负载均衡

负载均衡监听器

产品文档

腾讯云

文档目录

负载均衡监听器
- 负载均衡监听器概述
- 协议支持及端口配置
- 轮询方式
- 会话保持
- 健康检查
- 证书配置
- 应用型 CLB 转发规则及转发组配置说明
- 应用型 CLB 支持 SNI 绑定多个证书
- 公网应用型 CLB 重定向配置

负载均衡监听器
负载均衡监听器概述

最近更新时间：2018-01-26 19:13:23

负载均衡服务主要由负载均衡监听器提供。监听器负责监听负载均衡实例上的请求，执行策略分发流量至后端服务器等服务。

负载均衡监听器使用前端（客户端到负载均衡器）连接的协议和端口及后端（负载均衡器到后端实例）连接的端口进行配置。同时，负载均衡监听器还可以配置相应的 会话保持 和 健康检查 策略。

负载均衡监听器可以通过监听负载均衡实例上的四层和七层请求，并将这些请求分发到后端服务器上进行处理。四层和七层负载均衡的区别，主要体现在对后台的服务器进行负载均衡时，是依据四层的信息还是七层的信息来决定如何转发流量。其中四层为传输层协议，主要通过 VIP+端口接受请求并分配流量到后端服务器；七层为应用层协议，则基于 URL、HTTP头部等 应用层信息进行流量分发。

支持的协议类型

典型的 Web 应用程序之间的通信需要经由网络的各个分层，每层都会提供特定的通信功能。依据 OSI 网络模型，各个分层中都有标准的通信格式。腾讯云负载均衡涉及网络模型中的 四层（传输层） 和 七层（应用层）。

腾讯云负载均衡支持以下协议的请求转发：

- HTTP（应用层）
- HTTPS（应用层）
- TCP（传输层）
- UDP（传输层）

四层协议

如果使用四层协议转发，负载均衡实例会直接将请求转发到后端实例，而不修改任何数据包。负载均衡收到请求之后，会尝试在监听器配置中指定的端口上打开与后端实例的 TCP 连接。

七层协议

如果前端和后端连接均使用七层协议转发，负载均衡器会解析请求中有意义的应用层内容，并根据其内容选择后端服务器。因此，七层负载均衡器需要先代理后端服务器和客户端建立连接(三次握手)后，才可能接受到客户端发送的

真正应用层内容的报文，然后再根据该报文中的特定字段，再加上负载均衡设备设置的服务器选择方式，决定最终选择的内部服务器。

因为负载均衡位于客户端和服务器之间，因此后端服务器访问日志中将仅含有负载均衡器的 IP 地址。如需查看客户端的实际 IP 地址，需要使用 X-Forwarded-For 请求头。具体内容可参考 获取客户端 IP。

HTTPS 监听器的安全机制

HTTPS 是一种安全的 HTTP 连接，通过使用 SSL 证书来确保服务器端和客户端的可信度。负载均衡通过证书解密来自客户端的请求，然后再将请求发送到后端实例上。有关更多信息，请参阅 SSL 原理说明。

四层负载均衡和七层负载均衡的区别

四到七层负载均衡，就是在对后台的服务器进行负载均衡时，依据四层的信息或七层的信息来决定怎么样转发流量。

四层的负载均衡，就是通过三层的IP地址（VIP），然后加四层的端口号，来决定哪些流量需要做负载均衡，对需要处理的流量进行NAT处理，转发至后台服务器。

七层的负载均衡，就是在四层的基础上，再考虑应用层的特征（如HTTP头部、URL等），比如同一个Web服务器的负载均衡，除了根据VIP加80端口辨别是否需要处理的流量，还可根据七层的URL、浏览器类别、语言来决定是否要进行负载均衡。七层负载均衡也称为“内容交换”，也就是主要通过报文中的真正有意义的应用层内容，再加上负载均衡设备设置的服务器选择方式，决定最终选择的内部服务器。负载均衡设备如果要根据真正的应用层内容再选择服务器，只能先代理最终的服务器和客户端建立连接(三次握手)后，才可能接受到客户端发送的真正应用层内容的报文，然后再根据该报文中的特定字段，再加上负载均衡设备设置的服务器选择方式，决定最终选择的内部服务器。负载均衡设备在这种情况下，更类似于一个代理服务器。负载均衡和前端的客户端以及后端的服务器会分别建立TCP连接。

协议支持及端口配置

最近更新时间：2018-01-26 19:14:11

腾讯云支持 四层转发 和 七层转发 两种监听器模式，针对不同的协议类型，分别作用于网络模型中的传输层和应用层。有关何种类型的负载均衡实例支持何种监听器模式的跟多内容，可以参考 公网负载均衡实例 和 内网负载均衡实例。

四层转发监听器协议及端口配置

负载均衡监听器监听负载均衡实例上的四层请求（即OSI网络协议模型第四层传输层），并将 TCP 及 UDP 请求分发至后端服务器进行处理。四层转发能力通过以下配置实现：

前后端协议	前端端口（负载均衡端口）	后端端口（服务器端口）	备注
四层协议，即 TCP 和 UDP	负载均衡器对外或对内提供服务时接收请求的前端端口。 用户可以为下列 TCP 端口执行负载均衡： 21（FTP）、 25（SMTP）、 80（Http）、 443（Https），以及 1024-65535等端口。	后端端口为云服务器提供服务的端口，会接受负载均衡分发的流量。在一个负载均衡实例中，同个负载均衡端口可以对应多个云服务器端口	**当前，前端端口在同一个负载均衡实例内不可以重复**，比如用户不可以创建 TCP：23 UDP：23 端口。 **后端端口在同一个负载均衡实例内不可重复** 四层转发情况下客户端来源 IP 将直接发送给后端服务器，获取客户端 IP 能力默认开启（不适用于私有网络中的内网型负载均衡实例）

七层转发监听器协议及端口配置

负载均衡监听器监听负载均衡实例上的七层请求（即OSI网络协议模型第七层应用层），并将 HTTP(s) 请求分发至后端服务器进行处理。七层转发能力通过以下配置实现：

前后端协议	前端端口（负载均衡端口）	后端端口（服务器端口）	备注

前后端协议	前端端口（负载均衡端口）	后端端口（服务器端口）	备注
七层协议，即HTTP(HTTPS)协议	负载均衡器对外或对内提供服务时接收请求的前端端口。 支持 1-65535 区间的端口	后端端口为云服务器提供服务的端口，会接受负载均衡分发的流量。在一个负载均衡实例中，同个负载均衡端口可以对应多个云服务器端口	**前端端口在同一个负载均衡实例内不可以和四层端口重复，比如客户不可以同时创建HTTP 80和TCP 80端口** **后端端口在同一个负载均衡实例内可以重复** 七层转发情况下客户端来源IP将直接发送给后端服务器（通过配置proxy_set_header X-Forwarded），获取客户端IP能力默认开启。（不适用于私有网络中的内网型负载均衡实例）

轮询方式

最近更新时间：2018-06-01 17:28:53

轮询方式是指负载均衡向后端服务器分配流量的算法，根据不同的轮询方式及后端服务器的权重设置，可以达到不同的效果。

加权轮询算法（Weighted Round-Robin Scheduling）

加权轮询算法就是以轮叫的方式依次将请求调度不同的服务器。加权轮询调度算法可以解决服务器间性能不一的情况，它用相应的权值表示服务器的处理性能，按权值的高低和轮询方式分配请求到各服务器。权值高的服务器先收到连接，权值高的服务器比权值低的服务器处理更多的连接，相同权值的服务器处理相同数目的连接数。

- **优势**：算法的优点是其简洁性和实用性。它无需记录当前所有连接的状态，所以它是一种无状态调度。
- **劣势**：加权轮询调度算法相对简单，但不适用于请求服务时间变化比较大，或者每个请求所消耗的时间不一致的情况，此时轮询调度算法容易导致服务器间的负载不平衡。
- **适用场景**：每个请求所占用的后端时间基本相同，负载情况最好。常用于短连接服务，例如 HTTP 等服务。
- **用户推荐**：用户可知每个请求所占用后端时间基本相同时，如已知后端服务器处理的都是同类型或者相似类型的请求时，推荐选择加权轮询的方式。当请求时间相差较小时也推荐使用加权轮询的方式，因为该实现方式消耗小，无需遍历，效率较高。

