

Using HP Serviceguard for Linux to Provide High Availability for LAMP

December 2005



Executive Summary	2
Introduction	2
Audience	3
Providing high availability to the LAMP stack	3
Background.....	3
Separate LAMP configuration	3
Consolidated LAMP configuration	3
Package Configuration.....	3
Approach	4
High availability for Consolidated LAMP configuration	4
High availability for Separate LAMP configuration	5
Configuring LAMP for high availability.....	5
Separate LAMP configuration	5
Consolidated LAMP configuration	6
Additional configuration steps	9
SELinux Considerations	10
Toolkit Script Modifications.....	11
Conclusion	12
Related Materials	12

Executive Summary

LAMP is an open source web server solution which is both powerful and stable that can be used to create business applications. The acronym LAMP stands for Linux, Apache, MySQL, and PHP/Perl/Python where Apache is the web server, MySQL the database, and PHP/Perl/Python are server side scripting languages for creating dynamic web pages. All these applications running on the open source OS, Linux, constitute the LAMP stack.

Since the applications based on LAMP may be critical to a business, many customers require that the LAMP components be “highly available”. HP Serviceguard for Linux can be used to run these components in a cluster that has no single point of failure so that application services are available despite hardware and software failure. This makes the overall business application “Highly Available”.

This whitepaper describes how to use the existing HP Serviceguard for Linux contributed toolkits to create a highly available and stable LAMP stack. This covers two configuration options:

- A Consolidated LAMP Configuration where both the web server and the database always run on the same server
- A Separate LAMP Configuration where the web server and the database run on separate servers.

Introduction

With the maturity of the Linux Operating System, organizations today are looking to deploy Linux for tasks that are more complex and more critical to their functioning. These include system and network management, databases, project management tools, and enterprise web applications. One such enterprise web application is the LAMP stack.

The individual components of the LAMP stack have been around for many years. But, the collection of components is being increasingly viewed by vendors and customers as a unified platform for building and running business applications. Such deployments require that the applications be highly available and not be affected by a single point of failure. The solution to this is to cluster multiple Linux servers together to build a highly available environment.

HP Serviceguard for Linux is used to form a cluster of Linux servers and makes application services available despite hardware or software failures or the planned downtime required for maintenance or system upgrades. HP Serviceguard for Linux monitors the LAMP stack and can respond to single or, sometimes, multiple failures within the cluster. For example, when Serviceguard detects the failure of a node, it fails over applications running on that node to other nodes in the cluster.

The existing Apache and MySQL toolkits for HP Serviceguard on Linux have been leveraged to provide high availability to the LAMP components. This whitepaper describes the procedure for setting up a highly available LAMP environment by using the toolkits. The configuration instructions listed below are in addition to those in the Apache and MySQL toolkit README's. The instructions in the toolkit README's must be followed

except where explicitly mentioned in this white paper. The instructions in this white paper are based on the tests that were conducted on various configurations.

Audience

This document is for users of HP Serviceguard on Linux who want to take advantage of a complete web based database solution in the form of the LAMP stack running on a cluster which is highly available.

It is assumed that the reader has a general understanding of HP Serviceguard for Linux and the LAMP components and features. Please see <http://www.hp.com/go/sglx> , <http://www.apache.org>, and <http://www.mysql.com/> for detailed information on each solution. It will be necessary to download the toolkits to get the READMEs. A link is available at the end of this document.

Providing high availability to the LAMP stack

Background

The LAMP stack can be deployed in various configurations. In one common configuration the web front end runs on servers which are tuned to handle higher network loads and the database runs on separate servers tuned for faster disk I/O. This separation allows for overall better performance. In another common configuration the web server and the database are consolidated on the same server. To provide high availability to the former setup, the web service and the database service is packaged separately while the latter requires that the consolidated LAMP application services be packaged as a single Serviceguard for Linux package. The table below lists the differences between the two configurations.

