Server Build

From Matchi Wiki

Contents

Common Server Build

The following series of build steps needs to be performed for each server, regardless of the server's function.

Conventions

- Builds are consistently referred to by their initial inception date in ISO date-format (YYYYMMDD). This is reflected in the names of the directories for source code, static content and databases.
- All account names and server names are in lower case
- All file names are either in lower case with "_" separators between terms, or in Camel-case with the first letter of each term being a capital letter
- Example commands that begin with a '$' are performed by a non-root user, and commands that begin with a '#' are performed by the root user.

First Access

From your terminal (use PuTTY if you are using Windows), open an SSH session to the server using the root account and default password. The password and server details are provided by the hosting provider and will be different for each server.

```
$ ssh root@s470071337.websitehome.co.uk
root@s470071337.websitehome.co.uk's password:
```

Type in the root password and hit Return. Ignore any warnings for now:

```
Warning: untrusted X11 forwarding setup failed: xauth key data not generated
Warning: No xauth data; using fake authentication data for X11 forwarding.
X11 forwarding request failed on channel 0
Last login: Fri May 31 15:42:09 2013 from host86-142-19-185.range86-142.btcentralplus.com
[root@s16972616 ~]#
```

You are now logged in as user 'root', who is the super user on the server.

Since this system does not host individual enterprise user-accounts, we only need one more system administrator user who is a less powerful than the 'root' user to perform basic management tasks. It will be possible to temporarily escalate this user's privileges to perform tasks that only the 'root' user would normally be able to perform (see the section on SUDO). From a security perspective, we will completely disable the ability to log into the server using the 'root' account (because everyone knows that such a user exists!) and allow solely through this **M**atchi system **AD**ministrator **MAN**anger account, **madman**,

```
[root@s16972616 ~]# useradd -g adm madman -c 'MWEB02 System Administrator'
```

Set the internal password for administrator user. If the password is weak, you will be warned, as in this example.

```
[root@s16972616 ~]# passwd madman
Changing password for user madman.
New password:
BAD PASSWORD: it is based on a dictionary word
Retype new password:
passwd: all authentication tokens updated successfully.
[root@s16972616 ~]#
```

Replace Password logins with Key logins

Passwords are by nature unsafe and are far more likely to fall into the wrong hands than other forms of authentication. We therefore use Public/Private Key authentication where possible on the system. All terminal-based network traffic between server and public network is encrypted since we are using the Secure Shell Version 2 (SSH2) protocol.

Setting up a Key-Pair

To set up a public private key-pair for the server, do this:

```
$ ssh-keygen -t rsa
Generating public/private rsa key pair.
Enter file in which to save the key (/home/madman/.ssh/id_rsa):
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /home/madman/.ssh/id_rsa.
Your public key has been saved in /home/madman/.ssh/id_rsa.pub.
The key fingerprint is:
2e:77:26:0c:28:08:ef:16:d9:93:ed:2f:e4:17:23:3a madman@mdev01.localdomain
The keys randomart image is:
+--[ RSA 1024]----+
|                 |
|                 |
|.                |
| .o o +          |
| . = = o S       |
| . o oo+o        |
|  o  +o.=oo      |
| .  E o+.+       |
|     . o.        |
+-----------------+
```

Accessing the Remote Server

To access the remote server from your local desktop machine, you will also need to have a public/private key-pair. If you do not have such a key-pair you can create your own key-pair on your own machine as follows. (Do not specify a passphrase for the Private Key when prompted)

```
[mymachine] $ ssh-keygen -t rsa
Generating public/private rsa key pair.
Enter file in which to save the key (/home/[myname]/.ssh/id_dsa)
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in id_rsa.
```

```
Your identification has been saved in id_rsa.
Your public key has been saved in id_rsa.pub.
The key fingerprint is:
03:5f:9f:fa:0d:47:33:6f:ca:bf:ee:15:67:c3:9f:09
The keys randomart image is:
+--[ RSA 1024]----+
|                 |
|                 |
|      .   .      |
|       o . . ..  |
|        S   oE++o|
|         . . ..=B|
|          . . .o=|
|           . = o.|
|            . *=o|
+-----------------+
```

The Key-pair for your machine is now created. You can list the public and private keys them with the command:

```
[mymachine] $  ls -al ~/.ssh
-rw-------   1 only_me only_me  668 Feb 17  2011 id_rsa
-rw-r--r--   1 only_me only_me  607 Feb 17  2011 id_rsa.pub
```

Where:

- id_rsa is the private key file. Keep it safe on your personal machine. Do not let it escape!
- id_rsa.pub is the public key file. It goes onto the keyring of every server that you want to connect to. It is OK to share this file with anyone in the public.

Adding your Key on the Server

To be able to access the server using your key, you need to get the *public* key onto the server first. For this, you *do* need to know what the password to the server is, and then only this once.

In the case where you are not a system administrator but need access to the server, you would email your public key to a system administrator who would then manually add it to the server's key ring, as explained in the section below.

The 'First-Principles' way

The basic way to put the public key file onto the server's key ring is to simply copy and paste it using a text editor. First we list the public key:

```
[mymachine] $ cat .ssh/id_rsa.pub
ssh-rsa AAAAB3NzaC1yc2EAAAADAQABAAABAQDI6DGYYu2Zzm2Gy5BFvgAt7qg8DDJJbnpVHu82mLCAqwx+fa2gWj+sc7S2utkhEwEJD5Qcs103gUBU4xwHwGMbHDPGY6cSmPTB6geo4cgWwjMB1UQG5j5zfoHRx/9gEZXiTGvdqy5R/gfo8P3
```

Note
Remember that it is OK to display a public key in public - hence the name "public". But don't ever let anyone ever see your private key.

Now select the text that starts from the `ssh-dss…` bit all the to the user name `bignose.the.third@terminator.com` and copy this. On the target server, open the following file using the `vi`-editor, type `i` for insert-mode, and paste the content into the bottom of the file. It is a long line of text, so it will wrap:

```
[madman@[server] $ vi .ssh/authorized_keys
```

After you have pasted the key value in, save and close the file (for the vi-editor at least): Hit the `Esc` key, then type `:wq`, then hit `Return`.

You have now successfully set up Public/Private-key authentication for the user madman from your personal machine to the server, albeit using the manual way.

A nicer way to disseminate your Public Key

There is a dedicated command that also accomplished the same as the above process: `ssh-copy-id`. You will of course still be prompted for a password:

```
[mymachine] ~ $  ssh-copy-id -i .ssh/id_dsa.pub madman@[servername]
madman@[servername]s password: <....>

Number of key(s) added: 1

Now try logging into the machine, with:   "ssh madman@[servername]"
and check to make sure that only the key(s) you wanted were added.
```

Testing your SSH connection

Test your new key authentication from a new terminal session on our personal machine. It would be helpful to keep the old session open in case you accidentally did something wrong.

```
[mymachine] $ ssh madman@[server]
Warning: untrusted X11 forwarding setup failed: xauth key data not generated
Warning: No xauth data; using fake authentication data for X11 forwarding.
X11 forwarding request failed on channel 0
madman@[server] ~$
```

Look! No password was required! You appear to have successfully logged on to the server. Check if this really is the user 'madman' who is logged on to the server:

```
$ id
uid=503(madman) gid=4(adm) groups=4(adm)
```

SUDO: Granting root-level functions to a mortal user

In order to govern a secure system, it should under normal circumstance not be possible to log in as the root-user. However, an administrator will occasionally need to perform root-level functions by momentarily elevating the user to root status and to execute root-level operations. For instance, installation of software can only be as user root, or as a user with temporarily-elevate rights through the sudo-process. This is done by making the user a "sudo-er" (from the concept pseudo-root) and granting previously-agreed functions that the admin user may need to execute. The granting of privileges is done by either making the admin user a member of a specific group (we do this here) or by allocating specific functions to the admin user (too labour-intensive).

The only person who can make sudo privilege grants is the root-user.

As user root, edit the `/etc/sudoers` file using the VI editor, using a special command:

```
root@s16972616 ~]# visudo
```

The VI-editor is a very sophisticated editor which you will you will either love or hate. In the case here where we need to start editing the file, hit the `i`-key (for insert mode in the vi-editor) and add this to the bottom of the file:

```
## MATCHi
%admin  ALL=(ALL)       ALL
```

By default, a sudo-user can only execute programs that are in the directories `/sbin`, `/bin`, `/usr/sbin` and `/usr/bin` - unless the full directory-path to the program is given, which is a pain. Since we have a number of our own system administration scripts deployed in `/usr/local/bin`, this directory needs to be added too. So change line 86 (more or less) and add this directory to the path so that it looks like this:

```
Defaults    secure_path = /sbin:/bin:/usr/sbin:/usr/bin:/usr/local/bin
```

Save and close the file: In vi, this is done by hitting the `Esc`-key, then hitting these 3 keys: `:` `w` `q` (for 'write', 'quit').

