8/ is where Tomcat is installed. Change it based on site requirements.

Figure 4-38 Restarting the Tomcat service

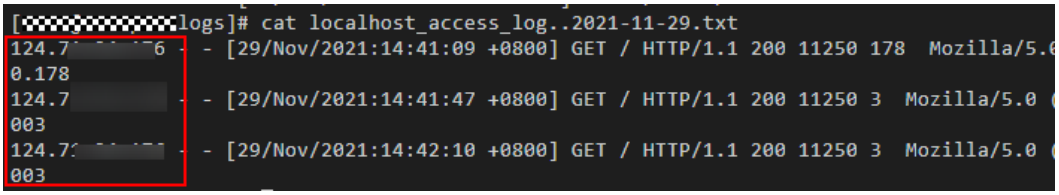
5. View the latest logs.

   As highlighted in the following figure, IP addresses shown in the red box are the client IP addresses.

```
cd /usr/tomcat/tomcat8/logs/
cat localhost_access_log.2021-11-29.txt
```

   In this command, **localhost_access_log.2021-11-29.txt** indicates the log path of the current day. Change it based on site requirements.

Figure 4-39 Checking the client IP addresses



Configuring Windows Servers with IIS Deployed

The following uses Windows Server 2012 with IIS7 as an example to describe how to obtain the client IP address.

1. Download and install IIS.
2. Download the **F5XForwardedFor.dll** plug-in and copy the plug-ins in the **x86** and **x64** directories to a directory for which IIS has the access permission, for example, **C:\F5XForwardedFor2008**.
3. Open Internet Information Services (IIS) Manager and choose **Modules** > **Configure Native Modules**.

Figure 4-40 Selecting **Modules**

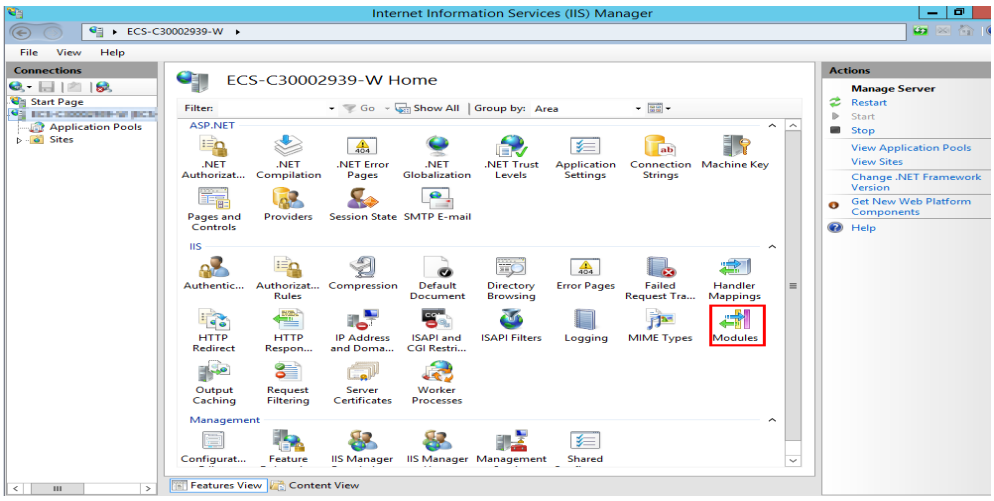
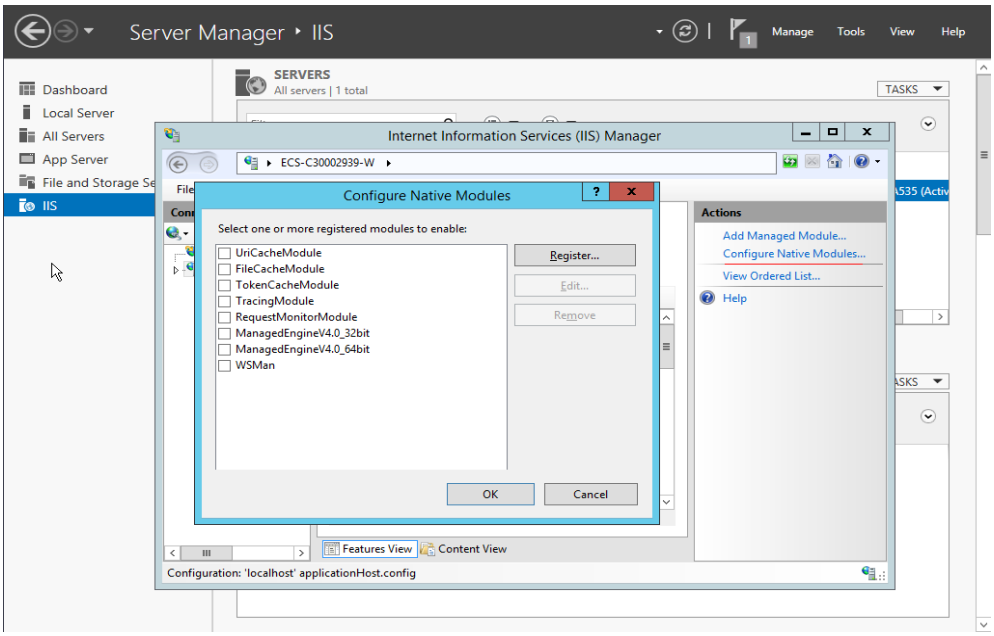
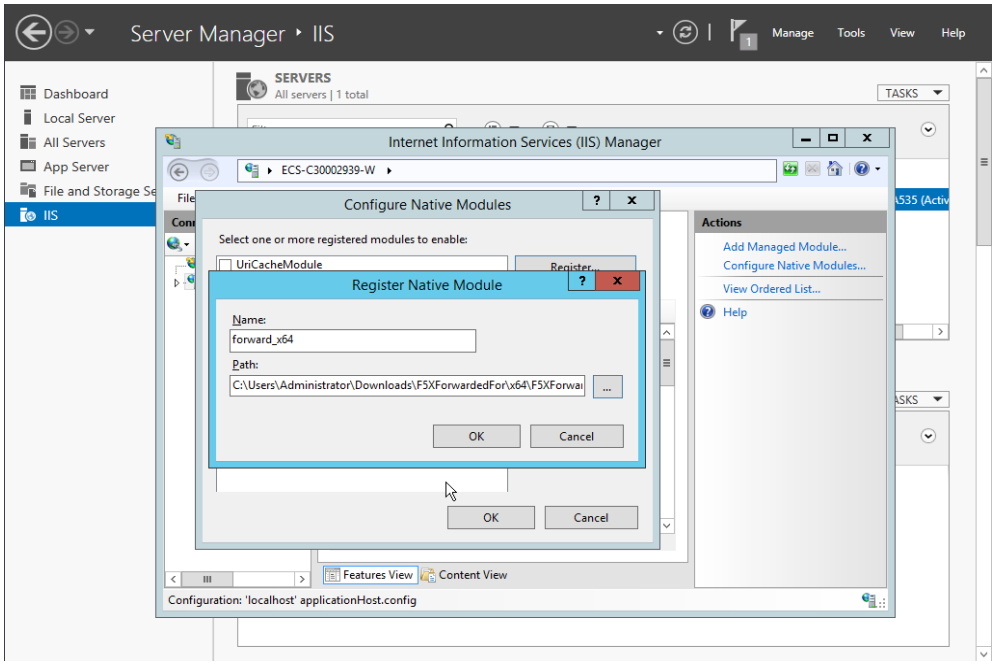


Figure 4-41 Configure Native Modules



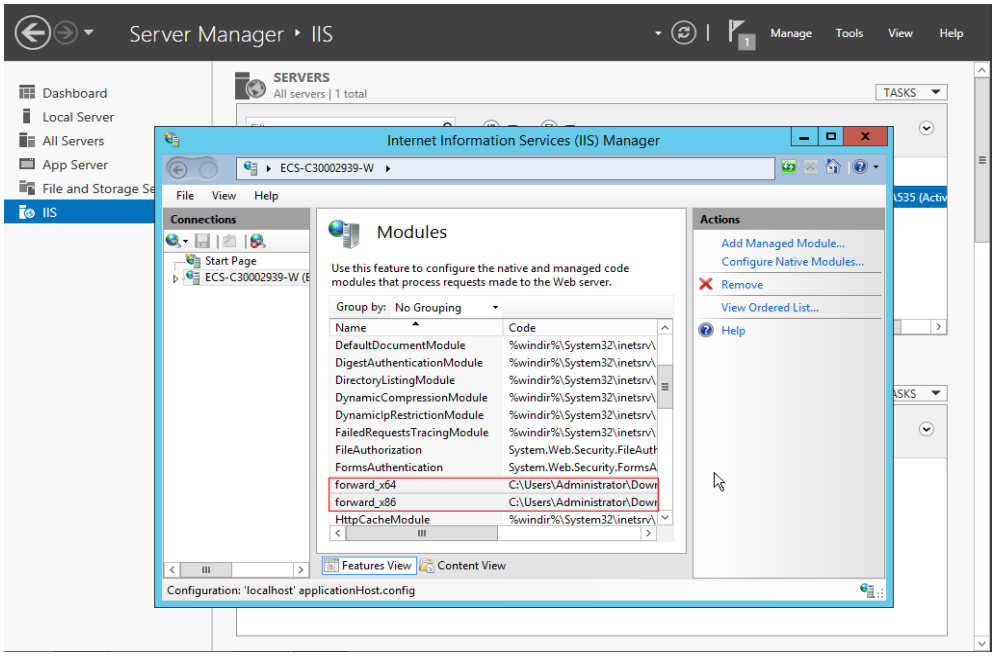
4. Click **Register** to register the x86 and x64 plug-ins.

Figure 4-42 Registering plug-ins



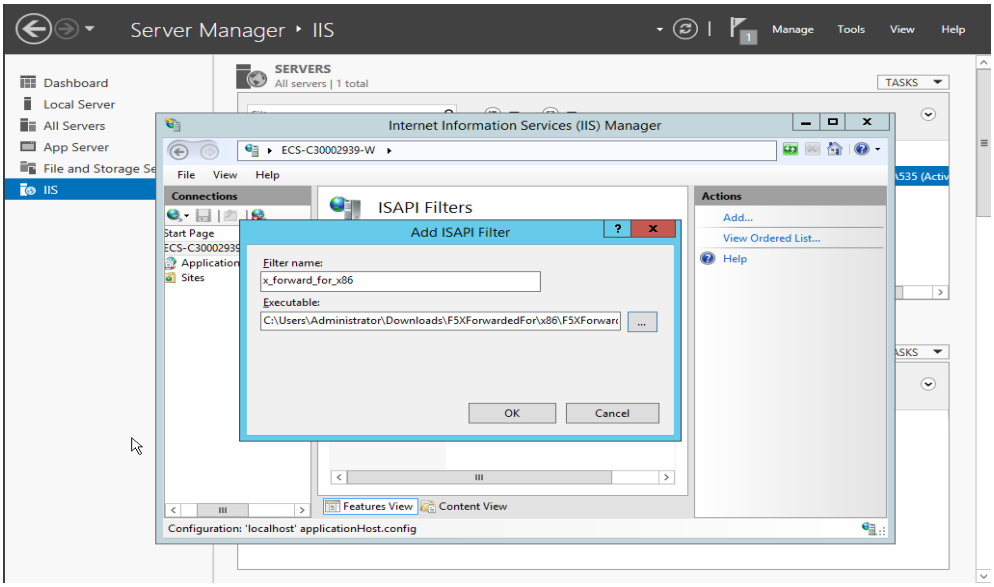
5. In the **Modules** dialog box, verify that the registered plug-ins are displayed in the list.

