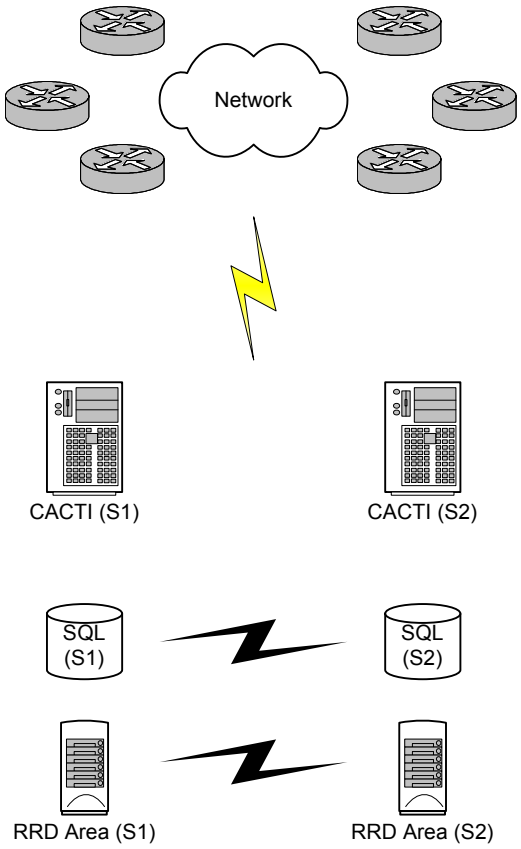
High Availability for Cacti

Author: Fausto Vetter @ DANTE / GÉANT (www.dante.net / www.geant.net)

This document describes a deployment of CACTI on high availability and load balancing mode. CACTI will be sharing the monitoring activity among the active nodes on the monitoring cluster. This setup is recommended for environments either requiring a high level of availability or when the number of monitored devices is high. In this document, two servers located on distinct networks were used. The final setup looks like the picture below:



The monitoring environment consist of a synchronized shared hard disk space, a master-master MySQL implementation and two instances of Cacti running on two separated Apache2 servers. In the end, the user has the feeling of running Cacti on its normal way, but gets the benefit of sharing the load and possibility to scale the monitoring as the environment grows.

Requirements

Hardware requirements:

- Two physical servers

Software requirements:

- Linux Operating System
- MySQL 5.x
- Unison
- SSH
- Apache2 Server
- RRDTool
- PHP 5
- Cacti 0.8.7e (patched)
- Cacti 0.8.7e plugin architecture
- Cacti 0.8.7e cluster architecture

Important Note: All the steps of this procedure should be undertaken using a **normal** user account and were accomplished under RedHat Enterprise (RHEL) 4 and 5. Whenever relevant on this procedure, both systems will be distinctly identified. This procedure can serve as the basis for other Linux implementations.

Installation Procedure

Linux Operating system installation is out of the scope of this document. It is expected that both systems are configured and running.

Requirements Installation

SSH Tunnel

This procedure should be undertaken on both servers. It is expected that SSH Server is pre-installed on both servers and allowing authentication using keys. Follow the next steps to configure the SSH tunnel between both servers:

1) Go to user's home folder: **cd ~/**
2) Create .keys folder: **mkdir .keys**

3) Create a password-free key pair running **ssh-keygen -t dsa** and storing them in the following path **~/.keys/cacti-[other_host]**:

```
Generating public/private dsa key pair.
Enter file in which to save the key (/home/[username]/.ssh/id_dsa):
/home/[username]/.keys/cacti-[other-hostname]
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /home/[username]/.keys/cacti-
[hostname].
Your public key has been saved in /home/[username]/.keys/cacti-
[hostname].pub.
The key fingerprint is:
de:c8:71:2b:52:cc:49:f0:8a:34:6b:30:d9:76:fa:ff
```

4) Add the content of the **~/.keys/cacti-[hostname].pub** to the **~/.ssh/authorized_keys** file of other host
5) Test the connection between the hosts adding the other host to the known hosts list:
   /usr/bin/ssh -i ~/.keys/cacti-[hostname] [username]@[other_hostname]

Unison

Unison is the software package used to synchronize folders among both servers and simulate a shared storage for the RRD files generated by Cacti. It was developed using OCAML interpreter. This tool is similar to the well-known RSync, keeping the files updated based on files' differences. The big difference to RSync is that it detects and merges changes done on both directions, keeping both sides updated to the latest files versions from either server.