Create a new group for sudo'ers and associate the administrator user, madman, to the newly-created group called admin:

```
[root@s16972616 ~]# groupadd admin
[root@s16972616 ~]# usermod -a -G admin madman
```

Test the sudo'ing capability of the administrator user. First, log in as the administrator user. Rather than create a new session, we can do a "substitute user" (su-command):

```
[root@s16972616 ~]# su - madman
[madman@s16972616 ~]$
```

Also test and ensure that the administrator can sudo to root-level:

```
madman@s16972616 ~ $ sudo su -
[sudo] password for madman:
root@s16972616 ~ # id
uid=0(root) gid=0(root) groups=0(root)
```

If this test failed, then you should fix the problem using the existing and open, root-logged-in terminal session to the server. Hopefully you have not closed it already.

Set up User and Server Identifiers in the command prompt

Since there are multiple servers in the solution, it is helpful for always clearly identify the user and the server to avoid the wrong command to be executed on the wrong server. Since access to the servers will mostly be via SSH-terminal session, the best-practice approach is to display this information on every command in the form of an elaborate command prompt. The standard convention is that the root-user command prompt ends in a '#'-character, and all other (non-root) users' command prompts end in a '$'-character.

- Non-root user command prompt:

For the current (non-root) user, set the command prompt and include the server alias name in there:

```
[madman@s16972616 ~]$ vi ~/.bash_profile
```

Add to the file for a green command and blue directory path, and set the traditional **$** character at the end of the non-root prompt. Replace the **mweb01** with the 'friendly' name of the server:

```
export PS1="\[\033[01;34m\]\u@mweb01 \[\033[01;34m\]\w $\[\033[00m\] "
```

- Root user command prompt:

When you do the same for the user **root**, the conventsion is to use a **#** in the command prompt, in which case you would need to do this:

```
export PS1="\[\033[01;31m\]\u@mweb01 \[\033[01;34m\]\w #\[\033[00m\] "
```

To make it more obvious that the root user is active, the prompt is also set to red. Again, do the same for user *root* later on the other servers.

- Command history

The command history holds the last 1000 commands entered by default. This is often not enough, so this can be extended to 10000 by setting this system variable in the file:

```
export HISTFILESIZE=10000
```

- Save and Close

Save and Close the file (hit `Esc` and then `:` `w` `q`). You can either open another console session to see the effect of the changes that you made in the .bash_profile file, or you can effect is immediatel with the **source** command. See how the command prompt changes:

```
$ source ./.bash_profile
madman@mweb01 ~ $
```

- Document Convention going forward

Going forward in the documentation, we don't show the full command prompt in console commands and simply indicate with a **$** in the first column to show whether the line was an input command or just output from the preceding command. When logged in as user **root**, we show the **#** in the command prompt of course.

Server aliases in hosts file

It convenient to refer to the servers by their local aliases instead of the formal instance names that were allocated to them by the hosting provider. Referring to servers via aliases makes the migration of infrastructure from one hosting provider to another easier. The current servers are known among each other by the following aliases, which are set on each server's hosts file:

- Web Server, alias mweb01
- Database Server, alias mdb01
- Application Server, alias mapp01
- Test Server, alias mtst01

Extend this pattern as further servers are added to the solution.

Server MWEB01

On the web server, edit the hosts file `/etc/hosts` and add the following:

```
127.0.0.1   localhost localhost.localdomain localhost4 localhost4.localdomain4 mweb01
::1         localhost localhost.localdomain localhost6 localhost6.localdomain6
82.165.16.42 mdb01
87.106.206.136 mapp01
```

Server MDB01

On the database server, edit the hosts file `/etc/hosts` and add the following:

```
127.0.0.1   localhost localhost.localdomain localhost4 localhost4.localdomain4 mdb01
::1         localhost localhost.localdomain localhost6 localhost6.localdomain6
```

```
87.106.201.248 mweb01
87.106.206.136 mapp01
```

Server MAPP01

On the application server, edit the hosts file `/etc/hosts` and add the following:

```
127.0.0.1   localhost localhost.localdomain localhost4 localhost4.localdomain4 mapp01
::1         localhost localhost.localdomain localhost6 localhost6.localdomain6
87.106.201.248 mweb01
82.165.16.42 mdb01
```

Any new Servers

```
127.0.0.1   localhost localhost.localdomain localhost4 localhost4.localdomain4 [friendly server name]
::1         localhost localhost.localdomain localhost6 localhost6.localdomain6
# Any the other friendly server names and their IP addresses
...
```

Your own Machine

On your own machine, add the following to your hosts file. If your machine is a Linux or Apple machine, add this to the file `/etc/hosts` using nano or vi.

If your machine runs Windows, add the following text to the file `C:\Windows\Drivers\etc\hosts` using Notepad or any similar tool:

```
82.165.16.42 mdb01
87.106.206.136 mapp01
87.106.201.248 mweb01
212.227.255.146 mtst01
# Any other friendly server names and their IP addresses
...
```

Setting up additional Server Repositories

The default code repository that comes with CentOS or RedHat is very conservative and does not contain leading-edge releases of many services and applications. PHP is a pertinent example: CentOS 6.7 in 2015 comes budled with PHP 5.3.0, which dates back to 2010.

The EPEL and REMI repositories

These two repositories have proven over time to be the most robust and stable repositories:

- EPEL: "Extra Packages for Enterprise Linux"
- REMI: Remi is a lovely person who devotes much time to the management of open source packages

These two additional "catalogs of software packages" are sufficient to serve the current needs. Follow the steps below to add more repositories if ever a need for it arises.

Install more repositories

This installs further repository catalogs on the server. Once the these are installed and enabled, they will automatically be updated with the core CentOS repository on the `yum update` command.

For CentOS 6.x, the commands are:

```
$ sudo su -
# wget https://dl.fedoraproject.org/pub/epel/epel-release-latest-6.noarch.rpm && rpm -Uvh epel-release-latest-6.noarch.rpm
# wget http://rpms.famillecollet.com/enterprise/remi-release-6.rpm && rpm -Uvh remi-release-6*.rpm
```

For CentOS 7.x, the commands are:

```
$ sudo su -
# wget https://dl.fedoraproject.org/pub/epel/epel-release-latest-7.noarch.rpm && rpm -Uvh epel-release-latest-7.noarch.rpm
# wget http://rpms.famillecollet.com/enterprise/remi-release-7.rpm && rpm -Uvh remi-release-7*.rpm
```

Enabe the REMI Repository

Set the `enable=1` flag for the REMI repository so that this repository does not need to be explicitly declared when installing software packages from there.

```
$ nano /etc/yum.repos.d/remi.repo
...
[remi]
name=Remi RPM repository for Enterprise Linux 6 - $basearch
#baseurl=http://rpms.remirepo.net/enterprise/6/remi/$basearch/
mirrorlist=http://rpms.remirepo.net/enterprise/6/remi/mirror
enabled=1
gpgcheck=1
gpgkey=file:///etc/pki/rpm-gpg/RPM-GPG-KEY-remi
```

Change it accordingly for CentOS 7.x

Enabe the EPEL Repository

Set the `enable=1` flag for the EPEL repository so that this repository does not need to be explicitly declared when installing software packages from there. Recently, this repository is enabled by default, so there may not be anything to do here.

```
$ nano /etc/yum.repos.d/epel.repo
...

[epel]
name=Extra Packages for Enterprise Linux 6/7 - $basearch
#baseurl=http://download.fedoraproject.org/pub/epel/6/$basearch
mirrorlist=https://mirrors.fedoraproject.org/metalink?repo=epel-6&arch=$basearch
failovermethod=priority
enabled=1
gpgcheck=1
gpgkey=file:///etc/pki/rpm-gpg/RPM-GPG-KEY-EPEL-6
```

Change it accordingly for CentOS 7.x

Update the new Repositories

Now would be a good time to update the new repositories:

```
# yum -y update
```

You should now have PHP Version 5.45 installed. Test it:

```
# php --version
PHP 5.4.45 (cli) ...
```

Operational Tools

These tools are required for performing general operations such as deploying new releases.

Subversion Client

We use Subversion at the moment for our code management, although the aim is to eventually use GIT.