Figure 4-43 Confirming the registration



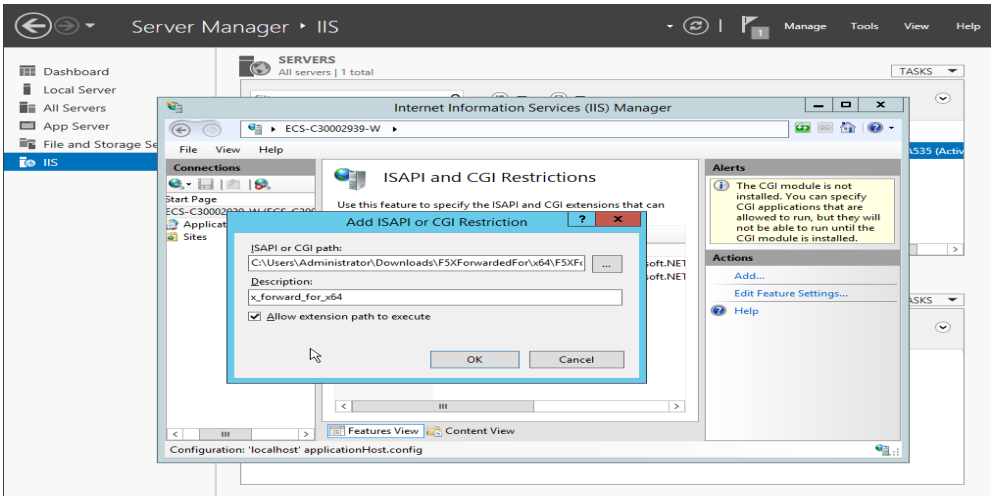
6. Select **ISAPI Filters** on the Internet Information Services (IIS) Manager homepage and authorize two plug-ins to run ISAPI and CGI extensions.

Figure 4-44 Adding authorization



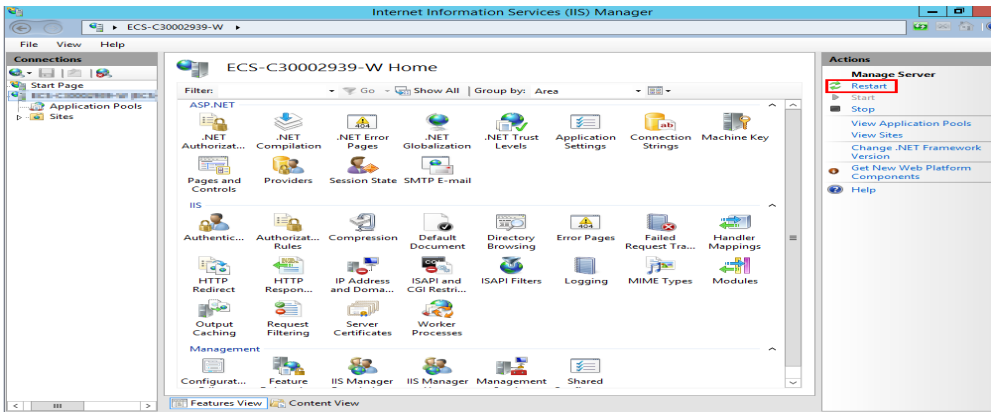
7. Select **ISAPI and CGI Restrictions** to set the execution permission for the two plug-ins.

Figure 4-45 Allowing the plug-ins to execute



8. Click **Restart** on the homepage to restart IIS. The configuration will take effect after the restart.

Figure 4-46 Restarting IIS



Step 3: Verify Transferring Client IP Addresses to the Backend Server

Run the following command to view the access logs of a Nginx server and check the client IP address:

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

In the logs, the first IP addresses in the **$http_x_forwarded_for** header are the client IP addresses.

Figure 4-47 Viewing access logs of a Nginx server

4.8 Querying Client IP Addresses in ELB Access Logs

Scenarios

When you are using ELB to distribute traffic, you may need to use the client IP addresses for analyzing data and protecting services.

To do this, you can enable access logging for Layer 7 listeners of a load balancer. Then, you can find the client IP addresses in the access logs of the load balancer.

Constraints

Access logging can be configured only for load balancers with HTTP, QUIC, TLS, or HTTPS listeners.

Preparations

- There is a dedicated load balancer with an EIP bound to it. If there is no such a load balancer, you can **buy one** and **bind an IPv4 EIP to the load balancer**.
- There is an HTTPS backend server group with an ECS (ECS01) running in it. The ECS is in the same VPC as the load balancer, and an application has been deployed on ECS01. For details about how to deploy an application for testing, see **Deploying an Application**.

- You have enabled LTS and have created a log group and log stream. For details, see **Ingesting ELB Logs to LTS**.
- You have set cloud structuring parsing for the log stream and have selected ELB as the system template. For details, see **Setting Cloud Structuring Parsing**.

Step 1: Upload the Server Certificate to ELB

Before adding an HTTPS listener to a load balancer, you need to upload your certificate to the ELB console.

1. Go to the **load balancer list page**.
2. In the navigation pane on the left, choose **Certificates**.
3. Click **Add Certificate** in the top right corner and set parameters by referring to **Table 4-6**.

   Table 4-6 Server certificate parameters

Parameter	Description
Certificate Type	Specifies the certificate type. Select **Server certificate**.
Source	Specifies the source of a certificate. There are two options: **SSL Certificate Manager** and **Your certificate**. **SSL Certificate Manager** is used in this example, so that you can select the SSL certificates you have purchased on the CCM console.
Certificate	Specifies the certificate that you want to upload to the ELB console.
Enterprise Project	Specifies an enterprise project by which cloud resources and members are centrally managed.
SNI Domain Name (Optional)	All domain names of the SSL certificate will be automatically selected. If the certificate is intended for SNI, you can select an SNI certificate based on the domain name in the HTTPS requests.
Description (Optional)	Provides supplementary information about the certificate.

4. Click **OK**.

Step 2: Add an HTTPS Listener and Configure One-Way Authentication

1. Go to the **load balancer list page**.
2. Locate the target load balancer and click **Add Listener** in the **Operation** column.

3. On the **Add Listener** page, select **HTTPS** for **Frontend Protocol** and **One-way authentication** for **SSL Authentication**.

   Select the server certificate uploaded to the ELB console in **Step 1**.

   Transfer Client IP Address is enabled by default for the HTTPS listener of the dedicated load balancer, which means that client IP addresses can be placed in the X-Forwarded-For header and transferred to the backend servers.

   Figure 4-48 Configuring one-way authentication

4. Click **Next: Configure Request Routing Policy** and select **Use existing** for **Backend Server Group**.

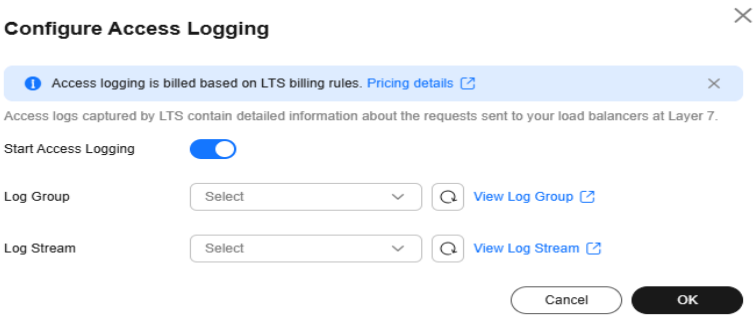
   Select an existing backend server group and click **Next: Confirm**.

5. Confirm the configurations and click **Submit**.

Step 3: Configure Access Logging

1. Go to the **load balancer list page**.
2. On the **Load Balancers** tab, locate the load balancer and click its name.
3. Under **Access Logs**, click **Configure Access Logging**.
4. Enable access logging and select the log group and log stream you have created.

Figure 4-49 Configuring access logging

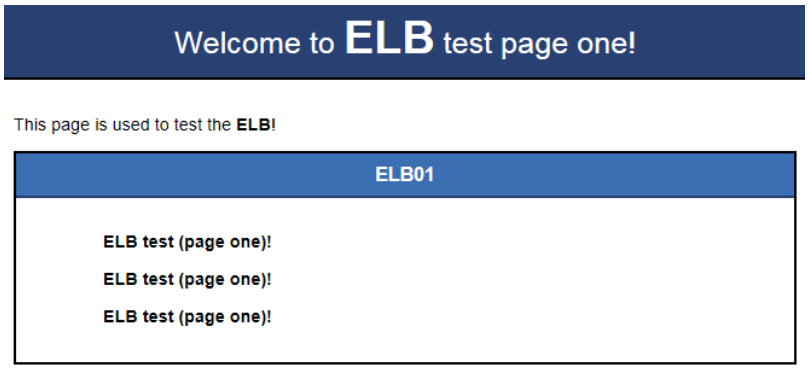


5. Click **OK**.

Step 4: Access the Load Balancer Through Its EIP

Enter the EIP bound to the load balancer in the address box of your browser to access the load balancer. If the page shown in **Figure 4-50** is displayed, the request is forwarded to ECS01, and the application is successfully deployed.

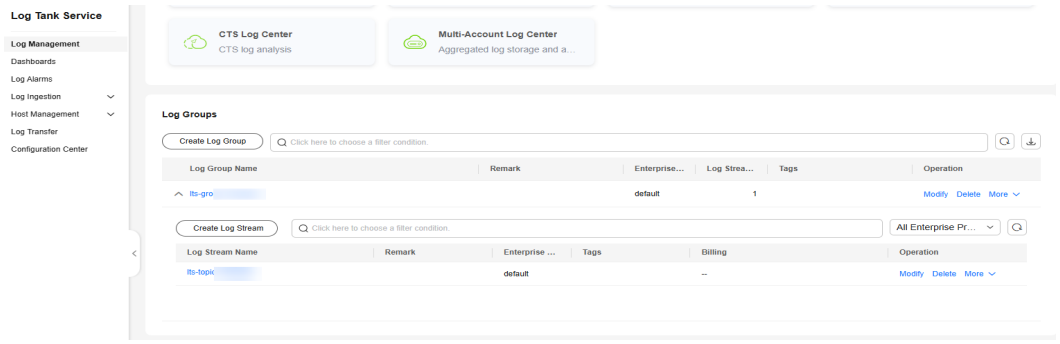
Figure 4-50 Nginx successfully deployed on **ECS01**



Step 5: Check the Client IP Address in Access Logs

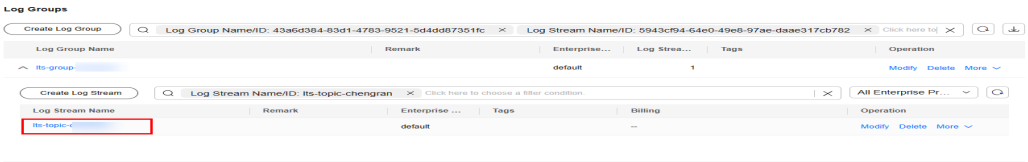
1. On the **Access Logs** tab of the load balancer, click **View Log Details**. The log management page of the LTS console is displayed.

Figure 4-51 Viewing the log group



2. On the **Log Groups** page, click the log stream name to go to its details page.

Figure 4-52 Viewing the log stream



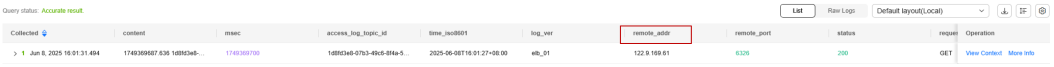
3. On the log stream details page, check the client IP address in the log list.
   a. If the client directly accesses the load balancer, the IP address in the **remote_addr** field indicates the client IP address.
   b. If the client accesses the load balancer through a proxy server, the first IP address in the **http_x_forwarded_for** field indicates the client IP address.

   You can search for logs by specific keyword or by time range. For details, see **Searching for Logs**.

