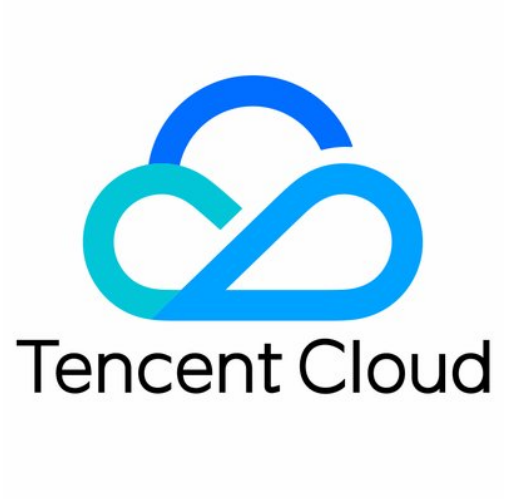
Gateway Load Balancer

Practical Tutorial

Product Documentation



Contents

Practical Tutorial
Easily Implementing Adaptation of a Third-Party Virtual Device with GWLB

Last updated：2024-12-16 17:21:44

Gateway Load Balancer (GWLB) is a load balancer running at the network layer. GWLB instances enable you to deploy, scale, and manage third-party virtual devices such as firewalls, intrusion detection and prevention systems, analysis systems, and visualization devices, with simpler operations and higher security. This document describes how to easily and efficiently implement adaptation of a third-party virtual device with GWLB.

Adaptation of the Third-Party Virtual Device

Supported Third-Party Virtual Devices

GWLB handles business traffic at the network layer and is independent of the device status. This design enables compatibility with a third-party virtual device, provided that the device supports GENEVE encapsulation/decapsulation and original data packets.

Adaptation Operation

To work with GWLB, the third-party virtual device needs to complete the following modifications:
Supports the Geneve protocol to exchange traffic with GWLB. Geneve encapsulation is necessary for transparent packet routing between GWLB and the device and sending of additional information (also known as metadata).
Supports encoding/decoding Geneve Type-Length-Value (TLV) pairs related to GWLB.
Responds to TCP health checks from GWLB.

Why Does the Third-Party Virtual Device Need to Support Geneve Encapsulation?

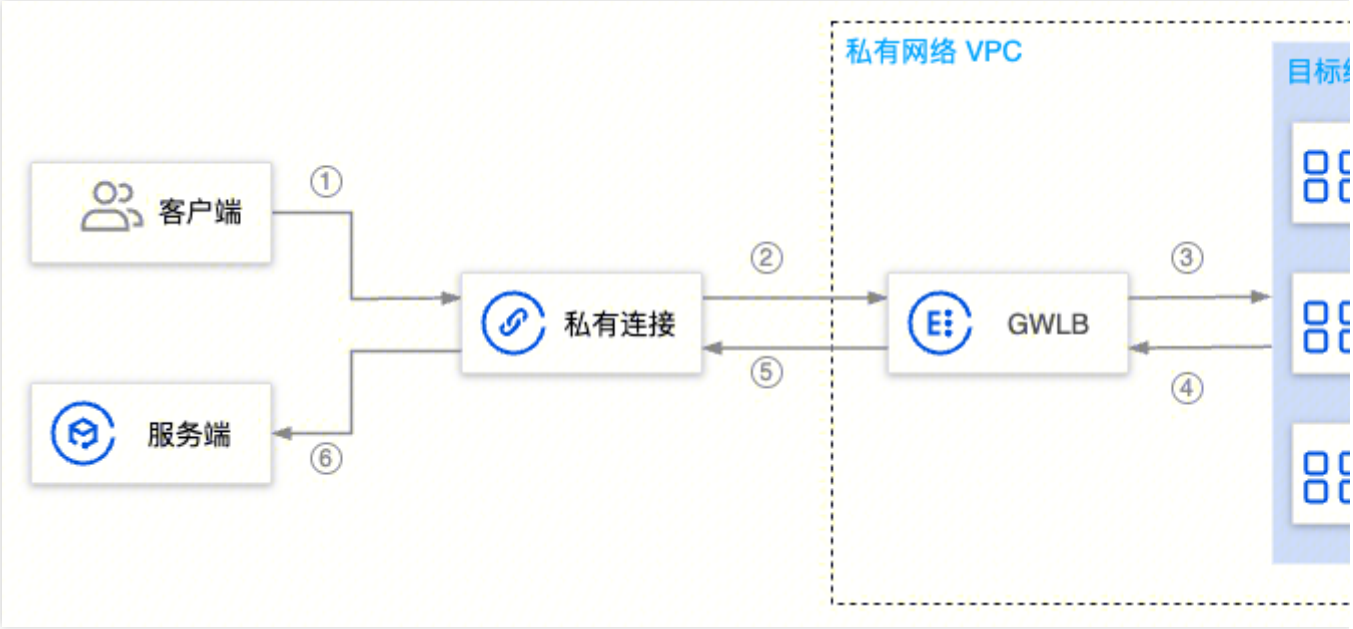
It is essential for maintaining the integrity of original packets and is a key feature provided by GWLB. Encapsulating the original packets into new L3 packets is the only feasible solution for routing packets between GWLB and the device. Because the source/destination IP address in such packets is different from the IP address of GWLB or the device, the packets will bypass GWLB or the device in normal VPC routing based on these IP addresses. Furthermore, to support multi-tenant devices with overlapping CIDRs, the devices need to identify the traffic source. GWLB also needs to track traffic and avoid mixing of the user traffic. GWLB can achieve this by using a TLV 3-tuple to send additional information (such as GWLBE/VPCE ID, Flow Direct, Flow Cookie, and Attachment ID) of each packet to the device.

