Red Hat Enterprise Linux 5

Virtual Server Administration

Linux Virtual Server (LVS) for Red Hat Enterprise Linux

Red Hat Enterprise Linux 5 Virtual Server Administration
Linux Virtual Server (LVS) for Red Hat Enterprise Linux
Edition 5

1801 Varsity Drive
Raleigh, NC 27606-2072 USA
Phone: +1 919 754 3700
Phone: 888 733 4281
Fax: +1 919 754 3701

Building a Linux Virtual Server (LVS) system offers highly-available and scalable solution for production services using specialized routing and load-balancing techniques configured through the **PIRANHA**. This book discusses the configuration of high-performance systems and services with Red Hat Enterprise Linux and LVS for Red Hat Enterprise Linux 5.

Introduction

This document provides information about installing, configuring, and managing Red Hat Virtual Linux Server (LVS) components. LVS provides load balancing through specialized routing techniques that dispatch traffic to a pool of servers. This document does not include information about installing, configuring, and managing Red Hat Cluster software. Information about that is in a separate document.

The audience of this document should have advanced working knowledge of Red Hat Enterprise Linux and understand the concepts of clusters, storage, and server computing.

This document is organized as follows:

- *Chapter 1, Linux Virtual Server Overview*
- *Chapter 2, Initial LVS Configuration*
- *Chapter 3, Setting Up LVS*
- *Chapter 4, Configuring the LVS Routers with **Piranha Configuration Tool***
- *Appendix A, Using LVS with Red Hat Cluster*

For more information about Red Hat Enterprise Linux 5, refer to the following resources:

- *Red Hat Enterprise Linux Installation Guide* — Provides information regarding installation of Red Hat Enterprise Linux 5.
- *Red Hat Enterprise Linux Deployment Guide* — Provides information regarding the deployment, configuration and administration of Red Hat Enterprise Linux 5.

For more information about Red Hat Cluster Suite for Red Hat Enterprise Linux 5, refer to the following resources:

- *Red Hat Cluster Suite Overview* — Provides a high level overview of the Red Hat Cluster Suite.
- *Configuring and Managing a Red Hat Cluster* — Provides information about installing, configuring and managing Red Hat Cluster components.
- *Logical Volume Manager Administration* — Provides a description of the Logical Volume Manager (LVM), including information on running LVM in a clustered environment.
- *Global File System: Configuration and Administration* — Provides information about installing, configuring, and maintaining Red Hat GFS (Red Hat Global File System).
- *Global File System 2: Configuration and Administration* — Provides information about installing, configuring, and maintaining Red Hat GFS2 (Red Hat Global File System 2).
- *Using Device-Mapper Multipath* — Provides information about using the Device-Mapper Multipath feature of Red Hat Enterprise Linux 5.
- *Using GNBD with Global File System* — Provides an overview on using Global Network Block Device (GNBD) with Red Hat GFS.
- *Red Hat Cluster Suite Release Notes* — Provides information about the current release of Red Hat Cluster Suite.

Red Hat Cluster Suite documentation and other Red Hat documents are available in HTML, PDF, and RPM versions on the Red Hat Enterprise Linux Documentation CD and online at *http://www.redhat.com/docs/*.

1. Document Conventions

This manual uses several conventions to highlight certain words and phrases and draw attention to specific pieces of information.

In PDF and paper editions, this manual uses typefaces drawn from the *Liberation Fonts*[1] set. The Liberation Fonts set is also used in HTML editions if the set is installed on your system. If not, alternative but equivalent typefaces are displayed. Note: Red Hat Enterprise Linux 5 and later includes the Liberation Fonts set by default.

1.1. Typographic Conventions

Four typographic conventions are used to call attention to specific words and phrases. These conventions, and the circumstances they apply to, are as follows.

`Mono-spaced Bold`

Used to highlight system input, including shell commands, file names and paths. Also used to highlight keycaps and key combinations. For example:

> To see the contents of the file **`my_next_bestselling_novel`** in your current working directory, enter the **`cat my_next_bestselling_novel`** command at the shell prompt and press **`Enter`** to execute the command.

The above includes a file name, a shell command and a keycap, all presented in mono-spaced bold and all distinguishable thanks to context.

Key combinations can be distinguished from keycaps by the hyphen connecting each part of a key combination. For example:

> Press **`Enter`** to execute the command.
>
> Press **`Ctrl`**+**`Alt`**+**`F2`** to switch to the first virtual terminal. Press **`Ctrl`**+**`Alt`**+**`F1`** to return to your X-Windows session.

The first paragraph highlights the particular keycap to press. The second highlights two key combinations (each a set of three keycaps with each set pressed simultaneously).

If source code is discussed, class names, methods, functions, variable names and returned values mentioned within a paragraph will be presented as above, in **`mono-spaced bold`**. For example:

> File-related classes include **`filesystem`** for file systems, **`file`** for files, and **`dir`** for directories. Each class has its own associated set of permissions.

Proportional Bold

This denotes words or phrases encountered on a system, including application names; dialog box text; labeled buttons; check-box and radio button labels; menu titles and sub-menu titles. For example:

> Choose **System** → **Preferences** → **Mouse** from the main menu bar to launch **Mouse Preferences**. In the **Buttons** tab, click the **Left-handed mouse** check box and click

[1] https://fedorahosted.org/liberation-fonts/

> **Close** to switch the primary mouse button from the left to the right (making the mouse suitable for use in the left hand).
>
> To insert a special character into a **gedit** file, choose **Applications** → **Accessories** → **Character Map** from the main menu bar. Next, choose **Search** → **Find…** from the **Character Map** menu bar, type the name of the character in the **Search** field and click **Next**. The character you sought will be highlighted in the **Character Table**. Double-click this highlighted character to place it in the **Text to copy** field and then click the **Copy** button. Now switch back to your document and choose **Edit** → **Paste** from the **gedit** menu bar.

The above text includes application names; system-wide menu names and items; application-specific menu names; and buttons and text found within a GUI interface, all presented in proportional bold and all distinguishable by context.

`Mono-spaced Bold Italic` or ***`Proportional Bold Italic`***

Whether mono-spaced bold or proportional bold, the addition of italics indicates replaceable or variable text. Italics denotes text you do not input literally or displayed text that changes depending on circumstance. For example:

> To connect to a remote machine using ssh, type **`ssh`** ***`username@domain.name`*** at a shell prompt. If the remote machine is **`example.com`** and your username on that machine is john, type **`ssh john@example.com`**.
>
> The **`mount -o remount`** ***`file-system`*** command remounts the named file system. For example, to remount the **`/home`** file system, the command is **`mount -o remount /home`**.
>
> To see the version of a currently installed package, use the **`rpm -q`** ***`package`*** command. It will return a result as follows: ***`package-version-release`***.

Note the words in bold italics above — username, domain.name, file-system, package, version and release. Each word is a placeholder, either for text you enter when issuing a command or for text displayed by the system.

Aside from standard usage for presenting the title of a work, italics denotes the first use of a new and important term. For example:

> Publican is a *DocBook* publishing system.

1.2. Pull-quote Conventions

Terminal output and source code listings are set off visually from the surrounding text.

Output sent to a terminal is set in **`mono-spaced roman`** and presented thus:

```
books        Desktop   documentation  drafts  mss    photos   stuff  svn
books_tests  Desktop1  downloads      images  notes  scripts  svgs
```

Source-code listings are also set in **`mono-spaced roman`** but add syntax highlighting as follows:

```
package org.jboss.book.jca.ex1;

import javax.naming.InitialContext;

public class ExClient
```

```
{
   public static void main(String args[])
       throws Exception
   {
      InitialContext iniCtx = new InitialContext();
      Object         ref    = iniCtx.lookup("EchoBean");
      EchoHome       home   = (EchoHome) ref;
      Echo           echo   = home.create();

      System.out.println("Created Echo");

      System.out.println("Echo.echo('Hello') = " + echo.echo("Hello"));
   }
}
```

1.3. Notes and Warnings

Finally, we use three visual styles to draw attention to information that might otherwise be overlooked.

Note

Notes are tips, shortcuts or alternative approaches to the task at hand. Ignoring a note should have no negative consequences, but you might miss out on a trick that makes your life easier.

Important

Important boxes detail things that are easily missed: configuration changes that only apply to the current session, or services that need restarting before an update will apply. Ignoring a box labeled 'Important' will not cause data loss but may cause irritation and frustration.

Warning

Warnings should not be ignored. Ignoring warnings will most likely cause data loss.

2. Feedback

If you spot a typo, or if you have thought of a way to make this manual better, we would love to hear from you. Please submit a report in Bugzilla (*http://bugzilla.redhat.com/bugzilla/*) against the component **Documentation-cluster**.

Be sure to mention the manual's identifier:

```
Virtual_Server_Administration(EN)-5 (2010-02-08T16:55)
```

By mentioning this manual's identifier, we know exactly which version of the guide you have.

If you have a suggestion for improving the documentation, try to be as specific as possible. If you have found an error, please include the section number and some of the surrounding text so we can find it easily.

Linux Virtual Server Overview

Linux Virtual Server (LVS) is a set of integrated software components for balancing the IP load across a set of real servers. LVS runs on a pair of equally configured computers: one that is an *active LVS router* and one that is a *backup LVS router*. The active LVS router serves two roles:

- To balance the load across the real servers.
- To check the integrity of the services on each real server.

The backup LVS router monitors the active LVS router and takes over from it in case the active LVS router fails.

This chapter provides an overview of LVS components and functions, and consists of the following sections:

1.1. A Basic LVS Configuration

Figure 1.1, "A Basic LVS Configuration" shows a simple LVS configuration consisting of two layers. On the first layer are two LVS routers — one active and one backup. Each of the LVS routers has two network interfaces, one interface on the Internet and one on the private network, enabling them to regulate traffic between the two networks. For this example the active router is using *Network Address Translation* or *NAT* to direct traffic from the Internet to a variable number of real servers on the second layer, which in turn provide the necessary services. Therefore, the real servers in this example are connected to a dedicated private network segment and pass all public traffic back and forth through the active LVS router. To the outside world, the servers appears as one entity.

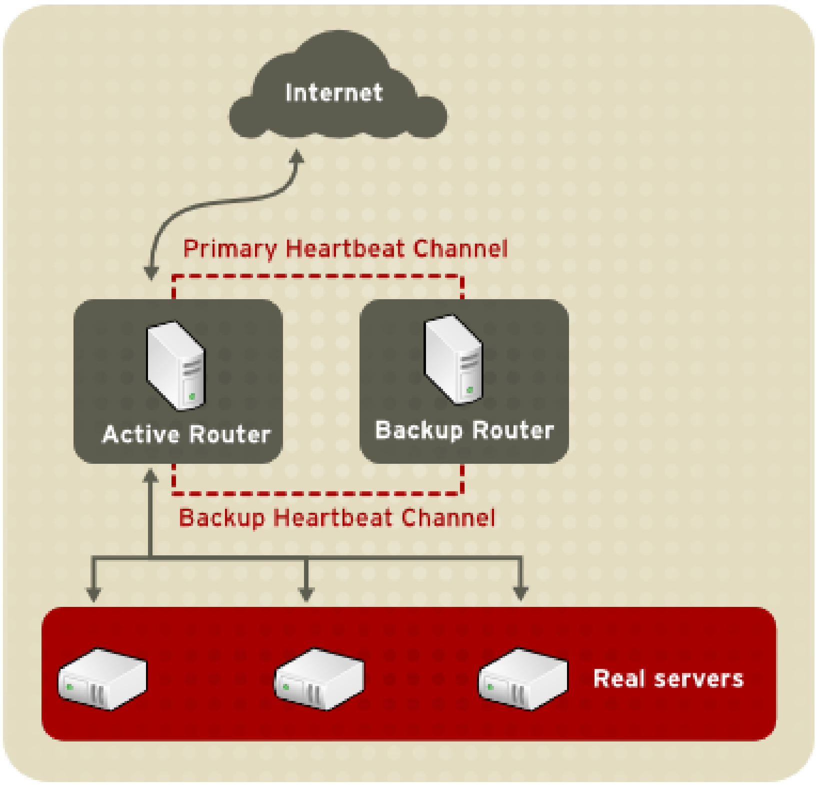


Figure 1.1. A Basic LVS Configuration

Service requests arriving at the LVS routers are addressed to a *virtual IP* address, or *VIP*. This is a publicly-routable address the administrator of the site associates with a fully-qualified domain name, such as www.example.com, and is assigned to one or more *virtual servers*. A virtual server is a service configured to listen on a specific virtual IP. Refer to *Section 4.6, "**VIRTUAL SERVERS**"* for more information on configuring a virtual server using the **Piranha Configuration Tool**. A VIP address migrates from one LVS router to the other during a failover, thus maintaining a presence at that IP address (also known as *floating IP addresses*).

VIP addresses may be aliased to the same device which connects the LVS router to the Internet. For instance, if eth0 is connected to the Internet, than multiple virtual servers can be aliased to **`eth0:1`**. Alternatively, each virtual server can be associated with a separate device per service. For example, HTTP traffic can be handled on **`eth0:1`**, and FTP traffic can be handled on **`eth0:2`**.

Only one LVS router is active at a time. The role of the active router is to redirect service requests from virtual IP addresses to the real servers. The redirection is based on one of eight supported load-balancing algorithms described further in *Section 1.3, "LVS Scheduling Overview"*.

The active router also dynamically monitors the overall health of the specific services on the real servers through simple *send/expect scripts*. To aid in detecting the health of services that require dynamic data, such as HTTPS or SSL, the administrator can also call external executables. If a service on a real server malfunctions, the active router stops sending jobs to that server until it returns to normal operation.

The backup router performs the role of a standby system. Periodically, the LVS routers exchange heartbeat messages through the primary external public interface and, in a failover situation, the private interface. Should the backup node fail to receive a heartbeat message within an expected interval, it initiates a failover and assumes the role of the active router. During failover, the backup router takes over the VIP addresses serviced by the failed router using a technique known as *ARP spoofing* — where the backup LVS router announces itself as the destination for IP packets addressed to the failed node. When the failed node returns to active service, the backup node assumes its hot-backup role again.

The simple, two-layered configuration used in *Figure 1.1, “A Basic LVS Configuration”* is best for serving data which does not change very frequently — such as static webpages — because the individual real servers do not automatically sync data between each node.

1.1.1. Data Replication and Data Sharing Between Real Servers

Since there is no built-in component in LVS to share the same data between the real servers, the administrator has two basic options:

- Synchronize the data across the real server pool
- Add a third layer to the topology for shared data access

The first option is preferred for servers that do not allow large numbers of users to upload or change data on the real servers. If the configuration allows large numbers of users to modify data, such as an e-commerce website, adding a third layer is preferable.

1.1.1.1. Configuring Real Servers to Synchronize Data

There are many ways an administrator can choose to synchronize data across the pool of real servers. For instance, shell scripts can be employed so that if a Web engineer updates a page, the page is posted to all of the servers simultaneously. Also, the system administrator can use programs such as **`rsync`** to replicate changed data across all nodes at a set interval.

However, this type of data synchronization does not optimally function if the configuration is overloaded with users constantly uploading files or issuing database transactions. For a configuration with a high load, a *three-tier topology* is the ideal solution.

1.2. A Three-Tier LVS Configuration

Figure 1.2, “A Three-Tier LVS Configuration” shows a typical three-tier LVS topology. In this example, the active LVS router routes the requests from the Internet to the pool of real servers. Each of the real servers then accesses a shared data source over the network.