OCAML Interpreter

This procedure should be undertaken on both servers. Follow the next steps to compile the OCAML interpreter:

1) Uncompress the downloaded file: **tar xvfz ocaml-3.11.1.tar.gz**
2) Move to the created folder: **cd ocaml-3.11.1**
3) Prepare the compilation process: **./configure**
4) Compile the interpreter: **make world**
5) Compile the interpreter for the hardware platform: **make opt**

6) Install the interpreter: **sudo make install**

Unison Synchronization Software

This procedure should be undertaken on both servers. Follow the next steps to compile Unison:

1) Uncompress the downloaded file: **tar xvfz unison-2.32.52.tar.gz**
2) Go to the created folder: **cd unison-2.32.52**
3) Compile Unison: **make UISTYLE=text**
4) Copy the generated binary file to a PATH folder **sudo cp ./unison /usr/bin**
5) Test Unison Connection by running the following commands (before testing, create a testing folder on both servers such as ~/unison-test):

```
unison /home/[username]/unison-test --sshargs '[path-to-key]'
ssh://[hostname-other-server]//home/[username]/unison-test -batch
```

MySQL

Note: This procedure was followed only on RHEL 4 system, since RHEL5 had MySQL5 installed. Basically, to install MySQL 5 under RHEL 5, the user can use the yum package repository available from RedHat.

1) Verify if any mysql older packages are currently installed: **rpm -qa | grep -i '^mysql-'**
2) Remove mysql database folder: **sudo rm -rf /var/lib/mysql/mysql/**
3) Remove all existing mysql packages… **--nodeps** option might be required
4) Install the shared libraries: **sudo rpm -ivh MySQL-shared-community-5.1.40-0.rhel4.x86_64.rpm**
5) Install the compatibility shared libraries: **sudo rpm -Uvh MySQL-shared-compat-5.1.40-0.rhel4.x86_64.rpm**
6) Install MySQL5 server: **sudo rpm -ivh MySQL-server-community-5.1.40-0.rhel4.x86_64.rpm**
7) Install MySQL5 client: **sudo rpm -ivh MySQL-client-community-5.1.40-0.rhel4.x86_64.rpm**
8) Install MySQL5 developer library: **sudo rpm -ivh MySQL-devel-community-5.1.40-0.rhel4.x86_64.rpm**
9) Add MySQL on the service list: **sudo /sbin/chkconfig -add mysql**

10) Change the startup configuration of MySQL: **sudo /sbin/chkconfig –level 2345 mysql on**
11) Set port **3307** on **/etc/my.cnf** (**port = 3307**)
12) Set Selinux enforce rules to Permissive (**sudo /usr/sbin/setenforce Permissive**)
13) Set Selinux enforce rules to Permissive on startup of server (edit **/etc/selinux/config** file and change **SELINUX** option to **SELINUX=permissive**)
14) Start MySQL if service wasn't started yet: **sudo /sbin/service mysql start**
15) Change MySQL password: **sudo mysqladmin password [password]**

Note: Since on our RHEL5 server there was already an instance of MySQL running and also doing replication, it was decided to implement a second instance of MySQL running on port 3307.

To configure a new process for running MySQL on port 3307, follow the next steps:

