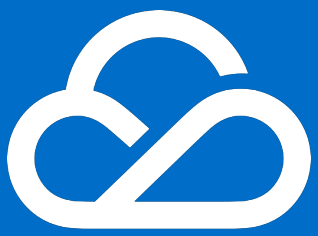
腾讯云私有网络

最佳实践

产品文档



腾讯云

【版权声明】

【商标声明】

【服务声明】

本文档意在向客户介绍腾讯云全部或部分产品、服务的当时的整体概况，部分产品、服务的内容可能有所调整。您所购买的腾讯云产品、服务的种类、服务标准等应由您与腾讯云之间的商业合同约定，除非双方另有约定，否则，腾讯云对本文档内容不做任何明示或模式的承诺或保证。

文档目录

最佳实践

使用Ipsec-tools搭建对端VPN网关连接VPC

腾讯云私有网络VPC可以通过加密的VPN通道连接客户IDC，只需在VPC及用户IDC中设置VPN网关及对端网关即可。客户如果暂时不打算使用cisco、juniper或H3C等厂家的硬件vpn设备，也可以使用开源软件在服务器上搭建对端网关。本文以在CentOS上安装ipsec-tools为例，介绍如何通过开源软件连接腾讯云VPC，建立混合云场景。

1. 环境说明

如上图所示，左边是您在腾讯云上建立的私有网络。为了将VPC与右边的客户IDC互通，可以利用公网在两者之间建立加密Ipsec VPN通道，保障传输数据的安全可靠。

首先您需要在腾讯云上建立您的私有网络，并根据您的需求规划子网、购买VPN网关，并设置好到需要互通的IDC网络的VP
C路由（注意路由设置中“
下一跳”请选择您购买的VPN网关）。具体操作步骤
请见创建私有网络及子网、向私有网络中添加云服务、关联子网路由、修改默认路由表和创建VPN网关
，将上图左边部分创建设置完成。

若您暂时不打算使用任何硬件厂家VPN设备，您也可按照本文档下面部分的操作方式，在位于IDC机房的一端使用Ipsec-tools开源工具搭建VPN网关。

2. 安装Ipsec-tools

系统环境要求，以CentOS 6.4为例：
Linux version 2.6.32-431.23.3.el6.x86_64
(mockbuild@c6b8.bsys.dev.centos.org) (gcc version 4.4.7 20120313
(Red Hat 4.4.7-4) (GCC)) #1 SMP Thu Jul 31 17:20:51 UTC 2014

根据系统平台不同选择使用ipsec-tools-0.8.0-25.3.x86_64.rpm包或者ipsec-tools-0.8.0-25.3.i686.rpm包，可以由下面的链接进入下载：

ftp://ftp.pbone.net/mirror/ftp5.gwdg.de/pub/opensuse/repositories/home:/aevseev/CentOS_CentOS-6/x86_64/ipsec-tools-0.8.0-25.3.x86_64.rpm

ftp://ftp.pbone.net/mirror/ftp5.gwdg.de/pub/opensuse/repositories/home:/aevseev/CentOS_CentOS-6/i686/ipsec-tools-0.8.0-25.3.i686.rpm

下载完成后，使用以下命令进行安装：

rpm -ivh ipsec-tools-0.8.0-25.3.x86_64.rpm

或

rpm -ivh ipsec-tools-0.8.0-25.3.i686.rpm

安装结束后，使用以下命令检查安装结果：

racoon –V

```
[root@VM_1_2_centos ~]# racoon -V
@(#)ipsec-tools 0.8.0 (http://ipsec-tools.sourceforge.net)

Compiled with:
- OpenSSL 1.0.0-fips 29 Mar 2010 (http://www.openssl.org/)
- IPv6 support
- Dead Peer Detection
- IKE fragmentation
- Hybrid authentication
- NAT Traversal
- Timing statistics
- Admin port
- Monotonic clock
```

3. 配置Ipsec-tools

需要配置的文件包括：

IPSec策略配置文件：/etc/racoon/setkey.conf
密钥配置文件：/etc/racoon/psk.txt
IKE配置文件：/etc/racoon/racoon.conf

3.1. 配置Ipsec策略

使用以下命令打开配置文件：

vi /etc/racoon/setkey.conf

设置如下信息：

假设您VPC的CIDR为10.100.2.0/24，VPC上VPN网关的IP地址为112.*.*.251。您IDC的CIDR为172.16.2.0/24，本地VPN设备的IP地址为112.*.*.152，则配置如下：

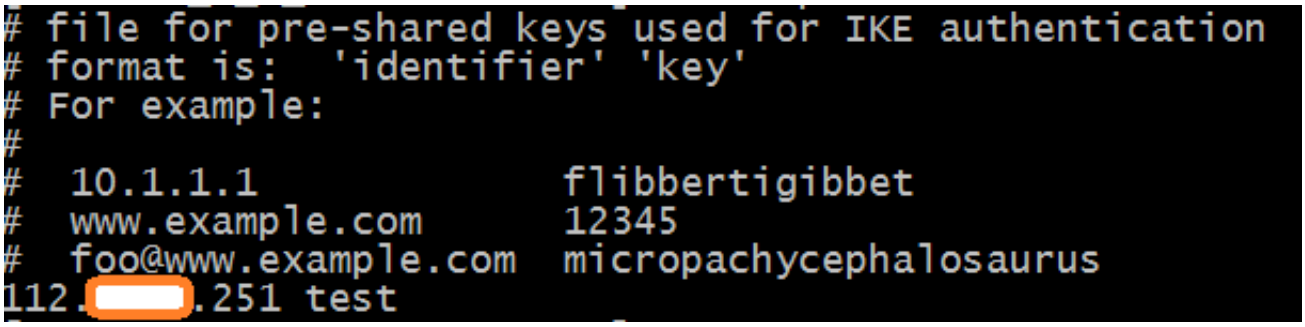
```
#flush SAD entries
flush;
#flush SPD entries
spdflush;

#add SP entries
spdadd 10.100.2.0/24 172.16.2.0/24 any -P in ipsec esp/tunnel/112.[redacted].251-112.[redacted].152/require;
spdadd 172.16.2.0/24 10.100.2.0/24 any -P out ipsec esp/tunnel/112.[redacted].152-112.[redacted].251/require;
```