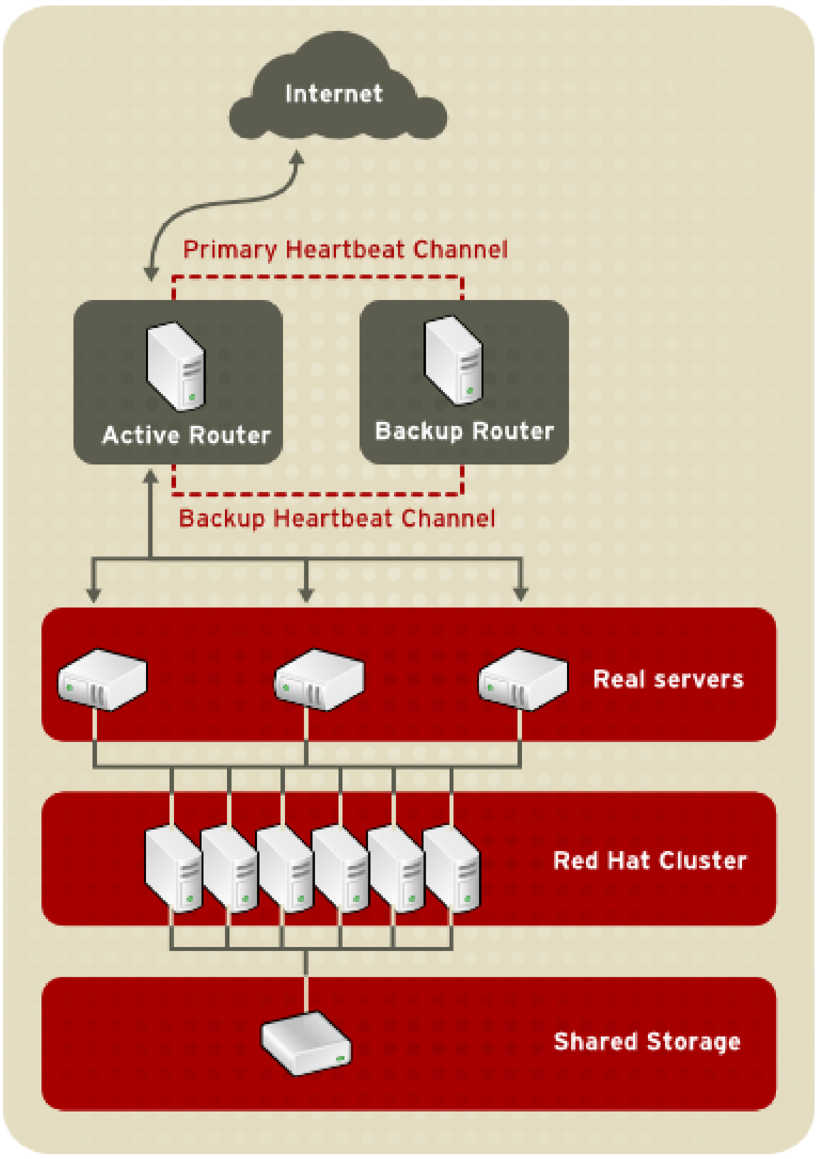


Figure 1.2. A Three-Tier LVS Configuration

This configuration is ideal for busy FTP servers, where accessible data is stored on a central, highly available server and accessed by each real server via an exported NFS directory or Samba share. This topology is also recommended for websites that access a central, highly available database for transactions. Additionally, using an active-active configuration with Red Hat Cluster Manager, administrators can configure one high-availability cluster to serve both of these roles simultaneously.

The third tier in the above example does not have to use Red Hat Cluster Manager, but failing to use a highly available solution would introduce a critical single point of failure.

1.3. LVS Scheduling Overview

One of the advantages of using LVS is its ability to perform flexible, IP-level load balancing on the real server pool. This flexibility is due to the variety of scheduling algorithms an administrator can choose from when configuring LVS. LVS load balancing is superior to less flexible methods, such as *Round-Robin DNS* where the hierarchical nature of DNS and the caching by client machines can lead to load imbalances. Additionally, the low-level filtering employed by the LVS router has advantages over application-level request forwarding because balancing loads at the network packet level causes minimal computational overhead and allows for greater scalability.

Using scheduling, the active router can take into account the real servers' activity and, optionally, an administrator-assigned *weight* factor when routing service requests. Using assigned weights gives arbitrary priorities to individual machines. Using this form of scheduling, it is possible to create a group of real servers using a variety of hardware and software combinations and the active router can evenly load each real server.

The scheduling mechanism for LVS is provided by a collection of kernel patches called *IP Virtual Server* or *IPVS* modules. These modules enable *layer 4* (*L4*) transport layer switching, which is designed to work well with multiple servers on a single IP address.

To track and route packets to the real servers efficiently, IPVS builds an *IPVS table* in the kernel. This table is used by the active LVS router to redirect requests from a virtual server address to and returning from real servers in the pool. The IPVS table is constantly updated by a utility called *ipvsadm* — adding and removing cluster members depending on their availability.

1.3.1. Scheduling Algorithms

The structure that the IPVS table takes depends on the scheduling algorithm that the administrator chooses for any given virtual server. To allow for maximum flexibility in the types of services you can cluster and how these services are scheduled, Red Hat Enterprise Linux provides the following scheduling algorithms listed below. For instructions on how to assign scheduling algorithms refer to *Section 4.6.1, “The **VIRTUAL SERVER** Subsection”*.

Round-Robin Scheduling

Distributes each request sequentially around the pool of real servers. Using this algorithm, all the real servers are treated as equals without regard to capacity or load. This scheduling model resembles round-robin DNS but is more granular due to the fact that it is network-connection based and not host-based. LVS round-robin scheduling also does not suffer the imbalances caused by cached DNS queries.

Weighted Round-Robin Scheduling

Distributes each request sequentially around the pool of real servers but gives more jobs to servers with greater capacity. Capacity is indicated by a user-assigned weight factor, which is then adjusted upward or downward by dynamic load information. Refer to *Section 1.3.2, “Server Weight and Scheduling”* for more on weighting real servers.

Weighted round-robin scheduling is a preferred choice if there are significant differences in the capacity of real servers in the pool. However, if the request load varies dramatically, the more heavily weighted server may answer more than its share of requests.

Least-Connection

Distributes more requests to real servers with fewer active connections. Because it keeps track of live connections to the real servers through the IPVS table, least-connection is a type of dynamic scheduling algorithm, making it a better choice if there is a high degree of variation in the request load. It is best suited for a real server pool where each member node has roughly the same capacity. If a group of servers have different capabilities, weighted least-connection scheduling is a better choice.

Weighted Least-Connections (default)

Distributes more requests to servers with fewer active connections relative to their capacities. Capacity is indicated by a user-assigned weight, which is then adjusted upward or downward by dynamic load information. The addition of weighting makes this algorithm ideal when the real server pool contains hardware of varying capacity. Refer to *Section 1.3.2, “Server Weight and Scheduling”* for more on weighting real servers.

Locality-Based Least-Connection Scheduling
: Distributes more requests to servers with fewer active connections relative to their destination IPs. This algorithm is designed for use in a proxy-cache server cluster. It routes the packets for an IP address to the server for that address unless that server is above its capacity and has a server in its half load, in which case it assigns the IP address to the least loaded real server.

Locality-Based Least-Connection Scheduling with Replication Scheduling
: Distributes more requests to servers with fewer active connections relative to their destination IPs. This algorithm is also designed for use in a proxy-cache server cluster. It differs from Locality-Based Least-Connection Scheduling by mapping the target IP address to a subset of real server nodes. Requests are then routed to the server in this subset with the lowest number of connections. If all the nodes for the destination IP are above capacity, it replicates a new server for that destination IP address by adding the real server with the least connections from the overall pool of real servers to the subset of real servers for that destination IP. The most loaded node is then dropped from the real server subset to prevent over-replication.

Destination Hash Scheduling
: Distributes requests to the pool of real servers by looking up the destination IP in a static hash table. This algorithm is designed for use in a proxy-cache server cluster.

Source Hash Scheduling
: Distributes requests to the pool of real servers by looking up the source IP in a static hash table. This algorithm is designed for LVS routers with multiple firewalls.

1.3.2. Server Weight and Scheduling

The administrator of LVS can assign a *weight* to each node in the real server pool. This weight is an integer value which is factored into any *weight-aware* scheduling algorithms (such as weighted least-connections) and helps the LVS router more evenly load hardware with different capabilities.

Weights work as a ratio relative to one another. For instance, if one real server has a weight of 1 and the other server has a weight of 5, then the server with a weight of 5 gets 5 connections for every 1 connection the other server gets. The default value for a real server weight is 1.

Although adding weight to varying hardware configurations in a real server pool can help load-balance the cluster more efficiently, it can cause temporary imbalances when a real server is introduced to the real server pool and the virtual server is scheduled using weighted least-connections. For example, suppose there are three servers in the real server pool. Servers A and B are weighted at 1 and the third, server C, is weighted at 2. If server C goes down for any reason, servers A and B evenly distributes the abandoned load. However, once server C comes back online, the LVS router sees it has zero connections and floods the server with all incoming requests until it is on par with servers A and B.

To prevent this phenomenon, administrators can make the virtual server a *quiesce* server — anytime a new real server node comes online, the least-connections table is reset to zero and the LVS router routes requests as if all the real servers were newly added to the cluster.

1.4. Routing Methods

Red Hat Enterprise Linux uses *Network Address Translation* or *NAT routing* for LVS, which allows the administrator tremendous flexibility when utilizing available hardware and integrating the LVS into an existing network.

1.4.1. NAT Routing

Figure 1.3, “LVS Implemented with NAT Routing”, illustrates LVS utilizing NAT routing to move requests between the Internet and a private network.

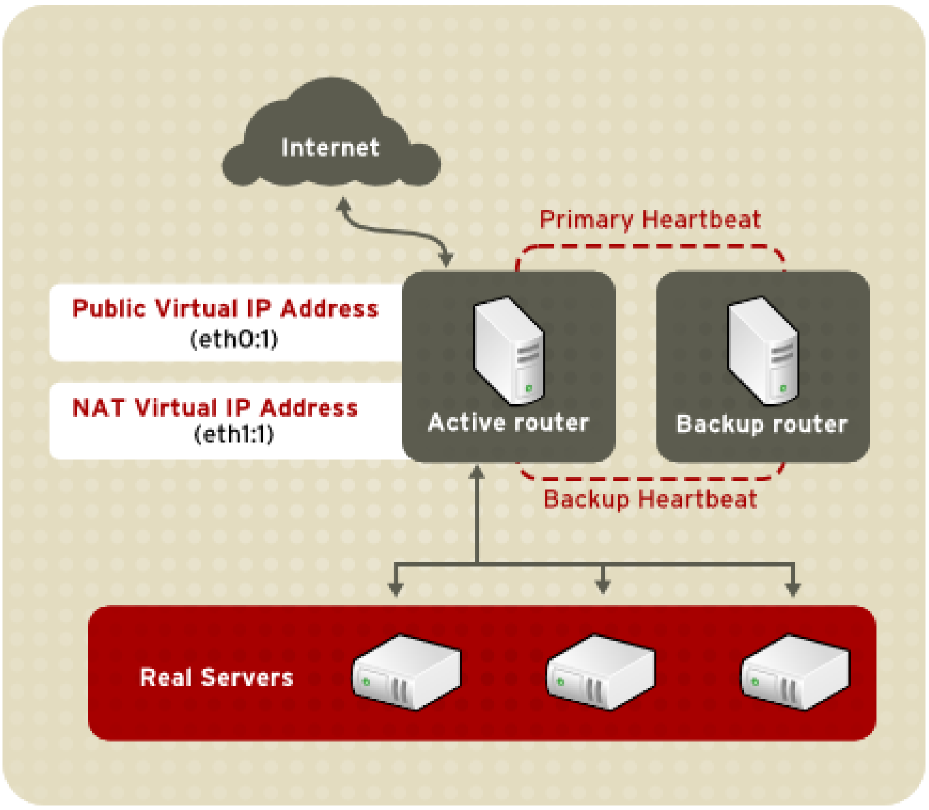


Figure 1.3. LVS Implemented with NAT Routing

In the example, there are two NICs in the active LVS router. The NIC for the Internet has a *real IP address* on eth0 and has a floating IP address aliased to eth0:1. The NIC for the private network interface has a real IP address on eth1 and has a floating IP address aliased to eth1:1. In the event of failover, the virtual interface facing the Internet and the private facing virtual interface are taken-over by the backup LVS router simultaneously. All of the real servers located on the private network use the floating IP for the NAT router as their default route to communicate with the active LVS router so that their abilities to respond to requests from the Internet is not impaired.

In this example, the LVS router's public LVS floating IP address and private NAT floating IP address are aliased to two physical NICs. While it is possible to associate each floating IP address to its own physical device on the LVS router nodes, having more than two NICs is not a requirement.

Using this topology, the active LVS router receives the request and routes it to the appropriate server. The real server then processes the request and returns the packets to the LVS router which uses network address translation to replace the address of the real server in the packets with the LVS routers public VIP address. This process is called *IP masquerading* because the actual IP addresses of the real servers is hidden from the requesting clients.

Using this NAT routing, the real servers may be any kind of machine running various operating systems. The main disadvantage is that the LVS router may become a bottleneck in large cluster deployments because it must process outgoing as well as incoming requests.

1.4.2. Direct Routing

Building an LVS setup that uses direct routing provides increased performance benefits compared to other LVS networking topologies. Direct routing allows the real servers to process and route packets directly to a requesting user rather than passing all outgoing packets through the LVS router. Direct routing reduces the possibility of network performance issues by relegating the job of the LVS router to processing incoming packets only.

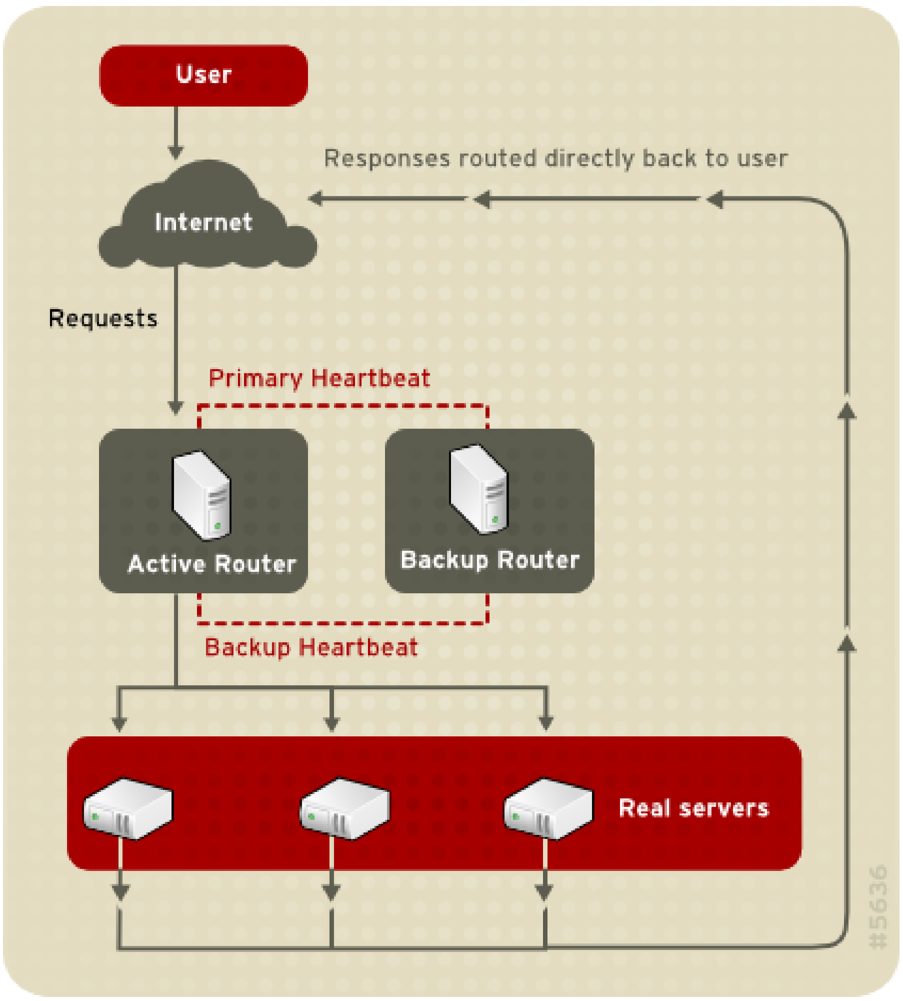


Figure 1.4. LVS Implemented with Direct Routing

In the typical direct routing LVS setup, the LVS router receives incoming server requests through the virtual IP (VIP) and uses a scheduling algorithm to route the request to the real servers. The real server processes the request and sends the response directly to the client, bypassing the LVS routers. This method of routing allows for scalability in that real servers can be added without the added burden on the LVS router to route outgoing packets from the real server to the client, which can become a bottleneck under heavy network load.

1.4.2.1. Direct Routing and the ARP Limitation

While there are many advantages to using direct routing in LVS, there are limitations as well. The most common issue with LVS via direct routing is with *Address Resolution Protocol* (ARP).

In typical situations, a client on the Internet sends a request to an IP address. Network routers typically send requests to their destination by relating IP addresses to a machine's MAC address with ARP. ARP requests are broadcast to all connected machines on a network, and the machine with the correct IP/MAC address combination receives the packet. The IP/MAC associations are stored in an ARP cache, which is cleared periodically (usually every 15 minutes) and refilled with IP/MAC associations.

The issue with ARP requests in a direct routing LVS setup is that because a client request to an IP address must be associated with a MAC address for the request to be handled, the virtual IP address of the LVS system must also be associated to a MAC as well. However, since both the LVS router and the real servers all have the same VIP, the ARP request will be broadcast ed to all the machines associated with the VIP. This can cause several problems, such as the VIP being associated directly to one of the real servers and processing requests directly, bypassing the LVS router completely and defeating the purpose of the LVS setup.

To solve this issue, ensure that the incoming requests are always sent to the LVS router rather than one of the real servers. This can be done by using either the **`arptables_jf`** or the **`iptables`** packet filtering tool for the following reasons:

- The **`arptables_jf`** prevents ARP from associating VIPs with real servers.
- The **`iptables`** method completely sidesteps the ARP problem by not configuring VIPs on real servers in the first place.

For more information on using **`arptables`** or **`iptables`** in a direct routing LVS environment, refer to *Section 3.2.1, “Direct Routing and **`arptables_jf`**”* or *Section 3.2.2, “Direct Routing and **`iptables`**”*.