The Geneve protocol (RFC 8926) is very flexible and allows transmitting such additional information. This scalable and customizable layer-3 encapsulation mechanism supports a wide range of use cases and simplifies the customer experience, for it does not require any changes to the source and destination devices. For details, see the Geneve TLV format below. As alternatives to Geneve encapsulation, VXLAN and GRE were evaluated, but due to fixed field sizes, they could not meet the above requirements.

How GWLB Works

Technology Architecture

As shown in the figure below, GWLB can be connected via Private Link to a GWLB endpoint in another VPC.



GWLB is divided into frontend and backend. The side connected to the Private Link is called the GWLB frontend, while the side connected to the destination device is called the GWLB backend. In the backend, GWLB acts as a CLB to route traffic to one of multiple equivalent destination devices. GWLB ensures the stickiness of bidirectional traffic to the destination device and also reroutes the traffic if the selected device becomes unhealthy. This document focuses on the backend feature, namely communication between GWLB and the destination device.
The packets sent from the source to the destination do not contain the GWLB IP as the destination IP address, but due to the route table configuration, they are routed to GWLB. To achieve transparent forwarding (that is, keep the original packets unchanged), GWLB uses Geneve to encapsulate the original packets and send them to the third-party virtual device or receive data packets from the third-party virtual device. The third-party virtual device also needs to decapsulate the Geneve TLV pairs.

GWLB is a packet-in/packet-out service. It does not maintain any application state or perform TLS/SSL decryption/encryption. These features are executed by the device itself.
When GWLB receives a new TCP/UDP flow, it uses a 3-tuple (source IP, destination IP, and transport protocol) for elastic hashing to select a healthy device from the target group. Subsequently, GWLB routes all (both forward and backward) packets of that flow to the same device (to maintain stickiness). For non-TCP/UDP flows, GWLB also uses the 3-tuple (source IP, destination IP, and transport protocol) to make forwarding decisions.

CLB Scheduling Algorithm

Elastic Hash

GWLB supports traffic scheduling with the symmetric hash algorithm based on the 3-tuple of source IP, destination IP address, and transport protocol. The traffic with the same 3-tuple will be scheduled to the same real server.

Load Balancing Flow Stickiness

Adding or removing instances in the target group: All traffic will be rehashed.
If a certain healthy instance in the target group becomes unhealthy: The traffic of the faulty node will be rehashed to other healthy nodes.
If a certain unhealthy instance in the target group becomes healthy: The traffic belonging to the recovered node will be redirected back to that node.

Health Examination

GWLB runs health checks periodically at a custom interval by sending TCP/PING packets to the device, which then responds to the TCP/PING packets. The details are as follows:
TCP: Successful connection is considered as passed.
PING: If the PING command is successful and an error message "port XX unreachable" is not returned by the real server within the response timeout period, the service is considered normal and the health check is successful.
The third-party virtual device should complete all the checks within the GWLB timeout. These checks assume that the device properly responding to TCP/PING packets (usually from its control plane) can also forward the packets to the destination through its data plane.

Data Forwarding

Step 1: When a private link (GWLBE) receives a packet from the source, it sends the packet to GWLB through the underlying PrivateLink technology. The packet stays on the VPC network and reaches GWLB.
Step 2: GWLB uses a 3-tuple (source IP, destination IP, and transport protocol) of the incoming packet and selects a specific device as the destination. Additionally, it generates a flow cookie and then stores the flow entry and its corresponding flow cookie in its flow table. Then, GWLB encapsulates the original packet with a Geneve header and embeds metadata in the form of a TLV 3-tuple.
Step 3: GWLB forwards the encapsulated packet to a specific device. It will forward bidirectional traffic with the same 3-tuple to this device during the lifecycle of the flow.