3.2. 配置密钥

使用以下命令打开配置文件：

vi /etc/racoon/psk.txt

仍然假设您VPC上VPN网关的IP地址为112.*.*.251，预共享密钥为test，则psk.txt配置内容如下：



并执行以下命令：

chmod 600 psk.txt

chown root psk.txt

3.3. 配置IKE

使用以下命令打开配置文件：

vi /etc/racoon/racoon.conf

仍然假设您VPC上VPN网关的IP地址为112.*.*.251，您本地VPN设备的IP地址为112.*.*.152，则racoon.conf配置内容如下：

```
path include "/etc/racoon";
path pre_shared_key "/etc/racoon/psk.txt";
path certificate "/etc/racoon/cert";

remote anonymous
{
        exchange_mode main,aggressive;
        my_identifier address 112.[redacted].152;
        peers_identifier address 112.[redacted].251;
        #nat_traversal on;

        proposal {
                encryption_algorithm des;
                hash_algorithm sha1;
                authentication_method pre_shared_key;
                dh_group 1;
                lifetime time 24 hour;
        }
}

sainfo anonymous {
{
        encryption_algorithm aes 128;
        authentication_algorithm hmac_sha1;
        compression_algorithm deflate;
}
```

4. 启动Ipsec-tools

执行以下命令启动Ipsec-tools：

echo 1 > /proc/sys/net/ipv4/ip_forward

/usr/sbin/setkey -f /etc/racoon/setkey.conf

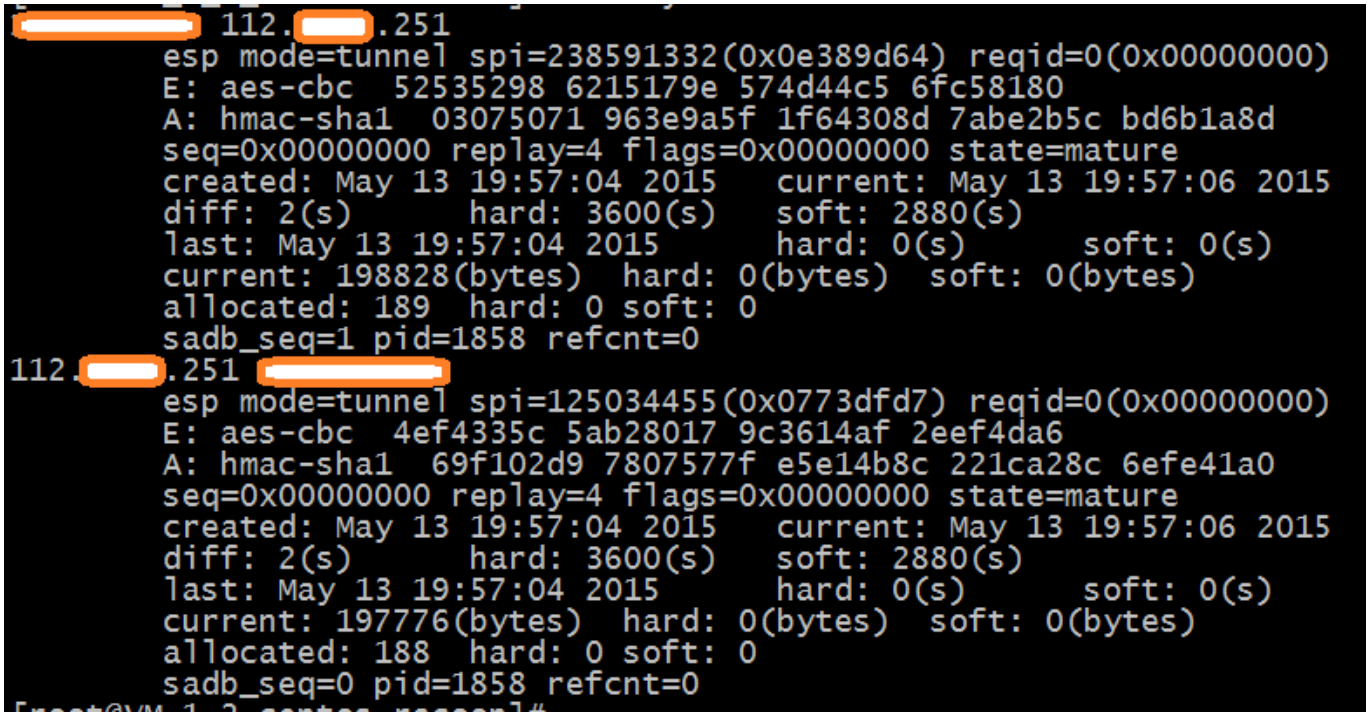
/usr/sbin/racoon -f /etc/racoon/racoon.conf