1.5. Persistence and Firewall Marks

In certain situations, it may be desirable for a client to reconnect repeatedly to the same real server, rather than have an LVS load balancing algorithm send that request to the best available server. Examples of such situations include multi-screen web forms, cookies, SSL, and FTP connections. In these cases, a client may not work properly unless the transactions are being handled by the same server to retain context. LVS provides two different features to handle this: *persistence* and *firewall marks*.

1.5.1. Persistence

When enabled, persistence acts like a timer. When a client connects to a service, LVS remembers the last connection for a specified period of time. If that same client IP address connects again within that period, it is sent to the same server it connected to previously — bypassing the load-balancing mechanisms. When a connection occurs outside the time window, it is handled according to the scheduling rules in place.

Persistence also allows the administrator to specify a subnet mask to apply to the client IP address test as a tool for controlling what addresses have a higher level of persistence, thereby grouping connections to that subnet.

Grouping connections destined for different ports can be important for protocols which use more than one port to communicate, such as FTP. However, persistence is not the most efficient way to deal with the problem of grouping together connections destined for different ports. For these situations, it is best to use *firewall marks*.

1.5.2. Firewall Marks

Firewall marks are an easy and efficient way to a group ports used for a protocol or group of related protocols. For instance, if LVS is deployed to run an e-commerce site, firewall marks can be used to bundle HTTP connections on port 80 and secure, HTTPS connections on port 443. By assigning the same firewall mark to the virtual server for each protocol, state information for the transaction can be preserved because the LVS router forwards all requests to the same real server after a connection is opened.

Because of its efficiency and ease-of-use, administrators of LVS should use firewall marks instead of persistence whenever possible for grouping connections. However, administrators should still add persistence to the virtual servers in conjunction with firewall marks to ensure the clients are reconnected to the same server for an adequate period of time.

1.6. LVS — A Block Diagram

LVS routers use a collection of programs to monitor cluster members and cluster services. *Figure 1.5, “LVS Components”* illustrates how these various programs on both the active and backup LVS routers work together to manage the cluster.

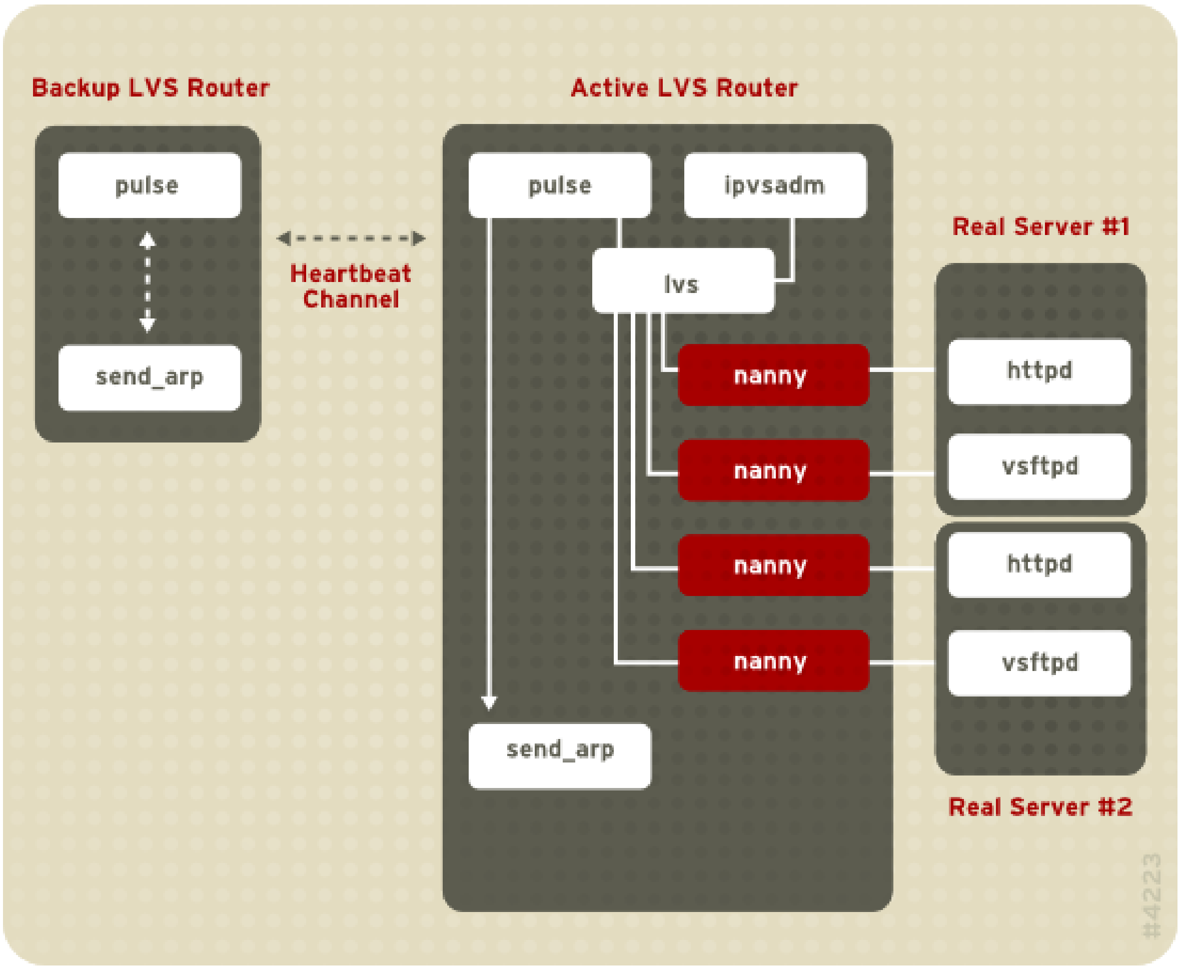


Figure 1.5. LVS Components

The **`pulse`** daemon runs on both the active and passive LVS routers. On the backup router, **`pulse`** sends a *heartbeat* to the public interface of the active router to make sure the active router is still properly functioning. On the active router, **`pulse`** starts the **`lvs`** daemon and responds to *heartbeat* queries from the backup LVS router.

Once started, the **`lvs`** daemon calls the **`ipvsadm`** utility to configure and maintain the IPVS routing table in the kernel and starts a **`nanny`** process for each configured virtual server on each real server. Each **`nanny`** process checks the state of one configured service on one real server, and tells the **`lvs`** daemon if the service on that real server is malfunctioning. If a malfunction is detected, the **`lvs`** daemon instructs **`ipvsadm`** to remove that real server from the IPVS routing table.

If the backup router does not receive a response from the active router, it initiates failover by calling **`send_arp`** to reassign all virtual IP addresses to the NIC hardware addresses (*MAC* address) of the backup node, sends a command to the active router via both the public and private network interfaces to shut down the **`lvs`** daemon on the active router, and starts the **`lvs`** daemon on the backup node to accept requests for the configured virtual servers.

1.6.1. LVS Components

Section 1.6.1.1, “`pulse`” shows a detailed list of each software component in an LVS router.

1.6.1.1. `pulse`

This is the controlling process which starts all other daemons related to LVS routers. At boot time, the daemon is started by the **`/etc/rc.d/init.d/pulse`** script. It then reads the configuration file **`/etc/sysconfig/ha/lvs.cf`**. On the active router, **`pulse`** starts the LVS daemon. On the backup router, **`pulse`** determines the health of the active router by executing a simple heartbeat at a user-configurable interval. If the active router fails to respond after a user-configurable interval, it initiates failover. During failover, **`pulse`** on the backup router instructs the **`pulse`** daemon on the active router to shut down all LVS services, starts the **`send_arp`** program to reassign the floating IP addresses to the backup router's MAC address, and starts the **`lvs`** daemon.

1.6.1.2. `lvs`

The **`lvs`** daemon runs on the active LVS router once called by **`pulse`**. It reads the configuration file **`/etc/sysconfig/ha/lvs.cf`**, calls the **`ipvsadm`** utility to build and maintain the IPVS routing table, and assigns a **nanny** process for each configured LVS service. If **nanny** reports a real server is down, **`lvs`** instructs the **`ipvsadm`** utility to remove the real server from the IPVS routing table.

1.6.1.3. `ipvsadm`

This service updates the IPVS routing table in the kernel. The **`lvs`** daemon sets up and administers LVS by calling **`ipvsadm`** to add, change, or delete entries in the IPVS routing table.

1.6.1.4. nanny

The **nanny** monitoring daemon runs on the active LVS router. Through this daemon, the active router determines the health of each real server and, optionally, monitors its workload. A separate process runs for each service defined on each real server.

1.6.1.5. `/etc/sysconfig/ha/lvs.cf`

This is the LVS configuration file. Directly or indirectly, all daemons get their configuration information from this file.

1.6.1.6. Piranha Configuration Tool

This is the Web-based tool for monitoring, configuring, and administering LVS. This is the default tool to maintain the **`/etc/sysconfig/ha/lvs.cf`** LVS configuration file.

1.6.1.7. `send_arp`

This program sends out ARP broadcasts when the floating IP address changes from one node to another during failover.

Chapter 2, Initial LVS Configuration reviews important post-installation configuration steps you should take before configuring Red Hat Enterprise Linux to be an LVS router.

Initial LVS Configuration

After installing Red Hat Enterprise Linux, you must take some basic steps to set up both the LVS routers and the real servers. This chapter covers these initial steps in detail.

Note

The LVS router node that becomes the active node once LVS is started is also referred to as the *primary node*. When configuring LVS, use the **Piranha Configuration Tool** on the primary node.

2.1. Configuring Services on the LVS Routers

The Red Hat Enterprise Linux installation program installs all of the components needed to set up LVS, but the appropriate services must be activated before configuring LVS. For both LVS routers, set the appropriate services to start at boot time. There are three primary tools available for setting services to activate at boot time under Red Hat Enterprise Linux: the command line program **`chkconfig`**, the ncurses-based program **`ntsysv`**, and the graphical **Services Configuration Tool**. All of these tools require root access.

Note

To attain root access, open a shell prompt and use the **`su -`** command followed by the root password. For example:

```
$ su - root password
```

On the LVS routers, there are three services which need to be set to activate at boot time:

- The **`piranha-gui`** service (primary node only)
- The **`pulse`** service
- The **`sshd`** service

If you are clustering multi-port services or using firewall marks, you must also enable the **`iptables`** service.

It is best to set these services to activate in both runlevel 3 and runlevel 5. To accomplish this using **`chkconfig`**, type the following command for each service:

```
/sbin/chkconfig --level 35 daemon on
```

In the above command, replace *`daemon`* with the name of the service you are activating. To get a list of services on the system as well as what runlevel they are set to activate on, issue the following command:

```
/sbin/chkconfig --list
```

Warning

Turning any of the above services on using **`chkconfig`** does not actually start the daemon. To do this use the **`/sbin/service`** command. See *Section 2.3, “Starting the **Piranha Configuration Tool** Service”* for an example of how to use the **`/sbin/service`** command.

For more information on runlevels and configuring services with **`ntsysv`** and the **Services Configuration Tool**, refer to the chapter titled *"Controlling Access to Services"* in the *Red Hat Enterprise Linux System Administration Guide*.

2.2. Setting a Password for the Piranha Configuration Tool

Before using the **Piranha Configuration Tool** for the first time on the primary LVS router, you must restrict access to it by creating a password. To do this, login as root and issue the following command:

```
/usr/sbin/piranha-passwd
```

After entering this command, create the administrative password when prompted.

Warning

For a password to be more secure, it should not contain proper nouns, commonly used acronyms, or words in a dictionary from any language. Do not leave the password unencrypted anywhere on the system.

If the password is changed during an active **Piranha Configuration Tool** session, the administrator is prompted to provide the new password.

2.3. Starting the Piranha Configuration Tool Service

After you have set the password for the **Piranha Configuration Tool**, start or restart the **`piranha-gui`** service located in **`/etc/rc.d/init.d/piranha-gui`**. To do this, type the following command as root:

```
/sbin/service piranha-gui start
```

or

```
/sbin/service piranha-gui restart
```

Issuing this command starts a private session of the Apache HTTP Server by calling the symbolic link **`/usr/sbin/piranha_gui -> /usr/sbin/httpd`**. For security reasons, the **`piranha-gui`** version of **`httpd`** runs as the piranha user in a separate process. The fact that **`piranha-gui`** leverages the **`httpd`** service means that:

1. The Apache HTTP Server must be installed on the system.

2. Stopping or restarting the Apache HTTP Server via the **`service`** command stops the **`piranha-gui`** service.

Warning

If the command **`/sbin/service httpd stop`** or **`/sbin/service httpd restart`** is issued on an LVS router, you must start the **`piranha-gui`** service by issuing the following command:

```
/sbin/service piranha-gui start
```

The **`piranha-gui`** service is all that is necessary to begin configuring LVS. However, if you are configuring LVS remotely, the **`sshd`** service is also required. You do *not* need to start the **`pulse`** service until configuration using the **Piranha Configuration Tool** is complete. See *Section 4.8, “Starting LVS”* for information on starting the **`pulse`** service.

2.3.1. Configuring the Piranha Configuration Tool Web Server Port

The **Piranha Configuration Tool** runs on port 3636 by default. To change this port number, change the line **`Listen 3636`** in Section 2 of the **`piranha-gui`** Web server configuration file **`/etc/sysconfig/ha/conf/httpd.conf`**.

To use the **Piranha Configuration Tool** you need at minimum a text-only Web browser. If you start a Web browser on the primary LVS router, open the location **`http://`*`localhost`*`:3636`**. You can reach the **Piranha Configuration Tool** from anywhere via Web browser by replacing *`localhost`* with the hostname or IP address of the primary LVS router.

When your browser connects to the **Piranha Configuration Tool**, you must login to access the configuration services. Enter **`piranha`** in the **Username** field and the password set with **`piranha-passwd`** in the **Password** field.

Now that the **Piranha Configuration Tool** is running, you may wish to consider limiting who has access to the tool over the network. The next section reviews ways to accomplish this task.

2.4. Limiting Access To the Piranha Configuration Tool

The **Piranha Configuration Tool** prompts for a valid username and password combination. However, because all of the data passed to the **Piranha Configuration Tool** is in plain text, it is recommended that you restrict access only to trusted networks or to the local machine.

The easiest way to restrict access is to use the Apache HTTP Server's built in access control mechanisms by editing **`/etc/sysconfig/ha/web/secure/.htaccess`**. After altering the file you do not have to restart the **`piranha-gui`** service because the server checks the **`.htaccess`** file each time it accesses the directory.

By default, the access controls for this directory allow anyone to view the contents of the directory. Here is what the default access looks like:

```
Order deny,allow
Allow from all
```

To limit access of the **Piranha Configuration Tool** to only the localhost change the **`.htaccess`** file to allow access from only the loopback device (127.0.0.1). For more information on the loopback device, see the chapter titled *Network Scripts* in the *Red Hat Enterprise Linux Reference Guide*.

```
Order deny,allow
Deny from all
Allow from 127.0.0.1
```

You can also allow specific hosts or subnets as seen in this example:

```
Order deny,allow
Deny from all
Allow from 192.168.1.100
Allow from 172.16.57
```

In this example, only Web browsers from the machine with the IP address of 192.168.1.100 and machines on the 172.16.57/24 network can access the **Piranha Configuration Tool**.

Warning

Editing the **Piranha Configuration Tool** **`.htaccess`** file limits access to the configuration pages in the **`/etc/sysconfig/ha/web/secure/`** directory but not to the login and the help pages in **`/etc/sysconfig/ha/web/`**. To limit access to this directory, create a **`.htaccess`** file in the **`/etc/sysconfig/ha/web/`** directory with **`order`**, **`allow`**, and **`deny`** lines identical to **`/etc/sysconfig/ha/web/secure/.htaccess`**.

2.5. Turning on Packet Forwarding

In order for the LVS router to forward network packets properly to the real servers, each LVS router node must have IP forwarding turned on in the kernel. Log in as root and change the line which reads **`net.ipv4.ip_forward = 0`** in **`/etc/sysctl.conf`** to the following:

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

The changes take effect when you reboot the system.

To check if IP forwarding is turned on, issue the following command as root:

`/sbin/sysctl net.ipv4.ip_forward`

If the above command returns a **`1`**, then IP forwarding is enabled. If it returns a **`0`**, then you can turn it on manually using the following command:

`/sbin/sysctl -w net.ipv4.ip_forward=1`

2.6. Configuring Services on the Real Servers

If the real servers are Red Hat Enterprise Linux systems, set the appropriate server daemons to activate at boot time. These daemons can include **`httpd`** for Web services or **`xinetd`** for FTP or Telnet services.

It may also be useful to access the real servers remotely, so the **`sshd`** daemon should also be installed and running.

Setting Up LVS

LVS consists of two basic groups: the LVS routers and the real servers. To prevent a single point of failure, each groups should contain at least two member systems.

The LVS router group should consist of two identical or very similar systems running Red Hat Enterprise Linux. One will act as the active LVS router while the other stays in hot standby mode, so they need to have as close to the same capabilities as possible.

Before choosing and configuring the hardware for the real server group, determine which of the three LVS topologies to use.

