Red Hat Enterprise Linux 7
Load Balancer 管理

Red Hat Enterprise Linux 的 Load Balancer 外掛程式

Red Hat Enterprise Linux 7 Load Balancer 管理

Red Hat Enterprise Linux 的 Load Balancer 外掛程式

法律聲明

摘要

建置一部 Load Balancer 外掛程式系統能為生產服務提供高可用性及可縮放調整的解決方案，它使用了專屬的 Linux Virtual Servers（LVS）來進行路由，並透過 Keepalived 和 HAProxy 來提供負載平衡技術。本書討論了高效能系統的配置與在 RHEL 7 中使用了 Load Balancer 技術的服務。

內容目錄

章 1. Load Balancer 總覽

Load Balancer 是個能平衡真實伺服器之間的 IP 流量的一組整合式軟體元件。它包含了兩項主要技術，以監控叢集成員與叢集服務：Keepalived 與 HAProxy。Keepalived 使用 LVS 來進行負載平衡與任務上的容錯移轉，HAProxy 則負責為 TCP 與 HTTP 應用程式進行負載平衡（load balancing）與高可用性（high-availability）服務。

1.1. keepalived

keepalived daemon 會在主動式和被動式的 LVS 路由器上執行。所有執行 **keepalived** 的路由器皆使用*虛擬路由冗餘協定*（Virtual Redundancy Routing Protocol，VRRP）。主動式路由器會以定時間隔傳送 VRRP 通告；若備份路由器無法接收這些通告，另一個新的主動式路由器將會被選出。

在主動式路由器上，**keepalived** 亦可為真實伺服器進行負載平衡任務。

Keepalived 是個與 LVS 路由器相關的控制程序。在開機時，daemon 會由 **systemctl** 指令所啓用，它會讀取 **/etc/keepalived/keepalived.conf** 這個配置檔案。在主動式路由器上，**keepalived** daemon 會啓用 LVS 服務並根據配置的拓樸來監控服務健康狀態。主動式路由器會透過使用 VRRP 定期傳送通告給備份路由器。在備份路由器上，VRRP instance 會判斷主動式路由器的運作狀態。若主動式路由器在經由使用者配置的時間間隔經過後依然無法傳送通告的話，Keepalived 便會啓用容錯移轉機制。在進行容錯移轉時，虛擬伺服器將會被清空。新的主動式路由器會接管控制 VIP、送出一則 ARP 訊息、設定 IPVS 表格項目（虛擬伺服器）、開始進行健康檢測，並開始傳送 VRRP 通告。

Keepalived 會在第 4 層或是傳輸層上執行容錯移轉，之後 TCP 便會進行基於連線的資料傳輸。當一部真實伺服器無法回應單純的逾時 TCP 連線時，**keepalived** 會偵測到伺服器已失效並將它由伺服器集區中移除。

1.2. haproxy

HAProxy 能為基於 HTTP 與 TCP 的服務（例如連至網際網路的服務和基於網站的應用程式）提供負載平衡服務。根據所選擇的負載平衡排程演算法則，**haproxy** 能夠處理以多重真實伺服器構成的單一虛擬伺服器之間的數千個連線事件。排程器會判斷連線的數量並以無權重的平等排程方式分配它們，或是以加權演算法給予能處理更高容量的伺服器更高的連線數量。

HAProxy 能讓使用者定義數個代理伺服器服務，並為這些代理伺服器的流量執行負載平衡服務。代理伺服器乃透過一個前端以及一或更多個後端所形成的。前端負責定義 IP 位址（VIP）和代理伺服器進行監聽的連接埠，以及定義特定代理伺服器所應使用的後端。

後端乃一群真實伺服器集區，並定義了負載平衡的演算法則。

HAProxy 會在第 7 層或是應用程式層上進行負載平衡管理。在大部份情況下，管理員會為基於 HTTP 的負載平衡（例如生產網站應用程式）建置 HAProxy，在此情況下，高可用性的基礎結構乃確保企業生產環境持續性的必備要素。

1.3. keepalived 和 haproxy

管理員可同時使用 Keepalived 和 HAProxy 以建立一個較為健全及可縮放的高可用性環境。透過 HAProxy 的高速與縮放性來為 HTTP，以及其它基於 TCP 的服務進行負載平衡並搭配 Keepalived 容錯移轉服務，管理員可藉由將負載分散在真實伺服器之間，以提高可用性並確保當路由器失效時，可藉由容錯移轉至備份路由器以提供持續性。

章 2. Keepalived 總覽

Keepalived 會在一個 *Active LVS 路由器*上，以及一或更多個選用性的 *Backup LVS 路由器*上運作。Active LVS 路由器負責兩項工作：

- 平衡真實伺服器之間的工作負載。
- 檢查各個真實伺服器上的服務完整性。

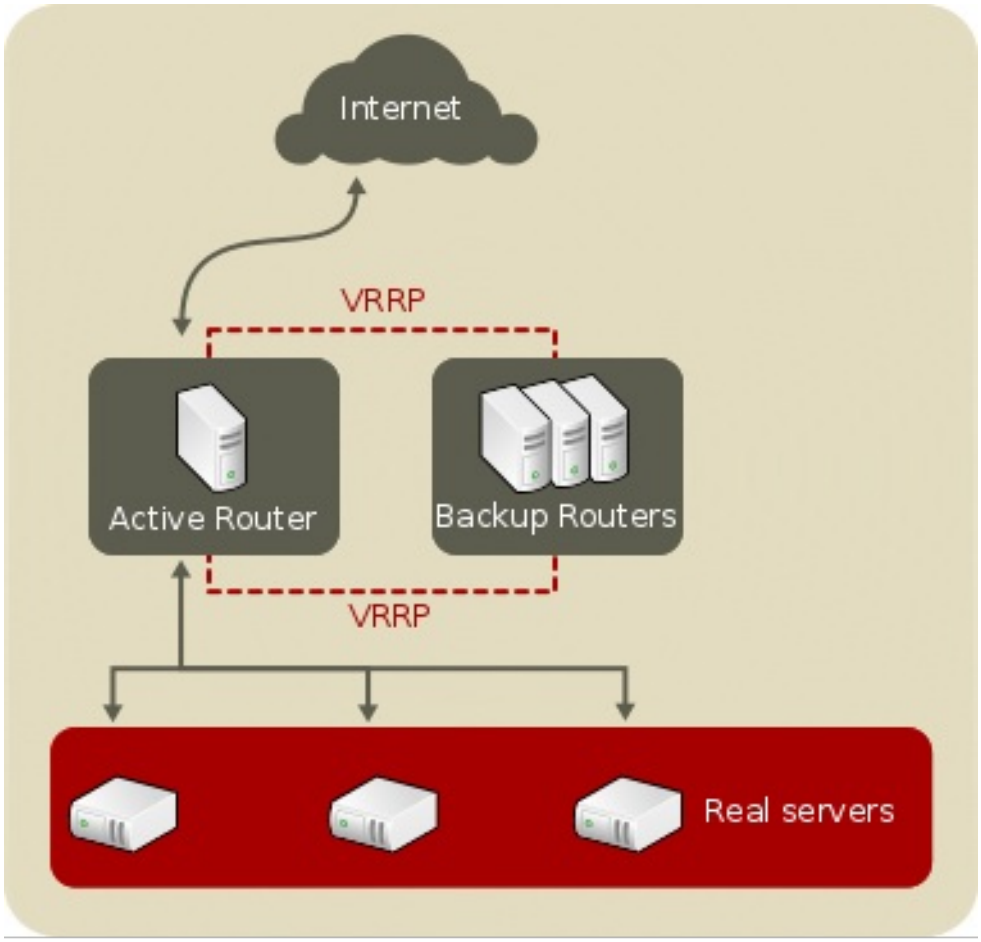
Active（master）路由器會透過*虛擬路由器冗餘協定*（VRRP）告知 backup 路由器其啓用狀態，master 路由器將必須以正常間隔送出通告。若 active 路由器停止送出通告，則新的 master 將會被選出。

本章節提供了 Load Balancer 元件及功能上的總覽，並包含了下列部分：

- 節 2.1, “基本的 Keepalived Load Balancer 配置”
- 節 2.2, “三層式的 keepalived Load Balancer 配置”
- 節 2.3, “keepalived 排程總覽”
- 節 2.4, “路由法則”
- 節 2.5, “Keepalived 的 Persistence 與 Firewall Mark”

2.1. 基本的 Keepalived Load Balancer 配置

〈圖形 2.1, “Load Balancer 的基本配置”〉顯示了包含兩個網路層的基本 Keepalived Load Balancer 配置。第一個網路層上包含了一個 active 及數個 backup LVS 路由器。各個 LVS 路由器皆擁有兩個網路介面卡，一個介面卡位於網際網路上，而另一個則位於私密網路上，這能讓它們管理兩個網路之間的流量。在此範例中，active 路由器使用了*網路位址轉譯*（或 *NAT*）來將流量由網際網路轉送至第二個網路層上的數個真實伺服器上，並依次提供必要的服務。因此，本範例中的真實伺服器連至了一個專屬的私密網路區段，並透過 active LVS 路由器來回傳送所有的公共流量。以外部的角度來看，這些伺服器會看似單一實體。



圖形 2.1. Load Balancer 的基本配置

抵達 LVS 路由器的服務請求會被指定一組*虛擬 IP*位址，亦稱為 *VIP*。這是組可公共路由、網站管理員用來與完整區域名稱（例如 www.example.com）相聯的位址，並且將會指定給一或更多個*虛擬伺服器*。虛擬伺服器乃一項配置來監聽特定虛擬 IP 的服務。當容錯移轉進行時，VIP 位址會由一個 LVS 路由器上遷移至另一個路由器上，並在該 IP 位址上保留其存在性（亦稱為*浮動 IP 位址*）。

VIP 位址能指定給將 LVS 路由器連至網際網路的相同裝置。比方說，若 **eth0** 已連至網際網路，那麼您將可分配多個虛擬伺服器給 **eth0**。此外視各項服務而定，各個虛擬伺服器皆可與獨立的裝置相聯。比方說，HTTP 流量能在 **eth0** 上於 192.168.1.111 處理，而 FTP 流量則能在 **eth0** 上於 192.168.1.222 處理。