1) Create a new username for the new MySQL: **sudo /usr/sbin/adduser -r -m -d /var/lib/mysql-cacti mysql-cacti**
2) Create the folder MySQL will use to store running files such as the PID file: **sudo mkdir /var/run/mysqld-cacti**
3) Change the ownership of this folder to the created user: **sudo chown mysql-cacti.mysql-cacti /var/run/mysqld-cacti**
4) Create the folder MySQL will use for logging: **sudo mkdir /var/log/mysql-cacti**
5) Change the ownership of this folder to the created user: **sudo chown mysql-cacti.mysql-cacti /var/log/mysql-cacti**
6) Create the initialization script for the new instance creating the following file **/etc/init.d/mysqld-cacti** and adding the following content to it:

```
#!/bin/bash
#
# mysqld       This shell script takes care of starting and stopping
#              the MySQL subsystem (mysqld).
#
# chkconfig: - 64 36
# description: MySQL database server.
```

```
# processname: mysqld
# config: /etc/my-$prefix.cnf
# pidfile: /var/run/mysqld-$prefix/mysqld.pid

# Source function library.
. /etc/rc.d/init.d/functions

# Source networking configuration.
. /etc/sysconfig/network


prog="MySQL"
prefix="cacti"
config_file=/etc/my-$prefix.cnf


# extract value of a MySQL option from config files
# Usage: get_mysql_option SECTION VARNAME DEFAULT
# result is returned in $result
# We use my_print_defaults which prints all options from multiple
files,
# with the more specific ones later; hence take the last match.
get_mysql_option(){
        result=`/usr/bin/my_print_defaults --defaults-
file="$config_file" "$1" | sed -n "s/^--$2=//p" | tail -n 1`
        if [ -z "$result" ]; then
            # not found, use default
            result="$3"
        fi
}

get_mysql_option mysqld datadir "/var/lib/mysql-$prefix"
datadir="$result"
get_mysql_option mysqld socket "$datadir/mysql.sock"
socketfile="$result"
get_mysql_option mysqld_safe log-error "/var/log/mysqld-$prefix.log"
errlogfile="$result"
get_mysql_option mysqld_safe pid-file "/var/run/mysqld-
$prefix/mysqld.pid"
mypidfile="$result"
```

```
start(){
        touch "$errlogfile"
        chown mysql-$prefix:mysql-$prefix "$errlogfile"
        chmod 0640 "$errlogfile"
        [ -x /sbin/restorecon ] && /sbin/restorecon "$errlogfile"
        if [ ! -d "$datadir/mysql" ] ; then
            action $"Initializing MySQL database: "
/usr/bin/mysql_install_db --defaults-file="$config_file" --
datadir="$datadir" --user=mysql-$prefix
            ret=$?
            chown -R mysql-$prefix:mysql-$prefix "$datadir"
            if [ $ret -ne 0 ] ; then
                return $ret
            fi
        fi
        chown mysql-$prefix:mysql-$prefix "$datadir"
        chmod 0755 "$datadir"
        # Pass all the options determined above, to ensure consistent
behavior.
        # In many cases mysqld_safe would arrive at the same
conclusions anyway
        # but we need to be sure.
        /usr/bin/mysqld_safe   --defaults-file="$config_file" --
datadir="$datadir" --socket="$socketfile" \
                --log-error="$errlogfile" --pid-file="$mypidfile" \
                --user=mysql-$prefix >/dev/null 2>&1 &
        ret=$?
        # Spin for a maximum of N seconds waiting for the server to
come up.
        # Rather than assuming we know a valid username, accept an
"access
        # denied" response as meaning the server is functioning.
        if [ $ret -eq 0 ]; then
            STARTTIMEOUT=30
            while [ $STARTTIMEOUT -gt 0 ]; do
                RESPONSE=`/usr/bin/mysqladmin --defaults-
file="$config_file" --socket="$socketfile" --user=UNKNOWN_MYSQL_USER
ping 2>&1` && break
                echo "$RESPONSE" | grep -q "Access denied for user"
&& break
                sleep 1
```

```
                let STARTTIMEOUT=${STARTTIMEOUT}-1
            done
            if [ $STARTTIMEOUT -eq 0 ]; then
                    echo "Timeout error occurred trying to start
MySQL Daemon."
                    action $"Starting $prog: " /bin/false
                    ret=1
            else
                    action $"Starting $prog: " /bin/true
            fi
        else
            action $"Starting $prog: " /bin/false
        fi
        [ $ret -eq 0 ] && touch /var/lock/subsys/mysqld-$prefix
        return $ret
}

stop(){
        MYSQLPID=`cat "$mypidfile"  2>/dev/null `
        if [ -n "$MYSQLPID" ]; then
            /bin/kill "$MYSQLPID" >/dev/null 2>&1
            ret=$?
            if [ $ret -eq 0 ]; then
                STOPTIMEOUT=60
                while [ $STOPTIMEOUT -gt 0 ]; do
                    /bin/kill -0 "$MYSQLPID" >/dev/null 2>&1 || break
                    sleep 1
                    let STOPTIMEOUT=${STOPTIMEOUT}-1
                done
                if [ $STOPTIMEOUT -eq 0 ]; then
                    echo "Timeout error occurred trying to stop MySQL
Daemon."
                    ret=1
                    action $"Stopping $prog: " /bin/false
                else
                    rm -f /var/lock/subsys/mysqld-$prefix
                    rm -f "$socketfile"
                    action $"Stopping $prog: " /bin/true
                fi
            else
                action $"Stopping $prog: " /bin/false
```

```
            fi
        else
            ret=1
            action $"Stopping $prog: " /bin/false
        fi
        return $ret
}

restart(){
    stop
    start
}

condrestart(){
    [ -e /var/lock/subsys/mysqld-$prefix ] && restart || :
}

# See how we were called.
case "$1" in
  start)
    start
    ;;
  stop)
    stop
    ;;
  status)
    status mysqld-$prefix
    ;;
  restart)
    restart
    ;;
  condrestart)
    condrestart
    ;;
  *)
    echo $"Usage: $0 {start|stop|status|condrestart|restart}"
    exit 1
esac

exit $?
```