加权最小连接数算法（Weighted Least-Connection Scheduling）

在实际情况中，客户端的每一次请求服务在服务器停留的时间可能会有较大的差异，随着工作时间的延伸，如果采用简单的轮询或随机均衡算法，每一台服务器上的连接进程数目可能会产生极大的不同，这样实际上并没有达到真正的负载均衡。

最小连接调度是一种动态调度算法，它通过服务器当前所活跃的连接数来估计服务器的负载情况。与轮询调度算法相反，最小连接调度是一种动态调度算法，它通过服务器当前所活跃的连接数来估计服务器的负载情况。调度器需要记录各个服务器已建立连接的数目，当一个请求被调度到某台服务器，其连接数加一; 当连接中止或超时，其连接数减一。

权重最小连接数调度算法是在最小连接数调度算法的基础上，根据服务器的不同处理能力，给每个服务器分配不同的权值，使其能够接受相应权值数的服务请求，是在最小连接数调度算法的基础上的改进。

1) 假设各台后端服务器的权值依次为 wi，当前连接数依次为 ci,依次计算 ci/wi,值最小的后端服务器实例作为下一个分配的实例。

2) 如果存在 ci/wi 相同的后端服务器实例，再使用加权轮询的方式调度。

- **优势**：此种均衡算法适合长时处理的请求服务，如 FTP 等应用。
- **劣势**：由于接口限制，目前最小连接数和会话保持功能不能同时开启。
- **适用场景**：每个请求所占用的后端时间相差较大的场景。常用于长连接服务。
- **用户推荐**：如果用户需要处理不同的请求，且请求所占用后端时间相差较大，如 3 ms 和 3s 这种数量级的差距时，推荐使用加权最小连接数算法实现负载均衡。

源地址散列调度算法（ip_hash）

根据请求的源 IP 地址，作为散列键（Hash Key）从静态分配的散列表找出对应的服务器，若该服务器是可用的且未超载，将请求发送到该服务器，否则返回空。

- **优势**：可以使某一客户端的请求通过哈希表一直映射在同一台后端服务器上。因此在不支持会话保持的场景可以使用 ip_hash 进行简单的会话保持实现。
- **用户推荐**：将请求的源地址进行哈希运算，并结合后端的服务器的权重派发请求至某匹配的服务器，这可以使得同一个客户端 IP 的请求始终被派发至某特定的服务器。该方式适合无 cookie 功能的 TCP 协议。

均衡算法选取及权重配置

为了让用户在不同场景下能够让后端服务器集群稳定地承接业务，我们给出几个负载均衡选择与权重配置的实例供用户进行参考。

- 场景1：

  设有3台配置相同（CPU / 内存）的后端服务器，由于性能一致，用户可以将后端服务器权重都设置为10。现在每台后端服务器与客户端建立了 100 个 TCP 连接，此时新增 1 台后端服务器。在此场景下，推荐用户使用最小连接数的均衡方式，这种配置能快速的让第四台后端服务器的负载提升，降低另外 3 台后端服务器的压力。

- 场景2：

  设用户首次接触云服务，且建站时间不长，网站负载较低，则建议购买相同配置的后端服务器，因此后端服

务器都是无差别的接入层服务器。在此场景下，用户可以将后端服务器权重都设为默认值 10，采用加权轮询的均衡方式进行流量分发。

- 场景3：

  用户有 5 台服务器，用于承载简单的静态网站访问，且 5 台服务器的计算能力的比例为 9：3：3：3：1（按CPU、内存换算）。在此场景下，用户可以依次将后端服务器权重比例设置为 90、30、30、30、和 10，由于静态网站访问大多数是短连接请求，因此可以采用加权轮询的均衡方式，让负载均衡实例按后端服务器的性能比例分配请求。

- 场景4：

  某用户有 10 台后端服务器用于承担海量的 Web 访问请求，且不希望多购置后端服务器增加支出。某台后端服务器经常会因为负载过高，导致服务器重启。在此场景下，建议用户根据后端服务器的性能进行相应的权重设置，给负载过高的后端服务器设置较小的权值。除此之外，可以采用最小连接数的负载均衡方式，将请求分配到活跃连接数较少的后端服务器上，从而解决某台后端服务器负载过高的问题。

- 场景5：

  某用户有 3 台后端服务器用于处理若干长连接请求，且这 3 台服务器的计算能力比例为 3：1：1（按CPU、内存换算）。此时性能最好的服务器处理请求较多，用户不希望过载此服务器，希望能够将新的请求分配到空闲服务器上。在此场景下，可以采用最小连接数的均衡方式，并适当降低繁忙服务器的权重，便于负载均衡将请求分配到活跃数较少的后端服务器上，实现负载均衡。

- 场景6：

  某用户希望后续客户端的请求可以分配到同一服务器上。而采用 加权轮询 或 加权最小连接数 的方式，不能保证相同客户端的请求被分到固定某台服务器上去。为了配合客户特定应用程序服务器的需求，保证客户端的会话具有“粘性”或是“持续性”，在此场景下，我们可以采用 ip_hash 的均衡方式进行流量分发。此方法可以确保来自同一客户端的请求总被定向分发到同一后端服务器上去。（服务器数量变化或是该服务器不可用时除外）

会话保持

最近更新时间：2018-09-10 15:18:49

会话保持可使得来自同一 IP 的请求被转发到同一台后端服务器上。默认情况下，负载均衡会将每个请求分别路由到不同后端服务器实例负载。但是，您可以使用会话保持功能使特定用户的请求被路由到同一台后端服务器实例上，这样可以使某些需要保持会话的应用程序（如购物车）合理地工作。

四层会话保持

四层转发情境支持简单会话保持能力，会话保持时间可设为 `30-3600` 秒中的任意整数值，超过该时间阈值，会话中无新请求则断开连接。

七层会话保持

七层转发情境支持基于 cookie 插入的会话保持能力（由负载均衡器向客户端植入 cookie），会话保持可选时间为 `30-3600` 秒，关于插入 cookie 会话保持的更多信息可以参考这里。

连接超时时间

当前HTTP连接超时时间（keepalive_timeout）暂时不支持调整，默认为 `75` 秒。超过该时间阈值，会话中无数据传输则断开连接。
当前TCP连接的超时时间暂时不支持调整，默认为 `900` 秒。超过该时间阈值，会话中无数据传输则断开连接。

配置会话保持

1. 登录 负载均衡控制台，单击需要配置会话保持的负载均衡实例 ID，进入详情页。
2. 单击操作栏中的【修改】。
3. 选择是否需要开启会话保持功能，单击按钮开启，输入保持时间，单击【确定】即可。

长连接和会话保持的关系

场景1：HTTP 七层业务

假设 Client 端访问是 HTTP / 1.1 协议，头部信息中设置 Connection:keep-alive。通过 CLB，再访问到后端 CVM，此时不开会话保持，下一次访问，能否访问到同一台 CVM？
答：不能。

- HTTP keep-alive 是指 TCP 连接在发送后将仍然保持打开状态，于是，浏览器可以继续通过相同的连接发送请求。保持连接节省了为每个请求建立新连接所需的时间，还节约了带宽。CLB 集群的默认超时时间是 75 秒（ 75 秒内无新请求刷新，则默认断开 TCP 连接）。
- HTTP keep-alive 是由 Client 端跟 CLB 建立的，若此时没有开启 cookie 会话保持，则下一次访问，CLB会根据轮询策略，随机挑选后端的一台CVM，此前的长连接等于白费了。

因此建议开启会话保持。当设置 cookie 会话保持的时间为 1000 秒时，Client 端再次发起请求。由于距离上一次请求，已经超过了 75 秒，TCP的连接要重新建立。应用层判断 cookie，找到同一台 CVM，Client 访问的 CVM 还是上一次访问的那一台。