在同時牽涉了一個 active 和一個 passive 路由器的建置情況下，active 路由器的工作就是將服務請求由虛擬 IP 位址重定向至真實伺服器。重定向乃基於〈節 2.3, “keepalived 排程總覽”〉中所詳述、八個受支援之一的負載平衡演算法來決定的。

Active 路由器也會透過三項內建式的健康檢測：基本的 TCP 連線、HTTP 以及 HTTPS 來動態式監控真實伺服器上的特定服務的整體健康狀態。關於 TCP 連線，active 路由器會定期檢查它是否能透過特定連接埠連上真實伺服器。至於 HTTP 與 HTTPS，active 路由器將會定期在真實伺服器上取一組 URL，並驗證其內容。

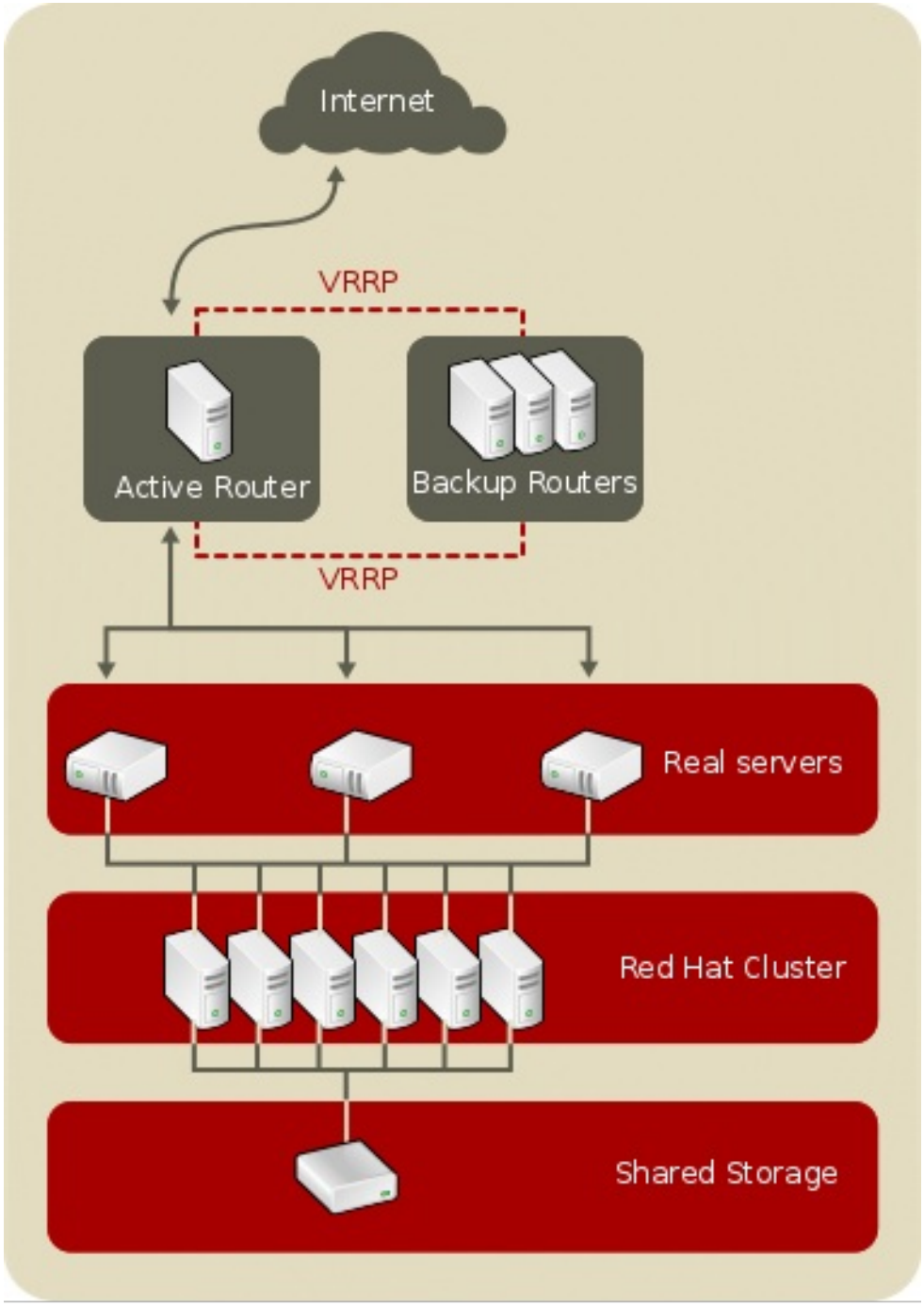
備用的路由器將負責進行預備系統的任務。路由器備援是由 VRRP 所負責處理的。開機時，所有路由器皆會加入一個 multicast 群組。此 multicast 群組會被使用來傳送與接收 VRRP 通告。因為 VRRP 是個基於優先順序的協定，擁有最高優先權的路由器將會被選擇作為 master。一旦路由器被選擇作為 master 之後，它便會負責每隔一定時間內負責傳送 VRRP 通告至 multicast 群組。

若備用的路由器無法在特定的一段時間內接收到通告（根據通告間隔）則新的 master 將會被選出。新的 master 將會接管 VIP 並傳送一則*位址解析協定*（Address Resolution Protocol，ARP）訊息。當路由器返回啓用服務時，它可能會成為一個 backup 或是 master。此特性將以路由器的優先順序而定。

使用於〈圖形 2.1, “Load Balancer 的基本配置”〉中的基本、兩層的配置，適合用來服務不會頻繁遭到改變的資料 — 例如靜態式的網頁 — 因為個別的真實伺服器不會自動同步各個節點之間的資料。

2.2. 三層式的 keepalived Load Balancer 配置

〈圖形 2.2, “三層式的 Load Balancer 配置”〉顯示了典型的三層式 Keepailved Load Balancer 拓樸。在此範例中，active LVS 路由器會將來自網際網路的路由請求導向到真實伺服器集區。接下來，真實伺服器的每台伺服器皆會透過網路存取共享的資料來源。



圖形 2.2. 三層式的 Load Balancer 配置

這項配置適用於繁忙的 FTP 伺服器，要存取的資料都儲存在單一、高可用性的伺服器上，由每個實體伺服器透過 NFS 或 Samba 來存取。這個拓樸也適用於存取單一、高可用性資料庫的網站。除此之外，使用 Load Balancer 外掛程式的「動態－動態」（active-active）設定，使用者可以設定一個高可用性的叢集，為其他類似情形提供服務。

上述範例的第三層並不一定要使用 Load Balancer 外掛程式，但不使用高可用性的解決方案可能會嚐到單點失效的苦果。

2.3. keepalived 排程總覽

使用 Keepalived 能在真實伺服器之間以極佳的靈活性分配流量，這也是所幸於各種受支援的排程演算法則。負載平衡優於其它靈活性較低的方式（例如 *Round-Robin DNS*，其 DNS 的階級特性與客戶端機器的快取可能會造成負載不平衡。此外， LVS 路由器所使用的低階請求轉送，優於應用程式階級的請求轉送，因為在網路封包階級上進行負載平衡，可大幅減少運算上的額外負載，並且能提供較佳的延展性。

使用指定的權重能為個別機器提供任意的優先權。若使用這種類型的排程方式，您亦可透過各種硬體和軟體組合來建立一組真實伺服器，並且主動式的路由器將可平均分配各個真實伺服器的負載。

Keepalived 的排程機制是由多個 kernel 修補程式 － 稱為「*IP 虛擬伺服器*」（*IPVS*，IP Virtual Server）模組 － 所提供。這些模組啓用了「*第四層*」（*L4*，layer 4）的傳輸層切換功能，能與單一 IP 位址上的多台伺服器運作。

為了更有效率地追蹤和路由轉送封包至真實伺服器，IPVS 會在 kernel 中建立一個 *IPVS 表格*。此表格會被 active LVS 路由器使用來將一個來自於虛擬伺服器位址的請求重定向至集區中的真實伺服器上，以及重定向來自於真實伺服器的請求。

2.3.1. Keepalived 排程演算法則

IPVS 表格的結構會依據管理者為任何選定的虛擬伺服器所選定的排程演算法則而定。為了讓您有最大的彈性可以使用叢集的服務類型、以及如何排程這些服務，Keepalived 支援以下列出的排程演算法則。

循環排程法

將請求依序發給集區中的伺服器。使用這演算法則時，所有真實伺服器不管能力為何，都會被視為平等。此排程模式類似循環 DNS（round-robin DNS）法，但更為細緻，因為這模式是以網路連線為基礎，而非以主機為基礎。 Load Balancer 的循環配置資源排程也不會因為快取了 DNS 的查詢項目而導致不平衡。

加權循環排程法

將請求依序發給集區中的真實伺服器，但把更多工作交給能力較強的電腦。能力的強弱會根據使用者所指定的權重而定，再根據動態負載的資訊來上下調整。

如果集區中的真實伺服器之能力相差懸殊，那麼加權循環排程會是較好的選擇。然而，如果請求所需的負載也相差懸殊，那麼權重更大的伺服器就可能會回覆過多的請求。

最少連線排程法

會把請求分散到連線數較少的電腦。因為最少連線法會透過 IPVS 表持續追蹤連至真實伺服器的現有連線，因此這是一種動態排程法則，使其成為請求負載變動高的更佳選擇。如果真實伺服器集區中的成員，都有著差不多的處理能力，那麼這方法就很適用。如果實體伺服器的能力不同，加權的最少連線排程法是更好的選擇。

加權最少連線排程法

把更多請求送到連線數少的伺服器上，對於連線數多寡的判斷，又與伺服器的能力有關。伺服器能力是以使用者指定的權重為準，權重可以由動態負載資訊來調整。當使用的伺服器配備不一時，權重功能可使最少連線法則成為理想的方案。

地區為主的最少連線排程法

有更少現有連線數（相對於目的地 IP 位址）的伺服器，將收到更多請求。這個演算法則是設計用在代理 / 快取伺服器叢集上。它會把送往一個 IP 位址的封包導向到擁有該位址的伺服器上；除非該伺服器已經超載，同時另一台伺服器的負載只有一半，這樣它就會把封包送往負載較輕的實體伺服上去。