Installing the Subversion Client

Install the Subversion Client if it is not already installed:

```
$ sudo yum -y install subversion
```

Connecting for the first time to Matchi's Subversion Repository

The default location for the local Subversion sandbox is ~/svn<code>. On the assumption that the Subversion Repository has been set up, do a *check out* of this repository into the server's workplace:

```
$ mkdir ~/svn
$ cd ~/svn
$ svn co http://mapp01/svn/matchi/trunk/php
```

You will be prompted for your subversion username and password:

```
Authentication realm: <http://mapp01:80> Subversion repositories
Password for 'madman':

-----------------------------------------------------------------------
ATTENTION!  Your password for authentication realm:

   <http://mapp01:80> Subversion repositories

can only be stored to disk unencrypted!  You are advised to configure
your system so that Subversion can store passwords encrypted, if
possible.  See the documentation for details.

You can avoid future appearances of this warning by setting the value
of the 'store-plaintext-passwords' option to either 'yes' or 'no' in
'/home/madman/.subversion/servers'.
-----------------------------------------------------------------------
Store password unencrypted (yes/no)? yes
A php/...
...
```

Do the same for the other pertinent directories - there is no need to get the entire repository, as it contains branches and archives which will just clutter up the server's hard drive.

```
$ svn co http://mapp01/svn/matchi/trunk/sql
$ svn co http://mapp01/svn/matchi/trunk/usr
```

Git Client

Some of the diagnostic utilities are only available from GIT repositories for which the GIT client is required. The default location for the local GIT sandbox is <code>~/git<code>.

Installing the Git Client

```
$ sudo yum -y install git
```

C-Compiler

In rare cases, a C/C++ compiler is required to build third-party utilities, and perhaps even internal utilities.

```
$ sudo yum -y install gcc
```

General Utility Tools

A consistent set of tools is required on each server for managing and monitoring the server. The following are assumed to be installed for moving forward with the server build, so install them first.

The tools are installed using the preferred package management application of the Red Hat and CentOS operating system, called YUM. Other package management tools such as RPM can also be used, if you have previously pulled down the correct version of the RPM file from the appropriate repository. So RPM sounds complicated? Stick to YUM then!

Install the NANO editor

VI is the preferred editor to many sysadmin people, but NANO is a simpler editor, although less sophisticated. Install it using YUM, and when prompted, enter the password for user madman (and not the password for user root):

```
[madman@server ~]$ sudo yum install nano
[sudo] password for madman:
...
Success.
```

Fine-tune the nano-editor settings for the type of application. Since this configuration files is in the /etc/ -directory, it can only be written to as root, or as we do here, a sudo-ed user and make the following changes:

```
[madman@server ~]$ sudo nano /etc/nanorc
...
set tabsize 2
...
set tabstospaces
...
include "/usr/share/nano/nanorc.nanorc"
include "/usr/share/nano/perl.nanorc"
include "/usr/share/nano/sh.nanorc"
include "/usr/share/nano/php.nanorc"
```

Save the changes in the file and Close the editor using the key-strokes `Ctrl`-`O` and then `Ctrl`-`X`.

Tree - A Directory Organization Tool

This comes by default on many other types of Linux servers but for some reason not with Red Hat / CentOs, yet is nice tool to have on hand for sysadmins. It displays a tree of directories and files, which is useful to visualize large directory trees and for producing documentation with.

```
$ sudo yum install tree
```

The tree of directories and files are shown like this:

```
$ tree /etc/yum
/etc/yum
```

```
/etc/yum
├── fssnap.d
├── pluginconf.d
│   ├── fastestmirror.conf
│   └── langpacks.conf
├── protected.d
│   └── systemd.conf
├── vars
│   └── infra
├── version-groups.conf
├── yum-cron.conf
└── yum-cron-hourly.conf

4 directories, 7 files
```

Figlet - Large Character Display

Figlet converts text into large characters that are made up of ordinary screen characters. It is mostly just a fun tool to have for making banners.

Install it like this:

```
$ sudo yum install figlet
```

Have some fun - try this:

```
$ figlet "Hello, World"
 _   _      _ _        __        __                _     _
| | | | ___| | | ___    \ \      / /__  _ __ ___| | __| |
| |_| |/ _ \ | |/ _ \    \ \ /\ / / _ \| '__/ _ \ |/ _` |
|  _  |  __/ | | (_) |    \ V  V / (_) | | |  __/ | (_| |
|_| |_|\___|_|_|\___/ )    \_/\_/ \___/|_|  \___|_|\__,_|
                     |/
```

Or something a little more complex with embedded string delimiters and a special execute-in-place character "!":

```
$ cat <<EOF | figlet
Hi y'all!
EOF

 _   _ _                 _ _ _
| | | (_)  _   _( ) __ _| | | |
| |_| | | | | |/ / _` | | | |
|  _  | | | |_| | | (_| | | | |_|
|_| |_|_|  \__, |  \__,_|_|_|_(_)
           |___/
```

Why do we need to do it like this? Read all about it in the section on the BASH shell.

System Diagnostic Tools

Install HTOP

HTOP is a server performance viewer with a rich set of features

Installation

```
$ sudo yum -y install htop
```

Usage

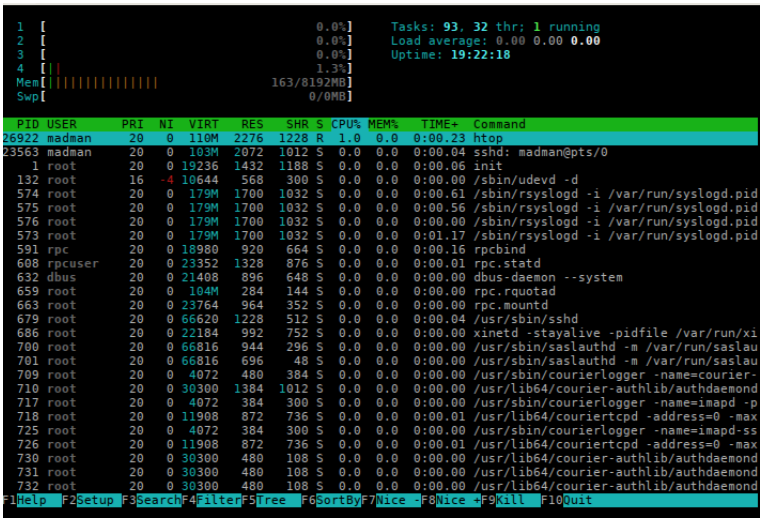
Test it with the command:

```
$ htop
```

This utility displays the following useful metrics:

- Server uptime
- CPU usage
- Memory usage
- Details of currently-running processes

It is possible to arrange the order of display using the function key shortcuts. Hit F10 or 'Q' to quit.



Install LYNX

LYNX is a text-based browser that you can run through a browser. It supports most browser features like cookies, however it does not support JavaScript and since it is a text-based browser, it also does not show graphic images but just leaves a hint that there is a graphic on a page, based on the <code>alt-tag.

It is useful for validating firewall configurations and for crude screen-scraping operations from other websites.

Installation

Simple!

```
$ sudo yum  -y install lynx
```

Usage

Usage

Test it with the command:

```
$ lynx http://matchi.biz

 Matchi Matchi
     * Home
     * Financial Institutions
     * Solution Providers
     * Challenges
     * News
     * Contact Us

   SIGN UP   LOGIN
   (BUTTON)
     * LOGIN
     * SIGN UP

   (BUTTON)
     * Home
     * Financial Institutions
     * Solution Providers
     * Challenges
     * News
     * Contact Us
```

Network Diagnostic Tools

Warning
The installation of these tools on servers can make the servers be used to attack other devices on the Internet, should contol of the servers end up in the wrong hands. If in doubt, do not install these tools, or at least un-install them when finished using them.

NMAP is a used to probe a remote server for various information, including what operating system it is running and what ports are open.

Installation

```
$ sudo yum install nmap
```

Usage

Examples:

- Check to see if MySQL/MariaDB access is allowed over a network connection on the database server `mdb02`:

```
$ nmap -p 3306 mdb02

Starting Nmap 6.40 ( http://nmap.org ) at 2016-10-23 14:59 BST
Nmap scan report for mdb02 (217.160.206.97)
Host is up (0.00095s latency).
PORT     STATE    SERVICE
3306/tcp closed   mysql

Nmap done: 1 IP address (1 host up) scanned in 0.04 seconds
```

We can see that this port is 'closed', which means that it is not possible to access a MySQL database from anywhere other than from the database server itself. This would of course present a problem to the webserver since it is hosted on a different device.

- Check to see what ports are open over a network connection on the database server `mdb02`:

```
$ nmap  mdb02

Starting Nmap 6.40 ( http://nmap.org ) at 2016-10-23 14:59 BST
Nmap scan report for mdb02 (217.160.206.97)
Host is up (0.68s latency).
Not shown: 995 filtered ports
PORT     STATE SERVICE
21/tcp   open  ftp
22/tcp   open  ssh
80/tcp   open  http
443/tcp  open  https
8443/tcp open  https-alt

Nmap done: 1 IP address (1 host up) scanned in 54.29 seconds
```

Install NTOP (Next Generation)

NTOP is a used to view network loading and bandwidth consumption.