场景2：TCP 四层业务

假设 Client 端发起访问，传输层协议是 TCP，启用长连接。但没有开基于源 IP 的会话保持。下一次访问，同一个 Client，能否访问到同一个机器？
答：不一定。
根据四层的实现机制，当 TCP 启用长连接时，如果该长连接一直没有断开，前后两次访问都是同一条连接，则可以访问到同一台机器。如果第二次访问时，第一条连接由于其他原因（网络重启、连接超时）被释放，这时第二次访问就有可能调度到其他后端云服务器上。且长连接默认全局的超时时间是 900 秒，即若没有新请求，则释放。

健康检查

最近更新时间：2018-09-10 14:57:07

- 腾讯云负载均衡实例可以定期向后端服务器发送 Ping、尝试连接或发送请求来测试后端服务器运行的状况，这些测试称为健康检查。
- 当后端服务器实例被判定为不健康时，负载均衡实例将不会把请求转发到该实例上。但健康检查会对所有后端服务器（不管是判定为健康的还是不健康的）进行，当不健康实例恢复正常状态时，负载均衡实例将恢复把新的请求转发给它。
- 弹性伸缩组会定期使用相似的方法确定每个组内实例的运行状况。有关更多信息，请参见 弹性伸缩。

健康检查配置字段说明

- **响应超时时间**
  健康检查响应的最大超时时间。如后端云服务器在相应时间内没有正确响应，则判定为健康检查失败。
- **健康检查间隔**
  进行健康检查的时间间隔。
- **不健康阈值**
  如果连续 n 次（n 为填写的数值）收到了健康检查结果失败状态，则识别为不健康，控制台显示为不健康。
- **健康阈值**
  如果连续 n 次（n 为填写的数值）收到了健康检查结果为成功状态，则识别为健康，控制台显示为健康。
- **正常状态码**
  仅在 HTTP 检查方式时才有。指定用户判断健康检查正常的 HTTP 状态码。可选值为 `http_1xx`、`http_2xx`、`http_3xx`、`http_4xx`、`http_5xx`，可多选。默认情况或不做任何选择时，该值置为 `http_2xx`。

四层转发健康检查配置

四层转发的健康检查机制由负载均衡器向配置中指定的服务器端口发起访问请求，如果端口访问正常则视为后端服务器运行正常，否则视为后端服务器运行异常。对于TCP的业务，使用 SYN 包进行探测。对于 UDP 业务，使用 Ping 进行检查。

- 响应超时时间：`2 - 60` 秒。
- 检查间隔：`5 - 300` 秒。
- 不健康阈值：`2 - 10` 次（健康后端服务器出现此指定次数响应超时后，视为不健康）。
- 健康阈值 `2 - 10` 次（不健康后端服务器出现此指定次数响应超时后，视为健康）。

七层转发健康检查配置

七层转发的健康检查机制由负载均衡器向后端服务器发送 HTTP 请求来检测后端服务，负载均衡器会通过 HTTP 返回值是否为 `http_2xx` 、 `http_4xx` 来判断服务是否正常。后续将推出用户自定义的方式，对响应代码所代表的状态进行描述。假定在某场景下，HTTP 返回值为 `http_1xx` 、 `http_2xx` 、 `http_3xx` 、 `http_4xx` 和 `http_5xx` 这几种，用户可以根据业务需要编辑 `http_1xx` 及 `http_2xx` 为服务正常状态，并设置 `http_3xx` 至 `http_5xx` 的返回值代表异常状态。

- 响应超时时间暂不能设置，默认响应超时时间为 `5` 秒。
- 检查间隔： `5 - 300` 秒，默认为 `6` 秒
- 不健康阈值： `2 - 10` 次，默认为 `3` 次（健康后端服务器出现此指定次数响应超时后，视为不健康）
- 健康阈值： `2 - 10` 次，默认为 `3` 次（不健康后端服务器出现此指定次数响应超时后，视为健康）
- HTTP 请求方式：默认使用 HEAD 方法，服务器仅返回 HTTP 头部信息，对应的后端服务需支持 HEAD，选择 HEAD 可降低后端开销，提升请求效率；若使用 GET 方法，后端服务则需支持 GET。

健康检查异常排查思路

四层排查

TCP 协议下，负载均衡使用 SYN 包进行探测；UDP 协议下，负载均衡使用 Ping 命令进行探测。
在页面查看后端服务器端口的健康状态，若不健康，排查思路如下：

- 确定后端服务器是否有配置有安全组影响了服务。有关如何控制后端服务器的访问来保证服务正常运行，请参考后端服务器的访问控制。
- 使用 `netstat` 命令，确定后端服务器的端口是否有进程在监听，若未发现进程则请重新启动服务。

七层排查

针对七层（HTTP / HTTPS 协议）服务，当某一监听出现健康检查异常时，可以通过如下方面进行排查：

1. 由于负载均衡的七层健康检查服务与后端 CVM 之间通过内网通信，您需要登录服务器检查应用服务器端口是否正常监听在内网地址上，如果没有监听在内网地址，请将应用服务器端口监听到内网上，从而确保负载均衡系统和后端 CVM 之间的通讯正常。
   假设负载均衡前端端口是 80，CVM 后端端口也是 80，CVM 的内网 IP 是：1.1.1.10，
   Windows 系统服务器使用如下命令：

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

   Linux 系统服务器使用如下命令：

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

如果能看到 1.1.1.10:80 的监听或 0.0.0.0:80 的监听则说明此配置正常。

2. 请确保后端服务器开启了您在负载均衡监听器中配置的后端端口。
   如果是四层负载均衡，只要后端端口 telnet 有响应即可，可以使用 `telnet 1.1.1.10 80` 来测试。如果是七层负载均衡，需要 HTTP 状态码是 200 等代表正常的状态码。检验方法如下：
   - Windows 系统可以直接在 CVM 内的浏览器输入内网 IP 测试是否正常，本例为： `http://1.1.1.10` ；
   - Linux 系统可以通过 `curl -I` 命令看看状态是否为 `HTTP/1.1 200 OK` ，本例使用 `curl -I 1.1.1.10` 命令。
3. 检查后端 CVM 内部是否有防火墙或其他安全类防护软件，这类软件很容易将负载均衡系统的本地 IP 地址屏蔽，从而导致负载均衡系统无法跟后端服务器进行通讯。
   检查服务器内网防火墙是否放行 80 端口，可以暂时关闭防火墙进行测试。
   - Windows 系统可以运行输入 `firewall.cpl` 操作关闭
   - Linux 系统可以输入 `/etc/init.d/iptables stop` 关闭
4. 检查负载均衡健康检查参数设置是否正确，建议参照本文档提供的健康检查参数默认值进行设置。
5. 健康检查指定的检测文件，建议是 HTML 形式的简单页面，只用于检查返回结果。不建议用 PHP 等动态脚本语言。
6. 检查后端是否有较高负载导致 CVM 对外提供服务响应慢。
7. 检查 HTTP 请求方式，如果使用 HEAD 方法，则后端服务一定要支持 HEAD；如果是 GET 方法，则后端服务一定要支持 GET。

关于健康检查探测频率过高的说明

健康检查探测包频率过高，控制台设置接受探测包 5 秒 1 次，实际后端 RS 发现 1 秒内收到 1 次甚至多次健康检查请求，这是什么原因呢？

- 当前，健康检查频率过高的问题，主要跟负载均衡后端健康探测实现机制有关。假设 100 万的 client 端请求，会分散在 4 台 CLB 后端物理机上，再转给云服务器。 健康检查探测是在 CLB 的后端物理机上，各自探测的。因此，CLB 实例设置 5 秒 1 次的探测请求，实际上 CLB 后端的每台物理机都会每 5 秒发送一次探测。因此在后端云服务器上，会收到多次探测请求。（假设 CLB 实例所在集群有 8 台物理机，那么每台机器 5 秒发送一次请求，后端主机可能会在 5 秒中收到 8 次探测）
- 该实现方案的优势是：效率高，探测精准，避免误剔除。比如 CLB 实例集群的 8 台物理机中，其中 1 台判断失败，仅那 1 台机器不再转发流量，另外 7 台的流量是正常的。

因此，如果您后端云服务器的探测频率过高，可以通过设置更长的探测间隔时间来解决（比如设置为 15 秒探测一次）。

证书配置

最近更新时间：2018-08-21 15:55:25

常用证书申请流程

- 本地生成私钥：openssl genrsa -out privateKey.pem 2048 ，其中 privateKey.pem 为您的私钥文件，请妥善保管。
- 生成证书请求文件：openssl req -new -key privateKey.pem -out server.csr ，其中 server.csr 是您的证书请求文件，可用其去申请证书。
- 获取请求文件中的内容前往 CA 等机构站点申请证书。