本地為主的最少連線排程法加上複製排程法

把更多請求送往與目的 IP 相比，較少連線數的伺服器上。這個演算法則是設計給代理 / 快取伺服器叢集使用。與前述不同的是，它會將目的地 IP 位址對應到實體伺服器節點的子集合裡。請求會被導向到這個集合裡，連線數目最少的伺服器上。如果用有這 IP 位址的所有節點都超載，那麼這演算法會複製新的伺服器到這個目標 IP 位址，方法是從整體伺服器群中，將實體伺服器加入該目標 IP 的伺服器子集合裡。最高負載的節點會從實體伺服器的子集合裡移除，避免過度複製。

目的地雜湊排程法

在靜態雜湊表中查詢目的地 IP 位址，將請求分散到真實伺服器集區中。這演算法則適用於代理快取（proxy-cache）伺服器叢集。

來源雜湊排程法

在靜態雜湊表中查詢來源 IP，把請求分散到真實伺服器集區中。這個演算法則是給多防火牆的 LVS 路由器使用的。

最短預期延遲

根據特定伺服器的連線除以其被分配的權重，來將連線請求發佈至預期延時最短的伺服器上。

永不排入佇列

一個雙叉的排程器，首先會尋找並傳送連線請求至一個閒置或無連線的伺服器上。若沒有閒置的伺服器，排程器就預設值會如*最短預期延時（Shortest Expected Delay）*一般將請求傳送至一個擁有最低延時的伺服器上。

2.3.2. 伺服器權重與排程

Load Balancer 的管理者可以對真實伺服器集區中的每個節點設定「*權重*」（weight）。權重是個整數值，用於任何「*需要權重*」的排程演算法則（例如加權最少連線法），並有助 LVS 路由器更平均地裝載擁有不同能力的硬體。

權重是以相對方式來運作。舉例來說，如果一台真實伺服器的權重為 1，另一台為 5，那麼前者每取得 1 個請求時，後者就會取得 5 個。真實伺服器的權重之預設值為 1。

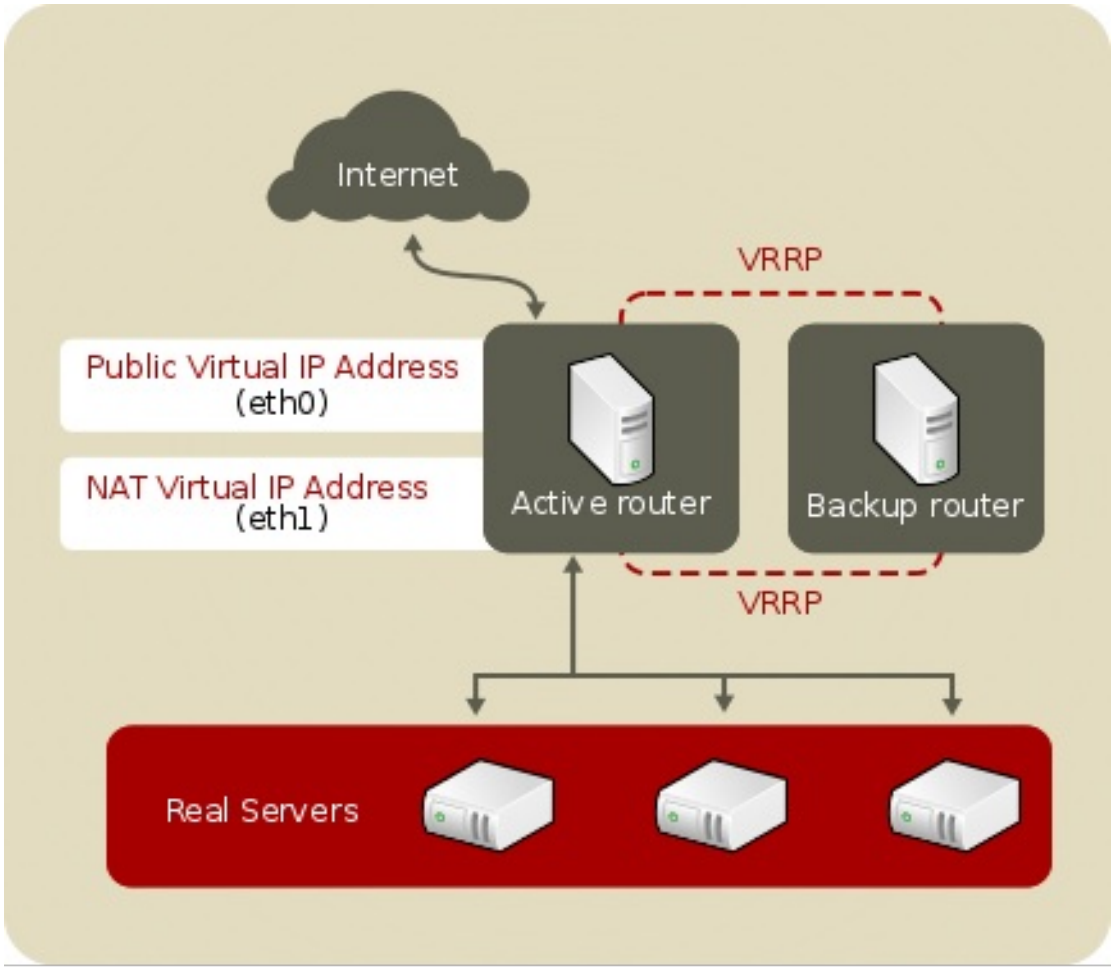
雖然針對真實伺服器集區中的不同硬體配置加入權重，能更有效率地平衡叢集的負載，但虛擬伺服器使用加權最少連線法來排程時，一台真實伺服器加入集區會短暫地導致負載不平衡。舉例來說，假設真實伺服器集區中有 3 台伺服器。伺服器 A 與 B 的權重為 1，而伺服器 C 為 2。如果 C 因故離線，其負載會平均分配給 A 與 B。然而，一旦 C 回復了，那麼 LVS 路由器會視 C 為 0 個連線，然後會把接下來的請求全部塞給 C，直到它的負載水準與 A 及 B 一樣為止。

2.4. 路由法則

Red Hat Enterprise Linux 會使用*網路位址轉譯*（*NAT 路由*）或是直接路由來啓用 Keepalived。這能在管理員使用可用硬體與整合 Load Balancer 入一個既有網路中的時候，提供極佳的靈活性。

2.4.1. NAT 路由

圖形 2.3, “實作了 NAT 路由的 Load Balancer”，描述了 Load Balancer 如何利用 NAT 路由來在網際網路與私密網路之間移動請求。



圖形 2.3. 實作了 NAT 路由的 Load Balancer

在此範例中，active LVS 路由器中有兩張 NIC。連上網際網路的 NIC 有位於 eth0 上的「*真實 IP 位址*」，同時亦有浮動 IP 位址。連到私有網路介面卡上的 NIC 在 eth1 上含有真實 IP 位址以及浮動 ID 位址。在備援情況發生時，網際網路與私有網路的虛擬介面卡會同時由 backup LVS 路由器所接手。所有位於私有網路上的真實伺服器皆會使用 NAT 路由器的浮動 IP 做為預設路徑，與 active LVS 路由器溝通，如此一來真實伺服器回應來自於網際網路上來的請求，便不會受到影響。

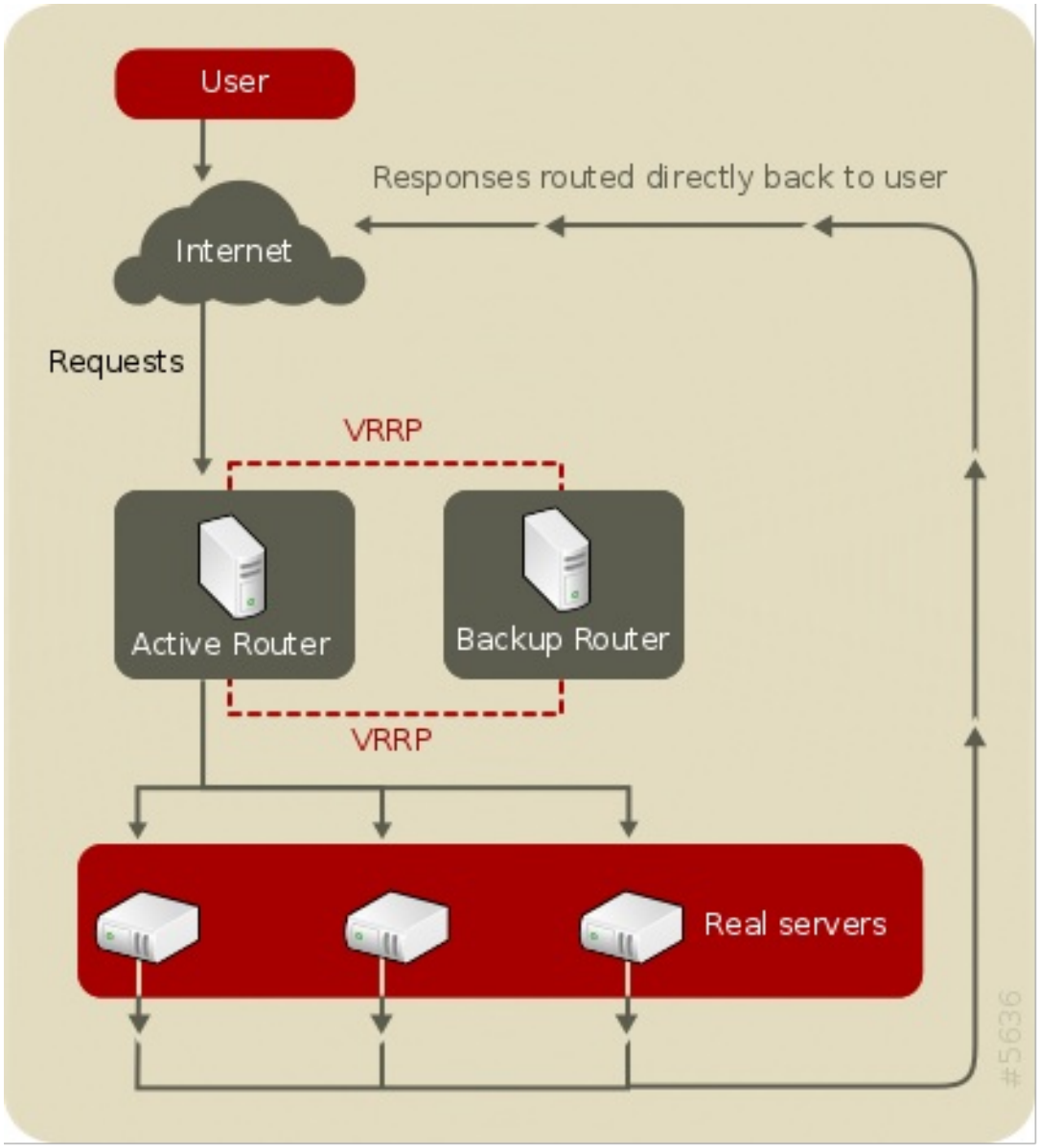
在此範例中，LVS 路由器的公開浮動 IP 位址，與私有 NAT 的浮動 IP 位址已指定給實體 NIC。儘管您能使各個浮動 IP 位址與其 LVS 路由器節點上的實體裝置相聯，不過卻不需要使用到超過兩張 NIC。