	Separate LAMP configuration	Consolidated LAMP configuration
Package Configuration	Simple and straightforward with no modifications of existing toolkits	Increased flexibility increases complexity. User must modify existing toolkit templates.
Package Administration	More challenging with different packages running on different nodes	Simple administration with both services run as part of the same package on the same node
Tuning	Nodes can be tuned to handle specific workloads thereby improving performance	Since both, front-end and database run on the same node, there are fewer options available for workload performance tuning.
OS & HW Upgrades	Rolling upgrades of the database and the web server can be done independently and one after another	Both the database and web service have to be moved to an alternate node before upgrades can be carried out
Security	The database servers and the web servers can be placed in different security zones. This would however require two different clusters separated by a firewall	Since the database and the web server run on the same node, protection via SELinux can be used.

Approach

The existing Toolkits for Apache and MySQL have been leveraged to package the LAMP setup with HP Serviceguard for Linux. The Apache and MySQL toolkits for HP Serviceguard for Linux consist of sets of shell scripts that startup, shutdown and monitor the Apache web server and the MySQL database applications. These two toolkits can be used to simplify the packaging of Apache and MySQL with HP Serviceguard for Linux. In the LAMP "Separate packages" configuration, these toolkits are employed as is. For the single package configuration however, one package is created that is common to both Apache and MySQL. This common package control script employs components from both the Apache and MySQL toolkits to start, monitor, and stop the respective services.

High availability for Consolidated LAMP configuration

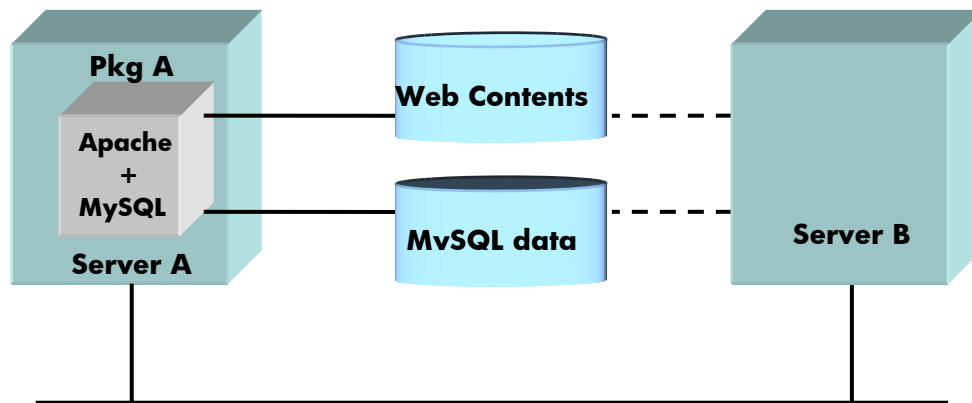


Figure 1: Apache and MySQL running as part of the same package in a Consolidated LAMP configuration

In the Consolidated LAMP configuration, MySQL database and the Apache Web Server (with PHP) always run on the same server node in the cluster, where Serviceguard continually monitors the health of these applications and also the node on which these applications run. On detecting a failure of any application or in the case of a node failure, both applications fail over to the alternate (adoptive) machine.

Here, Apache with PHP enabled and MySQL run on a single machine as a single Serviceguard for Linux package. If either one or both of these applications fails or if the node goes down, then both the applications switch to an alternate node. Such a configuration is the easiest to control.

High availability for Separate LAMP configuration

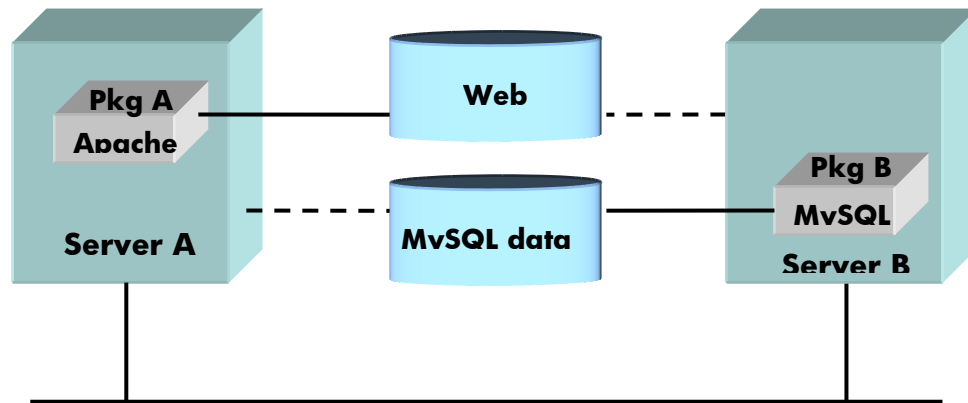


Figure 2: Apache and MySQL running as separate packages in a Separate LAMP configuration

In this configuration, the MySQL database and Apache Web server (with PHP) can run on separate machines. Failure of one application or node results in a restart of only that application on its adoptive node.