3.1. The NAT LVS Network

The NAT topology allows for great latitude in utilizing existing hardware, but it is limited in its ability to handle large loads because all packets going into and coming out of the pool pass through the LVS router.

Network Layout
: The topology for LVS using NAT routing is the easiest to configure from a network layout perspective because only one access point to the public network is needed. The real servers pass all requests back through the LVS router so they are on their own private network.

Hardware
: The NAT topology is the most flexible in regards to hardware because the real servers do not need to be Linux machines to function correctly. In a NAT topology, each real server only needs one NIC since it will only be responding to the LVS router. The LVS routers, on the other hand, need two NICs each to route traffic between the two networks. Because this topology creates a network bottleneck at the LVS router, gigabit Ethernet NICs can be employed on each LVS router to increase the bandwidth the LVS routers can handle. If gigabit Ethernet is employed on the LVS routers, any switch connecting the real servers to the LVS routers must have at least two gigabit Ethernet ports to handle the load efficiently.

Software
: Because the NAT topology requires the use of **`iptables`** for some configurations, there can be a fair amount of software configuration outside of **Piranha Configuration Tool**. In particular, FTP services and the use of firewall marks requires extra manual configuration of the LVS routers to route requests properly.

3.1.1. Configuring Network Interfaces for LVS with NAT

To set up LVS with NAT, you must first configure the network interfaces for the public network and the private network on the LVS routers. In this example, the LVS routers' public interfaces (**`eth0`**) will be on the 192.168.26/24 network (I know, I know, this is not a routable IP, but let us pretend there is a firewall in front of the LVS router for good measure) and the private interfaces which link to the real servers (**`eth1`**) will be on the 10.11.12/24 network.

So on the active or *primary* LVS router node, the public interface's network script, **`/etc/sysconfig/network-scripts/ifcfg-eth0`**, could look something like this:

```
DEVICE=eth0
```

```
BOOTPROTO=static
ONBOOT=yes
IPADDR=192.168.26.9
NETMASK=255.255.255.0
GATEWAY=192.168.26.254
```

The **`/etc/sysconfig/network-scripts/ifcfg-eth1`** for the private NAT interface on the LVS router could look something like this:

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

In this example, the VIP for the LVS router's public interface will be 192.168.26.10 and the VIP for the NAT or private interface will be 10.11.12.10. So, it is essential that the real servers route requests back to the VIP for the NAT interface.

Important

The sample Ethernet interface configuration settings in this section are for the real IP addresses of an LVS router and *not* the floating IP addresses. To configure the public and private floating IP addresses the administrator should use the **Piranha Configuration Tool**, as shown in *Section 4.4, “**GLOBAL SETTINGS**”* and *Section 4.6.1, “The **VIRTUAL SERVER** Subsection”*.

After configuring the primary LVS router node's network interfaces, configure the backup LVS router's real network interfaces — taking care that none of the IP address conflict with any other IP addresses on the network.

Important

Be sure each interface on the backup node services the same network as the interface on primary node. For instance, if eth0 connects to the public network on the primary node, it must also connect to the public network on the backup node as well.

3.1.2. Routing on the Real Servers

The most important thing to remember when configuring the real servers network interfaces in a NAT topology is to set the gateway for the NAT floating IP address of the LVS router. In this example, that address is 10.11.12.10.

Note

Once the network interfaces are up on the real servers, the machines will be unable to ping or connect in other ways to the public network. This is normal. You will, however, be able to ping the real IP for the LVS router's private interface, in this case 10.11.12.8.

So the real server's **`/etc/sysconfig/network-scripts/ifcfg-eth0`** file could look similar to this:

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

Warning

If a real server has more than one network interface configured with a **`GATEWAY=`** line, the first one to come up will get the gateway. Therefore if both **`eth0`** and **`eth1`** are configured and **`eth1`** is used for LVS, the real servers may not route requests properly.

It is best to turn off extraneous network interfaces by setting **`ONBOOT=no`** in their network scripts within the **`/etc/sysconfig/network-scripts/`** directory or by making sure the gateway is correctly set in the interface which comes up first.

3.1.3. Enabling NAT Routing on the LVS Routers

In a simple NAT LVS configuration where each clustered service uses only one port, like HTTP on port 80, the administrator needs only to enable packet forwarding on the LVS routers for the requests to be properly routed between the outside world and the real servers. See *Section 2.5, “Turning on Packet Forwarding”* for instructions on turning on packet forwarding. However, more configuration is necessary when the clustered services require more than one port to go to the same real server during a user session. For information on creating multi-port services using firewall marks, see *Section 3.4, “Multi-port Services and LVS”*.

Once forwarding is enabled on the LVS routers and the real servers are set up and have the clustered services running, use the **Piranha Configuration Tool** to configure LVS as shown in *Chapter 4, Configuring the LVS Routers with **Piranha Configuration Tool***.

Warning

Do not configure the floating IP for **`eth0:1`** or **`eth1:1`** by manually editing network scripts or using a network configuration tool. Instead, use the **Piranha Configuration Tool** as shown in *Section 4.4, "**GLOBAL SETTINGS**"* and *Section 4.6.1, "The **VIRTUAL SERVER** Subsection"*.

When finished, start the **`pulse`** service as shown in *Section 4.8, "Starting LVS"*. Once **`pulse`** is up and running, the active LVS router will begin routing requests to the pool of real servers.

3.2. LVS via Direct Routing

As mentioned in *Section 1.4.2, "Direct Routing"*, direct routing allows real servers to process and route packets directly to a requesting user rather than passing outgoing packets through the LVS router. Direct routing requires that the real servers be physically connected to a network segment with the LVS router and be able to process and direct outgoing packets as well.

Network Layout

In a direct routing LVS setup, the LVS router needs to receive incoming requests and route them to the proper real server for processing. The real servers then need to *directly* route the response to the client. So, for example, if the client is on the Internet, and sends the packet through the LVS router to a real server, the real server must be able to go directly to the client via the Internet. This can be done by configuring a gateway for the real server to pass packets to the Internet. Each real server in the server pool can have its own separate gateway (and each gateway with its own connection to the Internet), allowing for maximum throughput and scalability. For typical LVS setups, however, the real servers can communicate through one gateway (and therefore one network connection).

Important

It is not recommended to use the LVS router as a gateway for the real servers, as that adds unneeded setup complexity as well as network load on the LVS router, which reintroduces the network bottleneck that exists in NAT routing.

Hardware

The hardware requirements of an LVS system using direct routing is similar to other LVS topologies. While the LVS router needs to be running Red Hat Enterprise Linux to process the incoming requests and perform load-balancing for the real servers, the real servers do not need to be Linux machines to function correctly. The LVS routers need one or two NICs each (depending on if there is a back-up router). You can use two NICs for ease of configuration and to distinctly separate traffic — incoming requests are handled by one NIC and routed packets to real servers on the other.

Since the real servers bypass the LVS router and send outgoing packets directly to a client, a gateway to the Internet is required. For maximum performance and availability, each real server can be connected to its own separate gateway which has its own dedicated connection to the carrier network to which the client is connected (such as the Internet or an intranet).

Software
: There is some configuration outside of **Piranha Configuration Tool** that needs to be done, especially for administrators facing ARP issues when using LVS via direct routing. Refer to *Section 3.2.1, "Direct Routing and **arptables_jf**"* or *Section 3.2.2, "Direct Routing and **iptables**"* for more information.

3.2.1. Direct Routing and `arptables_jf`

In order to configure direct routing using **arptables_jf**, each real server must have their virtual IP address configured, so they can directly route packets. ARP requests for the VIP are ignored entirely by the real servers, and any ARP packets that might otherwise be sent containing the VIPs are mangled to contain the real server's IP instead of the VIPs.

Using the **arptables_jf** method, applications may bind to each individual VIP or port that the real server is servicing. For example, the **arptables_jf** method allows multiple instances of Apache HTTP Server to be running bound explicitly to different VIPs on the system. There are also significant performance advantages to using **arptables_jf** over the **iptables** option.

However, using the **arptables_jf** method, VIPs can not be configured to start on boot using standard Red Hat Enterprise Linux system configuration tools.

To configure each real server to ignore ARP requests for each virtual IP addresses, perform the following steps:

1. Create the ARP table entries for each virtual IP address on each real server (the real_ip is the IP the director uses to communicate with the real server; often this is the IP bound to **eth0**):

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

   This will cause the real servers to ignore all ARP requests for the virtual IP addresses, and change any outgoing ARP responses which might otherwise contain the virtual IP so that they contain the real IP of the server instead. The only node that should respond to ARP requests for any of the VIPs is the current active LVS node.

2. Once this has been completed on each real server, save the ARP table entries by typing the following commands on each real server:

   `service arptables_jf save`

   `chkconfig --level 2345 arptables_jf on`

   The **chkconfig** command will cause the system to reload the arptables configuration on bootup — before the network is started.

3. Configure the virtual IP address on all real servers using **ifconfig** to create an IP alias. For example:

   ```
   # ifconfig eth0:1 192.168.76.24 netmask 255.255.252.0 broadcast 192.168.79.255 up
   ```

   Or using the **iproute2** utility **ip**, for example:

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

As previously noted, the virtual IP addresses can not be configured to start on boot using the Red Hat system configuration tools. One way to work around this issue is to place these commands in **`/etc/rc.d/rc.local`**.

4. Configure Piranha for Direct Routing. Refer to *Chapter 4, Configuring the LVS Routers with* ***Piranha Configuration Tool*** for more information.

3.2.2. Direct Routing and `iptables`

You may also work around the ARP issue using the direct routing method by creating **`iptables`** firewall rules. To configure direct routing using **`iptables`**, you must add rules that create a transparent proxy so that a real server will service packets sent to the VIP address, even though the VIP address does not exist on the system.

The **`iptables`** method is simpler to configure than the **`arptables_jf`** method. This method also circumvents the LVS ARP issue entirely, because the virtual IP address(es) only exist on the active LVS director.

However, there are performance issues using the **`iptables`** method compared to **`arptables_jf`**, as there is overhead in forwarding/masquerading every packet.

You also cannot reuse ports using the **`iptables`** method. For example, it is not possible to run two separate Apache HTTP Server services bound to port 80, because both must bind to *`INADDR_ANY`* instead of the virtual IP addresses.

To configure direct routing using the **`iptables`** method, perform the following steps:

1. On each real server, run the following command for every VIP, port, and protocol (TCP or UDP) combination intended to be serviced for the real server:

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

   This command will cause the real servers to process packets destined for the VIP and port that they are given.

2. Save the configuration on each real server:

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

   The commands above cause the system to reload the **`iptables`** configuration on bootup — before the network is started.

3.3. Putting the Configuration Together

After determining which of the preceding routing methods to use, the hardware should be linked together on the network.

Important

The adapter devices on the LVS routers must be configured to access the same networks. For instance if **`eth0`** connects to public network and **`eth1`** connects to the private network, then these same devices on the backup LVS router must connect to the same networks.

Also the gateway listed in the first interface to come up at boot time is added to the routing table and subsequent gateways listed in other interfaces are ignored. This is especially important to consider when configuring the real servers.

After physically connecting together the hardware, configure the network interfaces on the primary and backup LVS routers. This can be done using a graphical application such as **system-config-network** or by editing the network scripts manually. For more information about adding devices using **system-config-network**, see the chapter titled *Network Configuration* in the *Red Hat Enterprise Linux Deployment Guide*. For the remainder of the chapter, example alterations to network interfaces are made either manually or through the **Piranha Configuration Tool**.

3.3.1. General LVS Networking Tips

Configure the real IP addresses for both the public and private networks on the LVS routers before attempting to configure LVS using the **Piranha Configuration Tool**. The sections on each topology give example network addresses, but the actual network addresses are needed. Below are some useful commands for bringing up network interfaces or checking their status.

Bringing Up Real Network Interfaces

To bring up a real network interface, use the following command as root, replacing *N* with the number corresponding to the interface (**`eth0`** and **`eth1`**).

```
/sbin/ifup ethN
```

Warning

Do *not* use the **`ifup`** scripts to bring up any floating IP addresses you may configure using **Piranha Configuration Tool** (**`eth0:1`** or **`eth1:1`**). Use the **`service`** command to start **`pulse`** instead (see *Section 4.8, “Starting LVS”* for details).

Bringing Down Real Network Interfaces

To bring down a real network interface, use the following command as root, replacing *N* with the number corresponding to the interface (**`eth0`** and **`eth1`**).

```
/sbin/ifdown ethN
```

Checking the Status of Network Interfaces

If you need to check which network interfaces are up at any given time, type the following:

```
/sbin/ifconfig
```

To view the routing table for a machine, issue the following command:

/sbin/route

3.3.1.1. Troubleshooting Virtual IP Address Issues

There may be instances when an administrator encounters issues during an automatic failover from an active LVS host to the standby host. All of the virtual IP addresses may not activate on the standby host upon failover. This issue can also occur when the standby host is stopped and the primary host is activated. Only when the **pulse** service is manually restarted do all virtual IP addresses activate.

To remedy this issue temporarily, you can run the following command at the root shell prompt:

```
echo 1 > /proc/sys/net/ipv4/conf/all/promote_secondaries
```

Note that this will only *temporarily* remedy the issue and that the command will not hold through a system reboot.

To permanently remedy this issue, open the **/etc/sysctl.conf** file and add the following line:

```
net.ipv4.conf.all.promote_secondaries = 1
```

3.4. Multi-port Services and LVS

LVS routers under any topology require extra configuration when creating multi-port LVS services. Multi-port services can be created artificially by using firewall marks to bundle together different, but related protocols, such as HTTP (port 80) and HTTPS (port 443), or when LVS is used with true multi-port protocols, such as FTP. In either case, the LVS router uses firewall marks to recognize that packets destined for different ports, but bearing the same firewall mark, should be handled identically. Also, when combined with persistence, firewall marks ensure connections from the client machine are routed to the same host, as long as the connections occur within the length of time specified by the persistence parameter. For more on assigning persistence to a virtual server, see *Section 4.6.1, "The **VIRTUAL SERVER** Subsection"*.

Unfortunately, the mechanism used to balance the loads on the real servers — IPVS — can recognize the firewall marks assigned to a packet, but cannot itself assign firewall marks. The job of *assigning* firewall marks must be performed by the network packet filter, **iptables**, outside of **Piranha Configuration Tool**.

3.4.1. Assigning Firewall Marks

To assign firewall marks to a packet destined for a particular port, the administrator must use **iptables**.

This section illustrates how to bundle HTTP and HTTPS as an example; however, FTP is another commonly clustered multi-port protocol. If an LVS is used for FTP services, refer to *Section 3.5, "Configuring FTP"* for configuration details.

The basic rule to remember when using firewall marks is that for every protocol using a firewall mark in **Piranha Configuration Tool** there must be a commensurate **iptables** rule to assign marks to the network packets.

Before creating network packet filter rules, make sure there are no rules already in place. To do this, open a shell prompt, login as root, and type:

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

If **`iptables`** is not running, the prompt will instantly reappear.

If **`iptables`** is active, it displays a set of rules. If rules are present, type the following command:

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

If the rules already in place are important, check the contents of **`/etc/sysconfig/iptables`** and copy any rules worth keeping to a safe place before proceeding.

Below are rules which assign the same firewall mark, 80, to incoming traffic destined for the floating IP address, *`n.n.n.n`*, on ports 80 and 443.

```
/sbin/modprobe ip_tables
```

```
/sbin/iptables -t mangle -A PREROUTING -p tcp -d n.n.n.n/32 --dport 80 -j
MARK --set-mark 80
```

```
/sbin/iptables -t mangle-A PREROUTING -p tcp -d n.n.n.n/32 --dport 443 -j
MARK --set-mark 80
```

For instructions on assigning the VIP to the public network interface, see *Section 4.6.1, "The **VIRTUAL SERVER** Subsection"*. Also note that you must log in as root and load the module for **`iptables`** before issuing rules for the first time.

In the above **`iptables`** commands, *`n.n.n.n`* should be replaced with the floating IP for your HTTP and HTTPS virtual servers. These commands have the net effect of assigning any traffic addressed to the VIP on the appropriate ports a firewall mark of 80, which in turn is recognized by IPVS and forwarded appropriately.

Warning

The commands above will take effect immediately, but do not persist through a reboot of the system. To ensure network packet filter settings are restored upon reboot, refer to *Section 3.6, "Saving Network Packet Filter Settings"*

3.5. Configuring FTP

File Transport Protocol (FTP) is an old and complex multi-port protocol that presents a distinct set of challenges to an LVS environment. To understand the nature of these challenges, you must first understand some key things about how FTP works.

3.5.1. How FTP Works

With most other server client relationships, the client machine opens up a connection to the server on a particular port and the server then responds to the client on that port. When an FTP client connects to an FTP server it opens a connection to the FTP control port 21. Then the *client* tells the FTP *server* whether to establish an *active* or *passive* connection. The type of connection chosen by the client determines how the server responds and on what ports transactions will occur.

The two types of data connections are:

Active Connections
: When an active connection is established, the *server* opens a data connection to the client from port 20 to a high range port on the client machine. All data from the server is then passed over this connection.