证书格式要求

- 用户要申请的证书为：Linux 环境下 PEM 格式的证书。负载均衡不支持其他格式的证书，如是其它格式的证书请参考本文 “负载均衡支持的证书格式及转换方式” 部分内容。
- 如果是通过 root CA 机构颁发的证书， 您拿到的证书为唯一的一份，不需要额外的证书，配置的站点即可被浏览器等访问设备认为可信。
- 如果是通过中级 CA 机构颁发的证书，您拿到的证书文件包含多份证书，需要人为的将服务器证书与中间证书合并在一起上传。
- 当您的证书有证书链时，请将证书链内容，转化为 PEM 格式内容，与证书内容合并上传。
- 拼接规则为：服务器证书放第一份，中间证书放第二份，中间不要有空行。

> **注意：**
> 一般情况下，机构在颁发证书的时候会有对应说明， 请注意规则说明。

以下为证书格式和证书链格式范例，请确认格式正确后上传：

1. root CA机构颁发的证书：证书格式为 Linux 环境下 PEM 格式。样例如下：

```
-----BEGIN CERTIFICATE-----
MIIE+TCCA+GgAwIBAgIQU306HIX4KsioTW1s2A2krTANBgkqhkiG9w0BAQUFADCB
tTELMAkGA1UEBhMCVVMxFzAVBgNVBAoTDlZlcmlTaWduLCBJbmMuMR8wHQYDVQQL
ExZWZXJpU2lnbiBUcnVzdCBOZXR3b3JrMTswOQYDVQQLEzJUZXJtcyBvZiB1c2Ug
YXQgaHR0cHM6Ly93d3cudmVyaXNpZ24uY29tL3JwYSoAYykwOTEvMC0GA1UEAxMm
VmVyaVNpZ24gQ2xhc3MgMyBTZWN1cmUgU2VydmVyIENBIC0gRzIwHhcNMTAxMDA4
MDAwMDAwWhcNMTMxMDA3MjM1OTU5WjBqMQswCQYDVQQGEwJVUzETMBEGA1UECBMK
V2FzaGluZ3RvbjEQMA4GA1UEBxQHU2VhdHRsZTEYMBYGA1UEChQPQW1hem9uLmNv
bSBJbmMuMRowGAYDVQQDFBFpYW0uYW1hem9uYXdzLmNvbTCBnzANBgkqhkiG9w0B
AQEFAAOBjQAwgYkCgYEA3Xb0EGea2dB8QGEUwLcEpwvGawEkUdLZmGL1rQJZdeeN
3vaF+ZTm8Qw5Adk2Gr/RwYXtpx04xvQXmNm+9YmksHmCZdruCrW1eN/P9wBfqMMZ
X964CjVov3NrF5AuxU8jgtw0yu//C3hWn0uIVGdg76626ggOoJSaj48R2n0MnVcC
AwEAAaOCAdEwggHNMAkGA1UdEwQCMAAwCwYDVR0PBAQDAgWgMEUGA1UdHwQ+MDww
OqA4oDaGNGh0dHA6Ly9TVlJTZWN1cmUtRzItY3JsLnZlcmlzaWduLmNvbS9TVlJT
ZWN1cmVHMi5jcmwwRAYDVR0gBD0wOzA5BgtghkgBhvhFAQcXAzAqMCgGCCsGAQUF
BwIBFhxodHRwczovL3d3dy52ZXJpc2lnbi5jb20vcnBhMB0GA1UdJQQWMBQGCCsG
AQUFBwMBBggrBgEFBQcDAjAfBgNVHSMEGDAWgBSl7wsRzsBBA6NKZZBIshzgVy19
RzB2BggrBgEFBQcBAQRqMGgwJAYIKwYBBQUHMAGGGGh0dHA6Ly9vY3NwLnZlcmlz
aWduLmNvbTBABggrBgEFBQcwAoY0aHR0cDovL1NWUlNlY3VyZS1HMi1haWEudmVy
aXNpZ24uY29tL1NWUlNlY3VyZUcyLmNlcjBuBggrBgEFBQcBDARiMGChXqBcMFow
WDBWFglpbWFnZS9naWYwITAfMAcGBSsOAwIaBBRLa7kolgYMu9BSOJsprEsHiyEF
GDAmFiRodHRwOi8vbG9nby52ZXJpc2lnbi5jb20vdnNsb2dvMS5naWYwDQYJKoZI
hvcNAQEFBQADggEBALpFBXeG782QsTtGwEE9zBcVCuKjrsl3dWK1dFiq3OP4y/Bi
ZBYEywBt8zNuYFUE25Ub/zmvmpe7p0G76tmQ8bRp/4qkJoiSesHJvFgJ1mksr3IQ
3gaE1aN2BSUIHxGLn9N4F09hYwwbeEZaCxfgBiLdEIodNwzcvGJ+2LlDWGJOGrNI
NM856xjqhJCPxYzk9buuCl1B4Kzu0CTbexz/iEgYV+DiuTxcfA4uhwMDSe0nynbn
1qiwRk450mCOnqH4ly4P4lXo02t4A/DI1I8ZNct/Qfl69a2Lf6vc9rF7BELT0e5Y
R7CKx7fc5xRaeQdyGj/dJevm9BF/mSdnclS5vas=
-----END CERTIFICATE-----
```

证书规则为：

- [———-BEGIN CERTIFICATE———-, ———-END CERTIFICATE———-] 开头和结尾，请将这些内容一并上传。
- 每行 64 字符，最后一行不超过 64 字符。

2. 中级机构颁发的证书链：

———-BEGIN CERTIFICATE———-
———-END CERTIFICATE———-
———-BEGIN CERTIFICATE———-
———-END CERTIFICATE———-
———-BEGIN CERTIFICATE———-
———-END CERTIFICATE———-

证书链规则：

- 证书之间不能有空行。
- 每一份证书遵守第一点关于证书的格式说明。

RSA 私钥格式要求

1. 样例如下：

RSA 私钥可以包括所有私钥（RSA 和 DSA）、公钥（RSA 和 DSA）和 (x509) 证书。它存储用 Base64 编码的 DER 格式数据，用 ascii 报头包围，因此适合系统之间的文本模式传输。

2. RSA 私钥规则：

- [———-BEGIN RSA PRIVATE KEY———-, ———-END RSA PRIVATE KEY———-] 开头结尾；请将这些内容一并上传。
- 每行 64 字符，最后一行长度可以不足 64 字符。

如果您不是按照上述方案生成私钥，得到[———-BEGIN PRIVATE KEY———-, ———-END PRIVATE KEY———-] 这种样式的私钥，您可以按照如下方式转换：

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

然后将new_server_key.pem的内容与证书一起上传。

证书转换为 PEM 格式说明

目前负载均衡只支持 PEM 格式的证书，其他格式的证书需要转换成 PEM 格式后才能上传到负载均衡中，建议通过 openssl 工具进行转换。下面是几种比较流行的证书格式转换为 PEM 格式的方法。

DER 转换为 PEM

DER 格式一般出现在 Java 平台中。

- 证书转换：`openssl x509 -inform der -in certificate.cer -out certificate.pem`
- 私钥转换：`openssl rsa -inform DER -outform PEM -in privatekey.der -out privatekey.pem`

P7B 转换为 PEM

P7B 格式一般出现在 Windows Server 和 tomcat 中。

- 证书转换：`openssl pkcs7 -print_certs -in incertificat.p7b -out outcertificate.cer`
  获取outcertificat.cer里面 [——-BEGIN CERTIFICATE——-， ——-END CERTIFICATE——-] 的内容作为证书上传。
- 私钥转换：无私钥。