Here Apache and MySQL would run as separate Serviceguard packages. This would mean that with this configuration in a production environment, one can do rolling upgrades of the database or web server independently. This configuration offers more flexibility.

Configuring LAMP for high availability

This section details the procedure for deploying the LAMP stack in a high availability environment. The setup described below makes a few assumptions. It assumes that Serviceguard for Linux is installed on all the nodes of the cluster and that both Apache and MySQL as well as their toolkit binaries are installed on all target nodes.

Separate LAMP configuration

Apache and MySQL in this configuration will run as separate Serviceguard for Linux packages. Both Apache and MySQL will have a separate set of package configuration and package control files.

Please follow the instructions in the Apache and MySQL toolkit READMEs and the steps listed in the section "Additional Configuration Steps" to install and configure Apache and MySQL as separate Serviceguard for Linux packages. In addition, the procedures described in the sections "SELinux Considerations" and "Toolkit Script Modifications" of this whitepaper must be completed before the packages can be run.

The PHP pages served by the Apache web server reside in Apache's Document Root directory. If the Document Root is local then synchronization across servers should be maintained by replicating the PHP files on all target nodes. In case of a shared Document root, synchronization across servers is automatically taken care of.

Consolidated LAMP configuration

In this configuration, since Apache and MySQL will run as a single Serviceguard for Linux package, only a single package configuration file and a single package control script need to be created. This package control script would make use of the toolkit scripts of both Apache and MySQL to start, monitor, and stop the respective services.

To configure Apache and MySQL as a consolidated LAMP package, follow all instructions in the Apache and MySQL toolkit README files except for the instructions which describe the creation and configuration of the package configuration file and the package control script for both Apache and MySQL (Section "C" in the Apache toolkit README and Section "D" in the MySQL toolkit README). These instructions in the Apache and MySQL README's are replaced by the procedure for configuring Apache and MySQL as part of the same Serviceguard for Linux package as explained below.

The following procedures provide an EXAMPLE for configuring a Serviceguard for Linux LAMP single package. In this example the package is named "*lamp_single*" and has a service each for Apache and MySQL. The Apache service is named "*http.monitor*" and the MySQL service is called "*mysql.monitor*". The package configuration file and the package control script are named "*lamp_single_pkg.config*" and "*lamp_single_pkg.cntl*" respectively.

In the EXAMPLE, the Apache Document Root is on a shared file system and the Server Root is configured locally. For MySQL however, both configuration files and the database are on a shared file system.

The Apache Document Root is on a file system "*/Apache*" directory, which resides on the logical volume "*lv01*" in the "*/dev/vg01*" volume group on shared storage. The MySQL configuration and database files are on a file system named "*/MySQL_1*" on "*lv01*" in the "*/dev/vg02*" volume group.

The volume structure chosen above is for the set of instructions described in this section. The user may choose another structure which is relevant to his setup. Please refer to the toolkit README's for a description of the possible volume structure configurations.

The first step before starting the package configuration is to create a directory for both Apache and MySQL. (E.g. */usr/local/cmcluster/conf/http_pkg*; */usr/local/cmcluster/conf/mysql_pkg*) and copy all the Apache and MySQL toolkit scripts from "*/usr/local/cmcluster/apachetoolkit*" and "*/usr/local/cmcluster/mysqltoolkit*" to the respective Apache and MySQL directories which were just created.

NOTE: the directory structures shown in this white paper are for RedHat installations. For SuSE installations replace "*/usr/local*" with "*/opt*". Check the documentation for the toolkits, Apache, and MySQL to determine any other directory differences.