More details:

- http://www.ntop.org/
- http://redis.io/

Installation

This installation guide covers RedHat 7.x only.

- Create a special repository file for NTOP and add the following details to it:

```
$ sudo nano /etc/yum.repos.d/ntop.repo

[ntop]
name=ntop packages
baseurl=http://www.nmon.net/centos-stable/$releasever/$basearch/
enabled=1
gpgcheck=1
gpgkey=http://www.nmon.net/centos-stable/RPM-GPG-KEY-deri
[ntop-noarch]
name=ntop packages
baseurl=http://www.nmon.net/centos-stable/$releasever/noarch/
enabled=1
gpgcheck=1
gpgkey=http://www.nmon.net/centos-stable/RPM-GPG-KEY-deri
```

Save and exit `Ctrl` - `O`, `Ctrl` - `X`.

- Update all the repositories

```
$ sudo yum -y update
```

- Install the necessary packages:

```
$ sudo yum -y install redis ntopng
$ sudo yum -y install hiredis-devel
```

Configuration of NTOP

- Basic configuration

This will remove all warnings that we are 'only' running the "Community Edition". Add the following to the file `/etc/ntopng/ntopng.conf`:

```
$ sudo nano /etc/ntopng/ntopng.conf

-G=/var/tmp/ntopng.pid\
-community
```

- Set up the services

The service `redis` and `ntopng` need to started and the set to restart again when the server is rebooted:

```
sudo systemctl start redis.service
sudo systemctl enable redis.service
```

```
sudo systemctl start ntopng.service
sudo systemctl enable ntopng.service
```

- Opening up the firewall Configuration for NTOP

This opens up port 3000 so that it can be accessed from other devices.

```
sudo firewall-cmd --permanent --add-port=3000/tcp
sudo firewall-cmd --reload
```

Usage

Test it by pointing your browser to this server over port 3000.

If this does not work, then try to browse directly from the server to the `ntopng` service using the `lynx` text-based browser, which should look like this.

```
$ lynx http://localhost:3000

Welcome to ntopng

  _________________ _________________
  (BUTTON) Login


  © 1998-2016 - ntop.org
  ntopng is released under GPLv3
```

By default, the access criteria is `admin`, `admin`. Change the password immediately to something more secure.

Nethogs

Use this tool to see which processes consume the most TCP-protocolled bandwidth. It is similar to HTOP, and lists processes in order of bandwidth consumption.

Installation

```
$ sudo yum -y install nethogs
```

Use

Specify the network interface for which bandwidth needs to be monitored. For a virtual servers, this is normally venet0, for tin servers this is either eth0, eth1. For Wifi interfaces, this can be wlan0 or ath0 or something else. Check the network device names for the server in `/proc/net/dev`:

```
$ cat /proc/net/dev
Inter-|Receive                                                           |Transmit
 face |bytes    packets errs drop fifo frame compressed multicast|bytes    packets errs drop fifo colls carrier compressed
    lo: 1539420091  6908499    0    0    0     0          0         0 1539420091 6908499    0    0    0     0       0          0
venet0: 97709451743 75682952   0    0    0     0          0         0 2856837870 32029423   0    0    0     0       0          0
```

Priviledged execution is required, so run this under sudo:

```
$ sudo nethogs venet0
```

```
NetHogs version 0.8.0

  PID USER     PROGRAM                                                                    DEV        SENT      RECEIVED
18165 madman   sshd: madman@pts/2                                                        venet0     0.000       1.681 KB/sec
24646 root     mysql                                                                      venet0     0.000       0.336 KB/sec
24636 root     mysql                                                                      venet0     0.000       0.335 KB/sec
24641 root     mysql                                                                      venet0     0.000       0.335 KB/sec
?     root     87.102.39.188:48180-87.106.206.136:23                                              0.000       0.026 KB/sec
?     root     121.12.125.103:6000-87.106.206.136:3306                                            0.000       0.023 KB/sec
24628 root     mysql                                                                      venet0     0.000       0.000 KB/sec
24622 root     mysql                                                                      venet0     0.000       0.000 KB/sec
24617 root     mysql                                                                      venet0     0.000       0.000 KB/sec
24612 root     mysql                                                                      venet0     0.000       0.000 KB/sec
?     root     151.227.118.73:50548-87.106.206.136:443                                            0.000       0.000 KB/sec
?     root     151.227.118.73:50547-87.106.206.136:443                                            0.000       0.000 KB/sec
?     root     151.227.118.73:50546-87.106.206.136:443                                            0.000       0.000 KB/sec
24603 root     mysql                                                                      venet0     0.000       0.000 KB/sec
?     root     151.227.118.73:50545-87.106.206.136:443                                            0.000       0.000 KB/sec
24595 root     mysql                                                                      venet0     0.000       0.000 KB/sec
24589 root     mysql                                                                      venet0     0.000       0.000 KB/sec
2189  madman   sshd: madman@pts/1                                                         venet0     0.000       0.000 KB/sec
24583 root     mysql                                                                      venet0     0.000       0.000 KB/sec
24576 root     mysql                                                                      venet0     0.000       0.000 KB/sec
24570 root     mysql                                                                      venet0     0.000       0.000 KB/sec
24560 root     mysql                                                                      venet0     0.000       0.000 KB/sec
```

```
24555 root     mysql                                                                    venet0     0.000       0.000 KB/sec
24550 root     mysql                                                                    venet0     0.000       0.000 KB/sec
?     root     108.160.163.108:443-87.106.206.136:47029                                            0.000       0.000 KB/sec
?     root     unknown TCP                                                                         0.000       0.000 KB/sec

  TOTAL                                                                                            0.000       2.736 KB/sec
```

To quit, hit `Ctrl`-`C`.

Install Internet forensic tools

We need tools such as `whois`, `dig` and other tools to help determine the source of web-based attacks.

Installation

```
$ sudu yum -y install whois
$ sudo yum -y install bind-utils
$ sudo yum -y install tcpdump
```

Usage

Read one of the many dodgy books on hacking. This is a big subject area.

Application Diagnostic Tools

ShellPic – a tool for viewing images on a text terminal

Instead of having to view image files on the server via an SSH tunnel, SCP'ing it to a desktop client first, or viewing in on a web browser (not always possible), it is possible to view a low-fidelity image directly on the terminal. This is very useful for quick verifications that the correct images are where they should be. Bear in mind that what you see will not provide any indication of the quality of the image, however.

Installation

To install thia utility, some system libraries and Python development libraries are required:

```
$ sudo yum -y install libjpeg-devel
$ sudo yum -y install python-devel
$ sudo yum -y install python-pip
```

Then the Python Imaging Library library *Pillow* needs to be installed using Python's package manager *pip*:

```
$ sudo pip install pillow
```

Note that Pillow replaces the legacy PIL library. Do not use PIL (Python Image Library) any more.

The source code for shellpic is pulled from its GIT repository:

```
$ cd ~/git
~/git $ git clone https://github.com/larsjsol/shellpic.git
~/git $ cd shellpic
~/git $ sudo python setup.py install
```

Installation errors or warnings

You might get an error message like this:

```
You are using pip version 7.1.0, however version 7.1.2 is available.
You should consider upgrading via the 'pip install --upgrade pip' command.
```

So do it!

```
$ sudo pip install --upgrade pip
```

Usage

You should be able to get a good but low-fidelity indication of what an image contains by viewing it on the terminal with the command:

```
$ shellpic [image_path]
```

For example:

Server notification email set-up

System notification events need to be communicated from the server to a real user via email. By default, system emails are sent to the internal user `root`, which needs to be passed to a real email address `sysadmin@matchi.biz`.

The /etc/aliases file

Edit the file `/etc/aliases` and set the real email addresses up for all server accounts, which all point to `root`, as `sysadmin@matchi.biz`.

```
# Person who should get root's mail
root:   sysadmin@matchi.biz
```

The result of this is that all system alerts emailed from the server end up in the same email address at `sysadmin@matchi.biz`, and the server that emitted the alert is identifiable from the friendly server name that appears in the recipient email address, e.g. `sysadmin@matchi.biz`.