PFX 转换为 PEM

PFX 格式一般出现在 Windows Server 中。

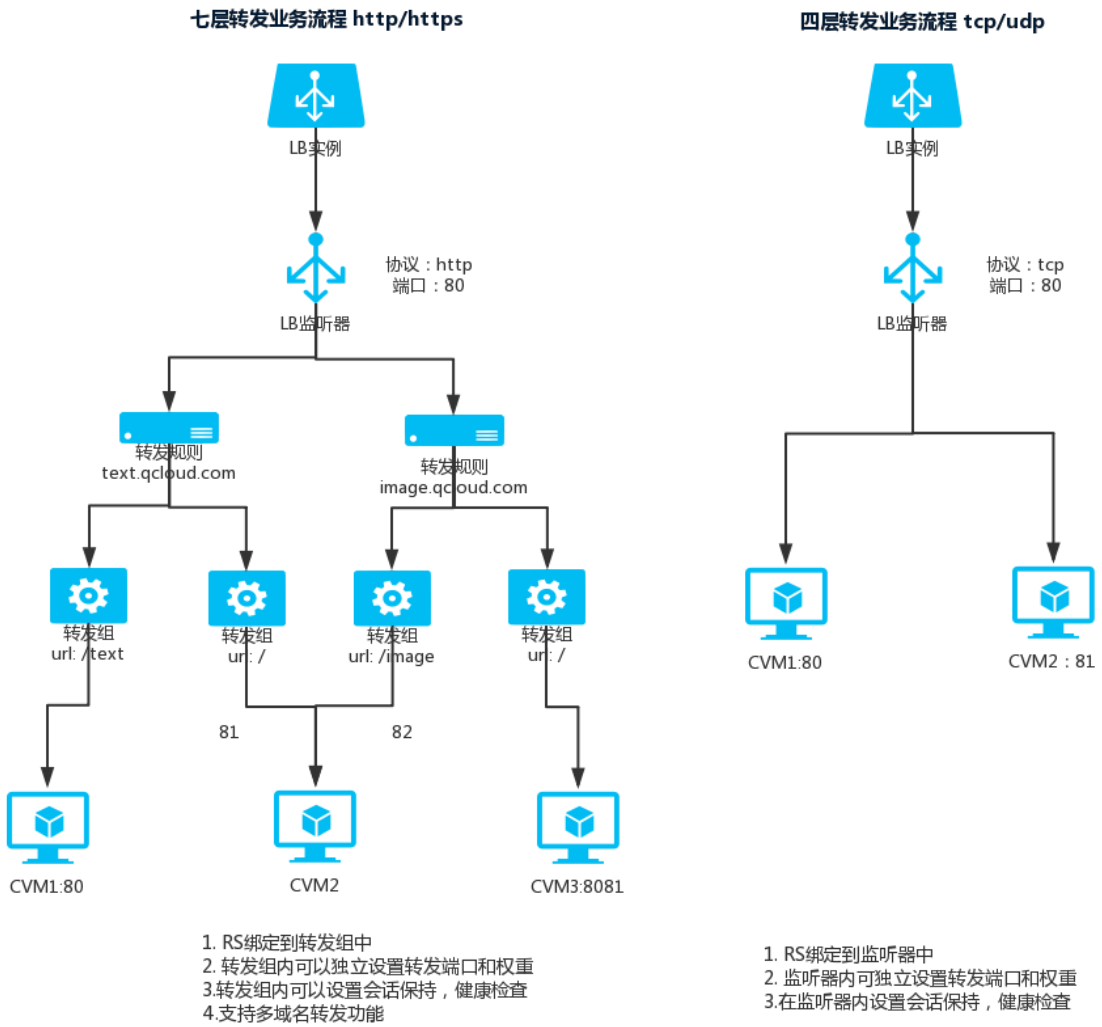
- 证书转换：`openssl pkcs12 -in certname.pfx -nokeys -out cert.pem`
- 私钥转换：`openssl pkcs12 -in certname.pfx -nocerts -out key.pem -nodes`

应用型 CLB 转发规则及转发组配置说明

最近更新时间：2018-10-08 17:39:32

业务流程图

公网应用型 CLB 的七层业务流程及四层业务流程如下所示：



使用公网应用型负载均衡的七层转发 HTTP / HTTPS 时，在一个 CLB 实例的监听器中新建转发规则，用户可以添加一个对应的域名。

- 当用户只建立了一条转发规则时，访问 VIP + URL 可以对应相应的转发规则，并正常访问服务。
- 当用户建立了多条转发规则时，将提示用户，此时访问 VIP + URL 不能保证访问到哪个域名 + URL，需要用户直接访问域名 + URL 来确保具体的转发规则生效。（即用户配置多条转发规则时，同一个 VIP 对应了多条域名，此

时不建议通过 VIP + URL 访问服务，而应该通过具体的域名 + URL 访问服务。）

转发规则配置说明

域名配置规则

公网应用型负载均衡，七层监听器的转发规则配置域名时，支持正则表达式，长度限制为 1 - 80。

- 非正则的域名支持的字符集如下：
  `a-z` `0-9` `.` `-`
- 通配的域名，目前只支持 `*.example.com` 或者 `www.example.*` 的形式，且单个域名中只支持 * 出现一次。
- 域名的正则表达式中不支持的字符集如下：
  `"` `{` `}` `;` `~` `'` `` ` `` `空格
- 应用型负载均衡支持的正则域名举例如下：
  `~^www\d+\.example\.com$`

健康检查配置规则

当用户填写域名为通配域名时，需要指定某一固定域名（非正则）为健康检查域名。该健康检查域名配置支持的字符集如下：
`a-z` `0-9` `.` `-`
公网应用型负载均衡，七层监听器配置健康检查的路径时，默认 /，必须以 / 开头。长度限制为 1 - 80。暂不支持正则表达，建议指定某个固定 URL 路径（静态页面）进行健康检查。其中，健康检查路径配置支持的字符集如下：
`a-z` `A-Z` `0-9` `.` `-` `/` `=` `?`

域名匹配规则示例

1. 转发规则中不配置域名，填写 IP 代替，并在转发组中配置多个 URL。该服务通过 VIP + URL 进行访问。
2. 通过在转发规则中配置完整域名，并在转发组中配置多个 URL。服务通过域名 + URL 进行访问。
3. 在转发规则中配置通配符域名，并在转发组中配置多个 URL，通过匹配请求域名 + URL 进行访问。当客户希望不同的域名能够指向相同的 URL 地址时，可以参照这种方式进行配置。以 `example.qcould.com` 为例，格式如下所示：
   `example.qcloud.com` 精确匹配 example.qcloud.com 域名
   `\*.qcloud.com` 匹配所有以 qcloud.com 结尾的域名
   `example.qcloud.\*` 匹配所有以 example.qcloud 开头的域名

> **注意：**
> 如果请求的域名匹配不到转发规则，则会默认匹配到第一个域名的配置。

4. 在转发规则中配置域名，并在转发组中配置模糊匹配的 URL。使用前缀匹配，可在最后加入通配符 $ 进行完整匹配。
   例如，客户通过配置转发组 URL ~* .(gif|jpg|bmp)$，希望匹配任何以 gif、 jpg 或 bmp 结尾的文件。
5. 建议用户设置默认访问域名，当监听器中所有域名均没有匹配成功时，系统会将请求指向默认访问域名，让默认规则可控。设置默认访问域名的操作如下图所示：