Table 4-7 Fields in the access log

Field	Description	Value Description
remote_addr: remote_port	IP address and port number of the client.	Records the IP address and port of the client.
http_x_forwarded_for	**http_x_forwarded_for** in the request header received by the load balancer, indicating the request is sent through a proxy server.	Records all the IP addresses along the path through which the request is sent. The first IP address is the client IP address.

Figure 4-53 Viewing the client IP address



Helpful Links

- **Access Logging**
- **Using a Dedicated Load Balancer at Layer 7 to Transfer Client IP Addresses**
- **Using a Dedicated Load Balancer at Layer 4 to Transfer Client IP Addresses**

4.9 Obtaining Client Certificate Information Through a Dedicated Load Balancer

Scenarios

Suppose you have deployed an application on the cloud, used a load balancer to distribute traffic to your application, and enabled mutual authentication on the

load balancer to offload client authentication from backend servers to the load balancer. This can save development and maintenance costs.

However, in certain cases like application-layer service authorization, custom services, and audits with high security requirements, backend servers may need to access the identity information of the client. In these cases, you can configure headers on the load balancer's listeners to include client certificate information in data packets sent to backend servers. This allows backend servers to access the client certificate information and enhance security control efficiently, without adding extra workload.

NOTE

This feature is available in certain regions. You can see which regions support this feature on the console. If you want to use this feature, **submit a service ticket**.

Preparations

- **Create a dedicated load balancer** and **bind an IPv4 EIP to it**.
- Create an HTTPS backend server group, add an ECS (ECS01) to the server group, and deploy an application on ECS01.
- **Create two ECSs** running CentOS in the same VPC as the load balancer. The first ECS (ECS_client) is used as the client to send requests, and the second ECS (ECS_server) is used as the backend server for deploying the application.
- Prepare certificates for the load balancer.
  - Either purchase a certificate or upload a third-party certificate to SSL Certificate Manager (SCM), and configure a public domain name for the certificate. It is recommended that you **purchase an SSL certificate** on the Cloud Certificate & Manager (CCM) console.
  - Purchase a CA certificate and export it to the local PC, or use a self-signed CA certificate. If you do not have such certificates, you can **purchase a private CA** from Huawei Cloud CCM and **export a private CA certificate**.
- Prepare certificates for the client. After creating and activating a private CA on the Huawei Cloud CCM console, you can apply for a private certificate and configure it on the client.
  - Set the private certificate file name to **client.crt** and the private key file name to **client.key**.
  - **Applying for a Private Certificate**
  - **Downloading a Private Certificate**
  - **Installing a Private Certificate on a Client**

Step 1: Upload the Server Certificate to ELB

Before adding an HTTPS listener to a load balancer, you need to upload your certificate to the ELB console.

1. Go to the **load balancer list page**.
2. In the navigation pane on the left, choose **Certificates**.
3. Click **Add Certificate** in the top right corner and set parameters by referring to **Table 4-8**.

Table 4-8 Server certificate parameters

Parameter	Description
Certificate Type	Specifies the certificate type. Select **Server certificate**.
Source	Specifies the source of a certificate. There are two options: **SSL Certificate Manager** and **Your certificate**. **SSL Certificate Manager** is used in this example, so that you can select the SSL certificates you have purchased on the CCM console.
Certificate	Specifies the certificate that you want to upload to the ELB console.
Enterprise Project	Specifies an enterprise project by which cloud resources and members are centrally managed.
SNI Domain Name (Optional)	All domain names of the SSL certificate will be automatically selected. If the certificate is intended for SNI, you can select an SNI certificate based on the domain name in the HTTPS requests.
Description (Optional)	Provides supplementary information about the certificate.

4. Click **OK**.

Step 2: Upload the CA Certificate to the ELB Console

Before adding an HTTPS listener to a load balancer, you need to upload your CA certificate to the ELB console.

1. Go to the **load balancer list page**.
2. In the navigation pane on the left, choose **Certificates**.
3. Click **Add Certificate** in the top right corner and set parameters by referring to **Table 4-9**.

Table 4-9 CA certificate parameters

Parameter	Description
Certificate Type	Specifies the certificate type. Select **CA certificate**.
Certificate Name	Specifies the name of the CA certificate.
Enterprise Project	Specifies an enterprise project by which cloud resources and members are centrally managed.

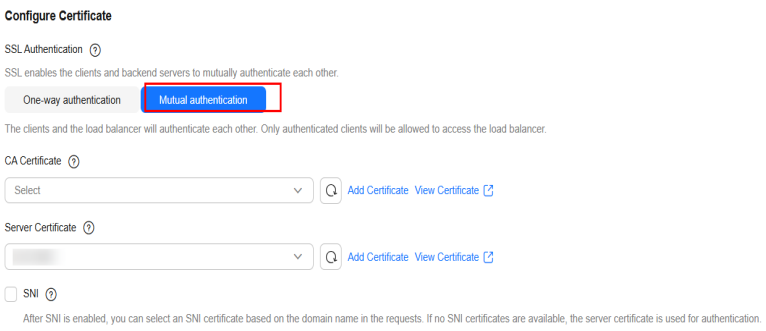
Parameter	Description
Certificate Content	Specifies the content of the CA certificate in PEM format. Click **Upload** and select the CA certificate to be uploaded. Ensure that your browser is the latest version. The format of the certificate body is as follows: -----BEGIN CERTIFICATE----- Base64-encoded certificate -----END CERTIFICATE-----
Description (Optional)	Provides supplementary information about the certificate.

4. Click **OK**.

Step 3: Add an HTTPS Listener and Configure Client Certificate Information

1. Go to the **load balancer list page**.
2. Locate the target load balancer and click **Add Listener** in the **Operation** column.
3. On the displayed page, set the following parameters:
   a. **Frontend Protocol**: **HTTPS**
   b. **Configure Certificate**
      i. **SSL Authentication**: **Mutual authentication**
      ii. **Server Certificate**: Select the server certificate uploaded to the ELB console in **Step 1**.
      iii. **CA Certificate**: Select the CA certificate uploaded to the ELB console in **Step 2**.

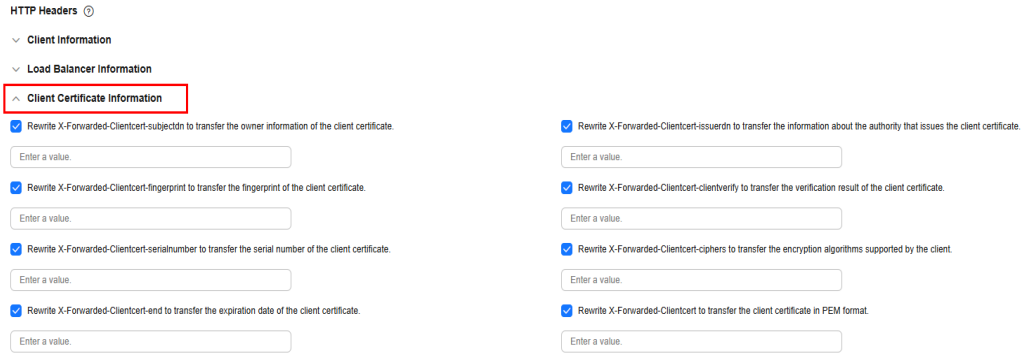
      Figure 4-54 Configuring mutual authentication for the HTTPS listener

   c. **HTTP Headers**
      i. Expand **More (Optional)**.
      ii. Expand **Client Certificate Information** under **HTTP Headers**.
      iii. Select the headers you want to rewrite to transfer the client certificate information to backend servers and define the header names.

Only lowercase letters, digits, underscores (_), and hyphens (-) are allowed.

Figure 4-55 HTTP headers that can transfer client certificate information



4. Click **Next: Configure Request Routing Policy** and select **Use existing** for **Backend Server Group**. Select the backend server group you have created and click **Next: Confirm**.
5. Confirm the configurations and click **Submit**.

Step 4: Deploy an Application on the Backend Server

1. Remotely log in to ECS_server.

   Multiple methods are available for logging in to an ECS. For details, see **Logging In to an ECS**.
2. Make sure the server uses Python 3 or a later version.
3. Create the **Test** directory on the backend server.
   ```
   mkdir Test
   ```
4. Create the **server.py** file in the **Test** directory and deploy the application in the file to print the request headers received by the backend server.
   a. Create the **server.py** file.
      ```
      vi server.py
      ```
   b. Press **i** to enter editing mode.

      Example Code

      - **server.py** script
        ```
        import http.server
        import socketserver
        import logging
        from datetime import datetime

        logging.basicConfig(
            level=logging.INFO,
            format='%(asctime)s - %(message)s',
            handlers=[
                logging.FileHandler('app.log', mode='a'),
                logging.StreamHandler()
            ]
        )

        class HeaderLoggerHTTPHandler(http.server.BaseHTTPRequestHandler):
            def do_GET(self):
                self.log_headers()
                self.send_response(200)
                self.send_header('Content-type', 'text/plain')
                self.end_headers()
        ```



```
        self.wfile.write(b"Headers logged to app.log")

    def log_headers(self):
        headers = {k: v for k, v in self.headers.items()}
        logging.info(f"{self.command} Request from {self.client_address[0]}")
        for key, value in headers.items():
            logging.info(f"Header: {key} = {value}")


if __name__ == '__main__':
    PORT = 443
    with socketserver.TCPServer(("", PORT), HeaderLoggerHTTPHandler) as httpd:
        print(f"Server started at http://localhost:{PORT}")
        print(f"Headers will be logged to app.log")
        try:
            httpd.serve_forever()
        except KeyboardInterrupt:
            print("\nServer stopped")
```

5. Press **Esc** and enter **:wq** to save the **server.py** file.
6. Run the **server.py** file.
   ```
   python3 server.py
   ```
7. Wait until the command output shown in **Figure 4-56** is displayed, which indicates the backend application is successfully deployed.

Figure 4-56 Backend application started

Step 5: Access the Backend Application from the Client

1. Remotely log in to **ECS_client**.
2. Upload the **client.crt** private certificate and **client.key** private key file downloaded to the local PC to the client.

   For details, see **How Do I Upload Files to My ECS?**
3. Run the curl command on the client to specify the client certificate file and the private key file of the client certificate to access the backend application.

   Example curl Command

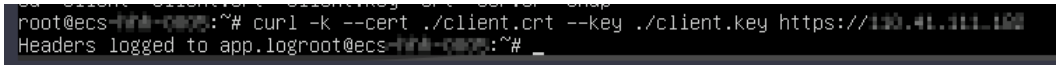
   – curl command for mutual authentication communication
   ```
   curl -k --cert <certificate> --key <private.key> https://<EIP-or-domain-name>
   ```
   --cert <certificate>: Specify the client certificate, which is used to authenticate the client.

   --key <private.key>: Specify the private key of the client certificate, which is used together with the client certificate for authentication.

   https://*<EIP-or-domain-name>*: Specify the destination IP address the client is trying to access.