Install the PostFix Mail server

Only install for RedHat / Centos 6.x Virtual Servers

Do this if the /etc/postfix directory does not yet exist.

```
# yum -y install postfix
```

The /etc/postfix/main.cf file

Add or replace the following lines to the file, and amend this according to the 'friendly server name', which in this case is *mweb01*:

```
myhostname = mweb01.matchi.biz
myorigin = $myhostname
```

Manage the Postfix Server

Restart the Postfix server:

On RedHat 6.x, do this:

```
$ sudo /etc/init.d/postfix restart
Shutting down postfix:                                     [FAILED]
Starting postfix:                                          [  OK  ]
```

Set the service so that it will automatically start each time the server is booted up:

```
$ sudo chkconfig --levels 2345 postfix on
```

On red Hat 7.x, Postfix will already be running. You can check it:

```
$ sudo systemctl status postfix
● postfix.service - Postfix Mail Transport Agent
   Loaded: loaded (/usr/lib/systemd/system/postfix.service; enabled; vendor preset: disabled)
   Active: active (running) since ...
```

Testing emailing from the server

Test the email capability on each server by sending an email from the server's console:

```
$ mail -s 'test' root <<!
This is a test.
!
```

Assuming that this test was done from server `mapp02`, then an email addressed to `sysadmin+mapp02@matchi.biz` should arrive at the inbox `sysadmin@matchi.biz`.

Setting up Secure Shell (SSH) channels between the servers

The simplest and most secure method to transfer files from one server to another is via SSH secure shell. While password-based access may still be possible a this stage, you can connect to a server and enter the password when prompted for it. However, for any automated process such as file transfers, a password-less approach is required, and to achieve this we use a public/private key pair.

Creating the public-private key-pair

For each server we create a key-pair:

```
madman@s16972616 mweb1 ~ $ ssh-keygen -t rsa
Generating public/private rsa key pair.
Enter file in which to save the key (/home/myname/.ssh/id_rsa)
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in id_rsa.
Your public key has been saved in id_rsa.pub.
The key fingerprint is:
03:5f:9f:fa:0d:47:33:6f:ca:bf:ee:15:67:c3:9f:09
The keys randomart image is:
+--[ RSA 1024]----+
|                 |
|                 |
|      .   .      |
|       o . . ..  |
|        S   oE++o|
|         . . ..=B|
|          . . .o=|
|           . = o.|
|            . *=o|
+-----------------+
```

Desimating the public key to target servers

Note
Only do this between servers where there SSH file transfers and SSH terminal sessions are likely to occur.

Now add the public key of each server to the key ring of all the other servers.

If password-based authentication is still enabled on the target server, use the BASH shell to pipe the public key over SHH. You will be prompted for a password one final time on each server. Do this on all 3 servers, changing the target server each time. Here we distribute the public key of server web01 to servers mapp01 and mdb01:

```
madman@mweb01 ~ $ cat ~/.ssh/id_rsa.pub | ssh mapp01 'cat >> ~/.ssh/authorized_keys'
The authenticity of host 'mdb01 (82.165.16.42)' cant be established.
RSA key fingerprint is e6:f1:c4:66:b6:a2:38:86:8b:61:9b:73:74:34:0f:c6.
Are you sure you want to continue connecting (yes/no)? yes
Warning: Permanently added 'mdb01,82.165.16.42' (RSA) to the list of known hosts.
madman@mapp01s password:
madman@mweb1 ~ $ cat ~/.ssh/id_rsa.pub | ssh mdb01 'cat >> ~/.ssh/authorized_keys'
The authenticity of host 'mdb01 (82.165.16.42)' cant be established.
RSA key fingerprint is e6:f1:c4:66:b6:a2:38:86:8b:61:9b:73:74:34:0f:c6.
Are you sure you want to continue connecting (yes/no)? yes
Warning: Permanently added 'mdb01,82.165.16.42' (RSA) to the list of known hosts.
madman@mdb01s password:
```

There is an even more elegant way of doing this with a single command:

```
madman@mweb01 ~ $ ssh-copy-id i ~/.ssh/id_rsa.pub mapp01
madman@mapp01s password:
```

If password-based access via SSH has already been disabled to the target server, the public key of the new server can only be distributed from a device that is already able to connect to the older servers, by copying and pasting the public key into the file .ssh/authorized_keys using a text editor.

Securing the Secure Shell (SSH) Service

Securing the Secure Shell (SSH) Service

Use the nano editor (if it already installed) or vi (installed by default) to edit the `/etc/ssh/sshd_config` file in order to secure the SSH service to achive the following objectives:

- The user 'root' can never log in remotely. Trying to log in as user root is a favourite attack vector, since every UNIX-based server has this user account.
- Only the specified user 'madman' can ever log in, even if there are other user accounts on the server (which there are - see the content of the `/etc/passwd` file)
- Password-based authentication is not possible and it is only possible to log in through the use of a public/private key-pair
- Add a warning banner to all those who attempt to log in to the server

```
$ sudo nano /etc/ssh/sshd_config
...
PasswordAuthentication no
PermitRootLogin no
AllowUsers madman
Banner /etc/warning
...
```

Save and Close the `/etc/ssh/sshd_config` file (`Ctrl`-`O`, `Ctrl`-`X`).

Secure Shell (SSH) Warning Banner

Warning Banner will display a no-trespassing warning message at the start of each SSH session. If password authentication were enabled, then this would be displayed even before the user is prompted for a password. This message will not deter a persistent adversary, but it does cause some basic attack scripts to break and it is also useful from a legal point of view to support the eventual prosecution of persistent attackers. It is not advised to keep this message simple and to not give too much information away about the server, the business or the people involved in this message. Create this standard message in the `/etc/warning` file:

```
$ sudo nano /etc/warning
[sudo] password for madman:
******************************************************************************
# You are entering a secured and monitored area.                             #
# Your IP, login time and username have now been noted.                      #
# This service is restricted to authorized users only.                       #
# All activities on this system are logged.                                  #
# Unauthorized access will be reported to relevant law enforcement agencies. #
# Disconnect immediately if you are not an authorized user.                  #
******************************************************************************
```

Save and Close the `/etc/warning` file (`Ctrl`-`O`, `Ctrl`-`X`).

These changes need to be effected ("actualized") on the server by restarting the SSH daemon. Open up another SSH terminal to the server as a failover before you perform the next step, in case something goes wrong and you do not completely loose your way to the server. Restart the SSH service to make these changes effective as follows:

On RedHat 6.x, do this:

```
$ sudo /etc/init.d/sshd restart
[sudo] password for madman:
Stopping sshd:                                             [  OK  ]
Starting sshd:                                             [  OK  ]
```

On RedHat 7.x, do this:

```
$ sudo systemctl restart sshd
[sudo] password for madman:
```

Test your settings:

- Try to login as user **root**:

```
[mydesktop] $ ssh root@mweb02
Permission denied (publickey,gssapi-keyex,gssapi-with-mic).
```

This is the desired results and already makes the server safe from a substantial part of the adversaries.

- Try to login as user **madman** from a device that does not have its public installed on this server:

```
[mydesktop] $ ssh madman@mweb02
Permission denied (publickey,gssapi-keyex,gssapi-with-mic).
```

Again, this is the desired results and already makes the server safe from another substantial part of the adversaries, who will attempt to run a dictionary attack.

Now open a new SSH session to ensure that the configuration changes are correct. If it turns out to be impossible to now connect to the target server from the new terminal, use the previously-opened fail-over session to recover from this situation by checking for possible errors in the `/etc/ssh/sshd_config` configuration file.

Work-around for the X11-based warning message:

if you encounter the warning message `xauth: file .Xauthority does not exist`, the workaround is to leave X11-forwarding over SSH enabled (even though we don't serve X11 here). Install the Xauthority package and set up an empty `~/.Xauthority` file:

```
$ sudo yum -y install xauth
$ touch ~/.Xauthority
```

Add a Welcome Message

This is a message that will only be displayed *after* a user has successfully logged on via an SSH session in a terminal, and displays the text in the file `/etc/motd`. (“motd” stands for “Message Of The Day”). By default, this file is blank as it is not essential. However, we use it here as it is useful to system administrators who are reminded which server it is that they have just logged on. It will be display whenever to log in like this:

```
$ ssh madman@server
... Display of the welcome message ...
... Do stuff on the server ...
```

If you don't want to see the welcome message, or your are logging in as part of a batch process, redirect the STDERR (file handle #2) to the null device:

```
$ ssh madman@server 2>/dev/null
... Do stuff on the server ...
```

Utilities, such as `fortune` exist that can randomly select and put a prithy little quote in the welcome message.

- **Manual creation of the `/etc/motd` file:**

Manually edit the `/etc/motd` file with NANO (or VI if you are well-hard):

```
$ sudo nano /etc/motd
[sudo] password for madman:
```

Have some fun and make a pretty welcome message. Generate some ASCII-text using this tool: http://patorjk.com/software/taag and then copy and paste the result into your edit on your terminal. Or copy this text below:

```
########################################################
#       Welcome to another happy server of             #
#  __  __    _  _____ ____ _   _ _   _     _           #
# |  \/  |  / \|_   _/ ___| | | (_) | |__ (_)____      #
# | |\/| | / _ \ | || |   | |_| | | | '_ \| |_  /      #
# | |  | |/ ___ \| || |___|  _  | | |_| |_) | |/ /____   #
# |_|  |_/_/   \_\_| \____|_| |_|_|_(_)_.__/|_/_______|  #
#                                                      #
########################################################
```

Save and Close the file.