藉由此拓樸，active LVS 路由器會收到請求，並將其路由導向至正確的伺服器上。真實伺服器會處理這些請求，然後傳回給 LVS 路由器，LVS 路由器會使用網路位址轉譯，把封包中的真實伺服器位址替代成 LVS 路由器的公開 VIP 位址。這項過程稱為「*IP 偽冒*」（IP masquerading），因為真實伺服器的位址被隱藏起來，用戶端是看不見的。

使用這種 NAT 路由時，實體伺服器可以是執行多種作業系統的任何電腦。其主要缺點是 LVS 路由器可能會成為大型佈建環境中的瓶頸，因為它必須處理進、出的請求。

2.4.2. 直接路由

跟其它 Load Balancer 的網路拓樸比起來，建立使用直接路由的 Load Balancer 的效能更好。直接路由能讓實體伺服器直接處理封包，並將其路由導向至請求的使用者；而非全部透過 LVS 路由器傳送連外的封包。直接路由透過了將 LVS 路由器的工作轉為僅處理進入的封包，ˇ以減少網路效能降低的可能性。



圖形 2.4. 以直接路由來實作 Load Balancer

在典型的直接路由 Load Balancer 的設置裡，LVS 路由器能透過虛擬 IP（VIP）收取進來的伺服器請求，使用排程演算法則將這些請求導向至實體伺服器。實體伺服器會處理請求，再將結果直接傳回給用戶端，跳過 LVS 路由器。這路由方式能實現可擴充性，因為實體伺服器可以在不需要增加 LVS 路由器的負擔之下加入，幫忙將連出的封包從實體伺服器送往用戶端；否則的話，這會在高網路負載的情形下，形成瓶頸。

2.4.2.1. 直接路由與 ARP 的限制

雖然在 Load Balancer 中使用直接路由有許多好處，但還是有限制。透過直接路由的 Load Balancer 最常見的問題就是使用「*位址解析通訊協定*」（ARP，Address Resolution Protocol）。

在一般的情形下，位於網際網路的使用者發送請求到一個 IP 位址。網路路由器會利用 ARP 取得目的地的 MAC 位址，並將 MAC 位址及 IP 位址做上關聯，以發送請求。ARP 會廣播詢問所有連上網路的電腦，擁有正確 IP/MAC 位址的電腦就會收到這封包。IP/MAC 的關聯會儲存在 ARP 快取裡，這快取會定期清除（通常是每十五分鐘），再被重新填入 IP/MAC 的關聯。

直接路由 Load Balancer 設定中的 ARP 請求問題，是因為用戶請求的 IP 必須與 MAC 位址相聯，因此 Load Balancer 系統的虛擬 IP 位址也必須跟某個 MAC 位址相聯。然而，因為 LVS 路由器與實體伺服器的 VIP 是一樣的，ARP 請求會廣播到與此 VIP 相聯的所有節點上。這會造成幾個問題，比方說 VIP 直接與一台實體伺服器相聯並直接處理需求，完全不透過 LVS 路由器，這完全違背了設定 LVS 的用途。

要解決這個問題，請確保所有進來的請求都會發送到 LVS 路由器，而非任何一台真實伺服器。這可以透過使用 **arptables_jf** 或 **iptables** 封包篩選工具來達成，理由如下：

- **arptables_jf** 會避免 ARP 將 VIP 與真實伺服器建立關連。
- **iptables** 法會一開始就配置真實伺服器上的 VIP，完全迴避 ARP 的問題。

2.5. Keepalived 的 Persistence 與 Firewall Mark

在某些情況下，我們會希望讓一台用戶端重複連線到同樣的實體伺服器上，而不是由負載平衡演算法則來決定。這樣的例子有多視窗的網頁表格、cookies、SSL 以及 FTP 連線。在這些例子裡，除非連線是由同一台機器所處理，否則用戶端多半會連線失敗。Keepalived 提供了兩種方法來處理這項問題：*persistence*（持續連線）與 *firewall mark*（防火牆標記）。

2.5.1. Persistence

當啓用時，persistence 會如計時器一般地運作。當客戶端連至一項服務時，Load Balancer 會在指定的一段時間內，記住最後的一次連線。若相同的客戶端 IP 在該時段之內再次連線的話，它將會被傳送到先前所連至的相同伺服器上 — 跳過負載平衡演算法。當連線在指定的時段之外發生時，它將會根據所配置的排程規則來被處理。

Persistence 亦可讓管理員指定子網路遮罩，以套用至客戶端 IP 位址測試，並作為一項控制哪組位址擁有較高等級持續性的工具，而藉此將連線分組在該子網路中。

對於使用超過一個連接埠來進行通訊的協定（比方說 FTP）來說，為連至不同連接埠的連線分組，有時是非常重要的。 然而，persistence 並非最適合用來將連至不同連接埠的連線分組的方式。在這種情況下，建議使用 *firewall marks*。

2.5.2. Firewall Marks

防火牆標記對於一組使用於通訊協定（或一組相關協定）的連接埠來說是個簡易且有效率的方式。比方說，若 Load Balancer 已被建置來管理一個電子商務網站，防火牆標記可被使用來將 port 80 上的 HTTP 連線和 port 443 上安全的 HTTPS 連線綁在一起。藉由將相同的防火牆標記指定至虛擬伺服器的各個協定，交易的狀態資訊將能被保留，因為 LVS 路由器會在一則連線開啓之後，將所有請求轉送至相同的真實伺服器。

因為其高效率以及其容易使用的特性，Load Balancer 的管理員應盡可能在為連線分組時，使用防火牆標記來取代永續性。然而，管理員還是應該搭配防火牆標記將永續性加入至虛擬伺服器，以確保客戶端能重新連到相同伺服器上一段足夠的時間。

章 3. 為 Keepalived 設置 Load Balancer 的先決條件

使用 Keepalived 的 Load Balancer 包含了兩個基本群組：LVS 路由器與真實伺服器。要避免單點失效的問題，每個群組都應該包含至少兩台成員系統。

LVS 路由器群組應該包含兩台完全相同、或非常近似的 Red Hat Enterprise Linux 系統。一台作為 active LVS 路由器，另一台則處於熱待命模式，因此兩台電腦的處理能力應該相仿。

選擇、配置真實伺服器群組之前，請先決定要使用三種 Load Balancer 拓樸中的哪一種。

3.1. NAT Load Balancer 網路

NAT 拓樸能讓使用者大幅使用既有硬體，但限於處理大量負載的能力，因為所有進出集區的封包都會通過 Load Balancer 路由器。

網路佈局

從網路佈局的角度來看，使用 NAT 路由的 Load Balancer 之拓樸是最容易配置的，因為僅需要單存取點即可存取公開網路。真實伺服器位於私有網路內，會透過 LVS 路由器將所有請求傳回。

硬體

就硬體來說，NAT 拓樸是最有彈性的配置，因為真實伺服器並不一定得是 Linux 伺服器才能正常運作。在 NAT 拓樸中，每台真實伺服器只需要一張網路卡，因為真實伺服器只需要回應 LVS 路由器即可。另一方面，LVS 路由器需要兩張網路卡連接兩組網路。因為這拓樸會在 LVS 路由器上造成網路瓶頸，因此不妨在每台 LVS 路由器上採用 gigabit 乙太網路卡。如果 LVS 路由器用了 gigabit 乙太網路卡，連接真實伺服器與 LVS 路由器的任何交換器，都必須有至少兩組 gigabit 乙太網路連接埠，以有效處理負載。

軟體

因為在某些配置下，NAT 拓樸需要使用 **iptables**，因此除了 Keepalived 以外，還需要不少軟體配置。尤其是 FTP 服務與使用防火牆標記，需要額外配置 LVS 路由器，路由功能才能正常運作。

3.1.1. 在 Load Balancer 搭配 NAT 情境下，配置網路介面

若要搭配 NAT 設定 Load Balancer，您必須先為 LVS 路由器的公開與私有網路配置網路介面卡。在此範例中，LVS 路由器的公開介面卡（**eth0**）是 192.168.26/24（這不是可路由的 IP 網段，但我們假設這台 LVS 路由器之前有防火牆），而連至真實伺服器的私有介面卡（**eth1**）位於 10.11.12/24 網段。

重要

請注意，編輯以下屬於 **network** 服務與 Load Balancer 的檔案，並不相容於 **NetworkManager** 服務。

在 active LVS 路由節點或「主」LVS 路由節點上，公開介面的網路 script **/etc/sysconfig/network-scripts/ifcfg-eth0** 看起來類似：

```
DEVICE=eth0
BOOTPROTO=static
ONBOOT=yes
IPADDR=192.168.26.9
```

```
NETMASK=255.255.255.0
GATEWAY=192.168.26.254
```

LVS 路由器上的私有 NAT 介面之檔案 **/etc/sysconfig/network-scripts/ifcfg-eth1** 則類似：

```
DEVICE=eth1
BOOTPROTO=static
ONBOOT=yes
IPADDR=10.11.12.9
NETMASK=255.255.255.0
```