   In this practice, you can run the following command to access the backend application:
   ```
   curl -k --cert ./client.crt --key ./client.key https://<EIP-of-the-load-balancer>
   ```

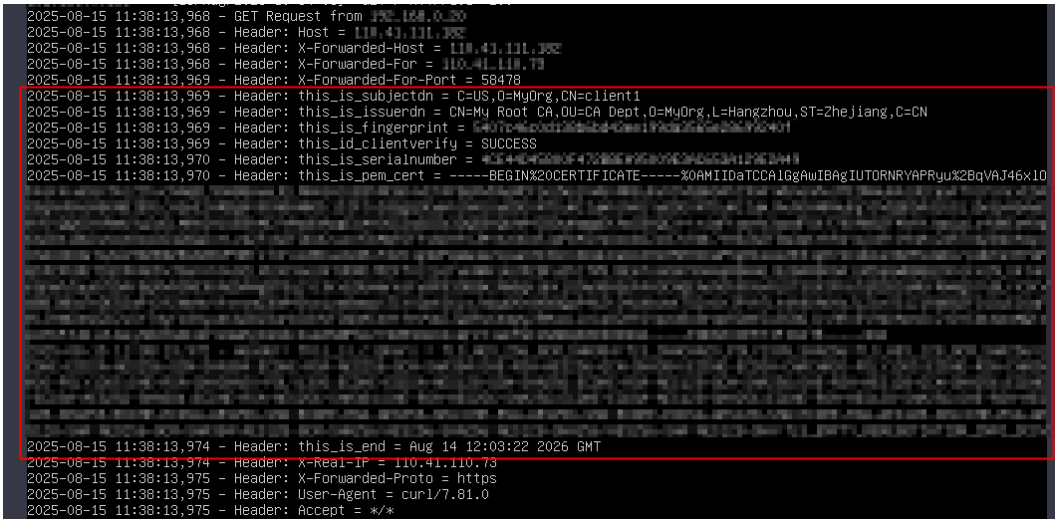
Figure 4-57 Accessing the backend application using the curl command



Step 6: Verify Client Certificate Information Transferring

1. Remotely log in to **ECS_server**.
2. Check the client certificate information transferred by the printed headers.

Figure 4-58 Checking the client certificate information obtained by the server



4.10 Using Deregistration Delay of a Dedicated Load Balancer to Ensure Smooth Service Termination

Scenarios

When a backend server is detected as unhealthy or gets removed, connections established with the backend server are not interrupted immediately. Client requests keep going to this server, which can lead to service termination failures or request issues.

To address this issue, you can use the deregistration delay feature of a dedicated load balancer. After deregistration delay is enabled, if a backend server is removed or the health check fails, ELB continues to route in-flight requests to this server until the deregistration delay timeout expires.

Preparations

- **Create a dedicated load balancer**.
- **Create two ECSs** in the same VPC as the load balancer.

  This practice uses CentOS ECSs as an example. An ECS (ECS_client) acts as the client to send requests, and it must support persistent connections. The other ECS (ECS_server) serves as the backend server for deploying backend applications.

Step 1: Create a Backend Server Group and Enable Deregistration Delay

In this practice, enable **Deregistration Delay** and set **Deregistration Delay Timeout (s)** to 30 seconds.

1. Go to the **backend server group list page**.
2. Click **Create Backend Server Group** in the upper right corner.
3. Configure the parameters based on **Table 4-3**. Retain the default values for other parameters.

Table 4-10 Parameters required for configuring a backend server group

Parameter	Example Value	Description
Backend Server Group Name	server_group	Specifies the name of the backend server group.
Type	Dedicated	Specifies the type of the load balancer that can use the backend server group.
Load Balancer	Associate existing	Specifies whether to associate a load balancer now. Click **Associate existing** and select the load balancer you have created.
Backend Protocol	TCP	Specifies the protocol that backend servers in the backend server group use to receive requests from the listeners. Select **TCP**.
Load Balancing Algorithm	Weighted round robin	Select **Weighted round robin**.
Deregistration Delay	Enable it.	This parameter is enabled by default if the backend protocol is TCP, UDP, or QUIC. If a backend server is removed or the health check fails, ELB stops routing new requests but continues to route in-flight requests to this server until the deregistration delay timeout expires.
Deregistration Delay Timeout (s)	30	ELB continues to route in-flight requests to the backend server until the deregistration delay timeout expires. In this practice, set it to 30 seconds.

4. Click **Next** to add backend servers and configure health check.
5. Click **Add Cloud Server**, select **ECS_server**, and retain the default values for other parameters.
6. Enable health check and retain the default values for other health check parameters.
7. Click **Next**.
8. Confirm the configurations and click **Create Now**.

Step 2: Add a TCP Listener and Associate It with the Backend Server Group

This practice uses a TCP listener as an example to distribute traffic.

1. Go to the **load balancer list page**.
2. Locate the target load balancer and click **Add Listener** in the **Operation** column.
3. On the **Add Listener** page, set **Protocol** to **TCP** and **Port** to **80**, and retain the default values for other parameters.
4. Click **Next: Configure Request Routing Policy** and configure the backend server group.

   Click **Use Existing** and select the backend server group created in **Step 1: Create a Backend Server Group and Enable Deregistration Delay**.
5. Click **Next: Confirm**, confirm the configurations, and submit your request.

Step 3: Deploy the Application on the Backend Server

1. Remotely log in to **ECS_server**.

   Multiple methods are available for logging in to an ECS. For details, see **Logging In to an ECS**.
2. Create the **Test** directory on the backend server.
   ```
   mkdir Test
   ```
3. Create the **server.py** file in the **Test** directory and deploy the service for testing deregistration delay in the file to print the connection status between the server and client per second.
   a. Create the **server.py** file.
      ```
      python server.py
      ```
   b. Press **i** to enter editing mode.

      Example Code

      - The following shows the **server.py** script and the default port is 80.
        ```
        import socket
        import threading
        def handle_client(client_socket, addr):
            print(f"[+] Connection request from {addr}")
            try:
                while True:
                    data = client_socket.recv(1024).decode()
                    if not data:
                        break
                    print(f"[{addr}] Received: {data}")
                    client_socket.send(f"ACK: {data}".encode())
            except ConnectionResetError:
                print(f"[-] {addr} Connection interrupted")
            finally:
                client_socket.close()
                print(f"[-] {addr} Connection closed")
        def start_server():
            server = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
            server.bind(('0.0.0.0', 80))
            server.listen(5)
            print("[*] Server started, waiting to be connected...")
            while True:
                client_sock, addr = server.accept()
                client_thread = threading.Thread(target=handle_client, args=(client_sock, addr))
                client_thread.start()
        if __name__ == "__main__":
            start_server()
        ```


4. Press **Esc** and enter **:wq** to save the **server.py** file.
5. Run the **server.py** file.
   ```
   python server.py
   ```
6. Wait until the command output shown in **Figure 4-59** is displayed, which indicates the backend service is successfully deployed.

   Figure 4-59 Backend service started

Step 4: Deploy the Application on the Client Server

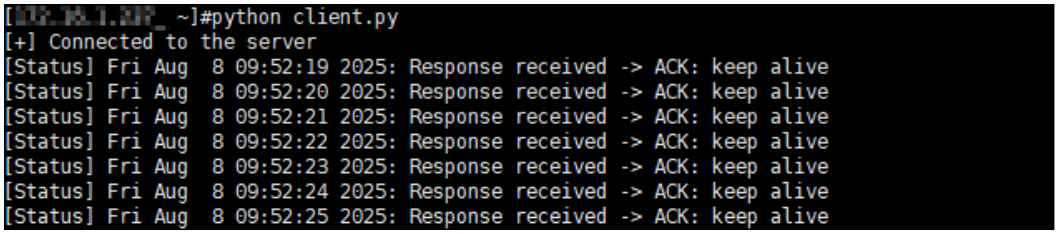
1. Remotely log in to **ECS_client**.
2. Create the **Test_client** directory on the client server.
   ```
   mkdir Test_client
   ```
3. Create the **python_client.py** file in the **Test_client** directory and deploy the application service in the file to print the connection status between the client and server every second.
   a. Create the **python_client.py** file.
      ```
      vi python_client.py
      ```
   b. Press **i** to enter editing mode.

      Example Code

      - The following shows the **python_client.py** script and the default port is 80.

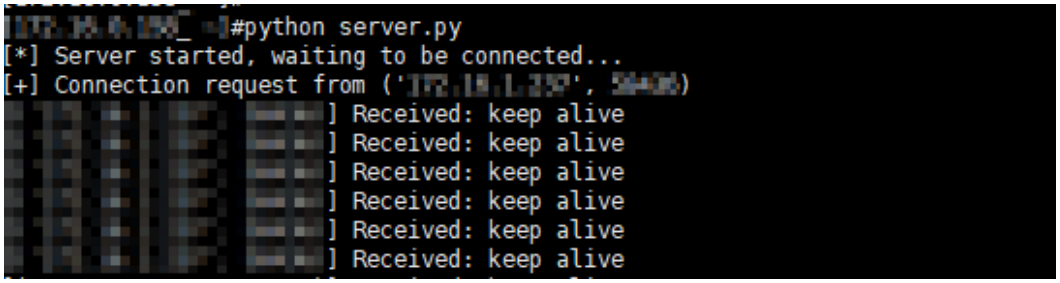
        ```
        import socket
        import time
        import threading
        def client_loop():
            try:
                s = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
                s.connect(('<your_elb_ip>', 80)) # <your_elb_ip>: Replace it with the private IP address of your load balancer.
                print("[+] Connected to the server")
                while True:
                    try:
                        s.send("keep alive".encode())
                        response = s.recv(1024).decode()
                        print(f"[Status] {time.ctime()}: Response received -> {response}")
                        time.sleep(1)
                    except (ConnectionResetError, ConnectionAbortedError):
                        print("[-] Connection to server interrupted")
                        break
            except Exception as e:
                print(f"[Error] Connection failed: {e}")
            finally:
                s.close()
        if __name__ == "__main__":
            client_thread = threading.Thread(target=client_loop)
            client_thread.start()
            client_thread.join()
        ```
4. Press **Esc** and enter **:wq** to save the **client.py** file.
5. Run the **client.py** file.
   ```
   python client.py
   ```
6. Wait until the command output shown in **Figure 4-60** is displayed, which indicates the backend service is deployed successfully.

Figure 4-60 Client started



7. Check whether **ECS_server** receives the client request and prints the connection status with the client as shown in **Figure 4-61**.

Figure 4-61 Client request received



Step 5: Remove the Backend Server and Record the Removal Time

In this practice, we remove a backend server to check how deregistration delay works.

1. Go to the **backend server group list page**.
2. On the backend server group list page, click the name of the target backend server group.
3. Switch to the **Backend Servers** tab and click **Cloud Servers**.
4. Select the backend servers you want to remove and click **Remove** above the backend server list.
5. In the displayed dialog box, click **OK**.
6. Record the time when the backend server is removed. The client prints the time **09:52:33**.

Step 6: Verify the Deregistration Delay Function

1. Observe the connection status printed by the client and server before deregistration delay timeout expires.

   If the connection status as shown in **Figure 4-62** and **Figure 4-63** is displayed, the persistent connection between the client and the server remains active.