- **Programatic creation of the `/etc/motd` file:**

Use `figlet` to create large text. Sudo to root-user first:

```
$ sudo su -
[sudo] password for madman:
# echo "Welcome to " > /etc/motd
# figlet "mweb02" >> /etc/motd
# echo "Another happy and secure server from" >> /etc/motd
# figlet "matchi.biz" >> /etc/motd
# exit
$
```

This is what it will look like - use the `cat`-command to list the content of the file:

```
# cat /etc/motd
Welcome to             _      ___ ____
 _ __ _____      _____| |__  / _ \___ \
| '_ ` _ \ \ /\ / / _ \ '_ \| | | |__) |
| | | | | \ V  V /  __/ |_) | |_| / __/
|_| |_| |_|\_/\_/ \___|_.__/ \___/_____|

Another happy and secure server from
                 _       _     _   _     _
 _ __ ___   __ _| |_ ___| |__ (_) | |__ (_)____
| '_ ` _ \ / _` | __/ __| '_ \| | | '_ \| |_  /
| | | | | | (_| | || (__| | | | |_| |_) | |/ /
|_| |_| |_|\__,_|\__\___|_| |_|_(_)_.__/|_/___|
```

Security Services

It is necessary to install a few additional processes that actively prevent attacks or at least let you know that your server may have been attacked, to allow you to take the necessary action.

FAIL2BAN

Ban unwelcome IP addresses that attempt to connect via SSH. This defence layer is in addition to the defence layer that the data centre itself offers.

Warning
Do not solely rely on the defence systems of others! In particular, the data centre's defence system is very weak. This is why we need to bolster our own local defenses here.

It is possible to ban unwelcome IP address from known adversaries and annoyances at the server level through the use of the fail2ban daemon, which adds the IP addresses to the server's firewall configuration, such that the IP addresses are simply ignored. Although it is far more optimal to ban IP addresses at the network switch level, only a server-based process is able to determine when an IP address repeatedly fails to log in to a server via SSH. We set this process so that 1 failed attempt to establish an SSH session from a given IP address will ban the IP address for a very long time.

The Symptoms

Here is how you know that your server is under attack and to find out what the attackers' IP addresses are:

You get notified when you log in

```
$ sudo su -
[sudo] password for madman:
Last login: Mon Feb 23 12:43:38 SAST 2015 on pts/0
Last failed login: Mon Feb 23 13:34:22 SAST 2xxx from 115.239.228.35 on ssh:notty
There were 15326 failed login attempts since the last successful login.
```

```
$ sudo su  -
# grep Failed /var/log/secure
Feb 23 13:05:26 mdev01 sshd[28450]: Failed password for root from 115.239.228.15 port 51645 ssh2
Feb 23 13:05:29 mdev01 sshd[28450]: Failed password for root from 115.239.228.15 port 51645 ssh2
Feb 23 13:05:38 mdev01 sshd[28455]: Failed password for root from 115.239.228.15 port 35855 ssh2
Feb 23 13:05:40 mdev01 sshd[28455]: Failed password for root from 115.239.228.15 port 35855 ssh2
Feb 23 13:05:43 mdev01 sshd[28455]: Failed password for root from 115.239.228.15 port 35855 ssh2
Feb 23 13:05:52 mdev01 sshd[28461]: Failed password for root from 115.239.228.15 port 46632 ssh2
...
# grep Failed /var/log/secure | awk '{print $11}' | sort -u
103.41.124.101
103.41.124.104
103.41.124.107
103.41.124.29
103.41.124.30
103.41.124.55
...
```

Most of these IP addresses originate from the Chinese mainland. Pick one of the IP addresses above to check:

```
$ whois 103.41.124.101 | grep "address:"
address:        INT'L TOWER 707-713 NATHAN RD MONGKOK  KLN HONG KONG, hongkong KLN 999077
address:        TOWER 707-713 NATHAN RD MONGKOK KLN HONG KONG
```

Installation

Fail2Ban is not part of the Red Hat Linux distribution, so it needs to be installed from the EPEL third-party library. First, check that this product is actually in the 3rd-party repository:

Install Fail2Ban using `yum`:

```
$ sudo yum -y install fail2ban
...
Total download size: 867 k
Installed size: 3.0 M
```

This installs all the dependencies and the eventual package.

Configuration

We configure the Fail2Ban daemon to:

- Ban all IP addresses that failed in their attempt to connect to the server over the SSH port
- Banned IP addresses are banned for a duration of 24 hours (86400 seconds)
- Send all banning notifications to root@localhost. The configuration of the `/etc/aliases` file will identify which server this notification originated from.

Edit the `/etc/fail2ban/jail.local` file, which will override any default settings in the `/etc/fail2ban/jail.conf` file. We may need to add future overrides of other protocols in the `/etc/fail2ban/jail.conf` file later on as we discover other types of attacks on the server.

```
$ sudo nano /etc/fail2ban/jail.local
```

Add the following overrides:

```
[DEFAULT]
ignoreip = 127.0.0.1/8
bantime = 84600
findtime = 600
maxretry = 1
```

Save and Exit.

Add a check specifically for the SSH protocol with the `/etc/fail2ban/jail.local` file:

```
$ nano /etc/fail2ban/jail.d/sshd.local
```

Add the following :

```
[sshd]
enabled = true
port = ssh
#action = firewallcmd-ipset
logpath = %(sshd_log)s
maxretry = 5
bantime = 86400
```

Save and Exit.

Starting the Service

Start the Fail2Ban service and also make it self-start on server boot:

On RedHat 6.x:

```
$ sudo service fail2ban start
Starting fail2ban:                                         [  OK  ]
$ sudo chkconfig --add fail2ban
```

Also, restart the IPTables firewall service. This also flushes all banned IP addresses, so only do this once after the installation of Fail2Ban is completed:

```
$ sudo service iptables restart && service iptables status
iptables: Flushing firewall rules:                         [  OK  ]
iptables: Setting chains to policy ACCEPT: nat mangle filte[  OK  ]
iptables: Unloading modules:                               [  OK  ]
```

On RedHat 7.x:

```
$ sudo systemctl start fail2ban
$ sudo systemctl enable fail2ban
```

The firewall service on Centos/RedHat 7.x is called *firewalld'* and does not need to be restarted.

Check which IP adddresses have been banned

Check which IP addresses have been banned. Nothing will show if you have just previously flushed the banned IP addresses:

```
$ sudo  iptables -L --line-numbers | grep DROP
1    DROP       all  --  host86-142-19-185.range86-142.btcentralplus.com  anywhere
2    DROP       all  --  220.221.iconpln.net.id  anywhere
```

Line 1 shows the IP address of an experimental ban. It is possible to unban IP addresses if there genuinely was a valid reason for the failed login attempts, as follows, including a check that this IP address does not get DROP'ed by the firewall:

Unban trusted IP-addresses

If the banned IP address is known to be a friendly one (i.e. someone you trust may have accidentally used the incorrect access criteria in succession), then you can unban their IP adddress:

```
$ sudo fail2ban-client set sshd unbanip 86.142.19.185

or

$ sudo fail2ban-client get ssh-iptables actionunban 86.142.19.185
sudo  iptables -L --line-numbers | grep DROP
```

You will know that your IP address has been banned when you get a message like this when remotely trying to log in:

```
$ ssh [user]@[server]
ssh: connect to host [server] port 22: Connection refused
```

Note
If you find that your IP address has been banned but it does not appear on iptable's DROP list, then it has been restricted by the data centre's security system. You will need to contact the data centre and ask them to remove the IP restriction.

References: http://centoshelp.org/security/fail2ban/

RKHUNTER (Rootkit Hunter)

Installation

This process checks that no Rootkits have been installed in on the server. In the unlikely event that it finds one, it sends and email alert to the system administrator. Install it as follows:

```
$ sudo yum  -y install rkhunter
```

This installation adds a job to the list of other daily batch processes. Daily batch jobs start their execution at 4:05 am UTC and can be be viewed like this:

```
$ ls /etc/cron.daily/
...
rkhunter
```

Cron jobs are performed in sequence and in alpha-numeric order, which is the same order that the `ls` command lists files.

The final step in setting up the rootkit hunter is to establish a reference of file hashes, against which all system files are be compared to on a daily basis:

```
$ sudo rkhunter --propupd
[sudo] password for madman:
[ Rootkit Hunter version 1.4.0 ]
File created: searched for 167 files, found 137
```

Using RKHUNTER to scan the server

This step is performed by the daily cron process, and can manually be performed at any time.

```
$  sudo rkhunter -c --report-warnings-only
```

You should not get any alerts after a fresh server install.