7) Change the permissions of the file: **sudo chmod 755 /etc/init.d/mysqld-cacti**

8) Create a new configuration file to MySQL (**/etc/my-cacti.cnf**) and add the following content to it:

```
[mysqld]
port=3307
datadir=/var/lib/mysql-cacti
socket=/var/lib/mysql-cacti/mysql.sock
old_passwords=1

[mysqld_safe]
log-error=/var/log/mysqld-cacti.log
pid-file=/var/run/mysqld-cacti/mysqld.pid
timezone = Etc/GMT
```

9) Add the new service to the known service list: **sudo /sbin/chkconfig --add mysqld-cacti**
10) Configure the service to start on boot:
    a. Enter the Setup tool: **sudo setup**
    b. Go to **System services**
    c. Choose **[*] mysqld-cacti** service on the list
    d. Go to **OK**
    e. And exit going to **Quit**
11) Start the service: **sudo /sbin/service mysqld-cacti start**
12) Change the MySQL root user's password for this new instance: **mysqladmin -S /var/lib/mysql-cacti/mysql.sock password [password]**

HTTP Server

Both servers had HTTP Apache 2 service installed. The only procedure was to start both servers: **sudo /sbin/service httpd start**

Also, do not forget to change the firewall changing the file **/etc/sysconfig/iptables** and adding the following rule:

```
-A RH-Firewall-1-INPUT -m state --state NEW -m tcp -p tcp --dport 80
-j ACCEPT
```

Finally, restart the service: **sudo /sbin/service iptables restart**

On the RHEL4, HTTP wasn't configured to start at the system boot. To configure this feature run the following command: **sudo /sbin/chkconfig --level 2345 httpd on**

RRDTool

This procedure should be undertaken on both servers. The procedures for installing RRDTool are:

1) Install required libraries and tools using yum for RHEL5 and rpm –ivh for RHEL4:
    a. GCC compiler: **gcc.x86_64**
    b. PCKConfig tool: **pkgconfig.x86_64**
    c. Libart development library: **libart_lgpl-devel.x86_64**
    d. ZLib development library: **zlib-devel.x86_64**
    e. LibPNG development library: **libpng-devel.x86_64**
    f. Freetype development library: **freetype-devel.x86_64**
2) Go to **/opt** folder: **cd /opt/**
3) Download the latest RRDTool 1.2.27 package: **wget http://oss.oetiker.ch/rrdtool/pub/rrdtool-1.2.27.tar.gz**
4) Uncompress RRDTool package: **tar xvfz rrdtool-1.2.27.tar.gz**
5) Change folder to the recently created RRDTool folder: **cd rrdtool-1.2.27**
6) Create the configuration file to compile RRDTool: **sh configure [--disable-ruby]**
7) Compile RRDTool: **make**
8) Install RRDTool: **sudo make install**
9) Move the binary files to the PATH folder: **sudo cp /usr/local/rrdtool-1.2.27/bin/* /usr/bin**

PHP 5 and Libraries

On RHEL5 system, use yum and install the following packages: **php.x86_64**, **php-mysql.x86_64**, **net-snmp-utils.x86_64** and **php-snmp.x86_64**.