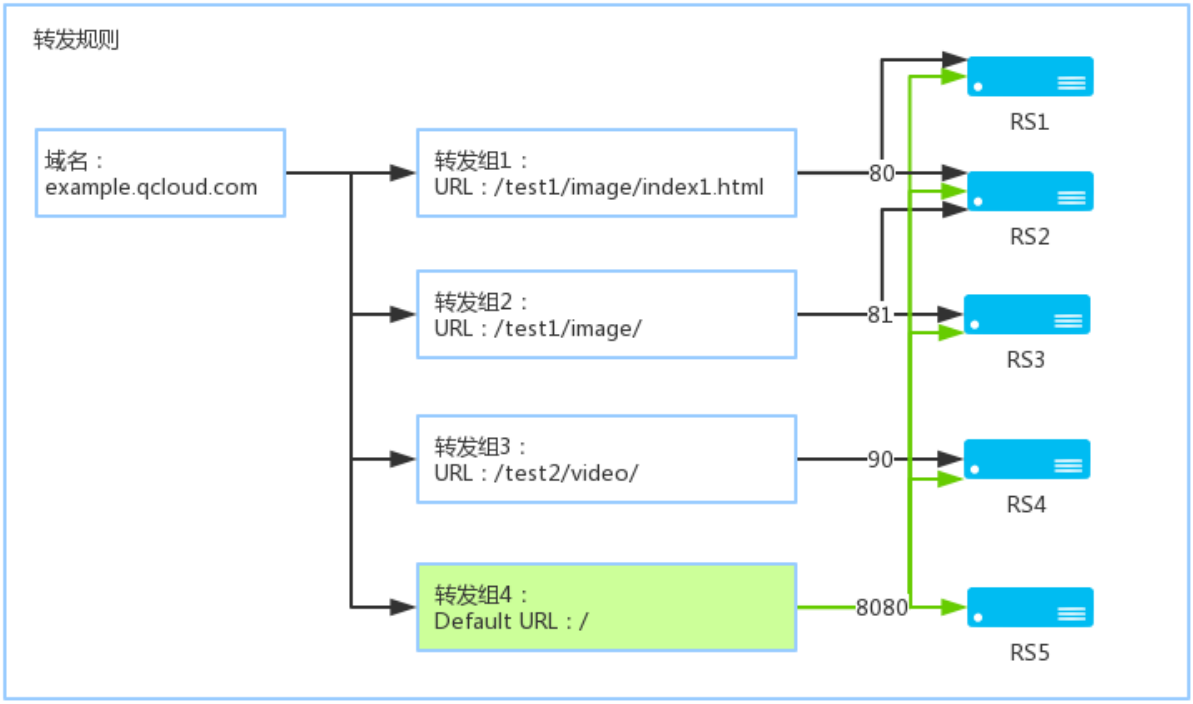
转发组 URL 匹配规则说明

URL 配置规则

应用型负载均衡七层监听器转发路径 URL，默认 /，必须以 / 开头。长度限制为 1 - 80。

- URL 支持正则表达，用如下方法判断：
  - `=` 开头表示精确匹配。
  - `^~` 开头表示 uri 以某个常规字符串开头，不是正则匹配。
  - `~` 开头表示区分大小写的正则匹配。
  - `~*` 开头表示不区分大小写的正则匹配。
  - `/` 通用匹配, 如果没有其它匹配,任何请求都会匹配到。
- 非正则的 URL 路径，以 / 开头，支持的字符集如下：
  `a-z` `A-Z` `0-9` `.` `-` `/` `=` `?`
- 正则的 URL，不支持的字符集如下：
  `"` `{` `}` `;` `\` `` ` `` `~` `'` 空格

URL 匹配规则示例



1. 匹配规则：优先精确匹配，之后依照规则模糊匹配。
   举例：依照下图配置转发规则及转发组后，如下几个请求将依次被匹配到不同的转发组中：
   a. example.qloud.com/test1/image/index1.html 由于精确匹配转发组 1 设置的 URL 规则，则该请求将被转发到转发组 1 所关联的后端云服务器中，图中即 RS1 和 RS2 的 80 端口。
   b. example.qloud.com/test1/image/hello.html 由于此请求无法精确匹配第一条规则，因此将继续匹配转发组 2 中的规则，发现模糊匹配成功。因此该请求将被转发到转发组 2 所关联的后端云服务器中，图中即 RS2 和 RS3 的 81 端口。
   c. example.qloud.com/test2/video/mp4/ 由于此请求无法精确匹配到前两条规则，因此将继续向下匹配，直至发现可以模糊匹配转发组 3 中的规则。因此该请求将被转发到转发组 3 所关联的后端云服务器中，图中即 RS4 的 90 端口。
   d. example.qloud.com/test3/hello/index.html 由于此请求无法匹配到前三个转发组中的规则，因此将匹配用户配置的最通用规则 default URL。这个时候应该是 nginx 转发请求给后端应用服务器，比如 FastCGI（php），tomcat（jsp），nginx 作为方向代理服务器存在
   e. example.qloud.com/test2/ 由于请求无法精确匹配到前三个转发组中的规则，因此将匹配用户所配置的通用规则 default URL。
2. 如果用户设置的 URL 规则中，服务不能正常运行，则匹配成功后，不会重定向到其他页面。
   举例：如客户端请求 example.qloud.com/test1/image/index1.html 匹配了转发组 1 的 URL 规则，但此时转发组 1 的后端服务器运行异常，出现 404 的页面时，用户进行访问时页面则会显示 404，不会跳转到其他页面。

3. **建议用户设置 default URL，将其指向服务稳定的页面（如静态页面、首页等），并绑定所有后端云服务器。**此时，如果所有规则均没有匹配成功时，系统会将请求指向 default URL 所在的页面，否则可能会出现 404 的问题。
4. 如果用户未设置 default URL，且所有转发规则都不匹配时，此时访问服务，会返回 404。

应用型 CLB 支持 SNI 绑定多个证书

最近更新时间：2018-08-21 16:07:26

SNI 简介

SNI（Server Name Indication，服务器名称指示）是用来改善服务器与客户端 SSL / TLS 的扩展，主要解决一台服务器只能使用一个证书的缺点，支持 SNI 表示服务器支持绑定多个证书。客户端使用 SNI，则需在与服务器建立 SSL / TLS 连接之前指定要连接的域名，这样服务器会根据这个域名返回一个合适的证书。
腾讯云应用型负载均衡的七层 HTTPS 监听器支持 SNI，即支持绑定多个证书，监听规则中的不同域名可使用不同证书。