为保障设备重启后ipsec服务自动开启，同时需要将这三条命令写入/etc/rc.local文件中。

5. 检查通道状态

通过以下命令查看是否完成通道sa的协商：

setkey -D



在您的IDC网络中需要将目的IP为VPC的CIDR的报文路由到您的对端VPN网关，即前面介绍的配置Ipsec-tools的机器。

在IDC中的机器上ping您在VPC里的子机IP，检测通信是否顺利建立。

6. 其他问题

若经过设置后VPC与IDC无法进行加密通信，请按照以下步骤检查：

1) 检测VPC网关IP是否可达

Ping您在腾讯云上购买的VPN网关公网IP地址是否可达。

2) 是否缺少路由

通过以下命令检查是否有到VPC网段的路由或默认路由。

route –n

3) 是否存在防火墙过滤规则

通过以下命令检查是否有防火墙将IKE或内网通信报文过滤掉。

```
iptables -nvL
```

4) 检查防火墙NAT规则

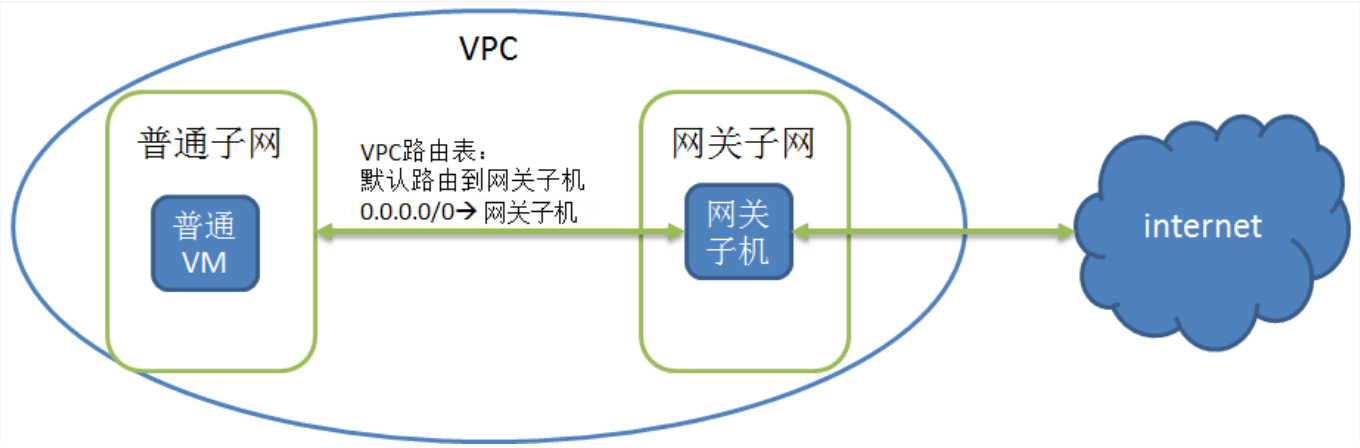
是否有将内网通信报文进行NAT导致命中不了Ipsec策略。

```
iptables -t nat -nvL
```

VPC内无外网IP主机通过公网网关访问外网的优化配置

1. 环境说明

客户在腾讯云VPC中的部分主机没有外网IP但需要访问外网时，可以通过购买公网网关主机作为其它没有外网IP的主机访问Internet的外网出口。公网网关主机将对出网流量进行源地址转换，所有其他主机访问外网的流量经过公网网关后，源IP都被转换为公网网关主机的IP地址，如下图：



普通CVM没有外网IP，但是可以借助公网网关访问Internet。

要完成上述架构，需要经过后文所述的几个步骤。

2. 架构搭建

2.1. 创建网关子网

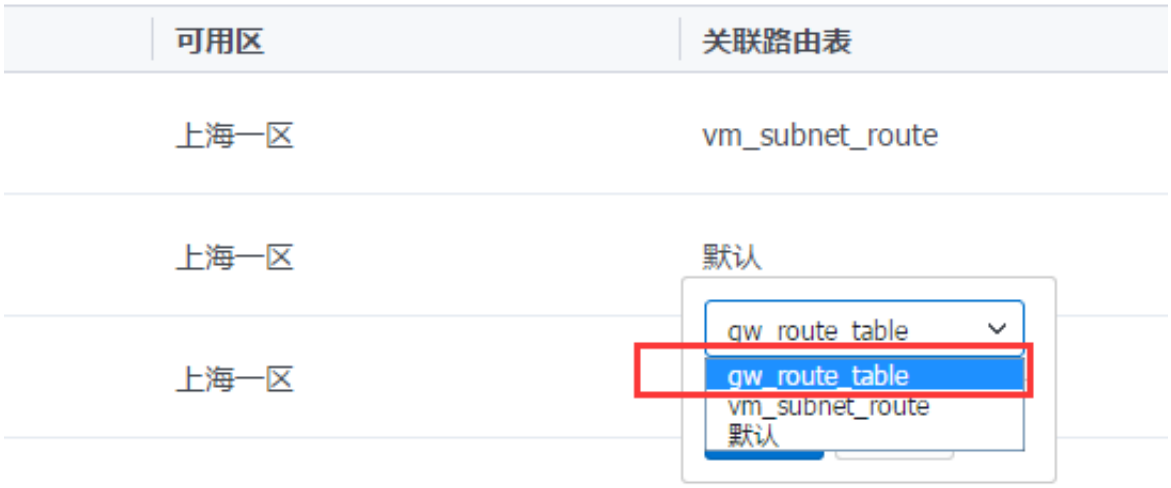
由于公网网关只能转发非所在子网的路由转发请求，因此公网网关主机不能和需要借助公网网关访问外网的CVM处于同一个子网下，所以需要先建立一个独立的网关子网。

2.2. 购买公网网关

在刚刚创建好的网关子网下购买公网网关，详细步骤请见购买私有网络的公网网关

2.3. 创建网关子网路由表

网关子网和普通子网不能关联同一张路由表，需要新建一张独立的网关路由表，且网关子网关联该路由表。



2.4. 配置普通子网路由表

配置普通子网的路由表，配置默认路由走公网网关主机，使得普通子网内主机能通过公网网关的路由转发能力访问外网。

3. 配置优化

公网网关主机会默认配置iptables的nat规则，以及打开kernel的ip_forward，基本的公网网关功能已经完全具备。建议经过下述配置，以达到更好的性能。

1) 通过以下命令将net.ipv4.ip_forward配置写到/etc/sysctl.conf文件中

```
echo "net.ipv4.ip_forward = 1" >> /etc/sysctl.conf
```

2) 通过以下命令将nf_conntrack配置参数调大

```
echo "echo 1048576 > /proc/sys/net/netfilter/nf_conntrack_max" >> /etc/rc.local
echo "echo 262144 > /sys/module/nf_conntrack/parameters/hashsize" >> /etc/rc.local
```