Dealing with RKHUNTER Alerts

You may receive the following notification email if either:

- The operating system was updated or patched
- The operating system has been compromised

```
Subject: [rkhunter] Warnings found for s16972617
Please inspect this machine, because it may be infected.
```

This is followed by a more descriptive alert notification:

File size has changed

And a following-on email notification that either the size or the file-hash has changed, e.g.

```
Subject: rkhunter Daily Run on [server]
---------------------- Start Rootkit Hunter Scan ----------------------
Warning: The file properties have changed:
         File: /usr/bin/size
         Current inode: 184336503    Stored inode: 233033071
Warning: The file properties have changed:
         File: /usr/bin/strings
         Current inode: 184336504    Stored inode: 233032818
---------------------- End Rootkit Hunter Scan ------------------------
```

In this case it means that the 2 files in question where relocated on the storage device but were not actually changed, so we can safely mute this alert.

File's inode has changed

```
Subject: rkhunter Daily Run on [server]
---------------------- Start Rootkit Hunter Scan ----------------------
Warning: The file properties have changed:
         File: /usr/bin/curl
         Current inode: 233045587    Stored inode: 233039515
---------------------- End Rootkit Hunter Scan ------------------------
```

This occasionally happens on virtual machines that use SANs. We can safely mute this alert.

File Hash has changed

Another case is where the file hash has been changed. This means that the content of the file has been altered but the size remains unchanged:

```
Warning: The file properties have changed:
         File: /usr/bin/chattr
         Current hash: 5c8123ff9c92c6a361b465517c397a699670bba8
         Stored hash : d327e24dbddacbd6f39c54619abf695fcbeb386f6419246b73299c0d3d7928ff
```

Investigate which RPM package this file belongs to:

```
$ sudo rpm -qf /usr/bin/chattr
e2fsprogs-1.41.12-21.el6.x86_64
```

Check if the file has been changed since that RPM packages was installed. A null response means that there are no changes.

```
$ sudo rpm -Vf /usr/bin/chattr
```

If still in doubt, re-install the RPM package:

```
$ sudo yum reinstall e2fsprogs-1.41.12-21.el6.x86_64
...
```

It should now be safe to mute the alert. See the next step.

Muting RKHUNTER Alerts

If you are sure that there are no real issues, then the list of file hashes can be refreshed:

```
$ sudo rkhunter --propupd
[ Rootkit Hunter version 1.4.0 ]
File updated: searched for 167 files, found 138
```

Warning
All RKHUNTER alerts should be thoroughly investigated and understood before muting them.

Permanently removing annoying RKHUNTER alerts

Unscheduled updates of the PLESK server management system cause harmless and false alerts, but they are annoying. There are a few other alerts that can be safely ignored too. Stop these alerts by making the following configuration changes:

```
$ sudo nano /etc/rkhunter.conf
```

At line 309:

```
ALLOW_SSH_ROOT_USER=no
```

At line 359:

```
ENABLE_TESTS="all"
DISABLE_TESTS="suspscan hidden_procs deleted_files packet_cap_apps apps loaded_modules"
```

At line 825, comment this line out:

```
# XINETD_ALLOWED_SVC=/etc/xinetd.d/echo
```

Add this if the server is supported by PLESK:

```
# Matchi sysadmin says: Add this if your server is supported by PLESK
XINETD_ALLOWED_SVC=/etc/xinetd.d/ftp_psa
XINETD_ALLOWED_SVC=/etc/xinetd.d/poppassd_psa
XINETD_ALLOWED_SVC=/etc/xinetd.d/smtp_psa
XINETD_ALLOWED_SVC=/etc/xinetd.d/smtps_psa
XINETD_ALLOWED_SVC=/etc/xinetd.d/submission_psa
```

Setting the System Clock

It is important that the clocks on all servers are synchronized so that the time-stamped logging of information between servers is consistent. The NTP Daemon is used to synchronize the server's clock to an atomic clock.

Virtual Servers vs Physical Servers

Running the NTP Daemon on a Virtual Machine is not possible since the time clock for all guest virtual machines is owned by the hosting VM hypervisor (Virtuozzo in our case). The VM hypervisor itself is synchronized to an atomic clock, so therefore all virtual servers are synchronized too. It is also not possible to set the clock time manually on a virtual server. The only thing that can be set is the particular virtual server's time zone.

Network Time Protocol Service

Only do this on a dedicated (*tin*) server:

The Network Time Protocol Daemon ensures that the server is synchronized to a remote atomic clock using the NTP network time protocol. This protocol includes triangulation algorithms that compensate for the latency in the actual network delay over which the time signal is sent. If all servers are synchronized to an atomic clock, then they will be synchronized with each other. By default, the servers have all the configurations set up to the external atomic clocks. All that is required, is to install and start the NTP daemon and to set the daemon that it restarts should the server ever get rebooted:

```
$ sudo yum -y install ntp ntpdate ntp-doc
...
$ sudo /etc/init.d/ntpd start
Starting ntpd:                                             [  OK  ]
$ sudo chkconfig --levels 2345 ntpd on
```

This will gracefully change the clock on the server to the correct time.

Manually Changing the Server's Wall-Clock Time

On a physical server, and for testing purposes usually, you can also force a server time change by manually turning the NTPD service and then changing the time and date with this command:

```
$ sudo date -s 'Mon Nov 17 11:14:21 UTC 2013'
```

More about the *date* command can be found with this command:

```
$ man date
```

Remember to turn the NTPD service back on again when testing is completed - this will gracefully change the wall-clock time to the correct time.

Setting the Server's Time Zone

Do this on both Virtual or Physical servers:

Check your server's current Wall-Clock time zone:

```
$ date
Fri Apr 22 05:21:56 CDT 2016
```

In this case it is CDT (Central Dailight Time in the USA - Wisconsin etc...). All Matchi's servers are set to 'Europe/London', which means that depending on when you run the *date* command, you either get BST (British Summer Time) or GMT (Greenwich Mean Time).

All possible timezone supported by UNIX are in /usr/share/zoneinfo/ - there are more than 1700 of them. We are interested in the file /usr/share/zoneinfo/Europe/London that contains everything that needs to be known about clocks in London, birthplace of time and also the coolest city in the world.

Remove the current symbolic link at /etc/localtime and re-point it to the correct timezone definition file:

```
$ sudo rm -fr /etc/localtime
$ sudo ln -s /usr/share/zoneinfo/Europe/London  /etc/localtime
```

Check that you now get the correct time zone:

```
$ date
Fri Apr 22 11:32:43 BST 2016
```

You also need to update the timezone file for the local SMTP mail daemon, Postfix. Postfix does not like symbolic links, so you need to copy the actual time zone file:

```
$ sudo mkdir /var/spool/postfix/etc
$ sudo cp /usr/share/zoneinfo/Europe/London /var/spool/postfix/etc/localtime
$ sudo chown -R postfix /var/spool/postfix/etc
```

Restart Postfix for this change to take effect. For RedHat 6.x, do:

```
$ sudo /etc/init.d/postfix restart
Shutting down postfix:                                     [  OK  ]
Starting postfix:                                          [  OK  ]
```

For RedHat 7.x, do:

```
$ sudo systemctl restart postfix
```

From this point onwards, all date-based calculations will be based on this time zone.

TRIPWIRE

Tripwire is an intrusion detection system (IDS) that monitors critical system files on the server and alerts if they have been destroyed or modified by an adversary, or by mistake. It allows the system administrator to know immediately what was compromised on the system to be able to repair it. The first time Tripwire is run, it stores checksums, exact sizes and other data of all the selected files in a database. The successive runs check whether every file still matches the information in the database and report all changes.

This has not yet been installed on the servers.

Server Mail Services

Emails sent from servers, whether they are customer notification emails or system alerts, are all routed through Google GMail service using the email domain of 'matchi.biz'. This improves the chances that emails will be delivered, in particular, where the target recipient's email host has been poorly configured. Furthermore, the MX domain 'matchi.biz' has its domain key registered (DKIM) and further supports deliverability of emails to spam-adverse recipients. To achieve this, the default mail delivery configuration of Postfix needs to be converted into a mail relay. To use GMail as the end delivery host, some stringent authentication settings need to be set.

The email address and Google account that forms the core of Matchi's automated system emailing is sysadmin@matchi.biz. This is a 'real' account on Google Apps.