Passive Connections
: When a passive connection is established, the *client* asks the FTP server to establish a passive connection port, which can be on any port higher than 10,000. The server then binds to this high-numbered port for this particular session and relays that port number back to the client. The client then opens the newly bound port for the data connection. Each data request the client makes results in a separate data connection. Most modern FTP clients attempt to establish a passive connection when requesting data from servers.

Note

The *client* determines the type of connection, not the server. This means to effectively cluster FTP, you must configure the LVS routers to handle both active and passive connections.

The FTP client/server relationship can potentially open a large number of ports that the **Piranha Configuration Tool** and IPVS do not know about.

3.5.2. How This Affects LVS Routing

IPVS packet forwarding only allows connections in and out of the cluster based on it recognizing its port number or its firewall mark. If a client from outside the cluster attempts to open a port IPVS is not configured to handle, it drops the connection. Similarly, if the real server attempts to open a connection back out to the Internet on a port IPVS does not know about, it drops the connection. This means *all* connections from FTP clients on the Internet *must* have the same firewall mark assigned to them and all connections from the FTP server *must* be properly forwarded to the Internet using network packet filtering rules.

3.5.3. Creating Network Packet Filter Rules

Before assigning any **`iptables`** rules for FTP service, review the information in *Section 3.4.1, "Assigning Firewall Marks"* concerning multi-port services and techniques for checking the existing network packet filtering rules.

Below are rules which assign the same firewall mark, 21, to FTP traffic. For these rules to work properly, you must also use the **VIRTUAL SERVER** subsection of **Piranha Configuration Tool** to configure a virtual server for port 21 with a value of **`21`** in the **Firewall Mark** field. See *Section 4.6.1, "The **VIRTUAL SERVER** Subsection"* for details.

3.5.3.1. Rules for Active Connections

The rules for active connections tell the kernel to accept and forward connections coming to the *internal* floating IP address on port 20 — the FTP data port.

The following **`iptables`** command allows the LVS router to accept outgoing connections from the real servers that IPVS does not know about:

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

In the **`iptables`** command, *`n.n.n`* should be replaced with the first three values for the floating IP for the NAT interface's internal network interface defined in the **GLOBAL SETTINGS** panel of **Piranha Configuration Tool**.

3.5.3.2. Rules for Passive Connections

The rules for passive connections assign the appropriate firewall mark to connections coming in from the Internet to the floating IP for the service on a wide range of ports — 10,000 to 20,000.

> **Warning**
>
> If you are limiting the port range for passive connections, you must also configure the VSFTP server to use a matching port range. This can be accomplished by adding the following lines to **`/etc/vsftpd.conf`**:
>
> **`pasv_min_port=10000`**
>
> **`pasv_max_port=20000`**
>
> You must also control the address that the server displays to the client for passive FTP connections. In a NAT routed LVS system, add the following line to **`/etc/vsftpd.conf`** to override the real server IP address to the VIP, which is what the client sees upon connection. For example:
>
> **`pasv_address=n.n.n.n`**
>
> Replace *`n.n.n.n`* with the VIP address of the LVS system.
>
> For configuration of other FTP servers, consult the respective documentation.

This range should be a wide enough for most situations; however, you can increase this number to include all available non-secured ports by changing **`10000:20000`** in the commands below to **`1024:65535`**.

The following **`iptables`** commands have the net effect of assigning any traffic addressed to the floating IP on the appropriate ports a firewall mark of 21, which is in turn recognized by IPVS and forwarded appropriately:

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

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

In the **`iptables`** commands, *`n.n.n.n`* should be replaced with the floating IP for the FTP virtual server defined in the **VIRTUAL SERVER** subsection of **Piranha Configuration Tool**.

Warning

The commands above take effect immediately, but do not persist through a reboot of the system. To ensure network packet filter settings are restored after a reboot, see *Section 3.6, “Saving Network Packet Filter Settings”*

Finally, you need to be sure that the appropriate service is set to activate on the proper runlevels. For more on this, refer to *Section 2.1, “Configuring Services on the LVS Routers”*.

3.6. Saving Network Packet Filter Settings

After configuring the appropriate network packet filters for your situation, save the settings so they get restored after a reboot. For **`iptables`**, type the following command:

```
/sbin/service iptables save
```

This saves the settings in **`/etc/sysconfig/iptables`** so they can be recalled at boot time.

Once this file is written, you are able to use the **`/sbin/service`** command to start, stop, and check the status (using the status switch) of **`iptables`**. The **`/sbin/service`** will automatically load the appropriate module for you. For an example of how to use the **`/sbin/service`** command, see *Section 2.3, “Starting the **Piranha Configuration Tool** Service”*.

Finally, you need to be sure the appropriate service is set to activate on the proper runlevels. For more on this, see *Section 2.1, “Configuring Services on the LVS Routers”*.

The next chapter explains how to use the **Piranha Configuration Tool** to configure the LVS router and describe the steps necessary to activate LVS.

Configuring the LVS Routers with Piranha Configuration Tool

The **Piranha Configuration Tool** provides a structured approach to creating the necessary configuration file for LVS — **`/etc/sysconfig/ha/lvs.cf`**. This chapter describes the basic operation of the **Piranha Configuration Tool** and how to activate LVS once configuration is complete.

Important

The configuration file for LVS follows strict formatting rules. Using the **Piranha Configuration Tool** is the best way to prevent syntax errors in the **`lvs.cf`** and therefore prevent software failures.

4.1. Necessary Software

The **`piranha-gui`** service must be running on the primary LVS router to use the **Piranha Configuration Tool**. To configure LVS, you minimally need a text-only Web browser, such as **`links`**. If you are accessing the LVS router from another machine, you also need an **`ssh`** connection to the primary LVS router as the root user.

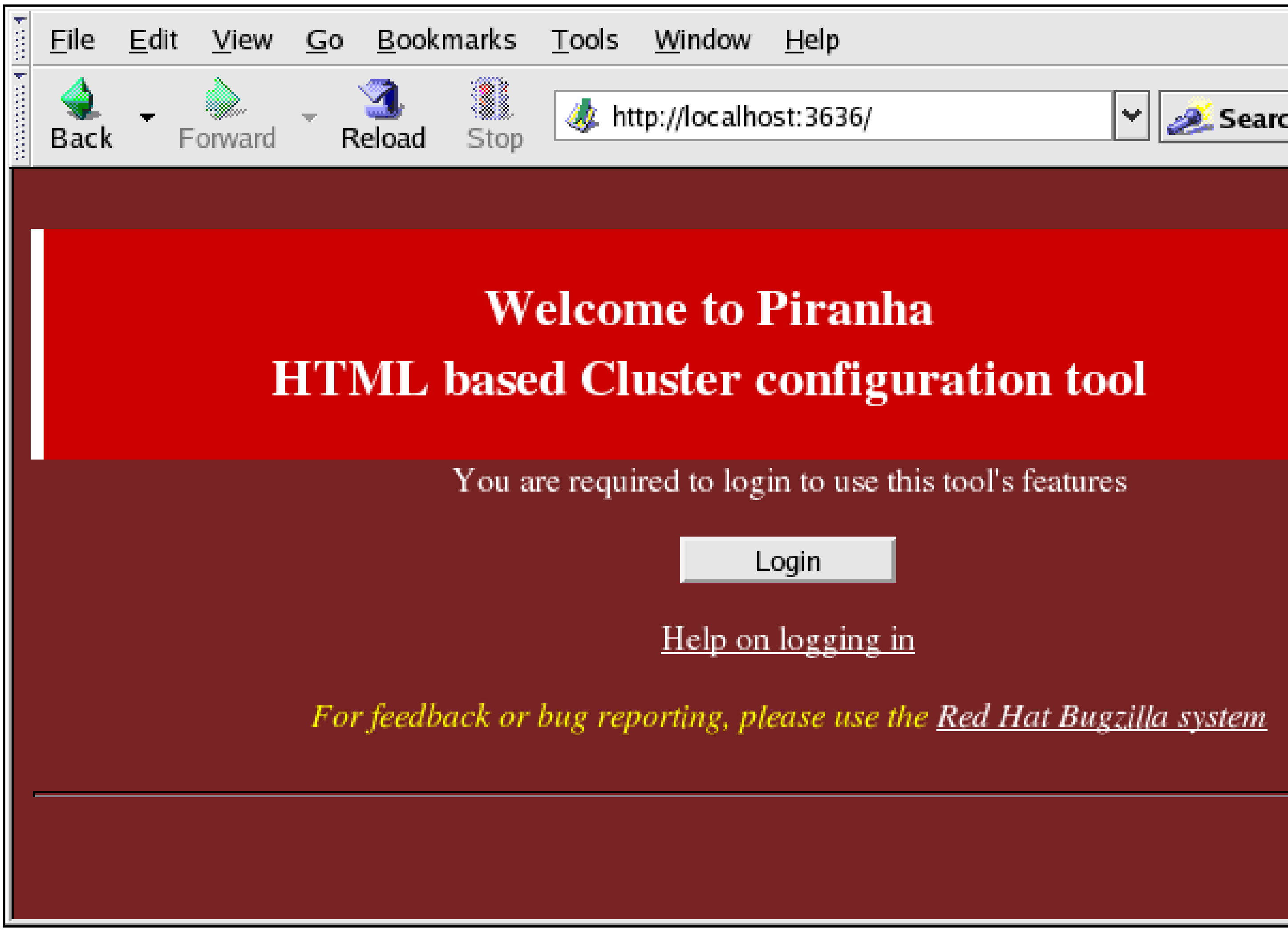
While configuring the primary LVS router it is a good idea to keep a concurrent **`ssh`** connection in a terminal window. This connection provides a secure way to restart **`pulse`** and other services, configure network packet filters, and monitor **`/var/log/messages`** during trouble shooting.

The next four sections walk through each of the configuration pages of the **Piranha Configuration Tool** and give instructions on using it to set up LVS.

4.2. Logging Into the Piranha Configuration Tool

When configuring LVS, you should always begin by configuring the primary router with the **Piranha Configuration Tool**. To do this,verify that the **`piranha-gui`** service is running and an administrative password has been set, as described in *Section 2.2, “Setting a Password for the **Piranha Configuration Tool**”*.

If you are accessing the machine locally, you can open **`http://localhost:3636`** in a Web browser to access the **Piranha Configuration Tool**. Otherwise, type in the hostname or real IP address for the server followed by **`:3636`**. Once the browser connects, you will see the screen shown in *Figure 4.1, “The Welcome Panel”*.



Welcome to Piranha
HTML based Cluster configuration tool

You are required to login to use this tool's features

Help on logging in

For feedback or bug reporting, please use the Red Hat Bugzilla system

Figure 4.1. The Welcome Panel

Click on the **Login** button and enter **`piranha`** for the **Username** and the administrative password you created in the **Password** field.

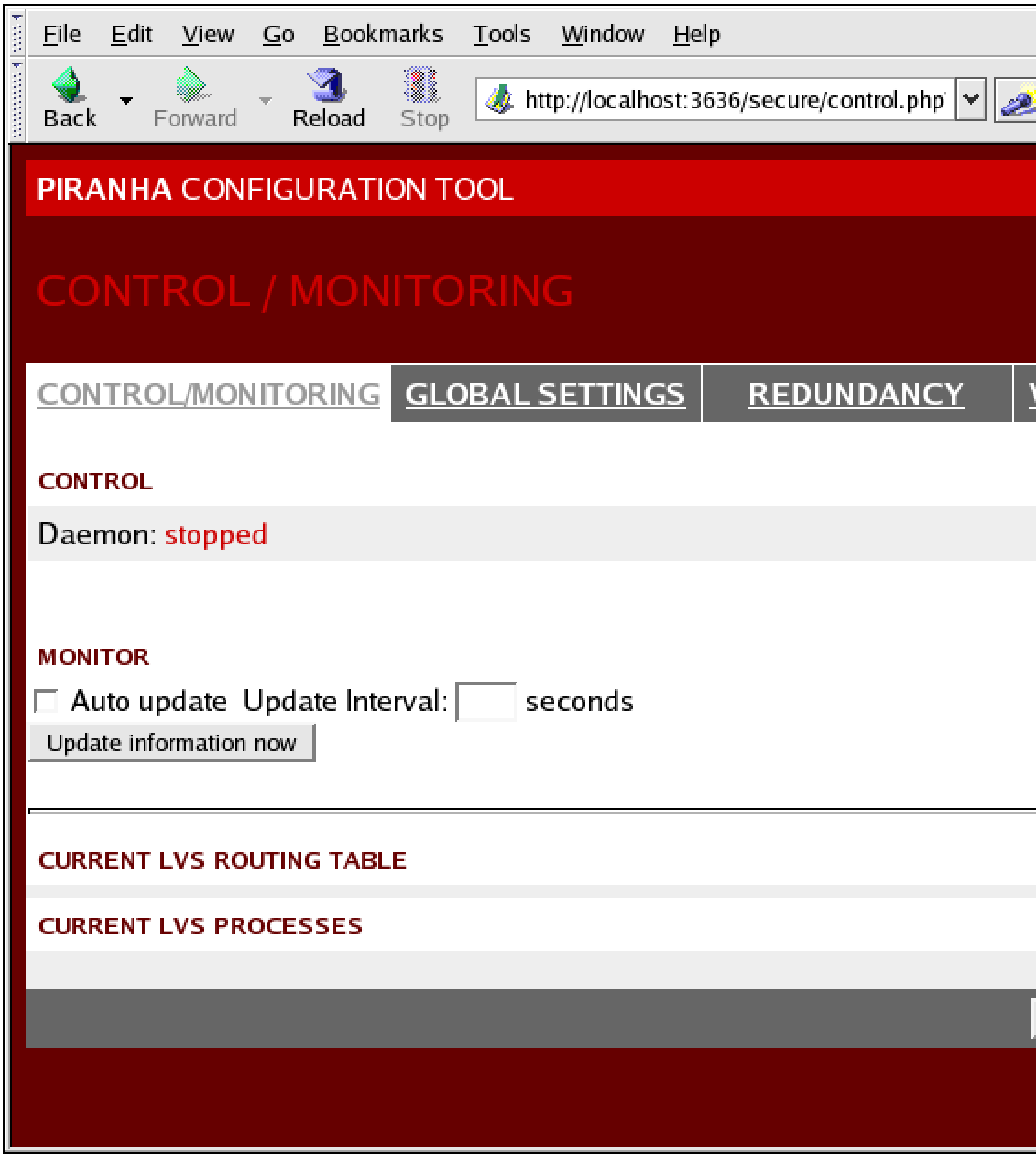
The **Piranha Configuration Tool** is made of four main screens or *panels*. In addition, the **Virtual Servers** panel contains four *subsections*. The **CONTROL/MONITORING** panel is the first panel after the login screen.

4.3. CONTROL/MONITORING

The **CONTROL/MONITORING** Panel presents the a limited runtime status of LVS. It displays the status of the **`pulse`** daemon, the LVS routing table, and the LVS-spawned **nanny** processes.

Note

The fields for **CURRENT LVS ROUTING TABLE** and **CURRENT LVS PROCESSES** remain blank until you actually start LVS, as shown in *Section 4.8, “Starting LVS”*.


File Edit View Go Bookmarks Tools Window Help
Back
Forward
Reload
Stop
http://localhost:3636/secure/control.php

PIRANHA CONFIGURATION TOOL

CONTROL / MONITORING

CONTROL

Daemon: stopped

MONITOR

☐ Auto update Update Interval: seconds

CURRENT LVS ROUTING TABLE

CURRENT LVS PROCESSES

Figure 4.2. The **CONTROL/MONITORING** Panel

Auto update

The status display on this page can be updated automatically at a user configurable interval. To enable this feature, click on the **Auto update** checkbox and set the desired update frequency in the **Update frequency in seconds** text box (the default value is 10 seconds).

It is not recommended that you set the automatic update to an interval less than 10 seconds. Doing so may make it difficult to reconfigure the **Auto update** interval because the page will update too frequently. If you encounter this issue, simply click on another panel and then back on **CONTROL/MONITORING**.