The package configuration file (e.g. *lamp_single_pkg.config*) and the package control script (e.g. *lamp_single_pkg.cntl*) templates must be created in the Apache package directory (the reason for keeping the files in the Apache package directory is explained later in the document).

To create the package configuration file and the package control script file:

```
# cmmakepkg -p lamp_single_pkg.config  
# cmmakepkg -s lamp_single_pkg.cntl
```

Edit the package configuration file (*lamp_single_pkg.config*) and change the `PACKAGE_NAME` and `NODE_NAME` fields according to the comments given in that file template. The next set of editable fields in the package configuration file defines the path to the package control file. Enter the path to the package control script for both the `RUN_SCRIPT` and `HALT_SCRIPT`. In the package configuration file used in the development of this white paper, these entries look as follows.

```
PACKAGE_NAME    lamp_single
NODE_NAME       node1
NODE_NAME       node2

RUN_SCRIPT      /usr/local/cmcluster/conf/http_pkg/lamp_single_pkg.cntl
RUN_SCRIPT_TIMEOUT    NO_TIMEOUT
HALT_SCRIPT     /usr/local/cmcluster/conf/http_pkg/lamp_single_pkg.cntl
HALT_SCRIPT_TIMEOUT    NO_TIMEOUT
```

The following sets of fields define the service variables required to start monitor services for both Apache and MySQL. Since in this configuration both Apache and MySQL run as a consolidated stack both MySQL and Apache monitor services need to be listed in this file. The sample field entries are shown below.

```
SERVICE_NAME                http.monitor
SERVICE_FAIL_FAST_ENABLED   NO
SERVICE_HALT_TIMEOUT        300

SERVICE_NAME                mysql.monitor
SERVICE_FAIL_FAST_ENABLED   NO
SERVICE_HALT_TIMEOUT        300
```

Next, edit the package control script (*lamp_single_pkg.cntl*). Other scripts from both original toolkits are called from this package control script. In a typical package control script, calls to the *toolkit.sh* file can be made in two ways. One way to do this is to enable the `HA_APP_SERVER` variable. This causes Serviceguard to make a call to the *toolkit.sh* file in the current directory, in this case the Apache package directory. This is one of the reasons for keeping the package configuration file and the package control file in the Apache package directory. Another way to call the *toolkit.sh* is by making explicit calls in the customer defined run and halt command functions. Since explicit paths can be used here, it is not mandatory for the *toolkit.sh* file to reside in the current directory. This feature can be used for calling the *toolkit.sh* for MySQL. The two calling methods shown above can be merged so that both Apache and MySQL *toolkit.sh* files can be called from the same package control script and both services can run as part of the same package.

Various lines in the package control script (*lamp_single_pkg.cntl*) need to change:

```
VGCHANGE="vgchange -a y"

VG[0] ="vg01"
VG[1] ="vg02"
```

Where, `VG[0]` specifies the volume group for the shared Apache DocumentRoot directory and `VG[1]` is the volume group for the MySQL database and configuration files.

```
LV[0]="/dev/vg01/lvol0"; FS[0]="/Apache"; FS_TYPE[0]="ext3";
FS_MOUNT_OPT[0]="-o rw"
```

```
LV[1]="/dev/vg02/lvol10"; FS[1]="/MySQL_1"; FS_TYPE[1]="ext3";
FS_MOUNT_OPT[1]="-o rw"
```

LV[0] is the logical volume of Apache's mount point and FS[0] is the mount point for the shared Apache directory, whereas LV[1] is the logical volume of MySQL's mount point and FS[1] is the mount point for the shared MySQL directory.

```
IP[0]="15.70.191.61"
SUBNET[0]="15.70.191.0"
```

IP[0] is the virtual IP for the package `lamp_single`

```
HA_APP_SERVER="post-IP"
```

Enabling this variable initiates a call to the `toolkit.sh` file in the current directory. In this case it is the Apache `toolkit.sh` file.

```
SERVICE_NAME[0]="mysql.monitor"
SERVICE_CMD[0]="/usr/local/cmcluster/conf/mysql_pkg/toolkit.sh monitor"
SERVICE_RESTART[0]="-r 0"
```

The three variables listed above must be configured in order to invoke the MySQL monitor script. The corresponding Apache service variables that invoke the Apache monitor are not defined here but are set in the `hahttp.conf` file that comes with the toolkit.