在此範例中，LVS 路由器介面的虛擬 IP 是 192.168.26.10，NAT 或私有介面的虛擬 IP 則是 10.11.12.10。因此，真實伺服器的路由請求會回到 NAT 介面的虛擬 IP。

重要

本節中的乙太網路介面卡配置之設定值範例，是給 LVS 路由器的真實 IP 位址使用，而「*不是*」給浮動 IP 位址使用的。

配置主 LVS 路由器節點的網路介面之後，配置 backup LVS 路由器的真實網路介面 — 請注意，不要有任何 IP 位址與網路上的其它 IP 位址相衝突。

重要

確定備用節點上的每個介面，都與主節點上的介面位於同一組網路上。舉例來說，如果 eth0 連接至主節點上的公開網路，那麼也必須連接到備用節點的公開網路上。

3.1.2. 真實伺服器上的路由

在 NAT 拓樸中配置真實伺服器網路介面時，最重要的是設定 LVS 路由器的閘道器之 NAT 浮動 IP 位址。在此範例中，IP 位址為 10.11.12.10。

注意

一旦真實伺服器上的網路介面都已啓動，電腦就無法透過其它方式 ping 或連接至公開網路。這很正常。然而您可以 ping LVS 路由器的私有介面之真實 IP，亦即此範例中的 10.11.12.9。

因此，真實伺服器的 **/etc/sysconfig/network-scripts/ifcfg-eth0** 檔案看起來類似：

```
DEVICE=eth0
ONBOOT=yes
BOOTPROTO=static
IPADDR=10.11.12.1
NETMASK=255.255.255.0
GATEWAY=10.11.12.10
```

警告

如果真實伺服器有超過一組網路介面配置有 **GATEWAY=** 一行，那麼第一組出現的會取得閘道器。因此如果 **eth0** 與 **eth1** 都已配置，且 **eth1** 用於 Load Balancer，那麼真實伺服器可能就無法正確處理路由的請求。

最好把多餘的網路介面卡關掉，作法是在 **/etc/sysconfig/network-scripts/** 目錄中的網路卡 script 裡，設置 **ONBOOT=no**，或確定第一張出現的網路卡之閘道器已被正確設定。

3.1.3. 在 LVS 路由器上啓用 NAT 路由設定

在簡易的 NAT Load Balancer 配置裡，每個叢集服務都僅使用一組連接埠，例如 HTTP 使用連接埠號 80，那麼管理者只需要在 LVS 路由器上啓用封包轉送功能，就可以讓這些請求在外在世界與真實伺服器之間正確地路由。然而，當叢集服務需要使用超過一組連接埠，以在同一個使用者 session 中連接同一台真實伺服器時，您便需要進行額外配置。

一旦 LVS 路由器啓用了封包轉送功能，且真實伺服器已經設定完成並執行叢集服務，請使用 **keepalived** 來配置 IP 資訊。

警告

請勿手動編輯網路 script 或使用網路 script 的編輯程式，來配置 **eth0** 或 **eth1** 的浮動 IP 位址。請透過 **keepalived.conf** 檔案來代替。

完成後，請啓動 **keepalived** 服務。一旦這項服務啓動並處於執行狀態時，active LVS 路由器便會開始將請求路由至真實伺服器集區中。

3.2. 透過直接路由進行負載平衡

直接路由（direct routing）能讓真實伺服器處理封包並將封包直接路由轉送至請求的使用者，而非透過 LVS 路由器傳送向外的封包。若要進行直接路由，真實伺服器需要透過 LVS 路由器實體連接至一個網路區段，並且也要能處理和轉送向外的封包。

網路佈局

在直接路由的 Load Balancer 設定中，LVS 路由器需要接收連入的請求，並將這些請求路由到適當的真實伺服器上以供處理。接下來真實伺服器就需要「*直接*」將回應路由至用戶端。因此，舉例來說，如果用戶端位於網際網路上，並透過 LVS 路由器發送封包到真實伺服器，那麼真實伺服器就必須透過網際網路直接連上用戶端。這可以透過為真實伺服器配置閘道器，將封包轉送到網際網路上來達成。集區中的每台真實伺服器都可以擁有自己的閘道器（並各自連上網際網路），提供最大的吞吐量與擴充性。然而，對於典型的 Load Balancer 來說，真實伺服器僅能透過單一的閘道器（也就是單一網路連線）來通訊。

硬體

使用直接路由的 Load Balancer 系統之硬體需求跟其它用於 Load Balancer 的拓樸類似。LVS 路由器需要執行 Red Hat Enterprise Linux 來處理連入的請求、並為真實伺服器進行負載平衡，而真實伺服器就不一定要是 Linux 機器才能正常運作。LVS 路由器需要 1 或 2 張網路卡（端視有沒有備用路由器而定）。您可以在每個配置中使用兩張網路卡，明確地分開網路流量 — 連入的請求由一張網路卡處理，路由封包則交給另一張網路卡處理。

因為真實伺服器會繞過 LVS 路由器，並直接發送向外的封包給用戶端，因此需要連往網際網路的閘道器。欲求最大效能與可用性，每台真實伺服器可以連上自己的獨立閘道器，透過專屬的連線連上用戶端所屬的網路（例如網際網路或企業內部網路）。

軟體

keepalived 以外還有一些配置工作要做，尤其是透過直接路由使用 Load Balancer 時，遇到 ARP 問題的使用者。詳情請參閱〈節 3.2.1, “直接路由與 **arptables_jf**”〉或〈節 3.2.2, “直接路由與 **iptables**”〉。

3.2.1. 直接路由與 arptables_jf

為了要使用 **arptables_jf** 配置直接路由，每台真實伺服器都必須配有虛擬 IP 位址，藉以直接路由封包。真實伺服器會忽視 VIP 的 ARP 請求，同時任何可能含有 VIP 的 ARP 封包都會被修改，以包含真實伺服器的 IP 位址，而非 VIP。

應用程式使用 **arptables_jf** 這方法就可以綁定到真實伺服器所服務的每個獨立 VIP 或連接埠。舉例來說，**arptables_jf** 方法能讓 Apache HTTP Server 的多個 instance 直接綁定在系統上的多個 VIP 上執行。使用 **arptables_jf** 而不是 **iptables** 指令，還有效能上的顯著好處。

然而，使用 **arptables_jf** 法，VIP 就不能透過標準的 Red Hat Enterprise Linux 系統配置工具，好在系統開機時啓動。

要配置每台真實伺服器忽略每個虛擬 IP 位址的 ARP 請求，請進行以下步驟：

1. 在每台真實伺服器上，為每個虛擬 IP 位址建立 ARP 表的條目（real_ip 是導向程式用來與真實伺服器溝通的 IP 位址；通常這是 **eth0** 的 IP 位址）：

   ```
   arptables -A IN -d <virtual_ip> -j DROP
   arptables -A OUT -s <virtual_ip> -j mangle --mangle-ip-s <real_ip>
   ```

   這會導致真實伺服器忽略虛擬 IP 位址的所有 ARP 請求，並改變任何向外、可能包含虛擬 IP 位址（而非伺服器的真實 IP 位址）的 ARP 回應。唯一應該回應任何 VIP 的 ARP 請求之節點，是目前的 active LVS 節點。

2. 在每台真實伺服器完成這項工作後，在每台真實伺服器上執行以下指令，以儲存 ARP 表的條目：

   service arptables_jf save

   chkconfig --level 2345 arptables_jf on

   chkconfig 指令會讓系統在開機時重新載入 arptables 的配置 — 在網路啓動之前。

3. 使用 **ip addr** 建立 IP 別名，在所有真實伺服器上配置虛擬 IP 位址。例如：

   ```
   # ip addr add 192.168.76.24 dev eth0
   ```

4. 配置 Keepalived 以進行直接路由。這能透過將 ***lb_kind DR*** 附加至 **keepalived.conf** 檔案。詳情請參閱〈*章 4, 搭配 Keepalived 進行 Load Balancer 的初始配置*〉。

3.2.2. 直接路由與 iptables

另一個解決 ARP 問題的方法，是建立 **iptables** 防火牆規則。要使用 **iptables** 配置直接路由，您必須新增規則，這規則會建立通透的代理，讓真實伺服器能為發送到這 VIP 位址的封包提供服務，即使這 VIP 位址不存在於系統上也無妨。

配置 **iptables** 比配置 **arptables_jf** 更容易。這方法也能完全避免 LVS ARP 問題，因為虛擬 IP 位址只存在於 active LVS 轉發者上。

然而，跟使用 **arptables_jf** 比起來，使用 **iptables** 會有效能上的問題，因為每個封包在轉送 / 偽冒時，都會產生負荷。

您也無法使用 **iptables** 來重新使用連接埠。舉例來說，要執行兩個獨立、使用 80 連接埠的 Apache HTTP Server 服務就是不可能的事情，因為兩者都必須與 ***INADDR_ANY*** 綁定，而非虛擬的 IP 位址。

要配置直接路由使用 **iptables** 法，請執行以下步驟：

1. 在每一台真實伺服器上，對要提供服務對象的 VIP、連接埠、以及通訊協定（TCP 或 UDP）組合上執行指令：

   iptables -t nat -A PREROUTING -p <tcp|udp> -d <vip> --dport <port> -j REDIRECT

   這指令會讓真實伺服器處理流向 VIP 與連接埠的封包。

2. 在每台真實伺服器上儲存配置：

   ```
   # service iptables save
   # chkconfig --level 2345 iptables on
   ```

   上述指令會讓系統在開機時重新載入 **iptables** 配置 — 在網路啓動之前。

3.2.3. 直接路由與 sysctl

當利用直接路由時，另一項可用來解決 ARP 限制上的方法就是透過 **sysctl** 介面。管理員可配置兩項 **systcl** 設定，以讓真實伺服器不會在 ARP 請求中宣布 VIP，並且不會回應 VIP 位址的 ARP 請求。若要啓用這項功能，請執行以下指令：

```
echo 1 > /proc/sys/net/ipv4/conf/eth0/arp_ignore
echo 2 > /proc/sys/net/ipv4/conf/eth0/arp_announce
```