Figure 4-62 Connection status printed by the client

```
[172.16.1.219 ~]#python client.py
[+] Connected to the server
[Status] Fri Aug  8 09:52:19 2025: Response received -> ACK: keep alive
[Status] Fri Aug  8 09:52:20 2025: Response received -> ACK: keep alive
[Status] Fri Aug  8 09:52:21 2025: Response received -> ACK: keep alive
[Status] Fri Aug  8 09:52:22 2025: Response received -> ACK: keep alive
[Status] Fri Aug  8 09:52:23 2025: Response received -> ACK: keep alive
[Status] Fri Aug  8 09:52:24 2025: Response received -> ACK: keep alive
[Status] Fri Aug  8 09:52:25 2025: Response received -> ACK: keep alive
[Status] Fri Aug  8 09:52:26 2025: Response received -> ACK: keep alive
[Status] Fri Aug  8 09:52:27 2025: Response received -> ACK: keep alive
[Status] Fri Aug  8 09:52:28 2025: Response received -> ACK: keep alive
[Status] Fri Aug  8 09:52:29 2025: Response received -> ACK: keep alive
[Status] Fri Aug  8 09:52:30 2025: Response received -> ACK: keep alive
[Status] Fri Aug  8 09:52:31 2025: Response received -> ACK: keep alive
[Status] Fri Aug  8 09:52:32 2025: Response received -> ACK: keep alive
[Status] Fri Aug  8 09:52:33 2025: Response received -> ACK: keep alive
[Status] Fri Aug  8 09:52:34 2025: Response received -> ACK: keep alive
[Status] Fri Aug  8 09:52:35 2025: Response received -> ACK: keep alive
[Status] Fri Aug  8 09:52:36 2025: Response received -> ACK: keep alive
[Status] Fri Aug  8 09:52:37 2025: Response received -> ACK: keep alive
[Status] Fri Aug  8 09:52:38 2025: Response received -> ACK: keep alive
[Status] Fri Aug  8 09:52:39 2025: Response received -> ACK: keep alive
[Status] Fri Aug  8 09:52:40 2025: Response received -> ACK: keep alive
[Status] Fri Aug  8 09:52:41 2025: Response received -> ACK: keep alive
[Status] Fri Aug  8 09:52:42 2025: Response received -> ACK: keep alive
```

Figure 4-63 Connection status printed by the server

```
[172.16.1.150 ~]#python server.py
[*] Server started, waiting to be connected...
[+] Connection request from ('172.16.1.219', 58446)
[('172.16.1.219', 58446)] Received: keep alive
[('172.16.1.219', 58446)] Received: keep alive
[('172.16.1.219', 58446)] Received: keep alive
[('172.16.1.219', 58446)] Received: keep alive
[('172.16.1.219', 58446)] Received: keep alive
[('172.16.1.219', 58446)] Received: keep alive
[('172.16.1.219', 58446)] Received: keep alive
[('172.16.1.219', 58446)] Received: keep alive
[('172.16.1.219', 58446)] Received: keep alive
[('172.16.1.219', 58446)] Received: keep alive
[('172.16.1.219', 58446)] Received: keep alive
[('172.16.1.219', 58446)] Received: keep alive
[('172.16.1.219', 58446)] Received: keep alive
[('172.16.1.219', 58446)] Received: keep alive
[('172.16.1.219', 58446)] Received: keep alive
[('172.16.1.219', 58446)] Received: keep alive
[('172.16.1.219', 58446)] Received: keep alive
[('172.16.1.219', 58446)] Received: keep alive
[('172.16.1.219', 58446)] Received: keep alive
[('172.16.1.219', 58446)] Received: keep alive
[('172.16.1.219', 58446)] Received: keep alive
[('172.16.1.219', 58446)] Received: keep alive
[('172.16.1.219', 58446)] Received: keep alive
```

2. Observe the connection status printed by the client and server after deregistration delay timeout (30s) expires.

   If the connection status as shown in **Figure 4-64** and **Figure 4-65** is displayed, the persistent connection between the client and server is disconnected. According to **Figure 4-64**, the connection was disconnected at **09:53:03**, 30s later than the time when the backend server was removed (**09:52:33**).

Figure 4-64 Connection status (disconnected) printed by the client

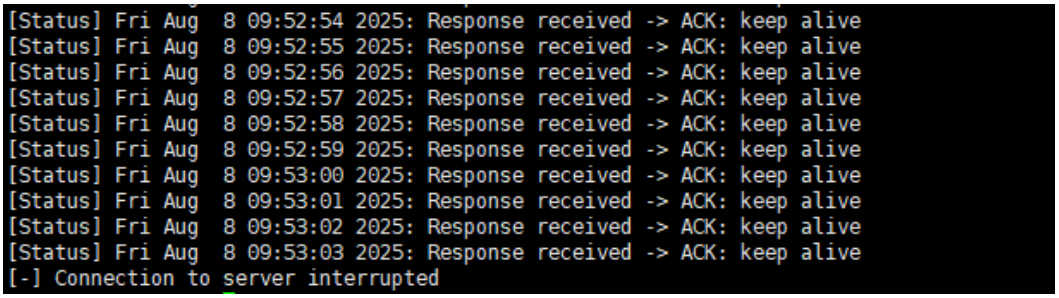


Figure 4-65 Connection status (disconnected) printed by the server

5 Forwarding Policies

5.1 Using Advanced Forwarding for Application Iteration

Scenarios

As the business grows, you may need to upgrade your application based on user feedback. In this process, you can use advanced forwarding to redirect requests from users to both the new and old version first. When the application of the new version runs stably, direct all the requests to the new version.

Prerequisites

Six ECSs are available, with three having the application of the old version deployed and the other three having the new version deployed.

Process for Configuring Advanced Forwarding

Figure 5-1 Flowchart

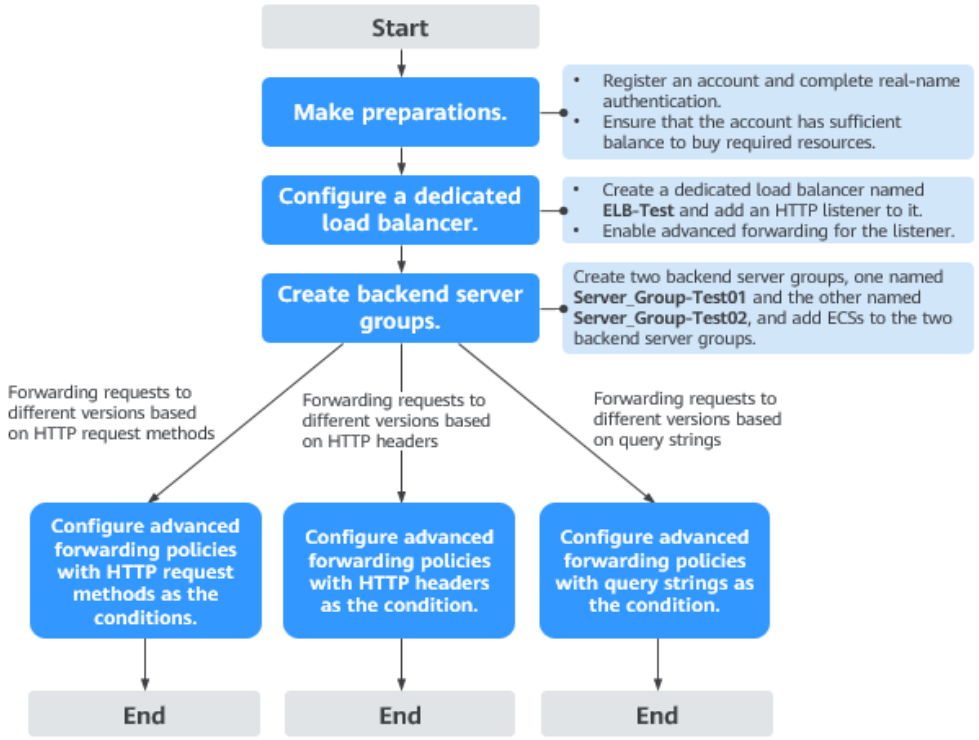


Table 5-1 Resource planning

Resource Name	Resource Type	Description
ELB-Test	Dedicated load balancer	Only dedicated load balancers support advanced forwarding.
Server_Group-Test01	Backend server group	Used to manage the ECSs where the application of the old version is deployed.
Server_Group-Test02	Backend server group	Used to manage the ECSs where the application of the new version is deployed.
ECS01	ECS	Used to deploy the application of the old version and added to **Server_Group-Test01**.
ECS02	ECS	Used to deploy the application of the old version and added to **Server_Group-Test01**.
ECS03	ECS	Used to deploy the application of the old version and added to **Server_Group-Test01**.

Resource Name	Resource Type	Description
ECS04	ECS	Used to deploy the application of the new version and added to **Server_Group-Test02**.
ECS05	ECS	Used to deploy the application of the new version and added to **Server_Group-Test02**.
ECS06	ECS	Used to deploy the application of the new version and added to **Server_Group-Test02**.

NOTE

In this practice, the dedicated load balancer is in the same VPC as the ECSs. You can also add servers in a different VPC or in an on-premises data center as needed. For details, see **Using IP as a Backend to Route Traffic Across Backend Servers**.

Step 1: Configure a Dedicated Load Balancer

1. Go to the **load balancer list page**.
2. In the upper right corner, click **Buy Elastic Load Balancer**.
3. Create a dedicated load balancer and configure the parameters as follows.
   - **Type**: **Dedicated**
   - **Name**: **ELB-Test**
   - Set other parameters as required. For details, see **Creating a Dedicated Load Balancer**.
4. Add an HTTP listener to **ELB-Test**. For details, see **Adding a Listener**.
5. Enable advanced forwarding. For details, see **Advanced Forwarding Policy**.

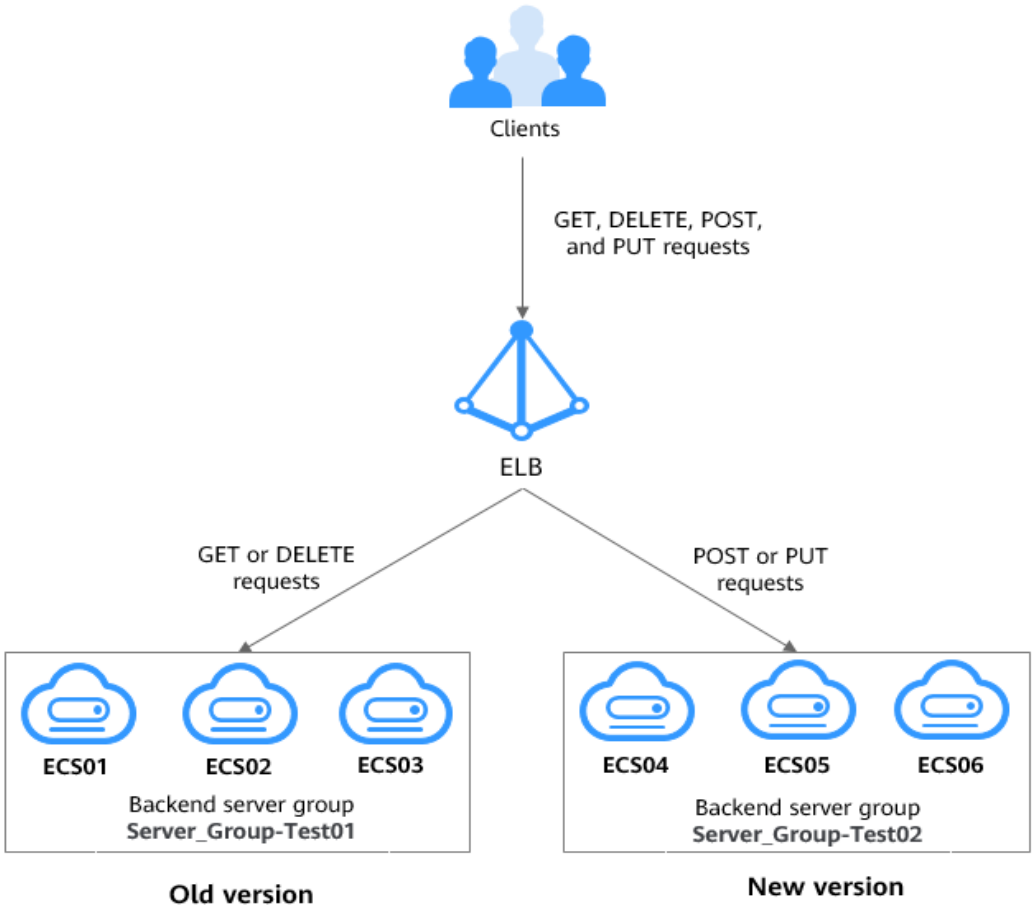
Step 2: Create Two Backend Server Groups and Add Backend Servers to Them

1. Go to the **load balancer list page**.
2. In the navigation pane on the left, choose **Elastic Load Balance** > **Backend Server Groups**.
3. Click **Create Backend Server Group** in the upper right corner.
   - Name: **Server_Group-Test01**
   - Load Balancer: Select **ELB-Test**.
   - **Backend Protocol**: **HTTP**
   - Configure other parameters as required.
4. Repeat **Step 5** to create backend server group **Server_Group-Test02**.
5. Add **ECS01**, **ECS02**, and **ECS03** to backend server group **Server_Group-Test01**.
6. Add **ECS04**, **ECS05**, and **ECS06** to backend server group **Server_Group-Test02**.