Note that the `SERVICE_NAME` entry must be the same as the one used in the package configuration file.

Next, make the changes in the control script that define the customer defined run and halt commands.

```
function customer_defined_run_cmds
{
# ADD customer defined run commands.
: # do nothing instruction, because a function must contain some
command.
/usr/local/cmcluster/conf/mysql_pkg/toolkit.sh start
#test_return 51
}

function customer_defined_halt_cmds
{
# ADD customer defined halt commands.
: # do nothing instruction, because a function must contain some
command.
/usr/local/cmcluster/conf/mysql_pkg/toolkit.sh stop
#test_return 52
}
```

In the customer defined run and halt commands, calls are made to the MySQL `toolkit.sh` file and start and stop parameters are passed to it. The MySQL toolkit requires separate calls to be made to monitor, start and stop the MySQL services. All calls to the Apache toolkit are made by enabling the `HA_APP_SERVER` variable which in turn enables the toolkit calling functions.

Next edit the `hahttp.conf` and `hamysql.conf` files residing in the respective toolkit directories as per the instructions in the respective toolkit `README` files. These files contain the environment variables required to run and monitor the respective services.

A sample *hahttp.conf* file and *hamysql.conf* changes for the above example is shown below. For a complete description of all variables listed in the *hahttp.conf* and *hamysql.conf* files and their sample values, please refer to the respective toolkit README files.

hahttp.conf

```
SERVER_ROOT="/etc/httpd"

PID_FILE="/var/run/httpd.pid"

HTTP_SERVICE_NAME="http.monitor"
HTTP_SERVICE_CMD="/usr/local/cmcluster/conf/http_pkg/hahttp.mon"
HTTP_SERVICE_RESTART="-r 0"
```

The HTTP_SERVICE_NAME entry must be the same as the SERVICE_NAME used in the package configuration file.

hamysql.conf

```
CONFIGURATION_FILE_PATH="/MySQL_1/my.cnf"
PID_FILE="/var/run/mysqld/mysqld.pid"
```

Now, copy the package directories to all nodes in the cluster running the respective packages. All the planned nodes should have identical file paths for all package files. Apply the Serviceguard for Linux package configuration using the "*cmapplyconf*" command with the "*P*" flag.

```
# cmapplyconf -P lamp_single_pkg.conf (After the cluster is configured)
```

In addition to all the preceding steps edit the Apache (*httpd.conf*) and MySQL (*my.cnf*) configuration files according to your particular setup. Follow the instructions in the toolkit README's for editing these files. In the Apache toolkit README these instructions are listed under the section "Setting up the Apache Web Server". The MySQL toolkit README lists these instructions under the section "Setting up MySQL with the Toolkit".

This is further explained under the section Additional configurations. After this step is completed and the SELinux Considerations are taken care of, the package would be ready to run.

Additional configuration steps

In addition to the steps mentioned above there are a few more steps that have to be completed before the packages can be run. These steps are explained in this section and apply to either case.

The files Apache configuration file "*httpd.conf*" which exists in the Apache Server Root directory and MySQL configuration file "*my.cnf*" need to be edited in accordance to the shared configuration of the user's setup. For the example LAMP configuration described above *httpd.conf* and *my.cnf* can have the following entries. Since the Apache Server Root directory is local in the example configuration, the file *httpd.conf* must be replicated across servers.

httpd.conf

```
ServerRoot "/etc/httpd"

PidFile run/httpd.pid
```

```
Listen 15.70.191.61:80
```

```
DocumentRoot "/Apache"
```

```
# This should be changed to whatever you set DocumentRoot to.  
#
```

```
<Directory "/Apache">  
    Options Indexes FollowSymLinks  
    AllowOverride None  
    Order allow, deny  
    Allow from all  
</Directory>
```

my.cnf

```
[mysqld]  
datadir=/MySQL_1/mysql  
socket=/MySQL_1/mysql/mysql.sock  
port=3306  
# Default to using old password format for compatibility with mysql 3.x  
# clients (those using the mysqlclient10 compatibility package).  
# old_passwords=1
```

```
[mysql.server]  
user=mysql  
basedir=/MySQL_1
```

```
[mysqld_safe]  
err-log=/MySQL_1/mysql/mysqld.err  
pid-file=/var/run/mysqld/mysqld.pid
```

```
[client]  
socket=/MySQL_1/mysql/mysql.sock
```