3) 设置转发的nat规则

```
echo "iptables -t nat -A POSTROUTING -o eth0 -j MASQUERADE" >> /etc/rc.local
```

4) 关闭时间戳选项

```
echo "iptables -t mangle -A POSTROUTING -p tcp -j TCPOPTSTRIP --strip-options timestamp"
>> /etc/rc.local
```

5) 设置公网网关的rps

在/usr/local/sbin/目录下新建脚本set_rps.sh，将以下代码写入脚本中：

```
#!/bin/bash
mask=0
i=0
cpu_nums=`cat /proc/cpuinfo |grep processor |wc -l`
if(($cpu_nums==0));then
  exit 0
fi

nic_queues=`cat /proc/interrupts |grep -i virtio0-input |wc -l`
if(($nic_queues==0));then
  exit 0
fi

echo "cpu number" $cpu_nums "nic queues" $nic_queues
```

```
mask=$(echo "obase=16;2^$cpu_nums - 1" |bc)
flow_entries=$(echo "$nic_queues * 4096" |bc)

echo "mask = "$mask
echo "flow_entries = "$flow_entries

#for i in {0..$nic_queues}
while (($i < $nic_queues))
do
  echo $mask > /sys/class/net/eth0/queues/rx-$i/rps_cpus
  echo 4096 > /sys/class/net/eth0/queues/rx-$i/rps_flow_cnt
  i=$(($i+1))
done

echo $flow_entries > /proc/sys/net/core/rps_sock_flow_entries
```

新建完成后执行以下命令：

```
chmod +x /usr/local/sbin/set_rps.sh
echo "/usr/local/sbin/set_rps.sh" >> /etc/rc.local
```

完成上述配置后，重启公网网关主机以使配置生效，并在无外网IP的子机上测试是否能够成功访问外网。

VPC 内通过 keepalived 搭建高可用主备集群

This document describes how to build a highly available master/slave cluster in Tencent Cloud VPC with keepalived.

Preface

To clearly describe how to implement keepalived on Tencent Cloud CVM, this document:

- Briefly introduce keepalived, and describe the difference between the applications of keepalived on CVMs and on the physical network.
- Explain how to implement the two modes below:
  - Non-standing-master mode: The priorities of the two devices to be chosen as the master device are the same.
  - Standing-master/slave mode: If no failure, try to keep one of the devices as the master. Compared to the non-standing-master mode, this mode limits the number of switchovers between the master and the slave. It is recommended to use the non-standing-master mode.
- This document provides multiple

  keepalived configuration and script files

  and

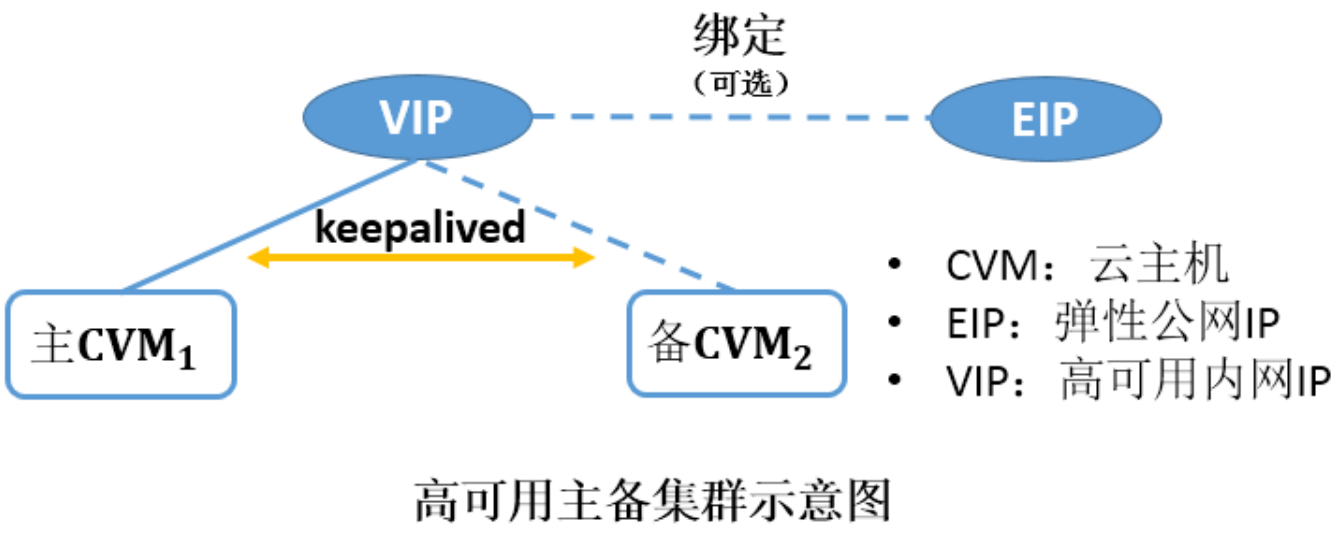
  configuration methods for different scenarios

  to help you carry out the implementation on the CVM.
- This document mainly introduces how to set up the VRRP Instance of keepalived for unicast VRRP messaging.

Basic Principle

Typically, the highly available master/slave cluster consists of two servers, with the master server being in the active state of a service (Active state) and the slave server being in the standby state of

the service (Standby state). The two servers share the same VIP (Virtual IP) which is only valid for the master device if no failure. In case of the master server failure, the slave server will take over the VIP to continue providing services. Highly available master/slave mode is widely used in MySQL master/slave switchover, Ngnix web access, and other scenarios.



高可用主备集群示意图

Keepalived on CVMs vs. Keepalived on Physical Network

In the traditional physical network, the master/slave state is determined by the keepalived VRRP protocol. Principle: the master device periodically sends gratuitous ARP messages to purge the MAC table or terminal ARP table of the uplink switch, triggering the VIP migration to the master device. The keepalived can be deployed in Tencent Cloud VPC to build a highly available master/slave cluster. The difference between this mode and the deployment in a physical network is:

- VIP migration through gratuitous ARP messages is not supported for now. VIP is bound to the master device by calling Cloud APIs.