On RHEL4, find PHP and its packages at **http://www.cyberciti.biz/files/lighttpd/rhel4-php5-fastcgi/** and install the following packages:

sudo rpm --nodeps -Uvh php-5.1.4-1.esp1.x86_64.rpm

sudo rpm -Uvh php-ldap-5.1.4-1.esp1.x86_64.rpm

sudo rpm -ivh php-pdo-5.1.4-1.esp1.x86_64.rpm

sudo rpm -ivh php-mysql-5.1.4-1.esp1.x86_64.rpm

On RHEL4, install Net-SNMP using the version 5.1.2-18 using the comand **rpm –ivh [package_name]**: **net-snmp-5.1.2-18.el4.x86_64.rpm**, **net-snmp-devel-5.1.2-18.el4.x86_64.rpm**, **net-snmp-libs-5.1.2-18.el4.x86_64.rpm**, **net-snmp-perl-5.1.2-18.el4.x86_64.rpm**, **net-snmp-utils-5.1.2-18.el4.x86_64.rpm** and **php-snmp-5.1.4-1.esp1.x86_64.rpm**.

Cacti Installation

This procedure should be undertaken on both servers. The procedures for installing Cacti are:

1) Go to **/opt folder**: **cd /opt**
2) Download Cacti: **wget http://www.cacti.net/downloads/cacti-0.8.7e.tar.gz**
3) Uncompress Cacti package: **sudo tar xvfz cacti-0.8.7e.tar.gz**
4) Create a logical link to the folder created: **sudo ln -s /opt/cacti-0.8.7e /opt/cacti**
5) Change user ownership of **/opt/cacti** to username: **sudo chown -R [username].[group] /opt/cacti**
6) Change user ownership of **/opt/cacti-0.8.7e** to username: **sudo chown -R [username].[username] /opt/cacti-0.8.7e**
7) Change folder to **/opt/cacti**: **cd /opt/cacti**
8) Create MySQL database for CACTI: **mysqladmin [-S /var/lib/mysql-cacti/mysql.sock] --user=root -p create cacti**
9) Import CACTI tables to the database: **mysql [-S /var/lib/mysql-cacti/mysql.sock] --user=root -p cacti < cacti.sql**
10) Create Cacti database username and password:
    a. Connect to database: **mysql [-S /var/lib/mysql-cacti/mysql.sock] --user=root -p mysql**
    b. Create username and password: **GRANT ALL ON cacti.* TO cactiuser@localhost IDENTIFIED BY '[password]';**
    c. Refresh the configuration: **FLUSH PRIVILEGES;**
    d. Exit MySQL client: **exit**
11) Edit **include/config.php** file and define the **$database_password** variable to the database password connection defined. E.g.: **$database_password = "password"**

12) Edit user crontab table running crontab -e adding the following line to it:

```
*/5 * * * * /usr/bin/php /opt/cacti/poller.php > /dev/null 2>&1
```

13) Move CACTI log folder and create the logrotate file:

a. Move CACTI logging folder to system log folder: **sudo mv /opt/cacti/log /var/log/cacti**

b. Create a symbolic link pointing to CACTI system logging folder in place of CACTI logging folder: **sudo ln -s /var/log/cacti /opt/cacti/log**

c. Change permissions on the logs symbolic link: **sudo chown -h -R [username].[group] /opt/cacti/log**

d. Give permissions to username on the logging folder: **sudo chown -R [username].[group] /var/log/cacti**

e. Create logrotate configuration file (**sudo vi /etc/logrotate.d/cacti**) file adding the following content to it:

```
/var/log/cacti/cacti.log {
        missingok
                compress
         notifempty
                size 10M
        postrotate
                endscript
                rotate 5
                create 0600 sd sd
}
```