Step 4: The device should be configured with an IP interface that can receive UDP/IP packets. All packets forwarded to the device are routed through this IP interface.
Step 5: The device encapsulates the original packet with a Geneve header and embeds the same metadata initially received for the flow.
Step 6: After receiving data packets from the device, GWLB will remove the GENEVE encapsulation and then verify, query, and forward the incoming (internal) packets along with the metadata extracted from GENEVE. If the forwarding query fails, GWLB will discard the incoming packets.
Step 7: The packet traverses to GWLBE through the underlying PrivateLink technology. The packet stays on the VPC network and reaches GWLBE, which then transmits it to the destination based on the next hop in the route table.

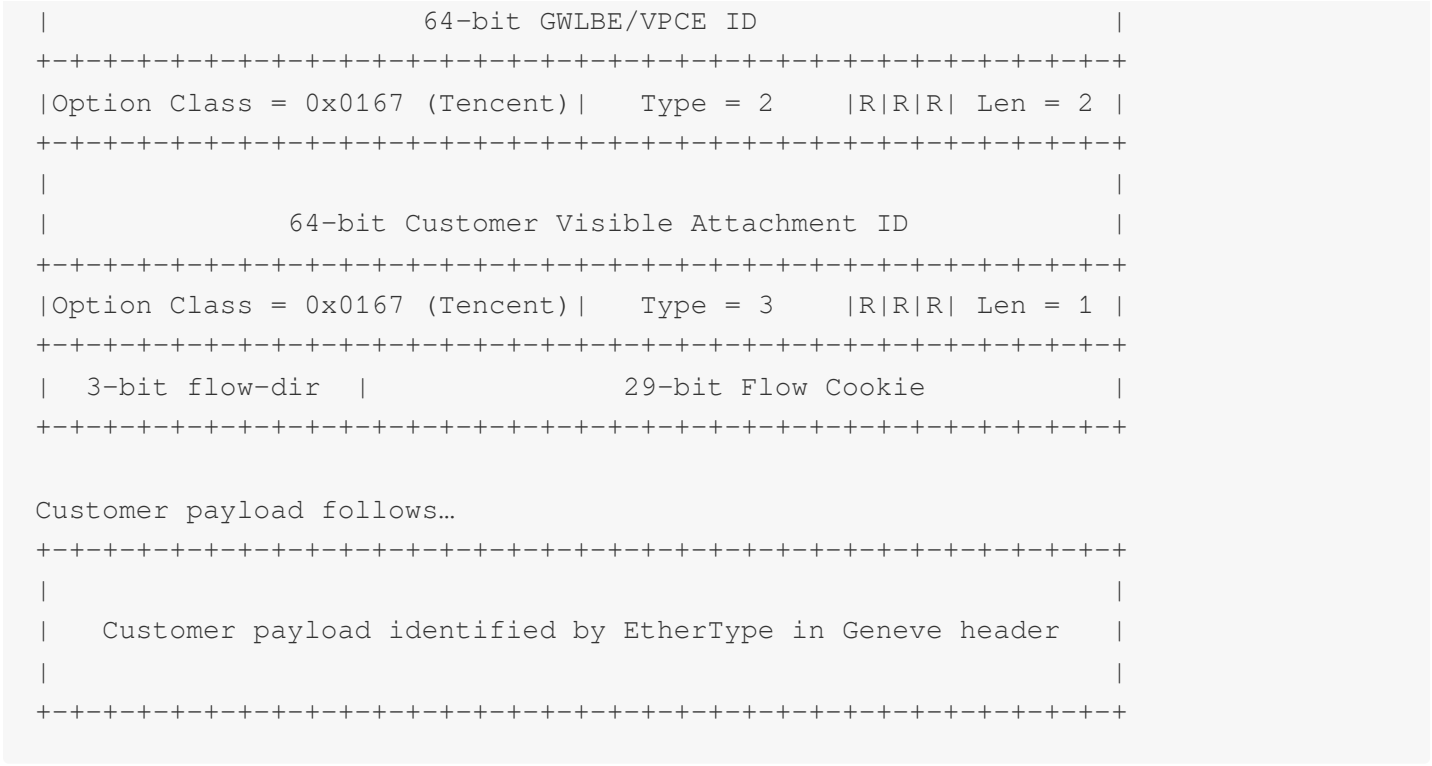
GWLB Data Format

The format of Geneve-encapsulated packets received by the device is shown as follows. For details of the Geneve header, see the Geneve protocol (RFC 8926).

```
Outer IPv4 Header:
0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|Version|  IHL  |Type of Service|          Total Length         |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|         Identification        |Flags|      Fragment Offset    |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|  Time to Live |Protocol=17 UDP|         Header Checksum       |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                     Outer Source IPv4 Address                 |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                   Outer Destination IPv4 Address              |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|       Source Port = xxxx      |    Dest Port = 6081 Geneve    |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|           UDP length          |         UDP Checksum          |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+

Outer Geneve Header:
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|V=0|Opt Len = 8|O|C|    Rsvd.  | EtherType = 0x0800 or 0x86DD  |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|    Virtual Network Identifier (VNI) = 0       |    Reserved   |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+

Outer Geneve Options: Tencent Gateway Load Balancer TLVs
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|Option Class = 0x0167 (Tencent)|   Type = 1    |R|R|R| Len = 2 |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                                                               |
```



Source Port: The Geneve source port is selected through 3-tuple hash.