Procedure Overview

1. Apply for a VIP, which can only migrate within a subnet (the master and slave servers must be in the same subnet).
2. Install and configure keepalived (Version 1.3.5 and above) on master and slave servers, and

modify the corresponding configuration files. If the primary IP of the local master and slave devices has only the private IP, modify the SDK host by following Step 9.

3. Edit the "notify" mechanism using keepalived, and use notify_action.sh and vip.py to call the Cloud APIs for the master/slave switchover.
4. Edit the track_script mechanism using keepalived, and use check_self.sh and vip.py to periodically check the script to improve its availability.
5. Assign a public IP to the VIP.(Optional)
6. Verify whether the VIP and the public IP are switched normally when the master/slave switchover occurs.

Note: This document provides several configuration and script files, so we first introduce the detailed modification steps for different scripts in this section for clear description later. Then you can solve problems you may encounter referring to the steps listed hereinafter, such as Cloud API usage, application for VIP. Here is the outlined modification steps:

```
/etc/keepalived/
|-- check_self.sh
|-- keepalived.conf
|-- notify_action.sh
|-- README
`-- vip.py
```

For the standing-master/slave mode:
Master device: (Standing master device)

1. Install keepalived and assign a public IP or EIP to the primary ENI.
2. Move the files in the current directory to the configuration directory /etc/keepalived/ used by keepalived, and add the executable permission with the command of chmod +x /etc/keepalived/*.sh; chmod -x /etc/keepalived/keepalived.conf.
3. Modify keepalived.conf:

0) state Initial role; Input Master for the master device, and Backup for the slave device.
1) interface Change it to the ENI name of the local device, such as eth0.
2) priority The value of the master should be greater than that of the slave, for example, 50 for the master and 30 for the slave.

3) unicast_src_ip Change it to the private IP of the local device.

4) unicast_peer Change it to the private IP of the peer device.

5) virtual_ipaddress Change it to the private network VIP.

6) track_interface Change it to the ENI name of the local device, such as eth0.

4. Modify vip.py

1) Line 12 interface Change it to the private IP of the local device. This IP must be bound with a public IP, otherwise modify SDK host by following Step 9.

2) Line 13 vip Change it to your VIP.

3) Line 14 thisNetworkInterfaceId Change it to the ID of the local CVM ENI.

4) Line 15 thatNetworkInterfaceId Change it to the ID of CVM ENI of the peer device.

5) Line 16 vpcId Change it to your VPC ID.

6) Line 19-22 Input your secretId and secretKey.

5 Modify check_self.sh:

1) Line 3 vip Change it to the private network VIP.

2) Line 4 interface Change it to the ENI name of the local device.

Slave device: (Standing slave device)

Similar to operations for the master device

==========

Steps for using stable: (the priorities of the two devices to be chosen as the master device are the same, non-standing-master/slave mode) (Recommended!)

The operations for the two devices are the same:

1. Install keepalived and assign a public IP or EIP to the primary ENI.

2. Move the files in the current directory to the configuration directory /etc/keepalived/ used by keepalived and modify the permission with the command chmod 744 /etc/keepalived/*.sh; chmod 644 /etc/keepalived/keepalived.conf.

3. Modify keepalived.conf:

0) state Initial role. Input Backup for both the master device and the slave device.

1) interface Change it to the ENI name of the local device, such as eth0.

2) priority The two devices are configured with the same integer, such as 50.

3) unicast_src_ip Change it to the private IP of the local device.

4) unicast_peer Change it to the private IP of the peer device.

5) virtual_ipaddress Change it to the private network VIP.

6) track_interface Change it to the ENI name of the local device, such as eth0.

4. Modify vip.py

1) Line 12 interface Change it to the private IP of the local device. This IP must be bound with a public IP, otherwise modify SDK host by following Step 9.

2) Line 13 vip Change it to your VIP.

3) Line 14 thisNetworkInterfaceId Change it to the ID of the local CVM ENI.

4) Line 15 thatNetworkInterfaceId Change it to the ID of CVM ENI of the peer device.

5) Line 16 vpcId Change it to your VPC ID.

6) Line 19-22 Input your secretId and secretKey.

5 Modify check_self.sh:

1) Line 3 vip Change it to the private network VIP.

2) Line 4 interface Change it to the ENI name of the local device.

Notes:

1. The script log will be written to /var/log/keealived.log, and takes up your disk space. You can clear the accumulated logs with logrotate and other tools.

2. The log related to Keepalived process will be written to /var/log/message.

Detailed Steps

Step 1. Apply for VIP

Apply for a VIP in a subnet (the IP applied by users within the VPC can be used as VIP). A VIP can be applied via the console or Cloud APIs. A VIP is bound to an ENI (including a primary ENI and a secondary ENI), and a primary ENI is assigned to each CVM in VPC by default when it's created, so you can apply for a VIP for the primary ENI bound to the master server:

- Via Console: Click to view ENI (Recommended).

Note:

1. After the subsequent configuration is completed and the keepalived service is enabled on the master/slave devices, the VIP can be found to be valid for the master device, and the VIP or the public VIP can respond to the ping command from other servers in the VPC. (Meanwhile, you can use security groups for the network isolation of your master/slave CVMs. It is recommended to set the all-pass security group for the master/slave CVM in the experimental stage).
2. After the applied VIP is available, it cannot be configured automatically on the ENI of the CVM, however, the VPC management platform has developed VIP-related features for you.
   1) If the VIP is not applied to keepalived, you need to configure the private IP in the CVM after it is assigned in order for this VIP to be visible in the CVM. Click to view ENI.
   2) The keepalived configured in this document can help you configure the VIP on the CVM ENI to make it visible in the CVM.

Step 2. Install keepalived (Version 1.3.5 and Above) on Master and Slave Servers.