Forwarding Requests to Different Versions of the Application Based on HTTP Request Methods

Configure two advanced forwarding policies with the HTTP request method as the condition to route GET and DELETE requests to the application of the old version and POST and PUT requests to the application of the new version. When the application of the new version runs stably, direct all the requests to the new version.

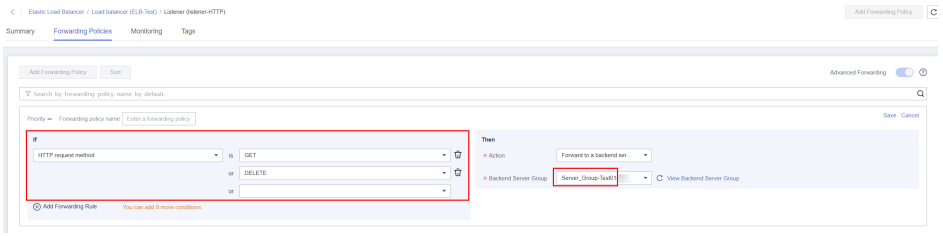
Figure 5-2 Forwarding requests based on HTTP request methods



1. Locate the dedicated load balancer and click its name **ELB-Test**.
2. On the **Listeners** tab, locate the HTTP listener added to the dedicated load balancer and click its name.
3. Switch to the **Forwarding Policies** tab on the right, and click **Add Forwarding Policy** to forward requests to the application of the old version.

   Select **GET** and **DELETE** from the **HTTP request method** drop-down list, select **Forward to a backend server group** for **Action**, and select **Server_Group-Test01** from the drop-down list.

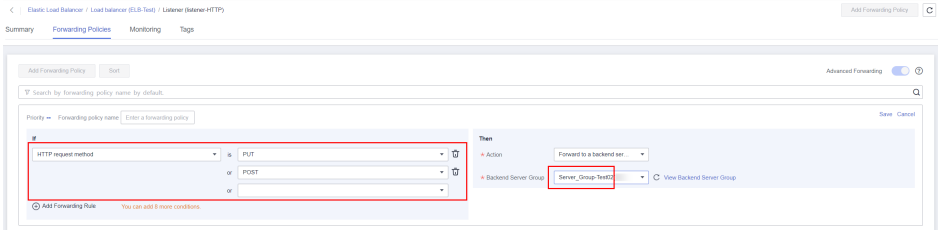
Figure 5-3 Forwarding GET and DELETE requests to the application of the old version



4. Click **Save**.
5. Repeat the preceding steps to add a forwarding policy to forward PUT and POST requests to the application of the new version.

   Select **PUT** and **POST** from the **HTTP request method** drop-down list, select **Forward to a backend server group** for **Action**, and select **Server_Group-Test02** from the drop-down list.

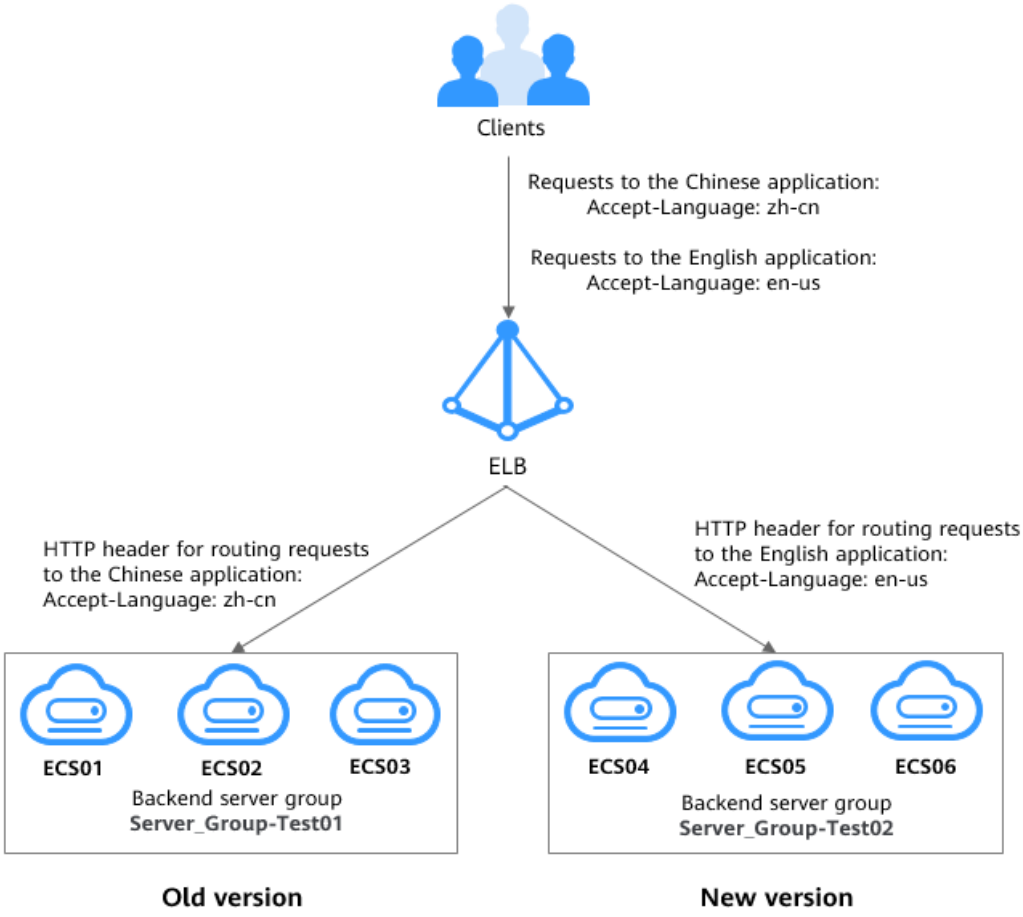
Figure 5-4 Forwarding PUT and POST requests to the application of the new version



Forwarding Requests to Different Versions of the Application Based on HTTP Headers

If the old version supports both Chinese and English, but the new version only supports English because the Chinese version is still under development, you can configure two advanced forwarding policies with the HTTP header as the condition to route requests to the Chinese application to the old version and requests to the English application to the new version. When the application of the new version runs stably, direct all the requests to the new version.

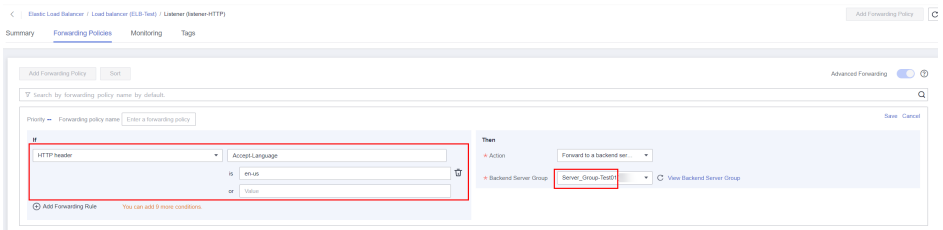
Figure 5-5 Smooth application transition between the old and new versions based on the HTTP request headers



1. Locate the dedicated load balancer and click its name **ELB-Test**.
2. On the **Listeners** tab, locate the HTTP listener added to the dedicated load balancer and click its name.
3. Switch to the **Forwarding Policies** tab on the right, and click **Add Forwarding Policy** to forward requests to the application of the old version.

   Select **HTTP header** from the drop-down list, set the key to **Accept-Language** and value to **en-us**, set the action to **Forward to a backend server group**, and select **Server_Group-Test01** as the backend server group.

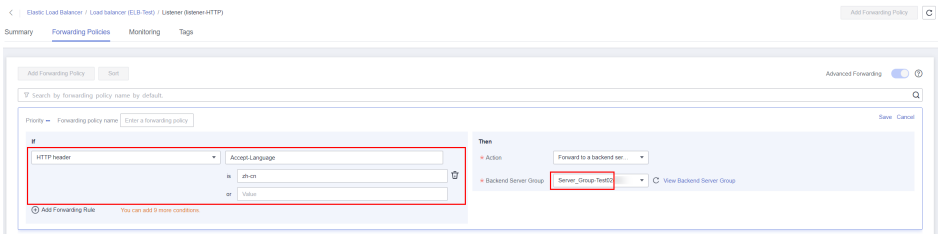
   Figure 5-6 Forwarding requests to the application of the old version

4. Click **Save**.
5. Repeat the preceding steps to add a forwarding policy to forward requests to the application of the new version.

Select **HTTP header** from the drop-down list, set the key to **Accept-Language** and value to **zh-cn**, set the action to **Forward to a backend server group**, and select **Server_Group-Test02** as the backend server group.

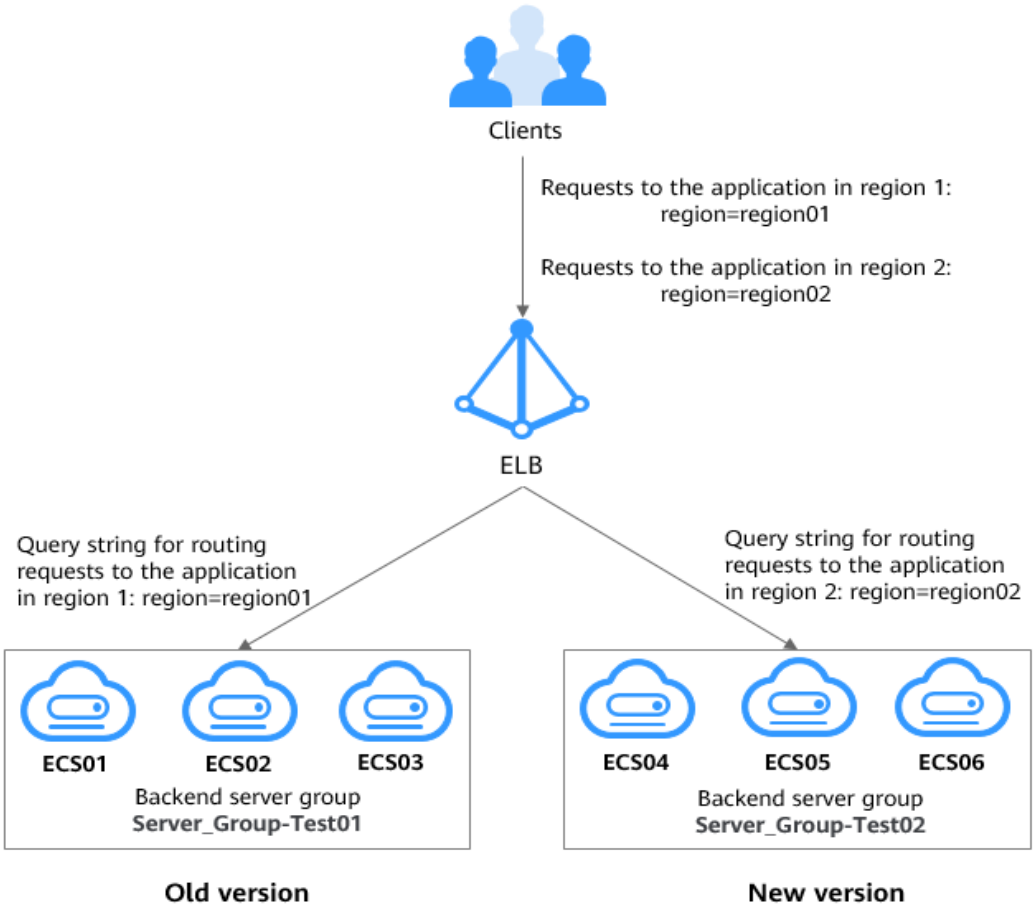
Figure 5-7 Forwarding requests to the application of the new version



Forwarding Requests to Different Versions of the Application Based on Query Strings

If the application is deployed across regions, you can configure two advanced forwarding policies with query string as the condition to forward requests to the application in region 1 to the old version and requests to the application in region 2 to the new version. When the application of the new version runs stably, direct all the requests to the new version.