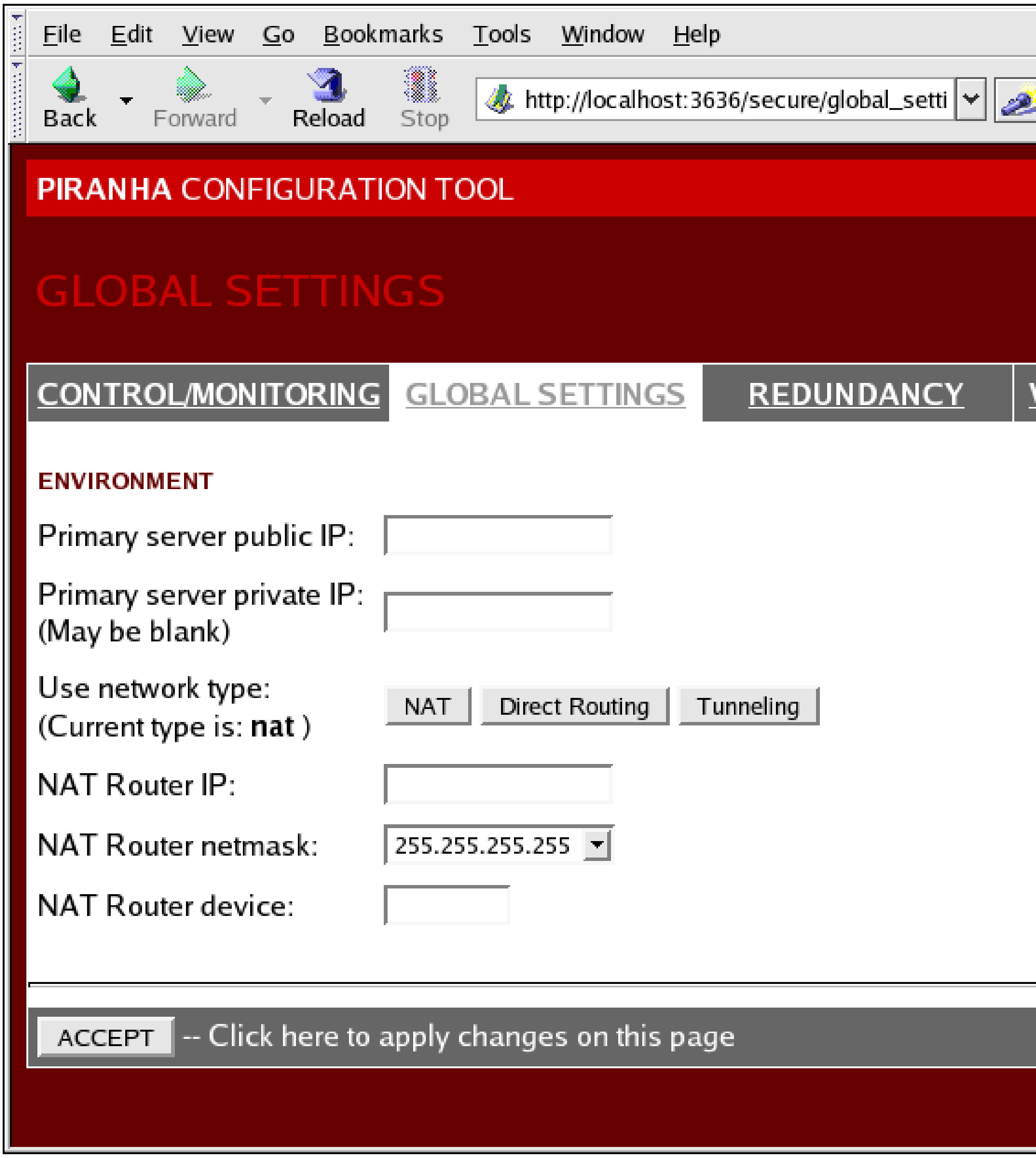
The **Auto update** feature does not work with all browsers, such as **Mozilla**.

Update information now
You can manually update the status information manually by clicking this button.

CHANGE PASSWORD
Clicking this button takes you to a help screen with information on how to change the administrative password for the **Piranha Configuration Tool**.

4.4. GLOBAL SETTINGS

The **GLOBAL SETTINGS** panel is where the you define the networking details for the primary LVS router's public and private network interfaces.


File Edit View Go Bookmarks Tools Window Help
Back Forward Reload Stop
http://localhost:3636/secure/global_setti
PIRANHA CONFIGURATION TOOL
CONTROL/MONITORING GLOBAL SETTINGS REDUNDANCY

GLOBAL SETTINGS

ENVIRONMENT

Primary server public IP:	
Primary server private IP: (May be blank)	
Use network type: (Current type is: **nat**)	NAT / Direct Routing / Tunneling
NAT Router IP:	
NAT Router netmask:	255.255.255.255
NAT Router device:	

ACCEPT -- Click here to apply changes on this page

Figure 4.3. The **GLOBAL SETTINGS** Panel

The top half of this panel sets up the primary LVS router's public and private network interfaces. These are the interfaces already configured in *Section 3.1.1, “Configuring Network Interfaces for LVS with NAT”*.

Primary server public IP
In this field, enter the publicly routable real IP address for the primary LVS node.

Primary server private IP

Enter the real IP address for an alternative network interface on the primary LVS node. This address is used solely as an alternative heartbeat channel for the backup router and does not have to correlate to the real private IP address assigned in *Section 3.1.1, “Configuring Network Interfaces for LVS with NAT”*. You may leave this field blank, but doing so will mean there is no alternate heartbeat channel for the backup LVS router to use and therefore will create a single point of failure.

> **Note**
>
> The private IP address is not needed for **Direct Routing** configurations, as all real servers as well as the LVS directors share the same virtual IP addresses and should have the same IP route configuration.

> **Note**
>
> The primary LVS router's private IP can be configured on any interface that accepts TCP/IP, whether it be an Ethernet adapter or a serial port.

Use network type

Click the **NAT** button to select NAT routing.

Click the **Direct Routing** button to select direct routing.

The next three fields deal specifically with the NAT router's virtual network interface connecting the private network with the real servers. These fields *do not* apply to the direct routing network type.

NAT Router IP

Enter the private floating IP in this text field. This floating IP should be used as the gateway for the real servers.

NAT Router netmask

If the NAT router's floating IP needs a particular netmask, select it from drop-down list.

NAT Router device

Use this text field to define the device name of the network interface for the floating IP address, such as **`eth1:1`**.

> **Note**
>
> You should alias the NAT floating IP address to the Ethernet interface connected to the private network. In this example, the private network is on the **`eth1`** interface, so **`eth1:1`** is the floating IP address.

Warning

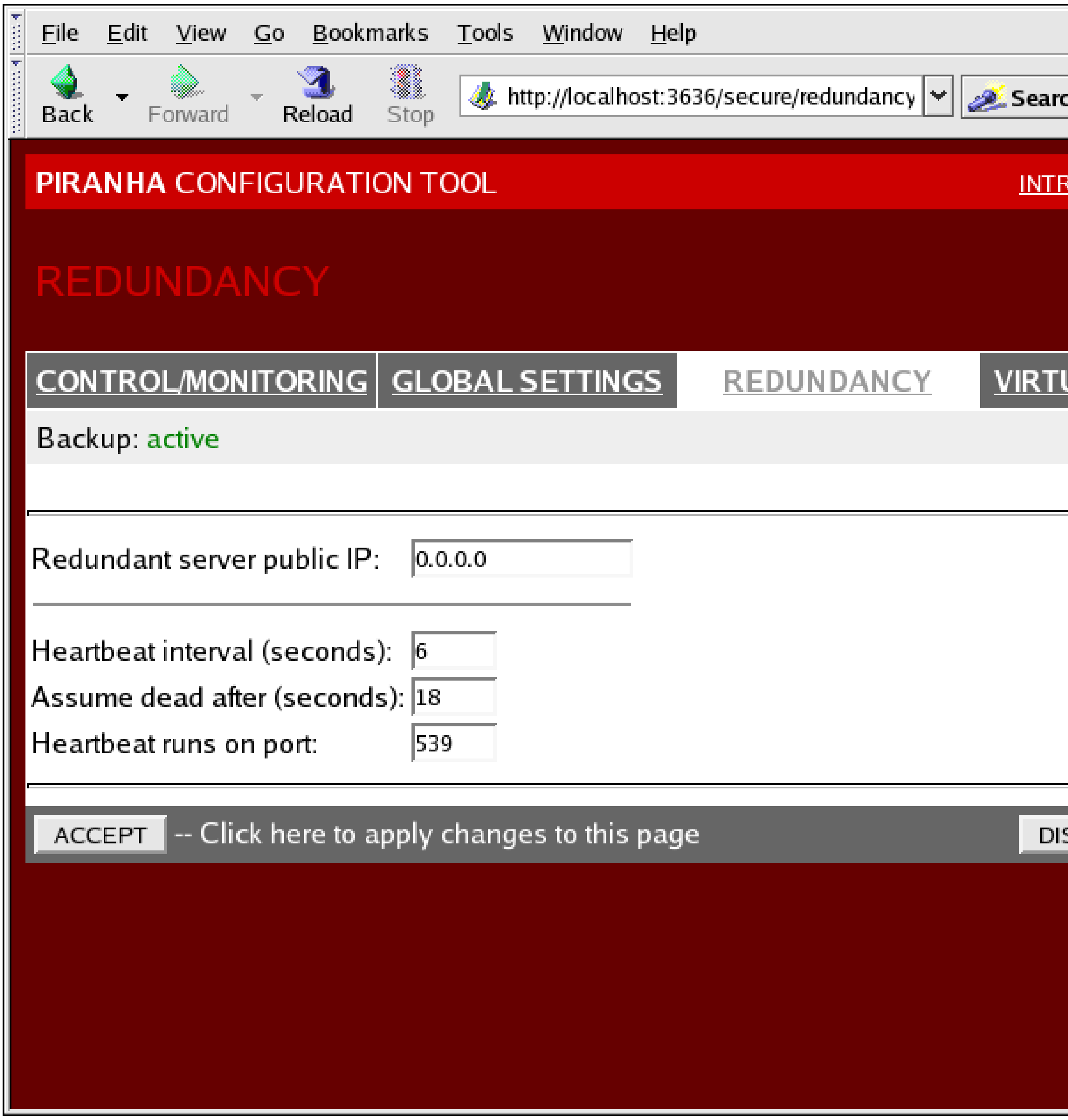
After completing this page, click the **ACCEPT** button to make sure you do not lose any changes when selecting a new panel.

4.5. REDUNDANCY

The **REDUNDANCY** panel allows you to configure of the backup LVS router node and set various heartbeat monitoring options.

Note

The first time you visit this screen, it displays an "inactive" **Backup** status and an **ENABLE** button. To configure the backup LVS router, click on the **ENABLE** button so that the screen matches *Figure 4.4, "The **REDUNDANCY** Panel"*.


File Edit View Go Bookmarks Tools Window Help
Back
Forward
Reload
Stop
http://localhost:3636/secure/redundancy
Searc
PIRANHA CONFIGURATION TOOL
INTR
REDUNDANCY
CONTROL/MONITORING
GLOBAL SETTINGS
REDUNDANCY
VIRTU
ACCEPT
-- Click here to apply changes to this page
DIS

Backup: active

Redundant server public IP: 0.0.0.0

Heartbeat interval (seconds): 6

Assume dead after (seconds): 18

Heartbeat runs on port: 539

Figure 4.4. The **REDUNDANCY** Panel

Redundant server public IP

Enter the public real IP address for the backup LVS router node.

Redundant server private IP

Enter the backup node's private real IP address in this text field.

If you do not see the field called **Redundant server private IP**, go back to the **GLOBAL SETTINGS** panel and enter a **Primary server private IP** address and click **ACCEPT**.

The rest of the panel is devoted to configuring the heartbeat channel, which is used by the backup node to monitor the primary node for failure.

Heartbeat Interval (seconds)
: This field sets the number of seconds between heartbeats — the interval that the backup node will check the functional status of the primary LVS node.

Assume dead after (seconds)
: If the primary LVS node does not respond after this number of seconds, then the backup LVS router node will initiate failover.

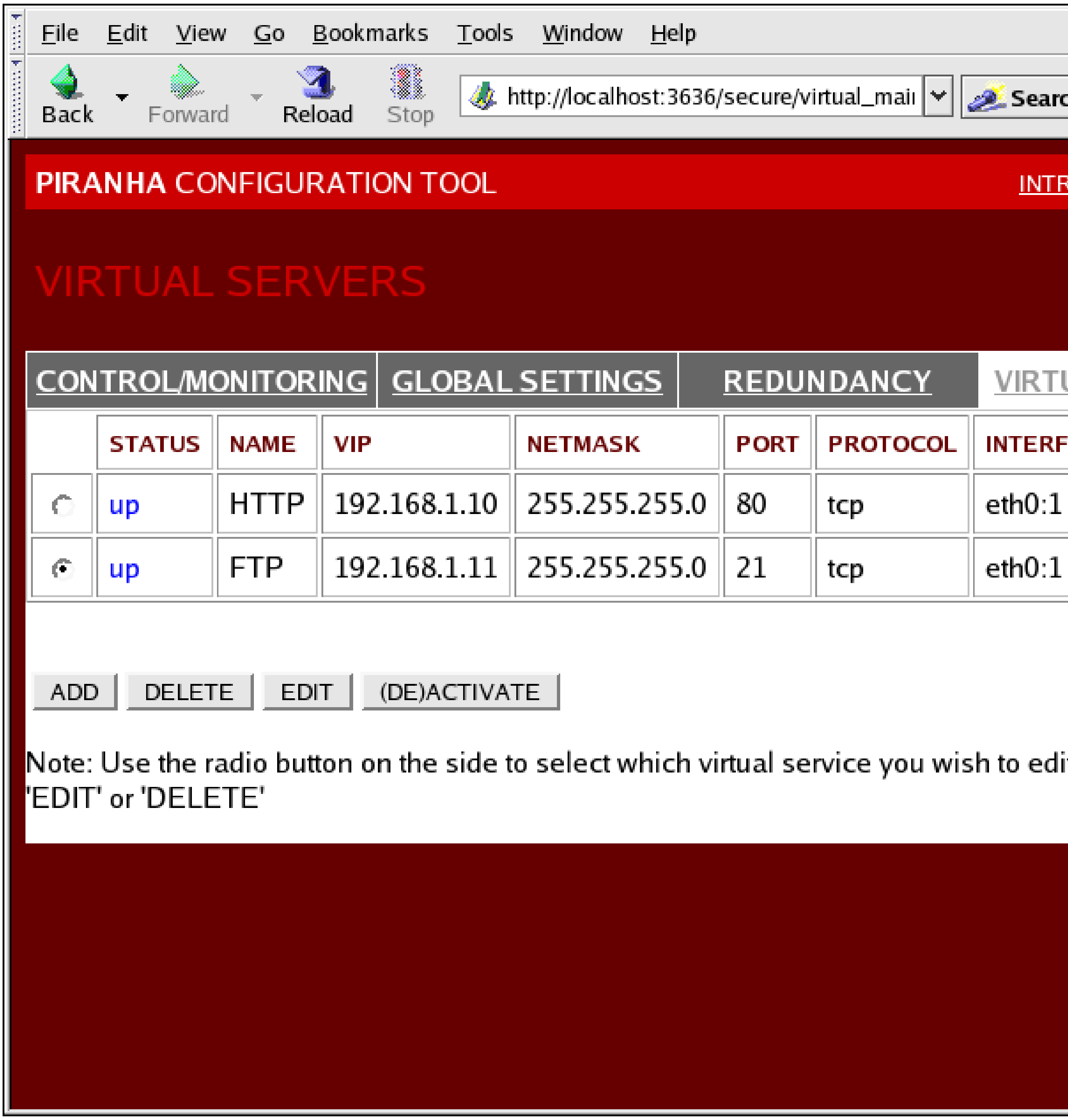
Heartbeat runs on port
: This field sets the port at which the heartbeat communicates with the primary LVS node. The default is set to 539 if this field is left blank.

Warning

Remember to click the **ACCEPT** button after making any changes in this panel to make sure you do not lose any changes when selecting a new panel.

4.6. VIRTUAL SERVERS

The **VIRTUAL SERVERS** panel displays information for each currently defined virtual server. Each table entry shows the status of the virtual server, the server name, the virtual IP assigned to the server, the netmask of the virtual IP, the port number to which the service communicates, the protocol used, and the virtual device interface.


File Edit View Go Bookmarks Tools Window Help
Back Forward Reload Stop
http://localhost:3636/secure/virtual_mai
Searc
PIRANHA CONFIGURATION TOOL
INTR
VIRTUAL SERVERS
CONTROL/MONITORING GLOBAL SETTINGS REDUNDANCY VIRTU
ADD DELETE EDIT (DE)ACTIVATE

	STATUS	NAME	VIP	NETMASK	PORT	PROTOCOL	INTERF
○	up	HTTP	192.168.1.10	255.255.255.0	80	tcp	eth0:1
●	up	FTP	192.168.1.11	255.255.255.0	21	tcp	eth0:1

Note: Use the radio button on the side to select which virtual service you wish to edi
'EDIT' or 'DELETE'

Figure 4.5. The **VIRTUAL SERVERS** Panel

Each server displayed in the **VIRTUAL SERVERS** panel can be configured on subsequent screens or *subsections*.