- Install
  Take CentOS as an example:

```
yum -y install keepalived
```

Step 3. Determine Requires of the Master/Slave Servers

This document will introduce the two modes in parallel:

- Non-standing-master mode: The priorities of the two devices to be chosen as the master device are the same;

- Standing-master/slave mode: If no failure, try to keep one of the devices as the master.

Compared to the non-standing-master mode, this mode limits the number of switchovers between the master and the slave. It is recommended to use the non-standing-master mode.

Step 4. Modify keepalived.conf

- Modifying the configuration file

For the standing-master/slave mode, take the keepalived.conf modifications for the master server as an example:

0) state Initial role; Input Master for the master device, and Backup for the slave device.

1) interface Change it to the ENI name of the local device, such as eth0.

2) priority The value of the master should be greater than that of the slave, for example, 50 for the master and 30 for the slave.

3) unicast_src_ip Change it to the private IP of the local device.

4) unicast_peer Change it to the private IP of the peer device.

5) virtual_ipaddress Change it to the private network VIP.

6) track_interface Change it to the ENI name of the local device, such as eth0.

For the non-standing-master/slave mode, the keepalived.conf modifications of two devices are the same:

0) state Initial role. Input Backup for both the master device and the slave device.

1) interface Change it to the ENI name of the local device, such as eth0.

2) priority The two devices are configured with the same integer, such as 50.

3) unicast_src_ip Change it to the private IP of the local device.

4) unicast_peer Change it to the private IP of the peer device.

5) virtual_ipaddress Change it to the private network VIP.

6) track_interface Change it to the ENI name of the local device, such as eth0.

> Note: It is important to configure the unicast mode, that is, to specify the IP address of the peer device.

```
! Configuration File for keepalived

global_defs {
  notification_email {
  acassen@firewall.loc
  failover@firewall.loc
  sysadmin@firewall.loc
  }
  notification_email_from Alexandre.Cassen@firewall.loc
  smtp_server 192.168.200.1
  smtp_connect_timeout 30
  router_id LVS_DEVEL
  vrrp_skip_check_adv_addr
  vrrp_garp_interval 0
  vrrp_gna_interval 0
}

vrrp_script checkhaproxy
{
  script "/etc/keepalived/check_self.sh"
  interval 5
}

vrrp_instance VI_1 {
  # Select proper parameters for the master and slave servers
  state MASTER #Master device #Modification item. "Master" is for the master device and "Backup" is for the slave device.
#state BACKUP #Slave device
  interface eth0 #Change it to the ENI name of the local device, such as eth0.
  virtual_router_id 51
  nopreempt #Non-preemptive mode
# preempt_delay 10
  priority 50 #The priority of the master should be greater than that of the slave, for example, 50 for the master and 30 for the slave.
```

```
  advert_int 1
  authentication {
  auth_type PASS
  auth_pass 1111
  }
  unicast_src_ip 10.0.1.17 #The private IP of the local device
  unicast_peer {
  10.0.1.16 #The IP address of the peer device, for example, 10.0.0.1.
  }
  virtual_ipaddress {
  10.0.1.100 #The private VIP
  }

  notify_master "/etc/keepalived/notify_action.sh MASTER"
  notify_backup "/etc/keepalived/notify_action.sh BACKUP"
  notify_fault "/etc/keepalived/notify_action.sh FAULT"
  notify_stop "/etc/keepalived/notify_action.sh STOP"
  garp_master_delay 1
  garp_master_refresh 5

  track_interface {
  eth0 #Change it to the ENI name of the local device, such as eth0.
  }

  track_script {
  checkhaproxy
  }
}
```

Step 5. Modify notify_action.sh to help the CVMs switch roles in case of failure

Modify notify_action.sh for the standing-master/slave mode:

1) N/A

Modify notify_action.sh for the non-standing-master/slave mode:

1) N/A

```
#!/bin/bash
#/etc/keepalived/notify_action.sh
log_file=/var/log/keepalived.log
log_write()
{
  echo "[`date '+%Y-%m-%d %T'`] $1" >> $log_file
}

[ ! -d /var/keepalived/ ] && mkdir -p /var/keepalived/

case "$1" in
  "MASTER" )
  echo -n "$1" > /var/keepalived/state
  log_write " notify_master"
  echo -n "0" > /var/keepalived/vip_check_failed_count
  python /etc/keepalived/vip.py migrate &
  ;;

  "BACKUP" )
  echo -n "$1" > /var/keepalived/state
  log_write " notify_backup"
  ;;

  "FAULT" )
  echo -n "$1" > /var/keepalived/state
  log_write " notify_fault"
  ;;

  "STOP" )
  echo -n "$1" > /var/keepalived/state
```

```
  log_write " notify_stop"
  ;;
  *)
  log_write "notify_action.sh: STATE ERROR!!!"
  ;;
esac
```

Step 6. Modify vip.py to help you migrate the VIP between CVMs and query the current IP of the local device.

Vip.py: Develop the master/slave switchover program using cloud APIs, and switch the IP addresses by calling the cloud API of private IP migration. Take Python as an example:

1) Download the Python SDK

- Installation via pip
  - yum install python-pip
  - pip install qcloudapi-sdk-python
- github source code download
  - Go to github to view Python SDK >>
  - Click to download Python SDK >>

Please read

README.md

carefully and download the SDK to the directory of

/etc/keepalived

.

2) Modify host for a complete private network environment

- If the primary IPs of your master and slave CVMs are both private IPs, modify the host used by SDK by following Step 9 to call the cloud API via the private network.

3) Obtain the cloud API key:

4) Develop vip.py switchover program for calling the cloud API based on the SDK, and save vip.py to the

/etc/keepalived

directory to call the cloud API for private IP migration:

- Find the primary ENI ID under the ENI tag on the CVM details page of the Console:

- Modify and use the code parameters (pay attention to the strict restrictions on indentation in python)

Modify vip.py for the standing-master/slave mode:

1) Line 12 interface Change it to the private IP of the local device. This IP must be bound with a public IP, otherwise modify SDK host by following Step 9.