14) Create CACTI alias on Apache by creating **sudo vi /etc/httpd/conf.d/cacti.conf** file and adding the following content to it:

```
Alias /cacti "/opt/cacti"

<Directory "/opt/cacti">
    Options Indexes MultiViews
    AllowOverride None
    Order deny,allow
    Allow from all
    Deny from all
</Directory>
```

15) Restart your web-server running the following command: **sudo /sbin/service httpd restart**

16) Point your web-browser to **http://A.B.C.D/cacti/**, where A.B.C.D is the server address;
17) Follow the installation screens, observing the following:
    a. Choose **Installation Type** as **New Install**
    b. Make sure CACTI find all required applications on the path
    c. After finishing installation, access CACTI using username admin and password admin, then change the admin password as you wish

Preparing Cacti to Run under Cluster Mode

The first desirable step is to make sure Cacti can run under Cluster Mode. Since any Cacti comes with the device **localhost** registered, so it is recommended that the user removes this instance on both servers to allow misinterpretation while running on cluster mode. Follow the next steps on both servers to prepare Cacti:

1) Connect as **admin** in the web interface of Cacti
2) Under **Management** tab, go to **Devices**
3) Check **localhost** device
4) Select **Delete** on **Choose an action:** and press **OK**
5) Confirm that you want to delete this device and you may logout the web interface for the moment
6) Connect the server using SSH using username running Cacti
7) Go to **/opt/cacti/rra**: **cd /opt/cacti/rra**
8) Delete all RRD files: **rm *.rrd**

Configure Synchronization of Cacti RRA Folder

Follow these steps on both servers to configure Unison to synchronize:

1) Create the script **~/scripts/sync.sh** adding the following content:

```
#!/bin/bash

/bin/sleep [time]
#/usr/bin/unison /var/lib/php/session -sshargs '-i
/home/[username]/.keys/cacti-[other_hostname]'
ssh://[other_hostname]//var/lib/php/session -batch
/usr/bin/unison /opt/cacti/rra -sshargs '-i
/home/[username]/.keys/cacti-[other_hostname]'
ssh://[other_hostname]//opt/cacti/rra -batch -times -force newer
```

* The only thing different beside [username] and [other_hostname] is [time], which on the RHEL5 was defined as 1m (1 minute) and on the RHEL4 as 3m (3 minutes)

2) Change the permissions on the file: **chmod 755 ~/scripts/sync.sh**;
3) Edit crontab (**crontab -e**) adding the following:

```
*/5 * * * * /home/[username]/scripts/sync.sh > /dev/null 2>&1
```

PS: **/var/lib/php/session** in this instance was commented out once the machines are not running on active failover, so they do not need to share the session information of the Apache server. If this is the case for another implementation, just uncomment this like on the user crontab.

Configure MySQL to run on Master-Master Mode

Once both Cacti systems are setup, running and sharing the same RRA folders, now is time to configure MySQL to run on Master-Master mode. Follow the next steps to configure the Master-Master mode:

1) Connect on both servers and create the replication user for the MySQL:
    a. Connect to the MySQL: **mysql [-S /var/lib/mysql-cacti/mysql.sock] --user=root -p**
    b. Run the following query on both servers:

```
GRANT REPLICATION SLAVE ON *.* TO 'slave'@'[other_host_ip]'
IDENTIFIED BY '[slavepass]';
FLUSH PRIVILEGES;
quit;
```

Change firewall settings to ensure each MySQL server can connect to the other one by adding to iptables configuration file (**/etc/sysconfig/iptables**) the following content before the firewall closure statement which rejects all the traffic (*-A RH-Firewall-1-INPUT -j REJECT --reject-with icmp-host-prohibited COMMIT*):

```
-A RH-Firewall-1-INPUT -p tcp -m tcp -s [other_host_ip] --dport 3307
-j ACCEPT
```