GWLBE/VPCE ID: This is the 64-bit endpoint service string ID assigned to GWLBE (for example, if the endpoint service string ID is vpce-12345678, then it would be 12345678 here. Service providers should add a prefix to this ID to obtain the final ID). The device can use this identifier to associate packets with its configured private link. This GWLBE/VPCE ID can be used to determine the source VPC of the traffic. Each GWLBE can only belong to one VPC and should be mapped to the source VPC through the application service provider's management software. The device should return this ID "as it is".

Customer Visible Attachment ID: currently not in use and specified as 0.

Flow Dir: identifies the flow direction. The RS should return this cookie "as it is".

1: Access from the public network to a VPC network. Under this scenario, in the original IP packet for Geneve encapsulation, the source IP is the IP address of the public network and the destination IP is the EIP address of the VPC network.

2: Access from a VPC network to the public network. Under this scenario, in the original IP packet for Geneve encapsulation, the source IP is the EIP address of the VPC network and the destination IP is the IP address of the public network.

3: Retain.

4: Access between VPCs.

Flow Cookie: The flow cookie is a 29-bit flow ID generated by GWLB during traffic forwarding. The RS should return this cookie "as it is".

Response Process of the Third-Party Virtual Device

1. Encapsulate the original packet in a Geneve header.

2. Swap the source IP address (IP address of the third-party virtual device) and the destination IP address (IP address of the GWLB instance) in the outer IPv4 header.
3. Retain the original ports, not swapping the source port and destination port in the outer IPv4 header.
4. Update the IP checksum in the outer IPv4 header.
5. Return all fields in the Geneve header without modification.
6. Return all TLV fields (GWLBE ID, Flow Direct, Flow Cookie, and Attachment ID) in the Geneve Option without modification.

Note:

The length of the packet returned by the third-party virtual device to GWLB cannot be greater than the length of the received packet.

Test Verification

When the third-party virtual device supports the Geneve protocol, GWLB TLV encoding/decoding, and responding to health checks, test verification can be conducted.
You can first use a single device as the simplest test case. To check whether the device responds to health checks, you can enable data packet capture on the device to view the packet flow and check whether the data packets are in the expected format.

Implementing HA Across Multiple AZs

Last updated：2024-12-16 17:22:19

GWLB guarantees high availability (HA) from multiple dimensions such as system architecture and product configuration. Based on the business scenarios and requirements, you can select one from a variety of HA solutions for cross-region disaster recovery or intra-region cross-AZ disaster recovery.

GWLB Cluster HA

GWLB instances adopt cluster deployment to eliminate single points of failure of servers and enhance system redundancy, thereby ensuring the service stability. All GWLB instances have the cluster HA.
Tencent Cloud's gateway load balancing is implemented based on its own GWLB gateway, which features high reliability, strong scalability, high performance, and strong anti-attack capability. A single cluster can handle Tbps-level traffic and support millions of QPS, easily responding to various traffic distribution scenarios.

Single GWLB Instance HA

GWLB provides load balancing services with the SLA of 99.99%. GWLB deploys clusters in multiple AZs and the same GWLB instance is deployed across multiple AZs simultaneously. When a client accesses this GWLB instance, the traffic is automatically directed to the AZ cluster with the lowest latency and then forwarded to the real servers.
If a GWLB cluster in a specific AZ becomes unavailable, GWLB can automatically switch to another AZ and restore services within a very short time (about 30 seconds). If only the GWLB cluster fails, it does not affect the access traffic.
If the entire AZ fails (including the real server), the GWLB instance will detect exceptions of the real server through health checks and will not forward traffic to the real server in the failed AZ.

Real Server HA

GWLB determines the availability of real servers through health checks to avoid real server exceptions from affecting frontend businesses, thereby improving the overall availability of businesses.
After the health check feature is enabled, health checks will be performed regardless of the real server weight (including 0). You can view the "health status" of real servers on the **Target Group Instances** tab of the target group details page, or check the number of unhealthy RSs on the Monitoring tab of the target group details page. For details of the health check mechanism, see Health Check Overview.