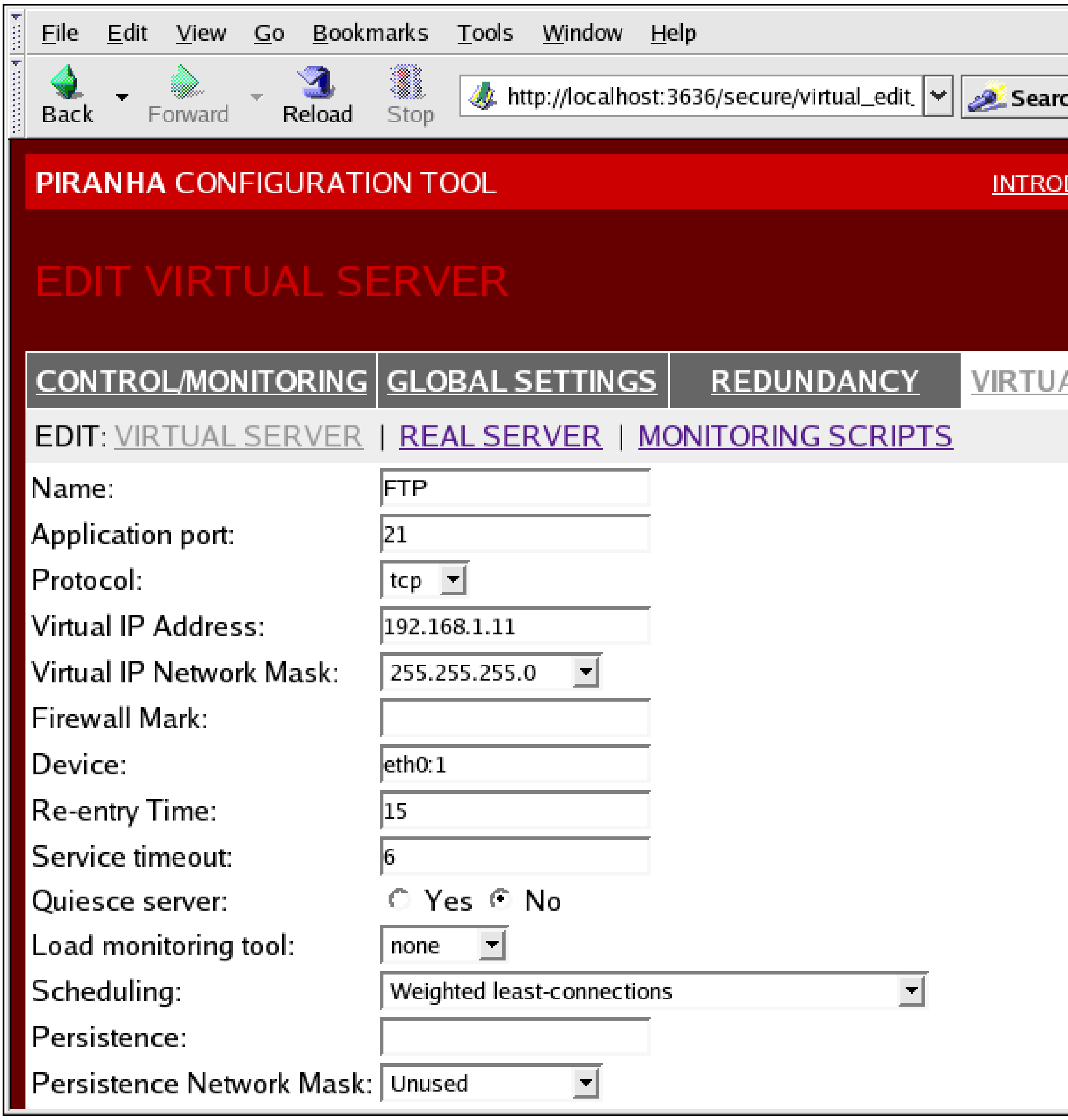
To add a service, click the **ADD** button. To remove a service, select it by clicking the radio button next to the virtual server and click the **DELETE** button.

To enable or disable a virtual server in the table click its radio button and click the **(DE)ACTIVATE** button.

After adding a virtual server, you can configure it by clicking the radio button to its left and clicking the **EDIT** button to display the **VIRTUAL SERVER** subsection.

4.6.1. The VIRTUAL SERVER Subsection

The **VIRTUAL SERVER** subsection panel shown in *Figure 4.6, “The **VIRTUAL SERVERS** Subsection”* allows you to configure an individual virtual server. Links to subsections related specifically to this virtual server are located along the top of the page. But before configuring any of the subsections related to this virtual server, complete this page and click on the **ACCEPT** button.


File Edit View Go Bookmarks Tools Window Help
Back
Forward
Reload
Stop
http://localhost:3636/secure/virtual_edit_
Searc
PIRANHA CONFIGURATION TOOL
INTRO
EDIT VIRTUAL SERVER
CONTROL/MONITORING
GLOBAL SETTINGS
REDUNDANCY
VIRTUA
EDIT: VIRTUAL SERVER | REAL SERVER | MONITORING SCRIPTS

Name:	FTP
Application port:	21
Protocol:	tcp
Virtual IP Address:	192.168.1.11
Virtual IP Network Mask:	255.255.255.0
Firewall Mark:	
Device:	eth0:1
Re-entry Time:	15
Service timeout:	6
Quiesce server:	Yes [✓ No]
Load monitoring tool:	none
Scheduling:	Weighted least-connections
Persistence:	
Persistence Network Mask:	Unused

Figure 4.6. The **VIRTUAL SERVERS** Subsection

Name

Enter a descriptive name to identify the virtual server. This name is *not* the hostname for the machine, so make it descriptive and easily identifiable. You can even reference the protocol used by the virtual server, such as HTTP.

Application port

Enter the port number through which the service application will listen. Since this example is for HTTP services, port 80 is used.

Protocol
: Choose between UDP and TCP in the drop-down menu. Web servers typically communicate via the TCP protocol, so this is selected in the example above.

Virtual IP Address
: Enter the virtual server's floating IP address in this text field.

Virtual IP Network Mask
: Set the netmask for this virtual server with the drop-down menu.

Firewall Mark
: Do *not* enter a firewall mark integer value in this field unless you are bundling multi-port protocols or creating a multi-port virtual server for separate, but related protocols. In this example, the above virtual server has a **Firewall Mark** of 80 because we are bundling connections to HTTP on port 80 and to HTTPS on port 443 using the firewall mark value of 80. When combined with persistence, this technique will ensure users accessing both insecure and secure webpages are routed to the same real server, preserving state.

Warning

Entering a firewall mark in this field allows IPVS to recognize that packets bearing this firewall mark are treated the same, but you must perform further configuration outside of the **Piranha Configuration Tool** to actually assign the firewall marks. See *Section 3.4, "Multi-port Services and LVS"* for instructions on creating multi-port services and *Section 3.5, "Configuring FTP"* for creating a highly available FTP virtual server.

Device
: Enter the name of the network device to which you want the floating IP address defined the **Virtual IP Address** field to bind.

You should alias the public floating IP address to the Ethernet interface connected to the public network. In this example, the public network is on the **`eth0`** interface, so **`eth0:1`** should be entered as the device name.

Re-entry Time
: Enter an integer value which defines the length of time, in seconds, before the active LVS router attempts to bring a real server back into the pool after a failure.

Service Timeout
: Enter an integer value which defines the length of time, in seconds, before a real server is considered dead and removed from the pool.

Quiesce server
: When the **Quiesce server** radio button is selected, anytime a new real server node comes online, the least-connections table is reset to zero so the active LVS router routes requests as if all the real servers were freshly added to the pool. This option prevents the a new server from becoming bogged down with a high number of connections upon entering the pool.

Load monitoring tool

The LVS router can monitor the load on the various real servers by using either **`rup`** or **`ruptime`**. If you select **`rup`** from the drop-down menu, each real server must run the **`rstatd`** service. If you select **`ruptime`**, each real server must run the **`rwhod`** service.

Warning

Load monitoring is *not* the same as load balancing and can result in hard to predict scheduling behavior when combined with weighted scheduling algorithms. Also, if you use load monitoring, the real servers must be Linux machines.

Scheduling

Select your preferred scheduling algorithm from the drop-down menu. The default is **`Weighted least-connection`**. For more information on scheduling algorithms, see *Section 1.3.1, "Scheduling Algorithms"*.

Persistence

If an administrator needs persistent connections to the virtual server during client transactions, enter the number of seconds of inactivity allowed to lapse before a connection times out in this text field.

Important

If you entered a value in the **Firewall Mark** field above, you should enter a value for persistence as well. Also, be sure that if you use firewall marks and persistence together, that the amount of persistence is the same for each virtual server with the firewall mark. For more on persistence and firewall marks, refer to *Section 1.5, "Persistence and Firewall Marks"*.

Persistence Network Mask

To limit persistence to particular subnet, select the appropriate network mask from the drop-down menu.

Note

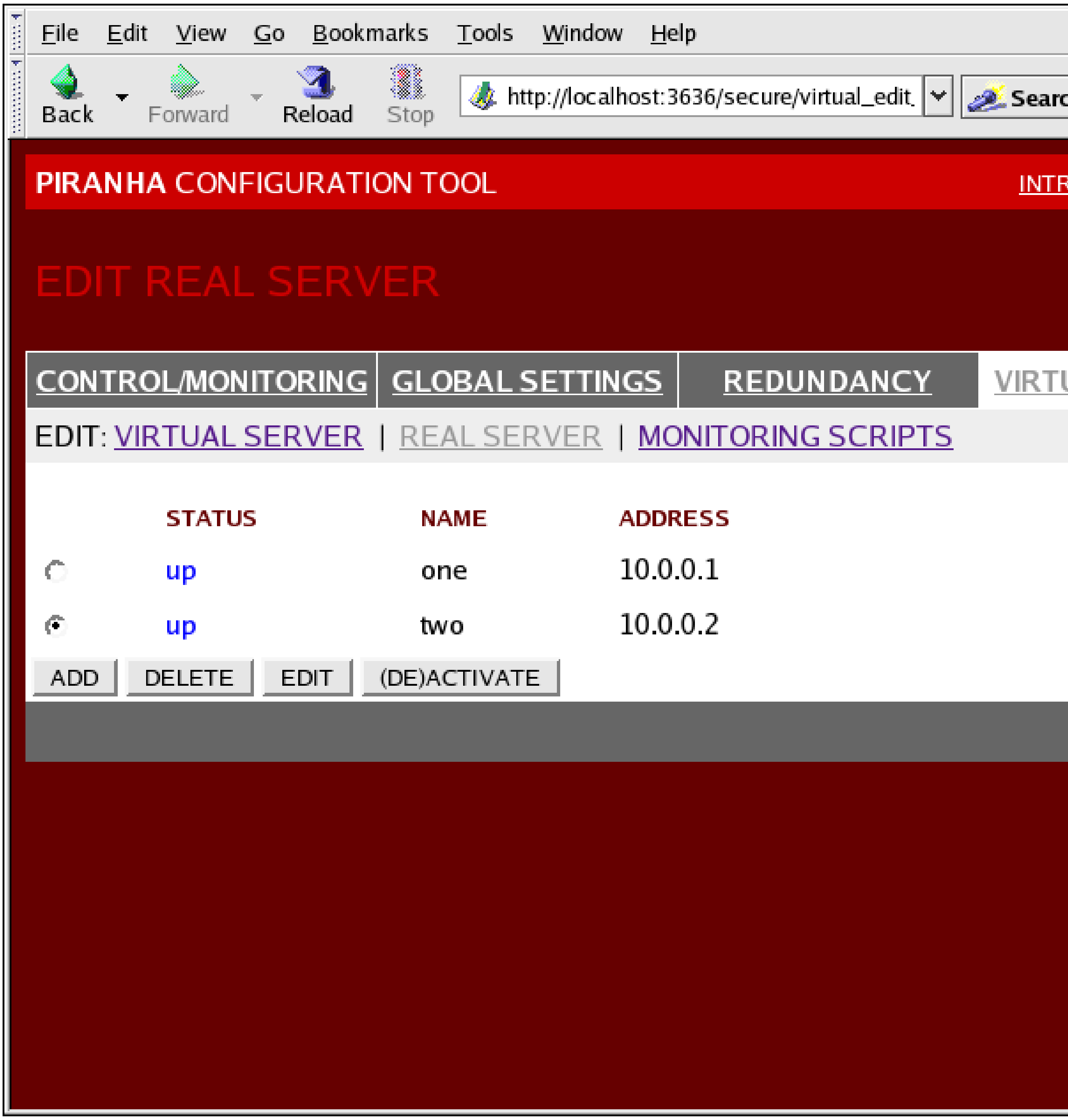
Before the advent of firewall marks, persistence limited by subnet was a crude way of bundling connections. Now, it is best to use persistence in relation to firewall marks to achieve the same result.

Warning

Remember to click the **ACCEPT** button after making any changes in this panel. To make sure you do not lose changes when selecting a new panel.

4.6.2. REAL SERVER Subsection

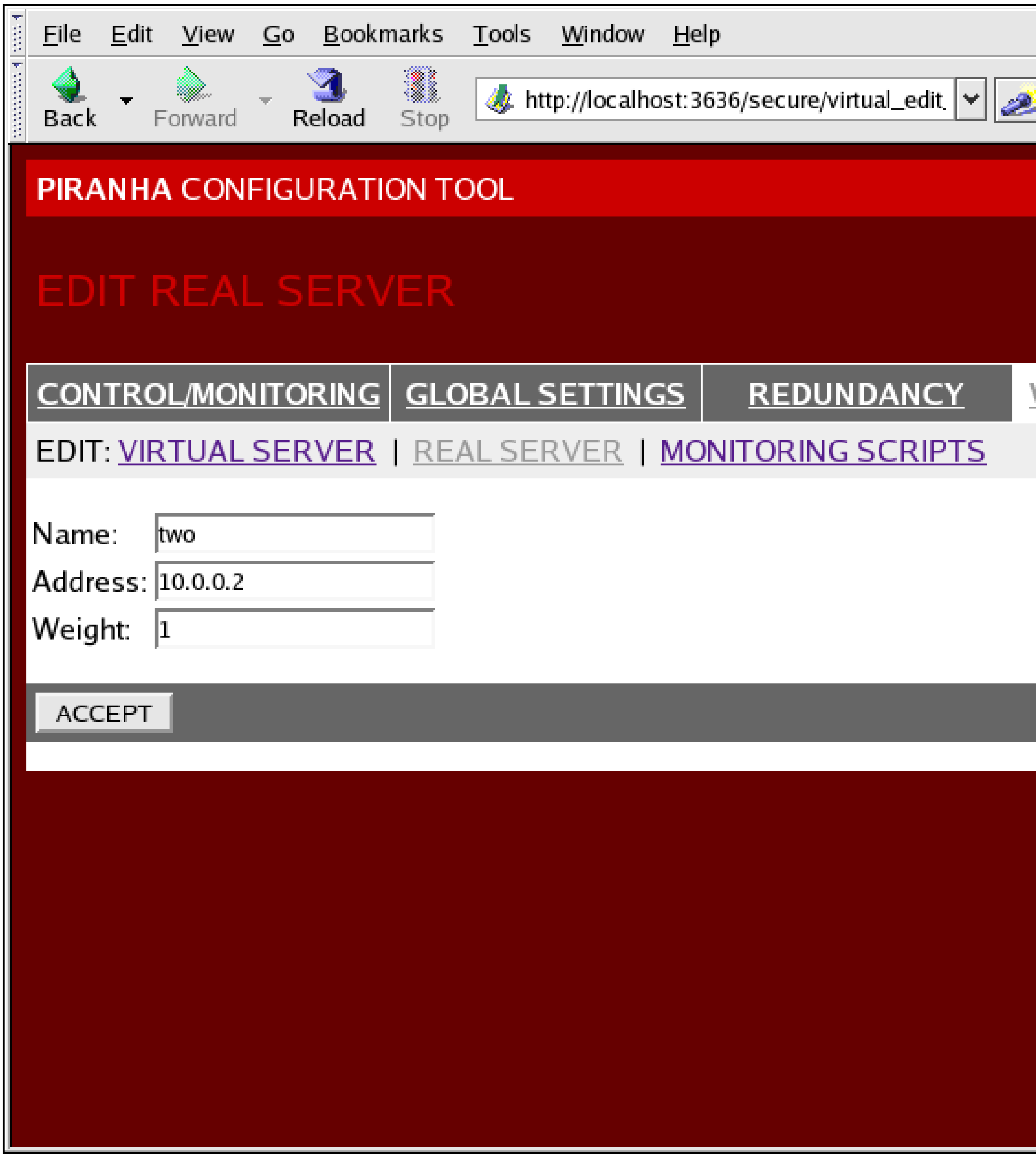
Clicking on the **REAL SERVER** subsection link at the top of the panel displays the **EDIT REAL SERVER** subsection. It displays the status of the physical server hosts for a particular virtual service.


File Edit View Go Bookmarks Tools Window Help
Back
Forward
Reload
Stop
http://localhost:3636/secure/virtual_edit_
Searc
PIRANHA CONFIGURATION TOOL
INTR
EDIT REAL SERVER
CONTROL/MONITORING
GLOBAL SETTINGS
REDUNDANCY
VIRTU
EDIT: VIRTUAL SERVER | REAL SERVER | MONITORING SCRIPTS
ADD
DELETE
EDIT
(DE)ACTIVATE

	STATUS	NAME	ADDRESS
○	up	one	10.0.0.1
●	up	two	10.0.0.2

Figure 4.7. The **REAL SERVER** Subsection

Click the **ADD** button to add a new server. To delete an existing server, select the radio button beside it and click the **DELETE** button. Click the **EDIT** button to load the **EDIT REAL SERVER** panel, as seen in *Figure 4.8, “The **REAL SERVER** Configuration Panel”*.