Figure 5-8 Forwarding requests based on query strings



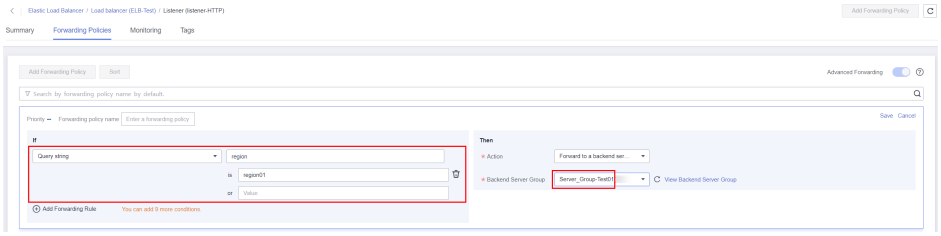
NOTE

- Dedicated load balancers can distribute traffic across regions or VPCs.
- In this example, you need to use Cloud Connect to connect the VPCs in two regions and then use a dedicated load balancer to route traffic to backend servers in the two regions.

1. Locate **ELB-Test** and click its name.
2. On the **Listeners** tab, locate the HTTP listener added to the dedicated load balancer and click its name.
3. Switch to the **Forwarding Policies** tab on the right, and click **Add Forwarding Policy** to forward requests to the application of the old version.

   Select **Query string** from the drop-down list, set the key to **region** and value to **region01**, set **Action** to **Forward to a backend server group**, and select **Server_Group-Test01** as the backend server group.

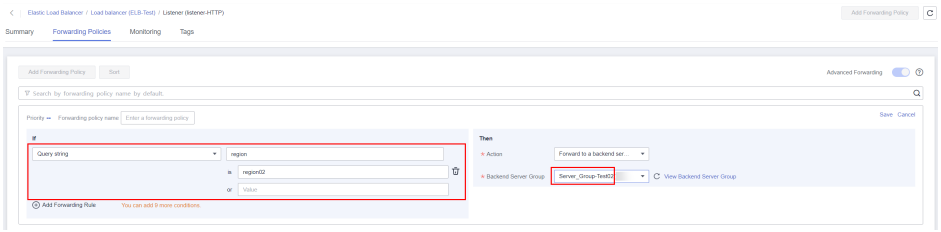
   Figure 5-9 Forwarding requests to the old version

4. Click **Save**.
5. Repeat the preceding steps to add a forwarding policy to forward requests to the application of the new version.

   Select **Query string** from the drop-down list, set the key to **region** and value to **region02**, set **Action** to **Forward to a backend server group**, and select **Server_Group-Test02** as the backend server group.

   Figure 5-10 Forwarding requests to the new version

5.2 Distributing Traffic Based on the Same Domain Name but Different Paths

Scenarios

Dedicated load balancers can distribute client requests to different backend server groups based on domain names and paths, allowing for efficient, reasonable, and refined traffic distribution.

- **Microservice architecture**: In a microservice architecture, a backend service consists of multiple independent subservices. Each subservice is deployed on a different backend server but uses the same domain name to provide services. In such a scenario, path-based forwarding policies can be used to distribute client requests to backend servers to handle the required service logic.
- **Dark launch or A/B testing**: Development, test, and production services are deployed under the same domain name but different paths. In such a scenario, forwarding policies with the same domain name but different paths can be used to forward traffic to the backend server where the target service is running.
- **Read/write separation**: Services with high concurrency and strict data consistency requirements, such as order processing, need read/write separation to optimize performance and enhance data security. In this case, read operations need to be forwarded to the read database server, and write operations forwarded to the write database server.

Overview of Domain Name- and Path-based Forwarding Policies

Dedicated load balancers allow you to configure forwarding rules and actions for advanced forwarding policies to route client requests to specific backend servers based on the information (such as domain name and path) in each request.

For details about the domain name- and path-based forwarding rules, see **Table 5-2**. For more forwarding rules, see **Advanced Forwarding**.

Table 5-2 Forwarding rules based on domain names and paths

Forwarding Rule	Description
Domain name	• Matching description Route requests based on domain names. You can configure multiple domain names with each consisting of at least two labels separated by periods (.). Each domain name can contain a maximum of 63 characters per label and a maximum total length of 100 characters. • Matching rules – Exact match and wildcard match: The domain name can contain only letters, digits, and special characters: .-?=~_+\^*!$&\|()[]. Asterisks (*) and question marks (?) can be used as wildcards. The domain name cannot start or end with a period (.) or contain two consecutive periods (..). – Regular expression match: The domain name can contain only letters, digits, and special characters: .-?=~_+\^*!$&|()[].

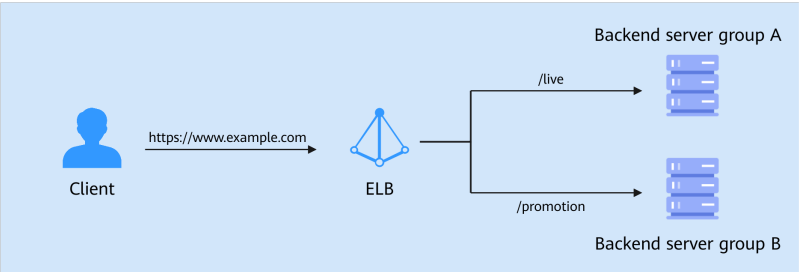
Forwarding Rule	Description
Path	• Matching description Route requests based on paths. You can configure multiple paths in a forwarding policy. Each path contains 1 to 128 characters, including letters, digits, and special characters: _~';@^-%#$.*+?,=!:|\/()[]{}. • Matching rules – Exact match: The request path is the same as the specified path and must start with a slash (/). – Prefix match: The request path starts with the specified path string and must start with a slash (/). – Regular expression match: The request path is matched against the specified path using a regular expression.

Solution Architecture

An e-commerce platform provides various microservices, including livestreaming and promotion services, through a domain name. Livestreaming requires high bandwidth and low latency, while promotions often face sudden traffic spikes, leading to heavy data read requests. If all services are deployed in the same backend server group, requests may be unevenly distributed. This impacts livestreaming performance during promotions because user surges overload the servers, causing unstable server running.

To address this issue, you can deploy each subservice of the e-commerce application on the server in different backend server groups, and configure forwarding policies with the same domain name but different paths for a dedicated load balancer to distribute client requests. This helps distribute traffic evenly and ensures each subservice can run efficiently and independently. As shown in **Figure 5-11**, requests matching the **/live** path are forwarded to backend server group A to provide enough bandwidths and processing capabilities for the livestreaming service. Requests matching the **/promotion** path are forwarded to backend server group B to optimize computing resources to handle queries and submissions faster.

Figure 5-11 Distributing traffic based on the same domain name but different paths



Constraints

- Only HTTP/HTTPS/QUIC listeners of dedicated load balancers support advanced forwarding policies.
- A maximum of 100 forwarding policies can be configured for a listener. If the number of forwarding policies exceeds the quota, the excess forwarding policies will not be applied.
- If the advanced forwarding policy is enabled, each forwarding rule has up to 10 forwarding conditions.

Preparations

- **Create a dedicated load balancer** and **bind an IPv4 EIP to it**.
- Create two HTTP backend server groups (A and B), add ECS01 to backend server group A and ECS02 to backend server group B, and deploy a service on each ECS.

  Deployment Commands of ECS01
  - yum install -y nginx
  - systemctl start nginx.service
  - cd /usr/share/nginx/html/
  - echo "Hello World ! This is live." > index.html

  Deployment Commands of ECS02
  - yum install -y nginx
  - systemctl start nginx.service
  - cd /usr/share/nginx/html/
  - echo "Hello World ! This is promotion." > index.html
- Register a domain name and complete the ICP filing. In this practice, domain name www.example.com is used.

Step 1: Enable Advanced Forwarding

1. Go to the **load balancer list page**.
2. On the displayed page, locate the load balancer you want to add forwarding policies for and click its name.
3. On the **Listeners** tab and click the target listener.
4. On the **Summary** tab, click **Enable** next to **Advanced Forwarding**.
5. Click **OK**.

Step 2: Configure Advanced Forwarding Policies

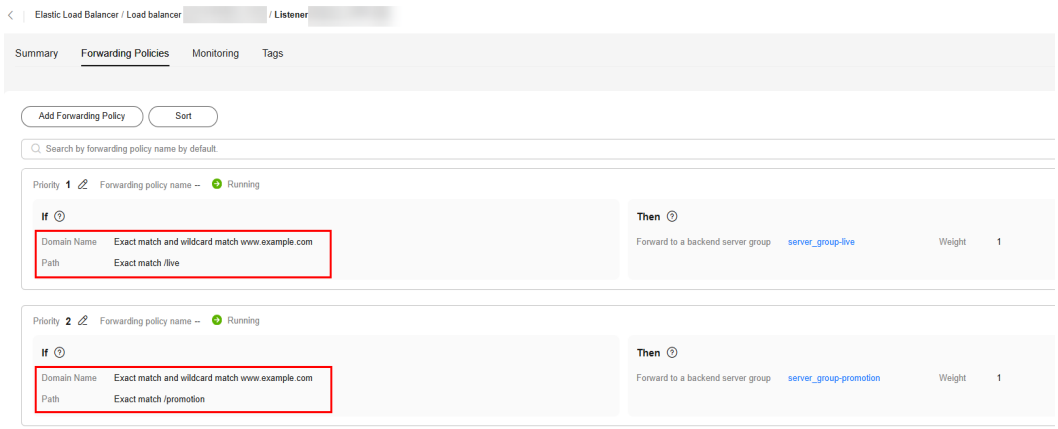
1. Go to the **load balancer list page**.
2. On the displayed page, locate the load balancer you want to add forwarding policies for and click its name.
3. On the **Listeners** tab, add forwarding policies in either of the following ways:
   - Locate the target listener and click **Add/Edit Forwarding Policy** in the **Forwarding Policies** column.
   - Locate the target listener, click its name, and click the **Forwarding Policies** tab.

4. Click **Add Forwarding Policy** and configure forwarding policies.

   Domain name: Enter the domain name used in this practice.

   Path: Requests matching the **/live** path are forwarded to backend server group A and those matching the **/promotion** path are forwarded to backend server group B.

5. Click **Save**.



Step 3: Configure Domain Name Resolution

You can add an A record set to resolve the domain name to the public IP address of the load balancer so that clients can access the load balancer using the public domain name.

The following provides an example for resolving a website domain name to an IPv4 address. For details about how to configure an A record set, see **Routing Internet Traffic to a Website**.

1. Go to the **DNS console**.
2. In the navigation pane on the left, choose **Public Zones**.

   The zone list is displayed.

3. Locate the public zone and click **Manage Record Sets** in the **Operation** column.
4. Click **Add Record Set**.
5. Configure the parameters based on **Table 5-3**.

Table 5-3 Parameters for adding an A record set

Parameter	Example Value	Description
Type	A – Map domains to IPv4 addresses	Type of the record set. In this example, set it to **A - Map domains to IPv4 addresses**.
Name	www	Prefix of the domain name to be resolved.

Parameter	Example Value	Description
Line	Default	Resolution line. The DNS server will return the IP address of the specified line, depending on where end users come from. The default value is **Default**. **Default**: returns the default resolution result irrespective of where the visitors come from.
TTL (s)	300	Cache duration of the record set on a local DNS server, in seconds. In this example, the default value 300 is used.
Value	192.168.12.2	IPv4 addresses mapped to the domain name. In this example, set this parameter to the EIPs bound to the load balancer.
Advanced Settings (Optional)	-	Click ⌄ to expand the advanced settings, set the alias and weight of the record set, and add a description and tags. In this example, the default settings are used.