2) Restart IP Tables (**sudo /sbin/service iptables** restart or **/etc/init.d/iptables restart**) on both servers
3) Test the firewall by running a telnet to port 3307: **telnet [other_hostname] 3307**
4) Create the MySQL logging folder for both systems:
    a. For RHEL4, create the folder MySQL (**sudo mkdir /var/log/mysql**) and change its ownership (**sudo chown mysql.mysql /var/log/mysql**)

b. For RHEL5, create the folder MySQL (**sudo mkdir /var/log/mysql-cacti**) and change its ownership (**sudo chown mysql-cacti.mysql-cacti /var/log/mysql-cacti**)

5) Change the MySQL configuration file of the RHEL5 system (**/etc/my-cacti.cnf**) and make sure it has the following content:

```
[mysqld]
server-id = 10
port=3307
replicate-same-server-id = 0
auto-increment-increment = 2
auto-increment-offset = 2
#master-host = [other_hostname]
#master-port = 3307
#master-user = slave
#master-password = [master_password]
#master-connect-retry = 60
replicate-do-db = cacti
log-bin = /var/log/mysql-cacti/mysql-bin.log
binlog-do-db = cacti
relay-log = /var/lib/mysql-cacti/slave-relay.log
relay-log-index = /var/lib/mysql-cacti/slave-relay-log.index
expire_logs_days = 10
max_binlog_size = 500M
datadir=/var/lib/mysql-cacti
socket=/var/lib/mysql-cacti/mysql.sock
old_passwords=1

[mysqld_safe]
log-error=/var/log/mysqld-cacti.log
pid-file=/var/run/mysqld-cacti/mysqld.pid
timezone = Etc/GMT
```

6) Restart MySQL: **sudo /sbin/service mysqld-cacti restart**

7) Change the MySQL configuration file of the RHEL4 system (**/etc/my.cnf**) and make sure it has the following content:

```
[mysqld]
server-id=20
port = 3307
datadir=/var/lib/mysql
socket=/var/lib/mysql/mysql.sock
```

```
old_passwords=1
master-host = [other_hostname]
master-user = slave
master-port = 3307
master-password = [master_password]

replicate-same-server-id = 0
auto-increment-increment = 2
auto-increment-offset = 1
replicate-do-db = cacti
log-bin = /var/log/mysql/mysql-bin.log
binlog-do-db = cacti
relay-log = /var/lib/mysql/slave-relay.log
relay-log-index = /var/lib/mysql/slave-relay-log.index
expire_logs_days = 10
max_binlog_size = 500M

[mysqld_safe]
log-error=/var/log/mysqld.log
pid-file=/var/run/mysqld/mysqld.pid
timezone = Etc/GMT
```

8) Restart MySQL: **sudo /sbin/service mysqld restart**
9) Import RHEL5 database into RHEL4 database by following the next steps:
    a. Connect to mysql: **mysql [-S /var/lib/mysql-cacti/mysql.sock] --user=root –p**
    b. Lock the tables:

```
USE cacti;
FLUSH TABLES WITH READ LOCK;
quit;
```

    c. Create a dump: **cd /tmp/ && mysqldump -S /var/lib/mysql-cacti/mysql.sock -u root –p --opt cacti > cactidump.sql**
    d. Connect to mysql (**mysql [-S /var/lib/mysql-cacti/mysql.sock] --user=root –p**) and unlock the tables:

```
UNLOCK TABLES;
quit;
```

    e. Transfer the dump to **/tmp** folder on the RHEL4 system;
    f. On RHEL4, import the dump running the following commands:

```
mysqladmin [-S /var/lib/mysql/mysql.sock] --user=root -p stop-slave
cd /tmp/
```

```
mysql [-S /var/lib/mysql/mysql.sock]  -u root -p cacti <
cactidump.sql
```

10) Go to HREL5 system, connect to mysql (**mysql [-S /var/lib/mysql-cacti/mysql.sock] --user=root -p**) and configure to run as a slave:

```
stop slave;
flush logs;
reset slave;
start slave;
```

11) Verify its master status:

```
USE cacti;
FLUSH TABLES WITH READ LOCK;
SHOW MASTER STATUS\G;
```

12) Take note of **File** and **Position** from last command and reset the master on the system:

```
mysql> CHANGE MASTER TO MASTER_HOST='[other_hostname]',
MASTER_USER='[master_user]', MASTER_PASSWORD='[master_pass]',
MASTER_LOG_FILE='[master_log_file]',
MASTER_LOG_POS=[master_log_position];
```

13) On HREL4 system, configure to run as a slave:

```
stop slave;
flush logs;
reset slave;
start slave;
```

14) On both systems, verify that both are running properly running on MySQL the following:

```
SHOW SLAVE STATUS\G;
```

15) Check that both systems have **Slave_IO_Running** and **Slave_SQL_Running** options set to **Yes**

16) Leave MySQL on both systems: **quit**

If everything have gone fine, master-master replication should be working. Check your logs on both systems if you encounter problems.