此外，您可新增下列幾行資訊至 /etc/sysctl.d/arp.conf 中：

```
net.ipv4.conf.eth0.arp_ignore = 1
net.ipv4.conf.eth0.arp_announce = 2
```

3.3. 把配置放在一起

決定要使用哪種前置路由方法後，硬體應該要透過網路連在一起。

重要

LVS 路由器的介面裝置必須配置好，存取同樣的網路。例如如果 **eth0** 連接到公開網路，而 **eth1** 連接到私有網路，那麼 backup LVS 路由器的同樣裝置就必須連上同樣的網路。

同時，開機時出現的第一個介面的閘道器會加入路由表，列在之後出現的其它介面之閘道器會被忽略。這在配置真實伺服器時，尤其重要。

在實體連接硬體後，請配置主要與備份 LVS 路由器上的網路介面卡。您可透過使用一個例如 **system-config-network** 的圖形化應用程式或是藉由手動編輯網路 script 來完成這項工作。欲取得更多有關於透過使用 **system-config-network** 來新增裝置的相關資訊，請參閱【*Red Hat Enterprise Linux 建置指南*】中的《*網路配置*》章節。

3.3.1. Load Balancer 的一般網路提示

在使用 Keepalived 配置 Load Balancer 之前，請先配置 LVS 路由器上的公開、私有網路之真實 IP 位址。每個拓樸的章節都會提供網路的範例，但需要您輸入實際的網路位址。以下是啓動網路介面或檢查網路介面之狀態時所能使用的一些指令。

啓動真實網路介面卡

要啓動真實網路介面卡，請以 root 身份使用以下指令，並以實際的介面卡數字（**eth0** 與 **eth1**）取代 *N*。

```
/usr/sbin/ifup ethN
```

警告

*請勿*使用 **ifup** script 來啓用任何您可能會使用到 Keepalived（**eth0:1** 或是 **eth1:1**）來配置的浮動 IP 位址。請使用 **service** 或是 **systemctl** 指令來啓用 **keepalived**。

停用真實網路介面卡

要停用真實網路介面卡，請以 root 身份使用以下指令，並以實際的介面卡數字（**eth0** 與 **eth1**）取代 *N*。

```
/usr/sbin/ifdown ethN
```

檢查網路介面卡的狀態

任何時候當您需要檢查網路介面狀態時，請輸入：

```
/usr/sbin/ifconfig
```

要檢視電腦的路由表，請執行以下指令：

```
/usr/sbin/route
```

3.4. 多連接埠服務與 Load Balancer

建立多連接埠的 Load Balancer 服務時，任何拓樸下的 LVS 路由器都需經過額外的配置。多連接埠服務可以用人工方式創見，方法是透過使用防火牆標記將不同、但相關的通訊協定（例如 HTTP 的 80 連接埠與

HTTPS 的 443 連接埠）放在一起，或當 Load Balancer 與真實的多連接埠通訊協定（例如 FTP）搭配使用時。不管在哪一種情況下，LVS 路由器皆會使用防火牆標記來以相同的方式處理連向不同連接埠，但卻有著同樣防火牆標記的封包。同時，與持續性相結合時，只要連線處於持續性參數所指定的時間內，防火牆標記便會確保來自用戶端機器的連線會導向至同樣的主機。

不幸的是，用來平衡真實伺服器負載的機制 — IPVS — 會辨識出指定給某個封包的防火牆標記，但自己無法指定防火牆標記。「*指定*」防火牆標記的工作必須由網路封包篩選程式，亦即 **iptables** 來進行。

3.4.1. 指定防火牆標記

要指定防火牆標記給連往某個特定連接埠的封包，管理者必須使用 **iptables**。

本節的範例描述如何處理 HTTP 與 HTTPS；然而，FTP 是另一個常見、叢集化的多連接埠通訊協定。

使用防火牆標記的基本規則，是每個在 Keepalived 中使用防火牆標記的每個通訊協定，都有相對應的 **iptables** 規則，以指定標記至網路封包上。

在建立網路封包篩選規則之前，請確定沒有其它既有規則。若要這麼做，請以 root 身份在終端機中輸入：

```
/usr/sbin/service iptables status
```

如果 **iptables** 不在執行中，終端機會立即出現提示符號。

如果 **iptables** 處於啓用狀態，那麼系統會顯示一連串的規則。如果出現規則的話，請輸入以下指令：

```
/sbin/service iptables stop
```

如果現有的規則很重要，請檢查 **/etc/sysconfig/iptables** 的內容，並在進行下一步之前，複製任何值得保存的規則到安全的地方。

第一項與 Load Balancer 相關的配置防火牆規則，就是允許 VRRP 流量，以讓 Keepalived 服務能運作。

```
/usr/sbin/iptables -I INPUT -p vrrp -j ACCEPT
```

以下是指定到同樣防火牆標記（80）到向內的流量、目的地為浮動 IP 位址 *n.n.n.n*（連接埠 80 與 443）的規則。

```
/usr/sbin/iptables -t mangle -A PREROUTING -p tcp -d n.n.n.n/32 -m multiport
--dports 80,443 -j MARK --set-mark 80
```

請注意，您必須以 root 身份登入，並在第一次發出規則前載入 **iptables** 模組。

在上述 **iptables** 指令中，*n.n.n.n* 是您 HTTP 與 HTTPS 虛擬伺服器的浮動 IP 位址。這些指令能指定任何送往 VIP、特定連接埠、且擁有防火牆標記 80 的網路流量，接下來會由 IPVS 所辨識、轉送。

警告

以上指令會即刻生效，不過將無法在系統重新開機時維持永續性。

3.5. 配置 FTP

FTP（檔案傳輸通訊協定，File Transport Protocol）是較老舊且複雜的多通訊埠通訊協定，對 Load Balancer 的環境來說，是項挑戰。要了解這項挑戰的本質，我們必須先了解 FTP 運作時的幾項關鍵要素。

3.5.1. FTP 如何運作

大部分伺服器與用戶端之間的關係，是用戶端機器會在特定連接埠上，開啓與伺服器之間的連線，然後伺服器會在該連接埠上回應用戶端。當 FTP 用戶端連上 FTP 伺服器時，會透過連接埠 21 開啓連線。然後「*用戶端*」會告訴 FTP 的「*伺服器*」端，要以「*主動*」（active）或「*被動*」（passive）方式來連接。此處用戶端選擇的連接方式會決定伺服器回應的方式、以及接下來要使用哪些連接埠。

兩種資料連線方式是：

主動連線

建立主動連線後，「*伺服器*」會開啓通往用戶端的資料連線，連接埠從 20 改為用戶端上的高端連接埠。所有來自伺服器的資料都會通過此連線傳遞。

被動連線

建立被動連線後，「*用戶端*」會要求 FTP 伺服器建立一個被動連線通訊埠，這可以是 10,000 以上的任何連接埠。伺服器接下來會綁定這個高端連接埠給這個特定的 session 使用，並將這連接埠號傳回給用戶端。然後用戶端會開啓新的綁定連接埠用以傳遞資料。用戶端要求的每個資料都會開啓新的資料連線。現今從伺服器要求資料時，FTP 用戶端多半採取被動連線。

注意

「*用戶端*」— 而非伺服器端 — 會決定連線類型。這表示要有效地將 FTP 叢集化，您必須配置 LVS 路由器處理主動與被動連線。

FTP 這種用戶端與伺服器之間的關係可能會開啓大量 Keepalived 所不知道的連接埠。

3.5.2. 對 Load Balancer 路由的影響

IPVS 封包轉送只會根據能辨識的連接埠號或防火牆標記，允許連線進出叢集。如果叢集外的用戶端試圖開啓 IPVS 並未配置的連接埠，IPVS 就會中斷此連線。同樣地，如果真實伺服器試著開啓 IPVS 所不知道的連線，向外連至網際網路，IPVS 也會中斷此連線。這表示「*所有*」來自網際網路的 FTP 連線都「*必須*」擁有同樣的防火牆標記，同時所有來自 FTP 伺服器的連線都「*必須*」使用網路封包篩選規則，適當地轉送到網際網路上。

注意

要啓用被動 FTP 連線，請確定您已經載入了 **`ip_vs_ftp`** kernel 模組，方法是在命令列中以 root 身份執行 **`modprobe ip_vs_ftp`**。

3.5.3. 建立網路封包篩選規則

在為 FTP 指定任何 **`iptables`** 規則之前，請查閱〈節 3.4.1, “指定防火牆標記”〉中關於多連接埠服務、以及檢查現有網路封包篩選規則的技巧。

以下為會將防火牆標記 21 指定至 FTP 流量的規則。

3.5.3.1. 主動連線的規則

主動連線的規則會通知 kernel 接受來自「*內部*」浮動 IP 位址的 20 號連接埠（FTP 資料埠）的連線，並加以轉送。

以下 **iptables** 指令能允許 LVS 路由器接受來自真實伺服器、且 IPVS 所不知道的向外連線：

```
/usr/sbin/iptables -t nat -A POSTROUTING -p tcp -s n.n.n.0/24 --sport 20 -j MASQUERADE
```

在 **iptables** 指令中，請以定義於 **keepalived.conf** 檔案的 ***virtual_server*** 部分中的 NAT 介面卡之內部網路介面卡的浮動 IP 前三碼來替換 *n.n.n*。

3.5.3.2. 被動連線的規則

被動連線的規則會指定正確的防火牆標記給來自網際網路、連往服務的浮動 IP 之連線到高端連接埠 — 10,000 到 20,000 之間。

警告

如果您想限制被動連線的連接埠範圍，就必須配置 VSFTP 伺服器使用相對應的範圍。這可以透過加入以下幾行到 **/etc/vsftpd.conf** 檔案中來達成：

```
pasv_min_port=10000

pasv_max_port=20000
```

請勿設定 **pasv_address** 來覆寫真實 FTP 伺服器位址，因為這會由 LVS 更新為虛擬 IP 位址。

欲知如何配置其它 FTP 伺服器，請查閱該產品的文件。

這範圍應該適用大部分情況，然而，您可以擴大這範圍，納入所有可用的非安全性連接埠，方法是將下述指令的 **10000:20000** 改為 **1024:65535**。