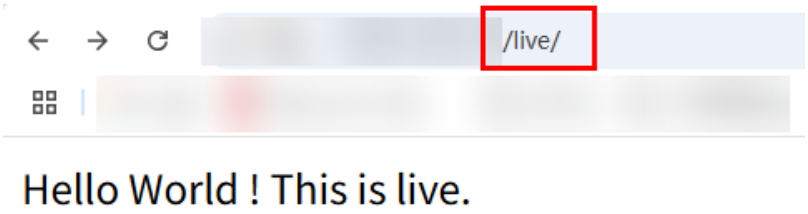
6. Click **OK**.
7. Switch back to the **Record Sets** tab.
   The added record set is in the **Normal** state.

Step 4: Verify Traffic Distribution

Use a browser to access **http://***<domain name>***/***<path>***/** to verify traffic routing.

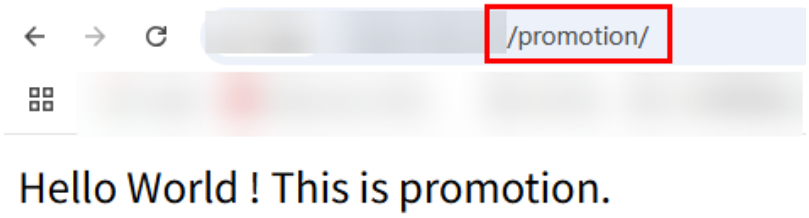
1. Access **http://www.example.com/live/**. The response shown in **Figure 5-12** is returned.

   Figure 5-12 Access to www.example.com/live/ succeeded

2. Access **http://www.example.com/promotion/**. The response shown in **Figure 5-13** is returned.

Figure 5-13 Access to www.example.com/promotion/ succeeded



Helpful Links

- **Using Advanced Forwarding for Application Iteration**
- **Using ELB to Redirect HTTP Requests to an HTTPS Listener for Higher Service Security**

6 ELB and Cloud-native Applications

A load balancer can work as an entry point for incoming traffic to CCE instances. It can distribute service requests from clients across CCE pods or containers.

A load balancer can be used as:

1. A LoadBalancer Service that can process TCP and UDP traffic at Layer 4 and HTTP and HTTPS traffic at Layer 7. Some Layer 7 functions can be used, such as certificate offloading, Layer 7 access logs, and Layer 7 Cloud Eye monitoring metrics.
2. An ELB ingress that can handle HTTP/HTTPS traffic at Layer 7 and support more advanced application-layer functions, such as Layer 7 routing, certificate offloading, Layer 7 access logs, and Layer 7 monitoring metrics.

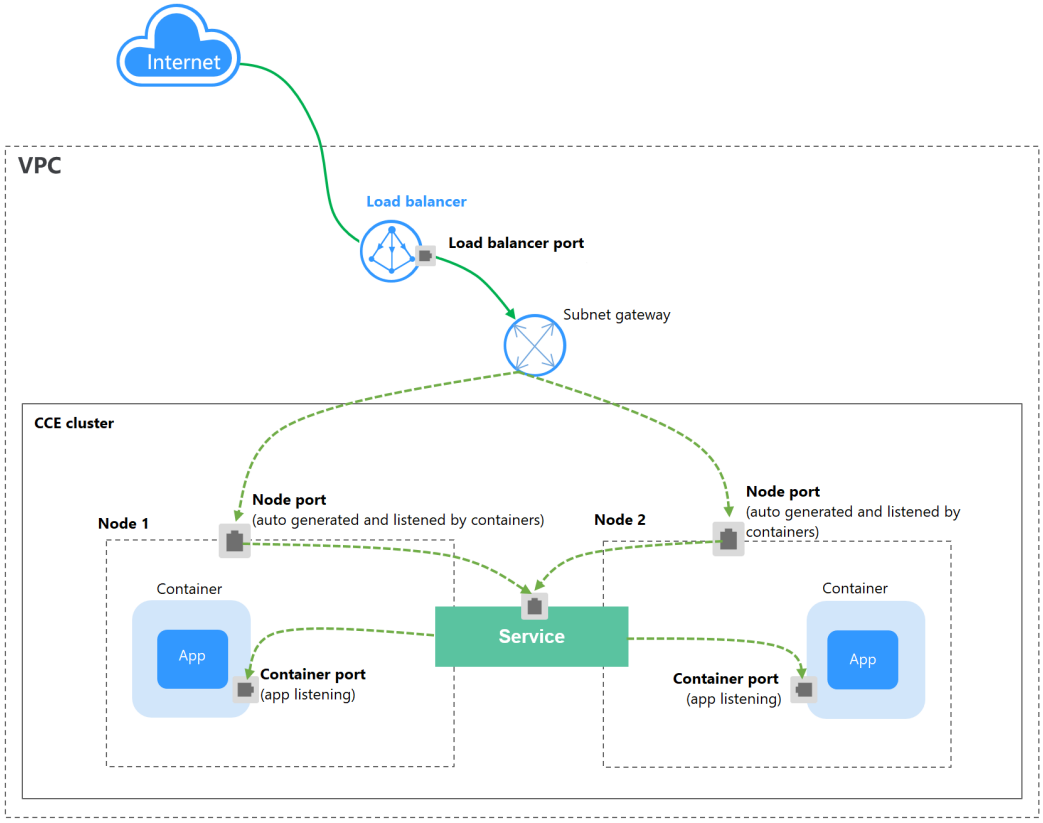
CCE provides powerful elasticity and automation capabilities, which can quickly start backend workloads. ELB and CCE can be used together for elastic and HA scenarios such as dark launches.

Using a LoadBalancer Service to Distribute External Requests Across Pods in CCE Cluster

A LoadBalancer Service adds an external load balancer on the top of a NodePort Service and distributes external traffic to multiple pods within a cluster. A LoadBalancer Service provides higher reliability than a NodePort Service. It automatically assigns an external IP address to allow access from the clients. LoadBalancer Services process TCP and UDP traffic at Layer 4 (transport layer) of the OSI model. They can be extended to support Layer 7 (application layer) capabilities to manage HTTP and HTTPS traffic.

If cloud applications require a stable, easy-to-manage entry for external access, you can create a LoadBalancer Service. For example, in a production environment, you can use LoadBalancer Services to expose public-facing services such as web applications and API services to the Internet. These services often need to handle heavy traffic while maintaining high availability. The access address of a LoadBalancer Service is in the format of *{EIP-of-load-balancer}*:*{access-port}*, for example, **10.117.117.117:80**.

Figure 6-1 LoadBalancer Service



If you need to configure a LoadBalancer Service, see .

Configuring Advanced Load Balancing Functions Using Annotations

LoadBalancer Services provide Layer 4 network access. You can add annotations to a YAML file to use some advanced CCE functions.

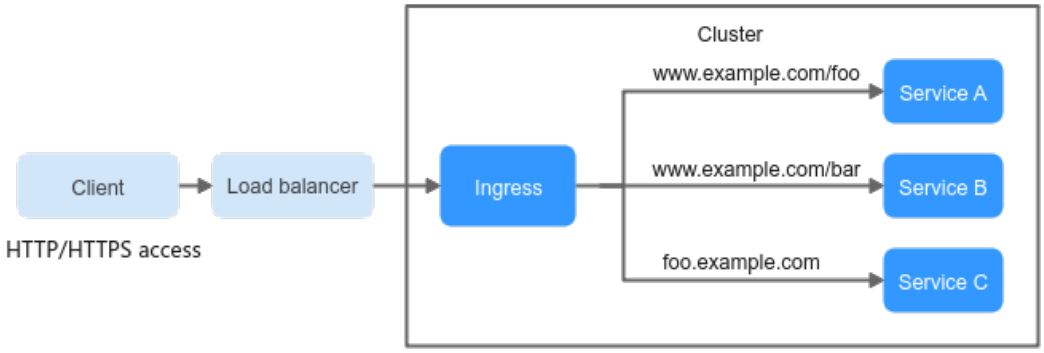
If you need to configure more advanced functions when creating a LoadBalancer Service, see .

Using ELB Ingresses in CCE

A Service is generally used to forward access requests based on TCP and UDP and provide layer-4 load balancing for clusters. However, in actual scenarios, if there is a large number of HTTP/HTTPS access requests on the application layer, the Service cannot meet the forwarding requirements. Therefore, the Kubernetes cluster provides an HTTP-based access mode, ingress.

An ingress is an independent resource in the Kubernetes cluster and defines rules for forwarding external access traffic. As shown in **Figure 6-2**, you can define forwarding rules based on domain names and paths to implement fine-grained distribution of access traffic.

Figure 6-2 Ingress diagram



For details about how to configure an ELB Ingress in a CCE cluster to control traffic, see

Configuring Advanced LoadBalancer Ingress Functions Using Annotations

You can add annotations to a YAML file for more advanced ingress functions. For details, see .

Migrating Data from a Bring-Your-Own Nginx Ingress to a LoadBalancer Ingress

LoadBalancer ingresses are implemented based on Huawei Cloud ELB. LoadBalancer ingresses offer better traffic management compared to bring-your-own Nginx ingresses because of the following advantages:

- Fully managed and O&M-free: ELB is a fully managed cloud service that requires no worker node, making it completely O&M-free.
- High availability: ELB enables active-active disaster recovery within a city across AZs. This ensures seamless traffic switchover in the event of a failure. Dedicated load balancers provide a comprehensive health check system to ensure that incoming traffic is only routed to healthy backend servers, improving the availability of your applications.
- Auto scaling: ELB can automatically scale to handle traffic spikes.
- Ultra-high performance: A single load balancer supports up to 1 million queries per second and tens of millions of concurrent connections.
- Integration with cloud services: ELB can run with various cloud services, such as WAF.
- Hot updates of configurations: Configuration changes can be updated in real-time without requiring a process reload. This helps to prevent disruptions to persistent connections.

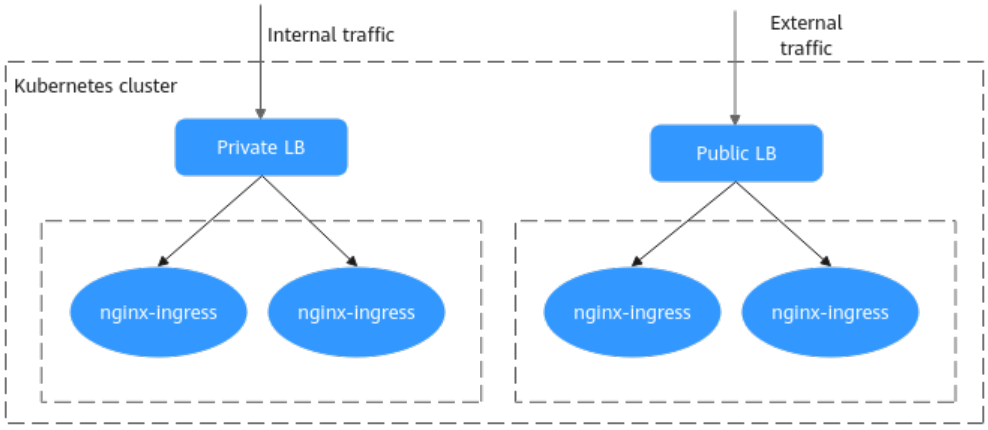
You can .

Deploying Nginx Ingress Controller in Custom Mode

NGINX Ingress Controller is a popular open-source ingress controller in the industry and is widely used. Large-scale clusters require multiple ingress controllers to distinguish different traffic. For example, if some services in a cluster

need to be accessed through a public network ingress, but some internal services cannot be accessed through a public network and can only be accessed by other services in the same VPC, you can deploy two independent Nginx Ingress Controllers and bind two different load balancers.

Figure 6-3 Application scenario of multiple Nginx ingresses



You can deploy multiple NGINX Ingress Controllers in the same cluster by referring to .