File Edit View Go Bookmarks Tools Window Help
Back
Forward
Reload
Stop
http://localhost:3636/secure/virtual_edit_
PIRANHA CONFIGURATION TOOL
EDIT REAL SERVER
CONTROL/MONITORING
GLOBAL SETTINGS
REDUNDANCY
EDIT: VIRTUAL SERVER | REAL SERVER | MONITORING SCRIPTS
ACCEPT

Name: two
Address: 10.0.0.2
Weight: 1

Figure 4.8. The **REAL SERVER** Configuration Panel

This panel consists of three entry fields:

Name
: A descriptive name for the real server.

Note

This name is *not* the hostname for the machine, so make it descriptive and easily identifiable.

Address

The real server's IP address. Since the listening port is already specified for the associated virtual server, do not add a port number.

Weight

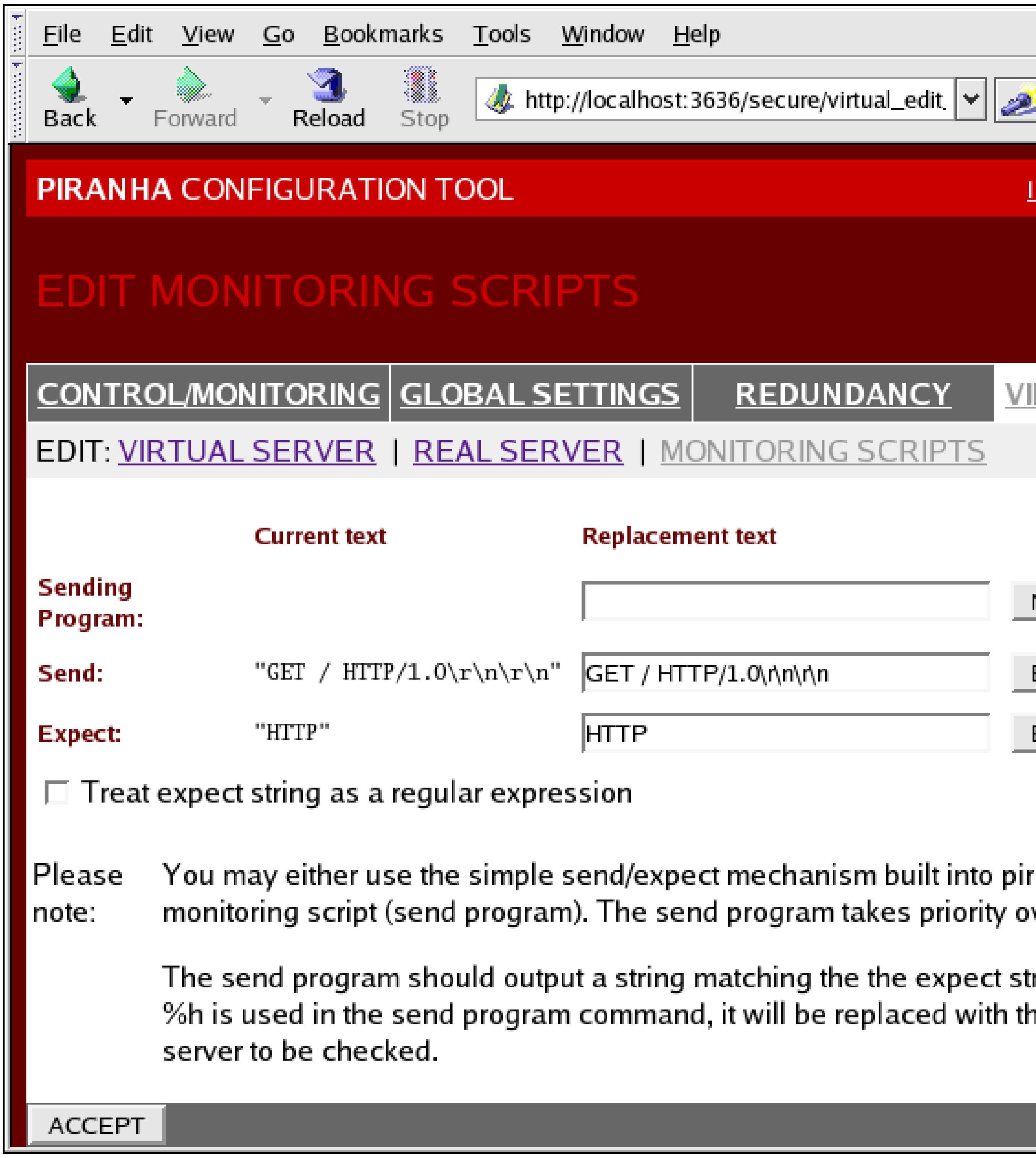
An integer value indicating this host's capacity relative to that of other hosts in the pool. The value can be arbitrary, but treat it as a ratio in relation to other real servers in the pool. For more on server weight, see *Section 1.3.2, “Server Weight and Scheduling”*.

Warning

Remember to click the **ACCEPT** button after making any changes in this panel. To make sure you do not lose any changes when selecting a new panel.

4.6.3. EDIT MONITORING SCRIPTS Subsection

Click on the **MONITORING SCRIPTS** link at the top of the page. The **EDIT MONITORING SCRIPTS** subsection allows the administrator to specify a send/expect string sequence to verify that the service for the virtual server is functional on each real server. It is also the place where the administrator can specify customized scripts to check services requiring dynamically changing data.


File Edit View Go Bookmarks Tools Window Help
Back
Forward
Reload
Stop
http://localhost:3636/secure/virtual_edit_
PIRANHA CONFIGURATION TOOL
CONTROL/MONITORING
GLOBAL SETTINGS
REDUNDANCY
VI
ACCEPT

EDIT MONITORING SCRIPTS

EDIT: VIRTUAL SERVER | REAL SERVER | MONITORING SCRIPTS

	Current text	Replacement text
Sending Program:		
Send:	"GET / HTTP/1.0\r\n\r\n"	GET / HTTP/1.0\r\n\r\n
Expect:	"HTTP"	HTTP

☐ Treat expect string as a regular expression

Please note: You may either use the simple send/expect mechanism built into pir monitoring script (send program). The send program takes priority ov

The send program should output a string matching the the expect str %h is used in the send program command, it will be replaced with th server to be checked.

Figure 4.9. The **EDIT MONITORING SCRIPTS** Subsection

Sending Program

For more advanced service verification, you can use this field to specify the path to a service-checking script. This functionality is especially helpful for services that require dynamically changing data, such as HTTPS or SSL.

To use this functionality, you must write a script that returns a textual response, set it to be executable, and type the path to it in the **Sending Program** field.

Note

To ensure that each server in the real server pool is checked, use the special token **%h** after the path to the script in the **Sending Program** field. This token is replaced with each real server's IP address as the script is called by the **nanny** daemon.

The following is a sample script to use as a guide when composing an external service-checking script:

```
#!/bin/sh

TEST=`dig -t soa example.com @$1 | grep -c dns.example.com

if [ $TEST != "1" ]; then
 echo "OK
else
 echo "FAIL"
fi
```

Note

If an external program is entered in the **Sending Program** field, then the **Send** field is ignored.

Send

Enter a string for the **nanny** daemon to send to each real server in this field. By default the send field is completed for HTTP. You can alter this value depending on your needs. If you leave this field blank, the **nanny** daemon attempts to open the port and assume the service is running if it succeeds.

Only one send sequence is allowed in this field, and it can only contain printable, ASCII characters as well as the following escape characters:

- \n for new line.
- \r for carriage return.
- \t for tab.
- \ to escape the next character which follows it.

Expect

Enter a the textual response the server should return if it is functioning properly. If you wrote your own sending program, enter the response you told it to send if it was successful.

Note

To determine what to send for a given service, you can open a `telnet` connection to the port on a real server and see what is returned. For instance, FTP reports 220 upon connecting, so could enter `quit` in the **Send** field and `220` in the **Expect** field.

Warning

Remember to click the **ACCEPT** button after making any changes in this panel. To make sure you do not lose any changes when selecting a new panel.

Once you have configured virtual servers using the **Piranha Configuration Tool**, you must copy specific configuration files to the backup LVS router. See *Section 4.7, “Synchronizing Configuration Files”* for details.

4.7. Synchronizing Configuration Files

After configuring the primary LVS router, there are several configuration files that must be copied to the backup LVS router before you start LVS.

These files include:

- `/etc/sysconfig/ha/lvs.cf` — the configuration file for the LVS routers.
- `/etc/sysctl` — the configuration file that, among other things, turns on packet forwarding in the kernel.
- `/etc/sysconfig/iptables` — If you are using firewall marks, you should synchronize one of these files based on which network packet filter you are using.

Important

The `/etc/sysctl.conf` and `/etc/sysconfig/iptables` files do *not* change when you configure LVS using the **Piranha Configuration Tool**.

4.7.1. Synchronizing `lvs.cf`

Anytime the LVS configuration file, `/etc/sysconfig/ha/lvs.cf`, is created or updated, you must copy it to the backup LVS router node.

Warning

Both the active and backup LVS router nodes must have identical **`lvs.cf`** files. Mismatched LVS configuration files between the LVS router nodes can prevent failover.

The best way to do this is to use the **`scp`** command.

Important

To use **`scp`** the **`sshd`** must be running on the backup router, see *Section 2.1, "Configuring Services on the LVS Routers"* for details on how to properly configure the necessary services on the LVS routers.

Issue the following command as the root user from the primary LVS router to sync the **`lvs.cf`** files between the router nodes:

```
scp /etc/sysconfig/ha/lvs.cf n.n.n.n:/etc/sysconfig/ha/lvs.cf
```

In the command, replace *`n.n.n.n`* with the real IP address of the backup LVS router.

4.7.2. Synchronizing `sysctl`

The **`sysctl`** file is only modified once in most situations. This file is read at boot time and tells the kernel to turn on packet forwarding.

Important

If you are not sure whether or not packet forwarding is enabled in the kernel, see *Section 2.5, "Turning on Packet Forwarding"* for instructions on how to check and, if necessary, enable this key functionality.

4.7.3. Synchronizing Network Packet Filtering Rules

If you are using **`iptables`**, you will need to synchronize the appropriate configuration file on the backup LVS router.

If you alter the any network packet filter rules, enter the following command as root from the primary LVS router:

```
scp /etc/sysconfig/iptables n.n.n.n:/etc/sysconfig/
```

In the command, replace *`n.n.n.n`* with the real IP address of the backup LVS router.

Next either open an **`ssh`** session to the backup router or log into the machine as root and type the following command:

```
/sbin/service iptables restart
```

Once you have copied these files over to the backup router and started the appropriate services (see *Section 2.1, “Configuring Services on the LVS Routers”* for more on this topic) you are ready to start LVS.

4.8. Starting LVS

To start LVS, it is best to have two root terminals open simultaneously or two simultaneous root open **`ssh`** sessions to the primary LVS router.

In one terminal, watch the kernel log messages with the command:

```
tail -f /var/log/messages
```

Then start LVS by typing the following command into the other terminal:

```
/sbin/service pulse start
```

Follow the progress of the **`pulse`** service's startup in the terminal with the kernel log messages. When you see the following output, the pulse daemon has started properly:

```
gratuitous lvs arps finished
```

To stop watching **`/var/log/messages`**, type **`Ctrl`**+**`c`**.

From this point on, the primary LVS router is also the active LVS router. While you can make requests to LVS at this point, you should start the backup LVS router before putting LVS into service. To do this, simply repeat the process described above on the backup LVS router node.

After completing this final step, LVS will be up and running.

Appendix A. Using LVS with Red Hat Cluster

You can use LVS routers with a Red Hat Cluster to deploy a high-availability e-commerce site that provides load balancing, data integrity, and application availability.

The configuration in *Figure A.1, “LVS with a Red Hat Cluster”* represents an e-commerce site used for online merchandise ordering through a URL. Client requests to the URL pass through the firewall to the active LVS load-balancing router, which then forwards the requests to one of the Web servers. The Red Hat Cluster nodes serve dynamic data to the Web servers, which forward the data to the requesting client.

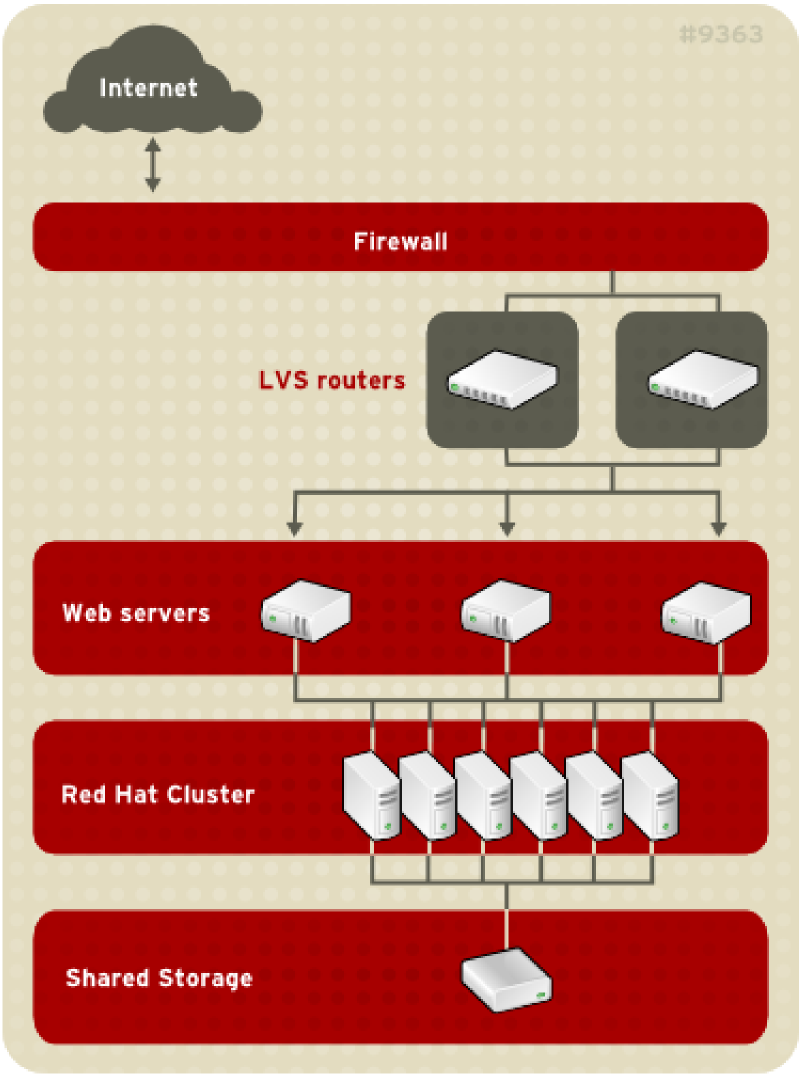


Figure A.1. LVS with a Red Hat Cluster

Serving dynamic Web content with LVS requires a three-tier configuration (as shown in *Figure A.1, “LVS with a Red Hat Cluster”*). This combination of LVS and Red Hat Cluster allows for the configuration of a high-integrity, no-single-point-of-failure e-commerce site. The Red Hat Cluster can run a high-availability instance of a database or a set of databases that are network-accessible to the Web servers.

A three-tier configuration is required to provide dynamic content. While a two-tier LVS configuration is suitable if the Web servers serve only static Web content (consisting of small amounts of infrequently changing data), a two-tier configuration is not suitable if the Web servers serve dynamic content. Dynamic content could include product inventory, purchase orders, or customer databases, which must be consistent on all the Web servers to ensure that customers have access to up-to-date and accurate information.

Each tier provides the following functions:

- First tier — LVS routers performing load-balancing to distribute Web requests.
- Second tier — A set of Web servers to serve the requests.
- Third tier — A Red Hat Cluster to serve data to the Web servers.

In an LVS configuration like the one in *Figure A.1, “LVS with a Red Hat Cluster”*, client systems issue requests on the World Wide Web. For security reasons, these requests enter a Web site through a firewall, which can be a Linux system serving in that capacity or a dedicated firewall device. For redundancy, you can configure firewall devices in a failover configuration. Behind the firewall are LVS load-balancing routers, which can be configured in an active-standby mode. The active load-balancing router forwards the requests to the set of Web servers.

Each Web server can independently process an HTTP request from a client and send the response back to the client. LVS enables you to expand a Web site's capacity by adding Web servers behind the LVS routers; the LVS routers perform load balancing across a wider set of Web servers. In addition, if a Web server fails, it can be removed; LVS continues to perform load balancing across a smaller set of Web servers.

Appendix B. Revision History

Revision 1-1   **Mon Feb 08 2010**   **Paul Kennedy** *pkennedy@redhat.com*

Resolves: 492000

Changes -d to -s in arptables "OUT" directive in "Direct Routing and arptables_jf" section.

Revision 1-0   **Tue Jan 20 2009**   **Paul Kennedy** *pkennedy@redhat.com*

Consolidation of point releases

Index