使用前提

使用 SNI 支持绑定多证书需以下前提条件：

1. 已购买**应用型**负载均衡实例。
2. 已拥有证书。

操作流程

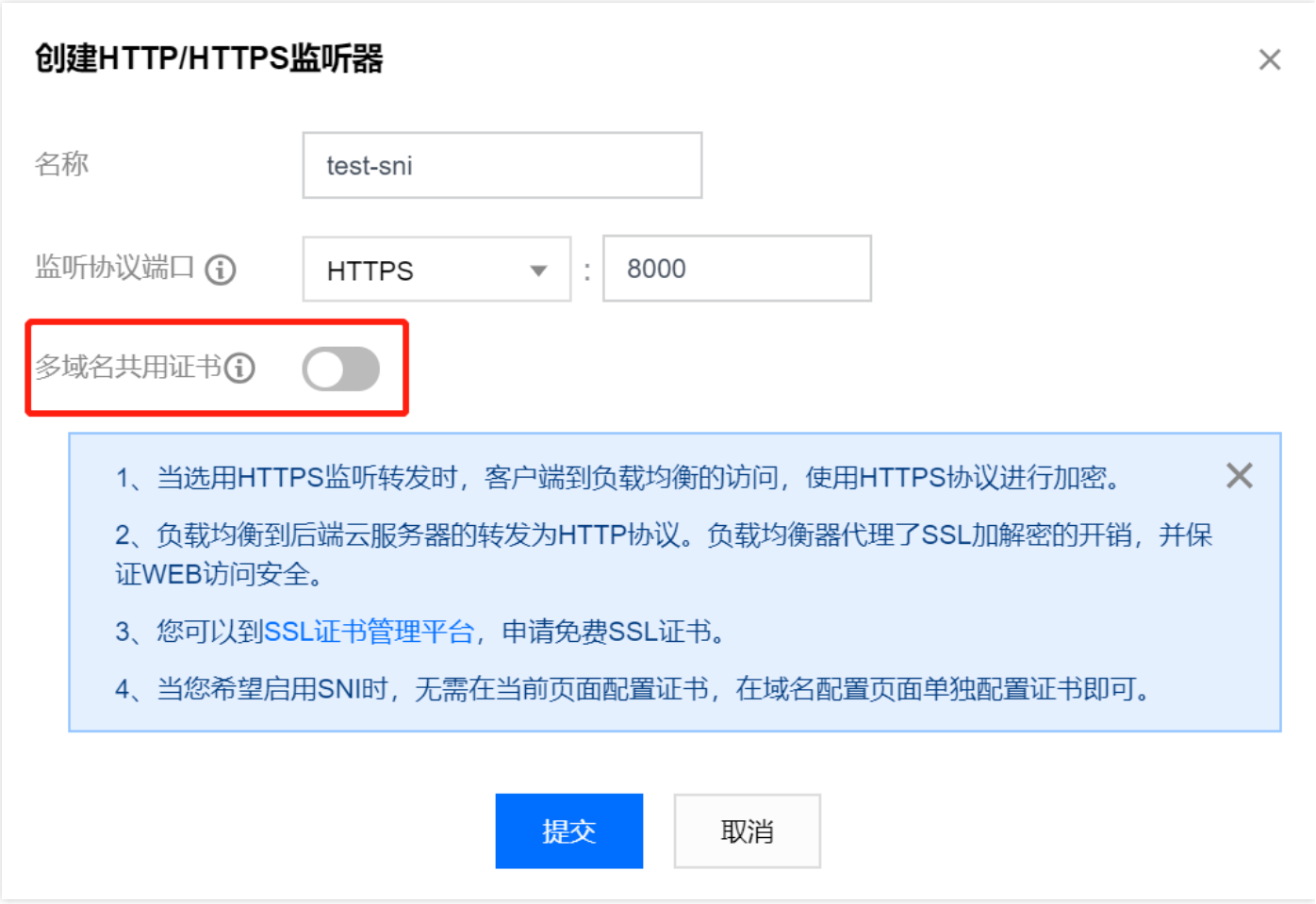
购买应用型负载均衡

1. 登录 负载均衡管理控制台。
2. 购买 应用型负载均衡

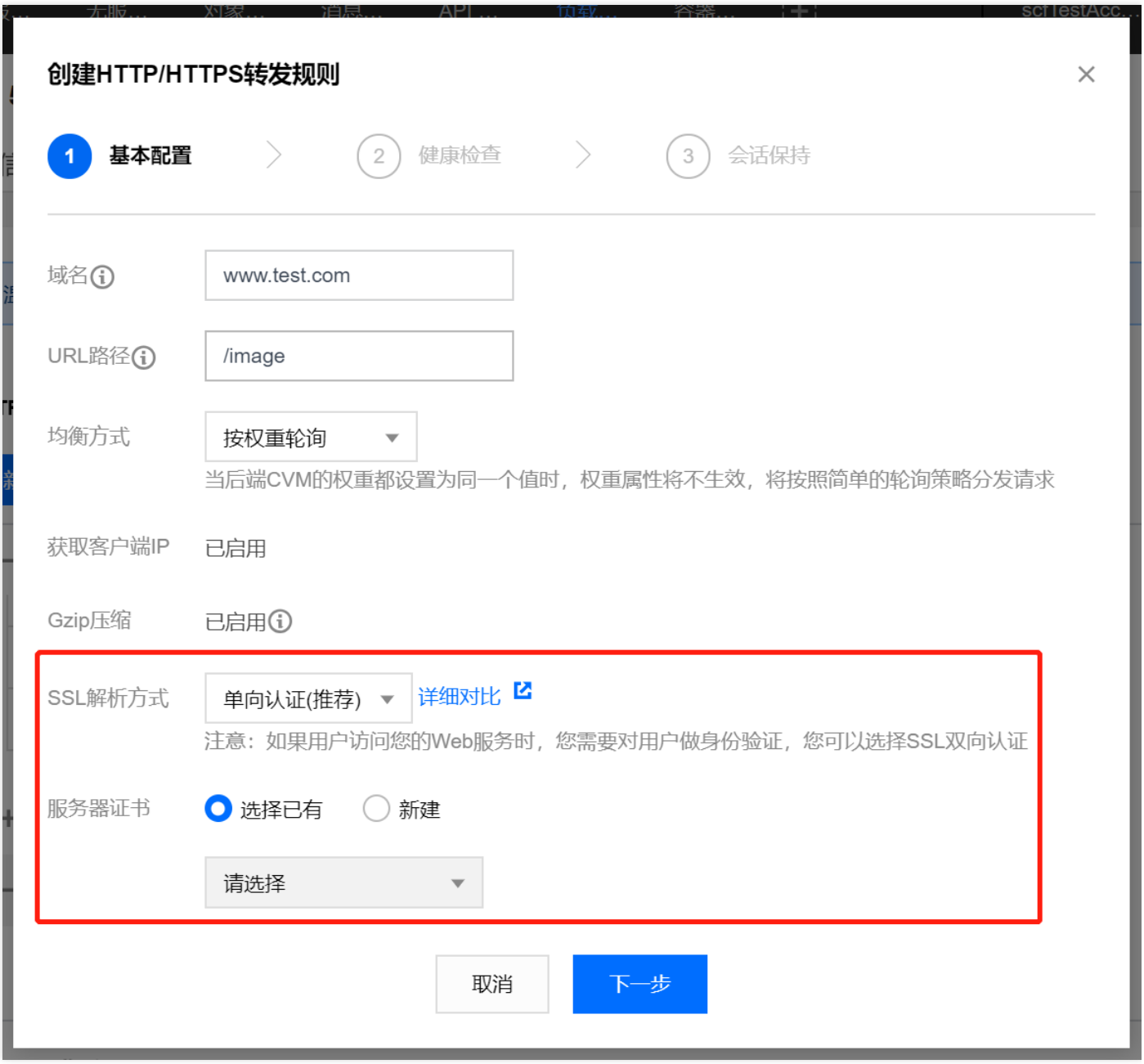
创建 HTTPS 监听器并配置 SNI

1. 创建 HTTPS 监听器时，关闭多域名共用证书。
   多域名共用证书表示服务器上多个域名共用一个证书，即为关闭 SNI；关闭多域名共用证书即为启用 SNI，服务

器支持不同域名使用不同证书。



2. 在该监听器中添加转发规则时，针对不同的域名可选用不同的服务器证书。



公网应用型 CLB 重定向配置

最近更新时间：2018-08-21 16:12:18

LoadBalance 团队在 4 月推出**公网应用型 LB**独家能力：自定义重定向。该能力可解决两大难题：

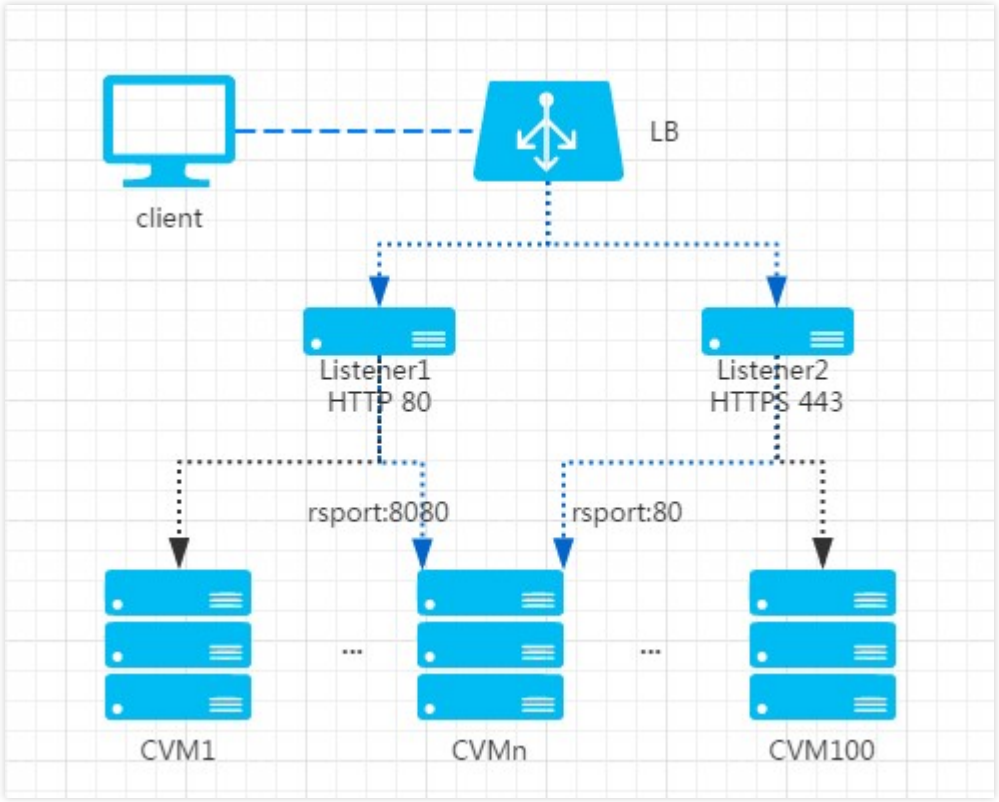
1. 强制 HTTPS
   PC、手机浏览器等以 HTTP 请求访问 Web 服务，LoadBalance 代理后，返回 HTTPS 的 respond。默认强制以 HTTPS 访问网页。
2. 自定义重定向
   当出现 Web 业务需要临时下线（如电商售罄、页面维护，更新升级时）会需要重定向能力。如果不做重定向，用户的收藏和搜索引擎数据库中的旧地址只能让访客得到一个 `404/503` 错误信息页面，降低了用户体验度，导致访问流量白白丧失。不仅如此，之前该页面积累的搜索引擎评分也浪费了。