以下 **iptables** 指令可以指定任何送往浮動 IP、且擁有防火牆標記 21 的網路流量，接下來會由 IPVS 辨識、轉送。

```
/usr/sbin/iptables -t mangle -A PREROUTING -p tcp -d n.n.n.n/32 --dport 21 -j MARK --set-mark 21
```

```
/usr/sbin/iptables -t mangle -A PREROUTING -p tcp -d n.n.n.n/32 --dport 10000:20000 -j MARK --set-mark 21
```

在 **iptables** 指令中，*n.n.n.n* 應替換為 **keepalived.conf** 檔案的 ***virutal_server*** 部分中所定義的 FTP 虛擬伺服器浮動 IP。

警告

以上指令會即刻生效，不過將無法在系統重新開機時維持永續性。

最後，您將需要確保正確的服務將在正確的 runlevel 上啓用。

3.6. 儲存網路封包篩選設定

配置好網路封包篩選規則之後，請儲存設定，好在重新開機之後依然適用。如使用的是 **iptables**，請輸入以下指令：

/usr/sbin/service iptables save

這會將設定儲存在 **/etc/sysconfig/iptables** 中，開機時會重新載入。

當此檔案寫入後，您便能使用 **/usr/sbin/service** 指令來啓用、停用和檢查 **iptables** 的狀態（使用狀態切換選項）。**/usr/sbin/service** 會為您自動載入正確的模組。

最後，您將需要確保正確的服務將在正確的 runlevel 上啓用。

3.7. 開啓封包轉送與 Nonlocal 綁定

若要確保 Keepalived 服務能正確轉送網路封包至真實伺服器上，各個路由器節點在 kernel 中的 IP 轉送功能皆必須開啓。請以 root 登入並將 **/etc/sysctl.conf** 中的 **net.ipv4.ip_forward = 0** 一行改為：

```
net.ipv4.ip_forward = 1
```

當您重新啓動系統時，變更便會生效。

HAProxy 中的 Load Balancing 也需要能綁定至一組*nonlocal（非本機）*的 IP 位址，代表它不會被分配至本機系統上的裝置。這能讓一個運作中的負載平衡 instance 綁定至一組非本機的的 IP 位址，以進行容錯移轉。

若要啓用，請將 **/etc/sysctl.conf** 中的 **net.ipv4.ip_nonlocal_bind** 一行編輯為：

```
net.ipv4.ip_nonlocal_bind = 1
```

當您重新啓動系統時，變更便會生效。

若要檢查 IP 轉送是否已開啓，請以 root 身份輸入以下指令：

/usr/sbin/sysctl net.ipv4.ip_forward

若要檢查 nonlocal 綁定是否已開啓，請以 root 身份輸入以下指令：

/usr/sbin/sysctl net.ipv4.ip_nonlocal_bind

若以上指令回傳了 **1**，則代表相應的設定已啓用。

3.8. 在真實伺服器上配置服務

若真實伺服器為 Red Hat Enterprise Linux 系統，請將相應的伺服器 daemon 設為在開機時啓用。這些 daemon 可包含用於網站服務的 **httpd**，或是用於 FTP 或 Telnet 服務的 **xinetd**。

有時您可能需要遠端存取真實伺服器，因此您應安裝並執行 **sshd** daemon。

章 4. 搭配 Keepalived 進行 Load Balancer 的初始配置

安裝了 Load Balancer 套件之後，您必須進行一些基礎步驟，以設定 LVS 路由器和真實伺服器搭配使用 Keepalived。本章節詳細涵蓋了這些初始步驟。

4.1. 基本的 Keepalived 配置

在這基本範例中，有兩台系統被配置為負載平衡器。LB1（啓用）與 LB2（備用）會將請求路由至一組集區，集區包含了執行 **httpd** 的四台網站伺服器，真實 IP 位址為 191.168.1.20 到 192.168.1.24，並共享一組虛擬 IP 位址 192.168.1.11。每台負載平衡器都有兩張網路卡（**eth0** 與 **eth1**），一張用來處理外部的網際網路流量，另一張用來將請求路由至真實伺服器。此處使用的負載平衡演算法則是循環法（Round Robin），路由方法為 NAT。

4.1.1. 建立 `keeaplived.conf` 檔案

Keepalived 是透過 **keepalived.conf** 檔案進行配置。要建立 load balancer 拓樸，如 節 4.1, “基本的 Keepalived 配置” 所示，請使用文字編輯器開啓 **keepalived.conf**。舉例來說：

```
vi /etc/keepalived/keepalived.conf
```

如〈節 4.1, “基本的 Keepalived 配置”〉所述，基本、配置好的負載平衡系統會有 **keepalived.conf** 檔案，如下所述：

4.1.1.1. 全域定義

keepalived.conf 檔案的「全域定義」一節能讓管理者指定負載平衡器發生變更時，進行通知的詳細資料。請注意「全域定義」是選用的，Keepalived 的配置不一定需要這一部分。

```
global_defs {

   notification_email {
       admin@example.com
   }
   notification_email_from noreply@example.com
   smtp_server 127.0.0.1
   smtp_connect_timeout 60
}
```

notification_email 參數指的是負載平衡器的管理者，而 ***notification_email_from*** 是負載平衡器狀態改變的發送者位址。SMTP 的相關配置指定了發送通知所使用的郵件伺服器。

4.1.1.2. VRRP Instance

```
vrrp_sync_group VG1 {
   group {
      RH_EXT
      RH_INT
```

```
    }

vrrp_instance RH_EXT {
    state MASTER
    interface eth0
    virtual_router_id 50
    priority 100
    advert_int 1
    authentication {
        auth_type PASS
 auth_pass password123
    }
    virtual_ipaddress {
    10.0.0.1
    }
}

vrrp_instance RH_INT {
   state MASTER
   interface eth1
   virtual_router_id 2
   priority 100
   advert_int 1
   authentication {
       auth_type PASS
       auth_pass password123
   }
   virtual_ipaddress {
       192.168.1.1
   }
}
```

在此範例中，***vrrp_sync_group*** 一節透過任何狀態上的變更（例如故障後復原），將 VRRP 群組定義在一起。在此範例中，有個 instance 是定義給用來與網際網路通訊的外部介面（RH_EXT），以及一組內部介面（RH_INT）。

vrrp_instance RH1 一行詳述了VRRP 服務 daemon 的虛擬介面配置。這會建立虛擬 IP instance。**state MASTER** 標示出啓用中的伺服器；配置中的備份伺服器也有上述的設置，但配置為 **state BACKUP**。

interface 參數指定了這虛擬 IP instnace 的實體介面名稱，而 **virtual_router_id** 是這 instance 的數字識別子。**priority 100** 指定了備援發生時，接手的介面之順序；數字越小，優先順序越高。

authentication 為伺服器指定備援同步時使用的身份認證類型（**auth_type**）與密碼（**auth_pass**）。**PASS** 指定了密碼認證；Keepalived 也支援 **AH**（認證表頭，Authentication Header）作為連結的完整性使用。

最後，**virtual_ipaddress** 選項會指定虛擬 IP 位址。

4.1.1.3. 虛擬伺服器的定義

```
virtual_server 192.168.1.11 80 {
    delay_loop 6
    lb_algo rr
    lb_kind NAT
```

```
    real_server 192.168.1.20 80 {
        TCP_CHECK {
                connect_timeout 10
        }
    }
    real_server 192.168.1.21 80 {
        TCP_CHECK {
                connect_timeout 10
        }
    }
    real_server 192.168.1.22 80 {
        TCP_CHECK {
                connect_timeout 10
        }
    }
    real_server 192.168.1.23 80 {
        TCP_CHECK {
                connect_timeout 10
        }
    }

}
```

在這一區裡，**virtual_server** 會以 IP 位址最先配置。然後 **delay_loop** 會配置每次健康檢查之間的秒數。**lb_algo** 選項會指定用於可用性的演算法則（在此例子中，**rr** 代表循環法 － Round-Robin）。**lb_kind** 選項會決定路由方式，此例使用 NAT（**nat**）。

配置虛擬伺服器的詳細資料後，就要配置 **real_server** 選項，同樣先配置 IP 位址。**TCP_CHECK** 一節會使用 TCP 檢查真實伺服器的可用性。**connect_timeout** 會配置逾時發生前所經過的秒數。

4.2. Keepalived 的直接路由配置

Keepalived 的直接路由配置與 NAT 的配置類似。以下範例會配置 Keepalived，為一群使用 HTTP（連接埠 80）的真實伺服器提供負載平衡。要配置直接路由（Direct Routing），請變更 ***lb_kind*** 參數為 **DR**。其它配置選項會在〈節 4.1, “基本的 Keepalived 配置”〉中討論。

```


global_defs {
   notification_email {
     admin@example.com
   }
   notification_email_from noreply_admin@example.com
   smtp_server 127.0.0.1
   smtp_connect_timeout 60
}

vrrp_instance RH_1 {
    state MASTER
    interface eth0
    virtual_router_id 50
    priority 1
    advert_int 1
    authentication {
```

```
        auth_type PASS
        auth_pass password123
    }
    virtual_ipaddress {
        172.31.0.1
    }
}

virtual_server 172.31.0.1 80
    delay_loop 10
    lb_algo rr
    lb_kind DR
    persistence_timeout 9600
    protocol TCP

    real_server 192.168.0.1 80 {
        weight 1
        TCP_CHECK {
          connect_timeout 10
          connect_port    80
        }
    }
    real_server 192.168.0.2 80 {
        weight 1
        TCP_CHECK {
          connect_timeout 10
          connect_port    80
        }
    }
    real_server 192.168.0.3 80 {
        weight 1
        TCP_CHECK {
          connect_timeout 10
          connect_port    80
        }
    }
}
```