SELinux Considerations

All the Apache and MySQL files and directories are protected by SELinux security policy. This is turned on by default in RHEL 4. All of these files need to be associated with a security context for them to work properly. SELinux protection allows processes access to only those files, which are set with the certain context. All the default directories of both Apache and MySQL have the right contexts associated with all its files. If any changes are made, such as moving the DocumentRoot of Apache or the MySQL database from local to a shared storage, then these new directories and all of the files in the directory need to be set to the same file context as the default installation.

In the case of multiple Apache and MySQL instances running, only one instance can use the default directory. All of the instances should bear the same security context for all its associated files and directories. It does not matter if these files reside on the local disk or the shared storage. SELinux will allow access to these files only after the right context is set. Configuring the context is mandatory if anything other than the default file locations of Apache and MySQL are used. Users should refer SELinux documents for detailed information on how to configure SELinux with these services.

Toolkit Script Modifications

The “*hamysql.sh*” also known as Toolkit Main Script contains the internal functions that support start/stop of a MySQL instance. While stopping the MySQL instance, this script terminates the process first and immediately afterwards unmounts the shared volume. Unmount occurs cleanly only if the instance is properly terminated. The time required to stop the MySQL instance however varies with the size of the database. To account for this variable termination duration, minor modifications are needed in the “*hamysql.sh*” script.

```
function stop_mysql_server
{
    echo "$(date '+%b %e %T') - Node \"$(hostname)\": Stopping MySQL
daemons"

    if [ -f $PID_FILE ]; then
        read pid < $PID_FILE
    else
        echo "$(date '+%b %e %T') - ERROR: Function stop_mysql_server:
The $PID_FILE does not exist."
        check_return 1 4
    fi
    if [ -n $pid ]; then
        kill -15 $pid
        return_value=$?
        if [ $return_value = 0 ]; then
            while [ 1 ]
            do
                if [ -d /proc/$pid ];then
                    sleep 1
                else
                    break
                fi
            done
            fi
            check_return $return_value 4
        else
            echo "$(date '+%b %e %T') - ERROR: Function stop_mysql_server:
The $PID_FILE was corrupted."
            check_return 1 4
        fi
    }
}
```

Since the SELinux security issues are taken care of by setting the correct security contexts as described in the section “SELinux Considerations”, the following lines from the *hamysql.sh* script should be removed.

```
# Bring SELinux to Permissive mode for MySQL Startup.
#setenforce 0
```

```
# Changed mode of SELinux to Enforcing Mode.
#setenforce 1 ;
```

The above two modifications to the toolkit script - *hamysql.sh* apply to both the Separate Package and the Single Package configurations. These changes must be made before running the LAMP package.

Conclusion

HP Serviceguard for Linux provides high availability to the LAMP stack in business critical environments. By following the recommended procedures one can get a highly available LAMP stack in an environment where both the database and the web server run in a consolidated environment on a single server and also in an environment where the web server and the database run on separate servers.

This white paper has given guidelines on two possible configurations. Variations are possible such as having Apache along with its Server Root and the Document Root on the shared file system. Typical configurations for Apache and MySQL are described in the respective toolkit README's. The choice of local or shared configuration largely depends on the cluster manageability and shared storage accessibility.

Related Materials

- Release notes, administration guide, Support matrices and White papers for Serviceguard for Linux are available at <http://docs.hp.com/en/ha.html>
 - All references on Serviceguard for Linux – *“Getting Started with HP Serviceguard for Linux”* and *“Managing HP Serviceguard for Linux”*.
- SELinux protection policy and configuration
<http://fedora.redhat.com/docs/selinux-apache-fc3/sn-getting-started.html>
- All references on MySQL - <http://www.mysql.com/>
- All references on Apache - <http://www.apache.org/>
- Download location for toolkits – <http://software.hp.com/go/softwaredepot/ha> - then select *“Serviceguard for Linux Contributed Toolkit Suite”*