一、CLB 代理 HTTPS 请求

nginx 的传统解决方案

方案说明

1. 假定开发者购买了负载均衡器，后端绑定了 100 台 CVM，配置网站 https://example.com 。开发者希望用户在浏览器中输入网址时，直接键入www.example.com 即可通过 HTTPS 协议安全访问（即无论 HTTP / HTTPS 请求，都返回 HTTPS，强制使用加密能力）。
2. 此时用户输入的 www.example.com 请求转发流程如下：
   该请求以 HTTP 协议传输，通过 VIP 访问 CLB 负载均衡监听器的 80 端口，并被转发到后端云服务器的 8080 端口。
3. 通过在腾讯云后端服务器的 nginx 上配置 rewrite 操作，该请求经过 8080 端口，并被重写到 https://example.com 页面。
4. 此时浏览器再次发送 https://example.com 请求到相应的 HTTPS 站点，该请求通过 VIP 访问负载均衡监听器的 443 端口，并被转发到后端云服务器的 80 端口。至此，请求转发完成。架构如下图所示：



具体配置

1. 当用户请求的 HTTP 和 HTTPS 服务的域名一样时，且 HTTPS 端口默认为 443 时，为实现以上请求转发操作，用户可以直接对后端服务器做如下配置：

```
server {
listen 80;
server_name example.com;

location / {
client_max_body_size 200m;
rewrite ^/.(.*) https://$host/$1 redirect; //在CVM上做rewrite配置
}
}
```

2. 当用户请求的 HTTP 和 HTTPS 服务的域名不同，或者端口不是 443 时，为实现以上请求转发操作，用户需要指定具体的 URL 和端口，对后端服务器做如下配置：

```
server {
listen 80;
server_name example.com;
```

```
location / {
client_max_body_size 200m;
rewrite ^/.(.*) https://xxx.xxx.xx:10011/x redirect; //在CVM上做rewrite配置
}
}
```

CLB 代理 HTTPS

1. 上述架构中，CLB 的主要作用是对 HTTPS 进行代理，因此无论是 HTTP 还是 HTTPS 请求，到了 CLB 转发给后端 CVM 时，都是 HTTP 请求。此时，**客户端到LB 时如果为 HTTPS 协议，则采用加密传输的方式，但 LB 到后端服务器依然是明文传输。**此时，开发者无法分辨出前端的请求是 HTTPS 还是 HTTPS。
2. 为了解决这个问题，腾讯云 CLB 在将请求转发给后端 CVM 时，头部 header 会植入 X-Client-Proto，从而便于开发者依据 header 内容判断请求类型：
   - X-Client-Proto: HTTP （前端为 HTTP 请求）。
   - X-Client-Proto: HTTPS （前端为 HTTPS 请求）。

存在问题

- 配置繁琐：假设客户有多个 domain + uri，有 100 台后端的 CVM 服务器，则需要在 100 台服务器上重复配置。且每增加一个 domain + uri，都需要在 100 台后端 CVM 上刷新一遍。
- 计算开销：重定向判断耗费后端 CVM 服务器的 CPU 资源。

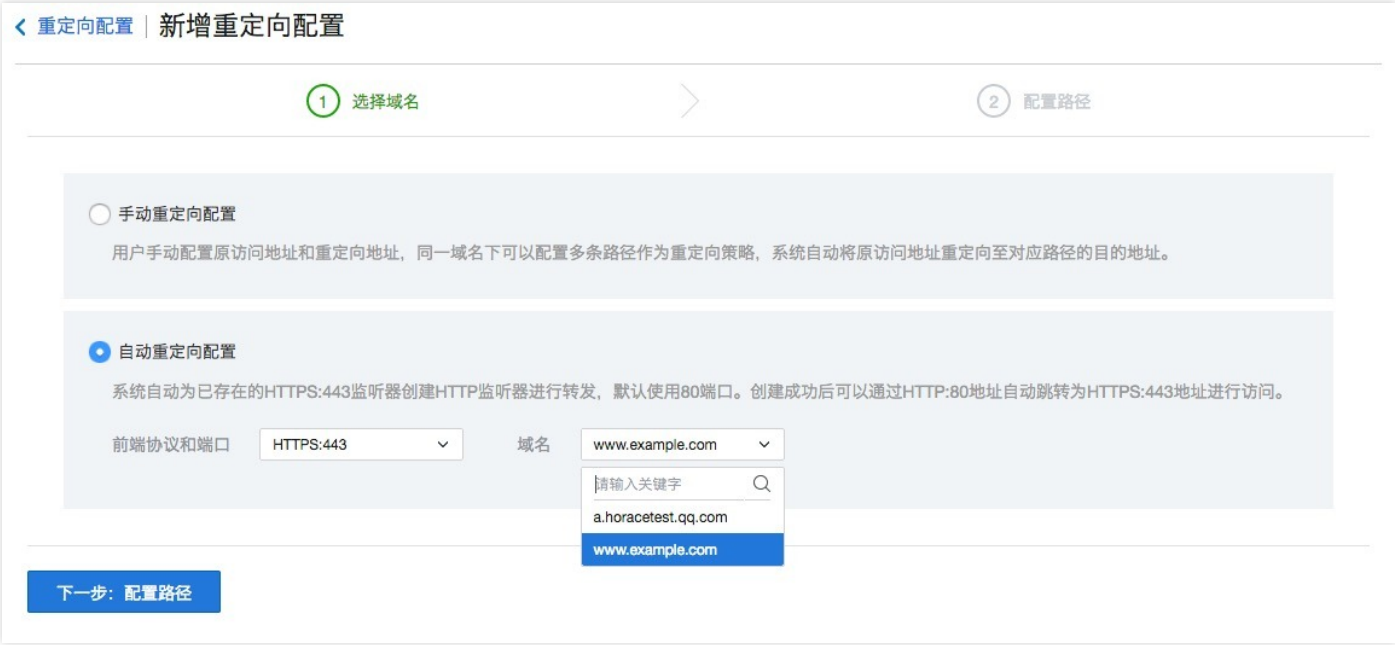
CLB 强制 HTTP 跳转方案

方案说明

假定开发者需要配置网站 https://example.com 。开发者希望用户在浏览器中输入网址时，直接键入 www.example.com 即可通过 HTTPS 协议安全访问。www.example.com 下，不仅仅是一个地址，后端关联的 URL 可能有数百的（用正则匹配），总的 real server 数量会有几百个，逐一配置难度太大。腾讯云支持一键式的，强制 HTTPS 跳转。

1. 第一步，先在 腾讯云负载均衡控制台 将 LB 的 HTTPS 监听器配置好，也就是将 https://example.com 的 Web 环境搭建好。

2. 第二步，到应用型负载均衡器控制台处启用重定向能力，目前支持域名级别，整体跳转。



方案优势

- 仅需 1 次配置：一个域名，一次配置即可完成强制 HTTPS。
- 更新：若 HTTPS 服务的 URL 有增减，只需要在控制台，重新使用该功能刷新一遍即可。

二、注意事项

- 会话保持：如 client 端访问了 example.com/bbs/test/123.html ，且后端 CVM 开启了会话保持。 当启用重定向，将流量导到 example.com/bbs/test/456.html 时，原会话保持机制将失效。
- TCP / UDP 重定向：暂不支持 IP + Port 级别的重定向，后续版本将提供。