2) Line 13 vip Change it to your VIP.

3) Line 14 thisNetworkInterfaceId Change it to the ID of the local CVM ENI.

4) Line 15 thatNetworkInterfaceId Change it to the ID of CVM ENI of the peer device.

5) Line 16 vpcId Change it to your VPC ID.

6) Line 19-22 Input your secretId and secretKey.

Modify vip.py for the non-standing-master/slave mode:

1) Line 12 interface Change it to the private IP of the local device. This IP must be bound with a public IP, otherwise modify SDK host by following Step 9.

2) Line 13 vip Change it to your VIP.

3) Line 14 thisNetworkInterfaceId Change it to the ID of the local CVM ENI.

4) Line 15 thatNetworkInterfaceId Change it to the ID of CVM ENI of the peer device.

5) Line 16 vpcId Change it to your VPC ID.

6) Line 19-22 Input your secretId and secretKey.

```
#!/usr/bin/python
# -*- coding: utf-8 -*-
#/etc/keepalived/vip.py

"""
# Note: The indention of python code must be consistent with that of this document.
Installation via pip:
```

After Python is installed, perform the following steps:
Step 1: yum install python-pip
Step 2: pip install qcloudapi-sdk-python
Step 3: Change "from src.QcloudApi.qcloudapi import QcloudApi" to "from QcloudApi.qcloudapi import QcloudApi" in the code.
Step 4: Edit and save the code in /etc/keepalived for use.

Via SDK source code download:
After Python is installed, perform the following steps:
Step 1: Download python-sdk from https://github.com/QcloudApi/qcloudapi-sdk-python
or execute "wget https://github.com/QcloudApi/qcloudapi-sdk-python/archive/master.zip" in Linux.
Step 2: Save the downloaded SDK package to the directory /etc/keepalived and decompress it. Change the decompressed folder name to src, and create a blank file named as __init__.py in the src folder.
Step 3: Save the following python code as vip.py to the same level directory with src of SDK, and edit its contents for use.

For more information on specific parameters, please see
https://cloud.tencent.com/doc/api/245/1361.
"""

```
#Installation via pip:
import os
import time
import json
import sys
from QcloudApi.qcloudapi import QcloudApi

# The primary ENI and IP of the current device
interface = {"eth0":"10.0.1.17"} #This IP must have a public IP
vip = "10.0.1.100" #Change it to the private VIP of the local device
thisNetworkInterfaceId = 'eni-pvsvph0u' #The ENI ID after IP migration (the ENI ID of the local device)
thatNetworkInterfaceId = 'eni-qnxioxyi' #The ENI ID before IP migration (the ENI ID of the peer
```

master device)

vpcId = 'vpc-1yxuk010' #vpcId

config = {
'Region': 'bj', #Your region
'secretId': 'Your secretId', #Your secretId
'secretKey': 'Your secretKey', #Your secretKey
'method': 'post'
}

log = open('/var/log/keepalived.log', 'a+')
state_file = open('/var/keepalived/state', 'r')

def get_now_time():
return time.strftime('[%Y-%m-%d %H:%M:%S]',time.localtime(time.time())) + '[pid' + str(os.getpid()) + ']'

def log_write(message=''):
log.write(get_now_time() + " " + str(message) + "\n")

def get_ip():
f = os.popen('ip addr show dev %s | grep %s | awk \'{print $2}\' | awk -F/ \'{print $1}\'' % (interface.keys()[0] , interface.values()[0]))
return f.read().strip()

def migrateVip():
module = 'vpc'
action = 'MigratePrivateIpAddress'
params = {
'vpcId': vpcId, #vpcId. No need to change this line.

```
'privateIpAddress': vip, #VIP. No need to change this line.
'oldNetworkInterfaceId': thatNetworkInterfaceId, #The ENI ID before IP migration (The ENI ID of the peer master device). No need to change this line.
'newNetworkInterfaceId': thisNetworkInterfaceId #The ENI ID after IP migration (The ENI ID of the local device). No need to change this line.
}

log_write(sys.argv[1])
log_write(" try set vip.")
retry_times_when_mgr_ip_got = 4
exceptimes = 0
get_ip_times = 0
time.sleep(0.5)
while get_ip_times < 5:
log_write(" get_ip=" + get_ip())
if get_ip()==interface.values()[0]:
log_write(" now set vip.")
try:
service = QcloudApi(module, config)
ret = service.generateUrl(action, params)
log_write(" generateUrl ret " + ret)
i = 0
while i < retry_times_when_mgr_ip_got:
check_vip_str = queryVip()
if check_vip_str == "true":
break
state_file.seek(0)
state = state_file.readline()
if state != 'MASTER':
break
ret = service.call(action, params)
ret_json = json.loads(ret)
log_write(" call ret " + ret)
#log_write(" last_code_mark: " + str(ret_json.get("code")))
```

```
  if ret_json.get("code") == 0:
  log_write(" set done")
  break
  if ret_json.get("code") == 6300:
  break
  i = i + 1
  time.sleep(2)
  if i >= retry_times_when_mgr_ip_got:
  log_write(" set vip failed")
  break
  except Exception, e:
  log_write(' exception:' + str(e))
  exceptimes = exceptimes + 1
  if exceptimes > 3:
  break
  time.sleep(0.5)
  get_ip_times = get_ip_times + 1
  log_write("vip.py checks vip: is this cvm holding the vip? " + queryVip())


def queryVip():
  module = 'vpc'
  action = 'DescribeNetworkInterfaces'
  params = {
  "networkInterfaceId": thisNetworkInterfaceId #The ENI ID of your local device. No need to change
this line.
  }

  result = 'true'
  return_json_str = None
  try:
  service = QcloudApi(module, config)
  ret = service.generateUrl(action, params)
  ret = service.call(action, params)
```

```
  return_json_str = ret
  ret_json = json.loads(ret)
  if ret_json.get("code") == 0:
  eni_data = ret_json['data']['data'][0]['privateIpAddressesSet']
  privateIpAddressSet = set([k['privateIpAddress'] for k in eni_data])
  if len(privateIpAddressSet) > 0 and vip not in privateIpAddressSet:
  log_write(" vip not in master in qcloud")
  result = 'false'
  else:
  log_write("call ret: " + return_json_str)
  log_write("attempt query vip failed")
  except Exception, e:
  log_write("call ret: " + return_json_str)
  log_write(' exception:' + str(e))
  exceptimes = exceptimes + 1
  return result



def print_help():
  log_write(
  '''
  ./vip.py migrate
  migrate your vip

  ./vip.py query
  query that if this cvm hold your vip in tencent cloud
  return: true or false
  ''')

if __name__ == '__main__':
  if len(sys.argv) == 1:
  log_write("vip.py: parameter num is 0")
  print_help()
```

```
elif sys.argv[1] == 'migrate':
migrateVip()
log_write()
elif sys.argv[1] == 'query':
print queryVip()
else:
log_write("vip.py: misMatched parameter")
print_help()
```