Cacti Plugin Architecture

This procedure should be undertaken on both servers. The procedures for installing Cacti Plugin Architecture are:

1) Extract the plugin architecture file on **/opt/cacti** folder: **tar -zvxf cacti-plugin-arch.tar.gz**

2) Test the patch: **patch -p1 -N --dry-run < cacti-plugin-arch.diff**
3) If everything goes fine, patch the code: patch **-p1 -N < cacti-plugin-arch.diff**
4) On **[cacti_path]/include/global.php**, change the variable **$config['url_path']** to:

```
$config['url_path'] = "/cacti/";
```

5) On only one of the servers, import the Plugin Architecture database: **mysql [-S /var/lib/mysql-cacti/mysql.sock] --user=root -p cacti < pa.sql**
6) Connect as admin to one of the web-interfaces of Cacti and enable the Plugin Architecture for the admin user:
    a. Under **Utilities** tab, go to **User Management**
    b. Click on **admin** user
    c. Under **Realm Permissions** tab, check **Plugin Management**
    d. Click on **Save** button

Setup Cacti for Cluster Mode

This procedure should be undertaken on both servers. The procedures for preparing Cacti to run on cluster mode are:

1) Make sure Cacti is patch with the following available patches:

```
patch -p1 -N < cli_add_graph.patch
patch -p1 -N < snmp_invalid_response.patch
patch -p1 -N < template_duplication.patch
patch -p1 -N < fix_icmp_on_windows_iis_servers.patch
```

2) Patch Cacti with the cluster infrastructure: **patch -p1 -N < cacti-0.8.7e-clustered.diff**
3) On only one of the servers, connect to MySQL (**mysql [-S /var/lib/mysql-cacti/mysql.sock] --user=root –p**) and run the following commands:

```
CREATE TABLE `cluster_nodes` (
  `uid` mediumint(8) unsigned NOT NULL AUTO_INCREMENT,
  `cluster_status` varchar(32) DEFAULT NULL,
  `uptime` varchar(255) DEFAULT NULL,
  `cluster_order` varchar(3) DEFAULT NULL,
  `hostname` varchar(32) DEFAULT NULL,
  `ipadd` varchar(32) DEFAULT NULL,
  PRIMARY KEY (`uid`)
) ENGINE=MyISAM AUTO_INCREMENT=3 DEFAULT CHARSET=latin1;
```

```
insert into plugin_realms(plugin, file, display)
values("cluster","cluster.php","Cacti Cluster");
```

7) Change permissions on the session folder of Apache to enable your username to have access: **sudo chmod 775 /var/lib/php/session**
8) Add the apache group to your username: **sudo /usr/sbin/usermod -a -G apache [username]**
9) Setup the crontab (**crontab -e**) on both servers for heartbeat check of the cluster:

```
* * * * * /usr/bin/php /opt/cacti/check_cluster.http.php
```

10) Connect as admin to one of the web-interfaces of Cacti and enable the Cluster Mode for the admin user:
    a. Under **Utilities** tab, go to **User Management**
    b. Click on **admin** user
    c. Under **Realm Permissions** tab, check **Cacti Cluster**
    d. Click on **Save** button
11) Add the cluster nodes on the Cacti:
    a. Under **Configuration**, go to **Cluster**
    b. Add the cluster nodes, by setting for each one the **hostname** (it has to be the same string that appear after typing the hostname command on each server) and **IP address**, and pressing the **Add** button
    c. Wait until **cluster_status**, **cron**, **mysql** and **httpd** background colour changes to **green** for both nodes