4.3. 啓動服務

執行以下指令以啓動服務：

```
systemctl start keepalived.service
```

要讓 Keepalived 服務在開機後自動執行，請執行以下指令：

```
systemctl enable keepalived.service
```

章 5. HAProxy 配置

本章解釋了基本設定的配置方式，點出系統管理者在高可用性環境中建置 HAProxy 服務時，會遇到的一般配置選項。

HAProxy 自己擁有一組進行負載平衡用的排程演算法則。這些演算法則詳述於〈節 5.1, “HAProxy 排程演算法則”〉中。

請透過 **/etc/haproxy/haproxy.cfg** 檔案配置 HAProxy。

使用 HAProxy 的 Load Balancer 配置包含了五個部分：

- 節 5.2, “***global* 設定”**
- 〈代理〉一節，包含四個小節：
  - 節 5.3, “***default* 設定”** 的設定
  - 節 5.4, “***frontend* 設定”** 的設定
  - 節 5.5, “***backend* 設定”** 的設定

5.1. HAProxy 排程演算法則

用來進行負載平衡的 HAProxy 排程演算法則可透過 **/etc/haproxy/haproxy.cfg** 配置檔案的 ***backend*** 部分中的 ***balance*** 參數來進行編輯。請注意，HAProxy 支援含有多重後端的配置，並且各個後端皆可透過一個排程演算法則來進行配置。

***循環法*（*roundrobin*）**

將請求依序發給集區中的伺服器。使用此演算法則時，所有真實伺服器不管能力為何，都會被視為平等。此排程模式類似循環 DNS（round-robin DNS）法，但更為細緻，因為這模式是以網路連線為基礎，而非以主機為基礎。 Load Balancer 的循環配置資源排程也不會因為快取了 DNS 的查詢項目而導致不平衡。然而，在 HAProxy 中，因為伺服器權重的配置能輕易透過此排程器來完成，因此各個後端的啓用中的伺服器數量會被限制為 4095。

***靜態循環法*（*static-rr*）**

如*循環法（Round-Robin）*一般地在一組伺服器之間循序性地分散各項請求，不過不允許動態式配置伺服器的權重。然而，基於伺服器權重的靜態本質，後端中不會有伺服器啓用數量上的限制。

***最少連線排程法*（*leastconn*）**

將請求分散至連線數較少的真實伺服器。當處於一個動態式且包含各種 session 或是連線長度的環境中，對管理員來說這個排程器可能會較適合其環境。這也適用於一個包含了一組擁有不同生產力的伺服器的環境中，因為透過此排程器管理員可輕易調整權重。

***來源*（*source*）**

透過雜湊請求的來源 IP 位址，並除以所有運作中的伺服器的權重，以判斷哪個伺服器將會收到請求。在所有伺服器皆在運作的情況下，來源 IP 的請求將會持續性地透過相同的真實伺服器處理。若運作中的伺服器之數量或權重改變的話，session 可能會被移至另一個伺服器上，因為雜湊/權重的結果已改變。

***URI*（*uri*）**

透過雜湊整個 URI（或是 URI 的可配置部分）並除以所有運作中的伺服器的權重，以判斷哪個伺服器將會收到請求。在所有啓用中的伺服器皆在運作的情況下，來源 IP 的請求將會持續性地由相同的

真實伺服器處理。此排程器亦可藉由配置 URI 起始的字元長度以運算雜湊結果，或藉由配置 URI 中的目錄的深度（以 URI 中的正斜線表明）來運算雜湊結果。

URL 參數（url_param）

藉由在來源 URL 請求中找尋特定參數字串，並執行一項雜湊計算除以所有運作中的伺服器的權重，以將請求分散至伺服器上。若 URL 中缺少了參數，排程器便會預設使用循環排程法。您可基於 POST 參數使用修飾符號，並根據管理員指定特定參數運算雜湊結果前，所需等待的最大八位組（octet）來決定等待限制。

表頭名稱（hdr）

透過選取各個來源 HTTP 請求中的特定表頭名稱來將請求發佈至伺服器上，並進行一項雜湊計算然後除以所有運作中伺服器的權重。若表頭不存在，排程器便會預設使用循環法（Round-robin）排程。

RDP Cookie（rdp-cookie）

透過查詢每項 TCP 請求的 RDP cookie 來將請求發佈至伺服器上，並執行一項雜湊計算然後除以所有運作中伺服器的權重。若表頭不存在的話，排程器便會預設使用循環法（Round-robin）排程。這方式適用於需保有永續性的情況下，因為它會保留 session 的完整性。

5.2. *global* 設定

global（全域）設定能配置用於執行 HAProxy 的所有伺服器。典型的 ***global*** 一節看起來像是：

```
global
    log 127.0.0.1 local2
    maxconn 4000
    user haproxy
    group haproxy
    daemon
```

在以上配置中，管理者已配置服務來 ***log***（記錄）所有條目至本地的 **syslog** 伺服器。預設上，這可以是 **/var/log/syslog** 或使用者指定的地方。

maxconn 參數會指定同步連上服務的最大連線數量。預設上，最大值是 2000。

user 與 ***group*** 參數能用來指定 **haproxy** 程序所屬的使用者名稱與群組名稱。

最後，***daemon*** 參數能讓 **haproxy** 在背景中執行。

5.3. *default* 設定

default（預設）設定能用來配置套用到 frontend（前端）、backend（後端）、listen（聆聽）等配置項目的所有代理小節。典型的 ***default*** 一節看起來像是：

注意

任何配置於 ***proxy*** 子節（***frontend***、***backend***、或 ***listen***）的參數，都會蓋過 ***default*** 中的參數值。

```
defaults
    mode                    http
    log                     global
    option                  httplog
    option                  dontlognull
    retries                 3
    timeout http-request    10s
    timeout queue           1m
    timeout connect         10s
    timeout client          1m
    timeout server          1m
```

mode 指定了 HAProxy instance 所使用的通訊協定。使用 ***http*** 模式會將來源請求連接至根基於 HTTP 的真實伺服器，試用於網頁伺服器的負載平衡。對於其它應用程式來說，請使用 ***tcp*** 模式。

log 會指定日誌條目將寫入的日誌位址與 **syslog** 的設施。***global*** 值參照了 ***global*** 一節中的 ***log*** 參數中所指定的 HAProxy instance。

option httplog 能紀錄 HTTP session 的多種值，包括 HTTP 請求、session 狀態、連線數、來源位址、以及連線計數器等等。

option dontlognull 停用了對 null 連線之記錄，表示 HAProxy 不會記錄未傳遞任何資料的連線。這不建議用在網際網路上的環境中（例如網頁應用程式），因為 null 連線可能表示惡意活動，例如開放式掃描連接埠弱點。

retries 指定了在無法第一次就連上時，真實伺服器會重試連線的次數。

多種 ***timeout*** 值會指定某個給定請求的非活動時間值（單位為秒或微秒）。***http-request 10s*** 會給來自用戶端的 HTTP 請求十秒的時間，已完成連線。***queue 10m*** 會設定連線被放棄、用戶端收到 503 或「Service Unavailable」錯誤之前的時間。***connect 10s*** 會指定等待成功連線至伺服器的秒數。***client 1m*** 會指定用戶端能保持非活動（既不接受也不傳送資料）的時間。***server 10m*** 會指定賦予伺服器在逾時值發生前，接受或傳送資料的時間，單位為微秒。

5.4. *frontend* 設定

frontend 設定會配置伺服器聆聽來自用戶端連線請求的聆聽 socket。***frontend*** 的典型 HAProxy 配置看起來像：

```
frontend  main
  bind 192.168.0.10:80
```

名為 ***main*** 的 ***frontend*** 透過 ***bind*** 參數將 IP 位址設為 192.168.0.10，並聆聽連接埠號 80。連上後，***use backend*** 會指定所有來自 session 的連線連往 ***app*** 後端。

5.5. *backend* 設定

backend 這設定會指定真實伺服器的 IP 位址，以及負載平衡程式的排程演算法。以下範例顯示了典型的 ***backend*** 一節：

```
backend app
   balance     roundrobin
   server  app1 192.168.1.1:80 check
   server  app2 192.168.1.2:80 check
   server  app3 192.168.1.3:80 check inter 2s rise 4 fall 3
   server  app4 192.168.1.4:80 backup
```

後端伺服器名為 ***app***。***balance*** 會指定負載平衡程式的排程演算法，此範例中使用的乃循環法（***roundrobin***），但可以是任何 HAProxy 所支援的排程程式。欲知如何在 HAProxy 中配置排程程式，請參閱〈節 5.1, “HAProxy 排程演算法則”〉。

server 行指定了後端可以使用的伺服器。***app1*** 到 ***app4*** 是內部指定到每台真實伺服器的名稱。位址是指定的 IP 位址。IP 位址冒號後面的是在伺服器上發生連線的連接埠號。***check*** 選項標誌著定期「健康檢查」的伺服器，以確保伺服器的可用性，得以收送資料並服務 session 的請求。伺服器 app3 也配置了健康檢查的間隔，時間為 2 秒（***inter 2s***）、app3 用來決定伺服器是否健康的檢查次數（***rise 4***）、以及伺服器直至認定為失敗的重試次數（***fall 3***）。

5.6. 啓動 haproxy

要啓動 HAProxy 服務，請執行以下指令：

```
systemctl start haproxy.service
```

要讓 HAProxy 服務在開機後自動執行，請執行以下指令：

```
systemctl enable haproxy.service
```

附錄 A. 修訂記錄

修訂 0.2-6.2 **Fri Nov 13 2015** **Terry Chuang**
翻譯、校閱完成

修訂 0.2-6.1 **Tue Nov 10 2015** **Terry Chuang**
讓翻譯檔案與 XML 來源 0.2-6 同步

修訂 0.2-6 **Mon Feb 16 2015** **Steven Levine**
7.1 GA 發行版本

修訂 0.2-5 **Thu Dec 11 2014** **Steven Levine**
7.1 Beta 發行版本

修訂 0.1-15 **Fri Dec 05 2014** **Steven Levine**
更新以在 Red Hat Enterprise Linux splash 頁面上實作新的排序順序。

修訂 0.1-12 **Tue Jun 03 2014** **John Ha**
7.0 GA 發行版本

修訂 0.1-6 **Mon Jun 13 2013** **John Ha**
為 Red Hat Enterprise Linux 7 beta 而建

修訂 0.1-2 **Mon Jun 13 2013** **John Ha**
初始反應的 Alpha 草稿

修訂 0.1-1 **Wed Jan 16 2013** **John Ha**
由本文件的 Red Hat Enterprise Linux 6 版本分支

索引

符號