Step 7. Modify check_self.sh to improve the troubleshooting capability of keepalived.

Modify check_self.sh for the standing-master/slave mode:
1) Line 3 vip Change it to the private network VIP.
2) Line 4 interface Change it to the ENI name of the local device.
Modify check_self.sh for the non-standing-master/slave mode:
1) Line 3 vip Change it to the private network VIP.
2) Line 4 interface Change it to the ENI name of the local device.

```
#!/bin/bash
#/etc/keepalived/check_self.sh
vip=10.0.1.100 #Change it to your private VIP
interface=eth0 #Your network API name

state_file=/var/keepalived/state
vip_last_check_result_file=/var/keepalived/vip_last_check_result
query_vip_asker=/etc/keepalived/query_vip.py
vip_migrater=/etc/keepalived/vip.py
vip_operater=/etc/keepalived/vip.py
state=`cat $state_file`


log_file=/var/log/keepalived.log
```

```
log_write()
{
  echo "[`date '+%Y-%m-%d %T'`] $1" >> $log_file
}

[ ! -d /var/keepalived/ ] && mkdir -p /var/keepalived/
[ ! -f $vip_last_check_result_file ] && touch $vip_last_check_result_file
[ ! -f $state_file ] && echo -n "FAULT" > $state_file

CMD=`ip addr show dev $interface | grep $vip | awk '{print $2}' | awk -F/ '{print $1}'| wc -l`

case $state in
  "MASTER")
  if [ ${CMD} -ne 1 ]; then
  log_write "it is detected no vip on nic in cvm in MASTER state, add vip on this nic"
  ip addr add $vip dev $interface
  echo -n "false" > $vip_last_check_result_file
  else
  is_vip_in_master=`timeout 3 python $vip_operater query`
  if [ "x$is_vip_in_master" == "xfalse" ]; then
  echo -n "false" > $vip_last_check_result_file
  python $vip_operater migrate &
  elif [ "x$is_vip_in_master" == "xtrue" ]; then
  vip_last_check_result=`cat $vip_last_check_result_file`
  [ "x$vip_last_check_result" == "xfalse" ] && log_write " vip_check pass"
  echo -n "true" > $vip_last_check_result_file
  else
  log_write "$vip_operater check vip time out"
  fi
  fi
  exit 0
  ;;

  *)
```

```
 if [ ${CMD} -ne 0 ]; then
 sleep 2
 CMD=`ip addr show dev eth0 | grep $vip | awk '{print $2}' | awk -F/ '{print $1}'| wc -l`
 if [ ${CMD} -ne 0 ]; then
 log_write "detect vip in non-MASTER status, so ystemctl restart keepalived"
 ip addr del $vip dev $interface
 systemctl restart keepalived &
 exit 1
 fi
 fi
 exit 0
 ;;
esac
```

Step 8. (Optional) Assign a public IP to the VIP

- Console: Apply for an EIP on the console, and then bind it to the private VIP applied in Step 1. The operation procedure is similar to Step 1.

Step 9. The primary IPs of your master and slave CVMs are both private IPs (complete private network environment)

The default host of the cloud API Python SDK is the public network host, so the SDK cannot be accessed via the private network. If the primary IPs of your master and slave CVMs are both private IPs, you need to modify Python SDK, that is, to change the host to the access domain name of the private cloud API. Here is the detailed modification steps:

- Check the SDK installation method. Is it installed using pip or by downloading the source code to the directory /etc/keepalived/?
- Determine the path in which the host file to be modified resides based on the installation method.
    - For the installation using source code, the path is /etc/keepalived/src/QcloudApi/modules/vpc.py.

    - For the installation using pip, the path is /usr/lib/pythonX.Y/site-packages/QcloudApi/modules/vpc.py (pythonX.Y needs to be changed to the actual value, such as python2.6).
- Change

  requestHost = 'vpc.api.qcloud.com'

  to

  requestHost = 'vpc.api.tencentyun.com'

  .

Step 10. Verify whether the VIP and public IP are switched normally when the master/slave switchover occurs

1. Enable keepalived:

   /etc/init.d/keepalived start

   or

   systemctl start keepalived

   or

   service keepalived start

2. Verify the disaster recovery capability of master/slave switchover: simulate the CVM failure by restarting the keepalived process/the server or using other methods to check whether the VIP can be migrated. The corresponding logs will be written in /var/log/keepalived.log. You can also view the interval from the network suspension to recovery by pinging VIP or its EIP.

Note:

1) It takes several seconds to migrate an IP to a new server because of the asynchronous method of cloud APIs. If the failure time of the master server is very short in the standing-master/slave mode, the master/slave switchover may occur twice for a short time, but it may take a long time (10s) for the VIP to be re-migrated to the recovered master server.

2) The script log will be written to

/var/log/keealived.log

, and takes up your disk space. You can clear the accumulated logs with logrotate and other tools. The log related to the keepalived process will be written to

/var/log/message

.