Setting up the Google Authentication

Add the following to the file /etc/postfix/sasl_passwd, where XXXXXX is the password that was set for the sysadmin@matchi.biz account on Google Appls. Note that the square brackets are required:

```
[smtp.gmail.com]:587 sysadmin@matchi.biz:XXXXXXXX
```

Convert the file into 'database file' that is readable by Postfix:

```
$ cd /etc/postfix
$ sudo postmap sasl_passwd
```

Setting up the TLS Certficated

Create file /etc/postfix/tls_policy containing this text:

```
[smtp.gmail.com]:587 encrypt
```

Convert the file also into 'database file' that is readable by Postfix:

```
$ cd /etc/postfix
$ sudo postmap tls_policy
```

Set up the SMTP Relay

Add the following to the file /etc/postfix/main.cf, and replace SERVERALIAS with the friendly server name, e.g. mweb01, mapp01, etc...

```
myhostname = SERVERALIS.matchi.biz

relayhost = [smtp.gmail.com]:587

smtp_sasl_auth_enable = yes
smtp_sasl_password_maps = hash:/etc/postfix/sasl_passwd
smtp_sasl_security_options = noanonymous
smtp_sasl_tls_security_options = noanonymous
smtp_sasl_mechanism_filter = plain

# TLS parameters
smtp_use_tls = yes
smtp_tls_CAfile = /etc/ssl/certs/ca-bundle.crt
smtpd_use_tls = yes
smtpd_tls_session_cache_database = btree:${data_directory}/smtpd_scache
smtp_tls_session_cache_database = btree:${data_directory}/smtp_scache
smtp_tls_note_starttls_offer = yes
smtp_tls_policy_maps = hash:/etc/postfix/tls_policy

# Debugging & Logging
debug_peer_list=smtp.gmail.com
debug_peer_level=2
```

Restart the Postfix Service

Restart the Postfix Service to actualize the settings:

```
$ sudo /etc/init.d/postfix restart
Shutting down postfix:                                     [  OK  ]
Starting postfix:                                          [  OK  ]
```

Test the SMTP Relay

Send a text email from the server to a test account and check that it arrives. Also look at the internal message details and ascertain that it arrived Google's GMail SMTP service, smtp.gmail.com.

```
$ mail -s 'Test for new Postfix configuration' test@matchi.biz <<!
This is a test
!
```

Securing the Server

The attack surface of the server needs to be reduced to a minimum. A simple way to achieve this (for now) is to keep the services that need to be accessed from the public internet to a minimum. You can view the open ports of a server from the outside world by running the *nmap* command from your local machine. In this example, the attack surface is larger than what it needs to be, with too many services running and potentially causing a vulnerability:

```
$ nmap [server]

Nmap scan report for [server]
Host is up (0.041s latency).
Not shown: 985 closed ports
PORT     STATE SERVICE
21/tcp   open  ftp
22/tcp   open  ssh
25/tcp   open  smtp
53/tcp   open  domain
80/tcp   open  http
106/tcp  open  pop3pw
110/tcp  open  pop3
143/tcp  open  imap
443/tcp  open  https
465/tcp  open  smtps
587/tcp  open  submission
993/tcp  open  imaps
995/tcp  open  pop3s
3306/tcp open  mysql
8080/tcp open  http-proxy
8443/tcp open  https-alt
```

Services that we definitely will never require are IMAP and POP services, so we can stop them on Red Hat 6.x as follows:

```
$ cd /etc/init.d
$ sudo ./courier-imapd stop
[sudo] password for madman:
Stopping Courier IMAP server:                              [  OK  ]
$ sudo ./courier-imaps stop
Stopping Courier IMAP server with SSL/TLS support:         [  OK  ]
$ sudo ./courier-pop3d stop
Stopping Courier POP3 server:                              [  OK  ]
$ sudo ./courier-pop3s stop
Stopping Courier POP3 server with SSL/TLS support:         [  OK  ]
```

On Red Hat 7.x use the `systemctl` command.

We then also ensure that these services will not be restarted when the server is rebooted, but permanently turnining then off, again for RedHat 6.x:

```
$ sudo chkconfig courier-imapd off
$ sudo chkconfig courier-imaps off
$ sudo chkconfig courier-pop3d off
$ sudo chkconfig courier-pop3s off
$ sudo chkconfig courier-authdaemon off
```

On Red Hat 7.x use the `systemctl` command:

```
$ sudo systemctl disable courier-imapd
$ sudo systemctl disable courier-imaps
$ sudo systemctl disable courier-pop3d
$ sudo systemctl disable courier-pop3s
$ sudo systemctl disable courier-authdaemon
```

You can view a summary of services that will start if the server is rebooted - for Red Hat 6.x:

```
$ chkconfig | grep on

cgconfig       0:off   1:off   2:off   3:off   4:off   5:off   6:off
crond          0:off   1:off   2:on    3:on    4:on    5:on    6:off
cups           0:off   1:off   2:on    3:on    4:on    5:on    6:off
fail2ban       0:off   1:off   2:on    3:on    4:off   5:off   6:off
httpd          0:off   1:off   2:off   3:on    4:off   5:off   6:off
ipset          0:off   1:off   2:on    3:on    4:on    5:on    6:off
iptables       0:off   1:off   2:on    3:on    4:on    5:on    6:off
maldet         0:off   1:off   2:on    3:on    4:on    5:on    6:off
messagebus     0:off   1:off   2:on    3:on    4:on    5:on    6:off
modules_dep    0:off   1:off   2:on    3:on    4:on    5:on    6:off
netconsole     0:off   1:off   2:off   3:off   4:off   5:off   6:off
netfs          0:off   1:off   2:off   3:off   4:on    5:on    6:off
network        0:off   1:off   2:on    3:on    4:on    5:on    6:off
nfslock        0:off   1:off   2:off   3:on    4:on    5:on    6:off
portreserve    0:off   1:off   2:on    3:off   4:on    5:on    6:off
psa            0:off   1:off   2:on    3:on    4:on    5:on    6:off
rabbitmq-server 0:off  1:off   2:on    3:on    4:on    5:on    6:off
restorecond    0:off   1:off   2:off   3:off   4:off   5:off   6:off
rpcbind        0:off   1:off   2:on    3:off   4:on    5:on    6:off
rpcgssd        0:off   1:off   2:off   3:on    4:on    5:on    6:off
rsyslog        0:off   1:off   2:on    3:on    4:on    5:on    6:off
saslauthd      0:off   1:off   2:off   3:on    4:off   5:off   6:off
sshd           0:off   1:off   2:on    3:on    4:on    5:on    6:off
sw-cp-server   0:off   1:on    2:off   3:off   4:off   5:off   6:off
sw-engine      0:off   1:off   2:on    3:on    4:on    5:on    6:off
udev-post      0:off   1:on    2:on    3:off   4:on    5:on    6:off
vzquota        0:on    1:on    2:on    3:on    4:on    5:on    6:on
vzreboot       0:off   1:off   2:off   3:off   4:off   5:off   6:on
xinetd         0:off   1:off   2:on    3:on    4:on    5:on    6:off
```

At the end of this rationalization of publicly-visible services, you can view which services are still visible to the public by looking at the server from your own machine with the `nmap` command:

```
nmap [server]

Nmap scan report for [server]
Host is up (0.044s latency).
Not shown: 989 closed ports
PORT     STATE SERVICE

22/tcp   open  ssh
53/tcp   open  domain
80/tcp   open  http
106/tcp  open  pop3pw
443/tcp  open  https
465/tcp  open  smtps
587/tcp  open  submission
3306/tcp open  mysql
8080/tcp open  http-proxy
8443/tcp open  https-alt
```

Sizing the Logical Volumes on a Dedicated Server

By default, a dedicated server ('tin') comes with the logical volumes sized very small, even though there is a plenty of capacity (1TB usually). You can see the default sizings:

```
$ df -h
Filesystem             Size  Used Avail Use% Mounted on
/dev/md1               4.0G  225M  3.7G   6% /
devtmpfs               7.8G     0  7.8G   0% /dev
tmpfs                  7.8G     0  7.8G   0% /dev/shm
tmpfs                  7.8G   33M  7.8G   1% /run
tmpfs                  7.8G     0  7.8G   0% /sys/fs/cgroup
/dev/mapper/vg00-usr   4.8G  2.4G  2.2G  53% /usr
none                   7.8G     0  7.8G   0% /tmp
/dev/mapper/vg00-var   4.8G  3.5G  1.1G  78% /var
/dev/mapper/vg00-home  4.8G  3.8G  846M  82% /home
tmpfs                  1.6G     0  1.6G   0% /run/user/10002
```

```
$ echo $(( `df -m | sed '1d' | awk '{print $2 "+"}' | tr -d '\n' | sed -e 's/.$//'` ))
60242
```

This totals 60G, and typically we have 1000G available. The `var` volume is impossibly small and the `home` volume could also so with a little more space. To extend the volumes and leave some space for future expansion, we do a 'physical volumne show' `pvs` to see how much physical storage is still available:

```
$ sudo pvs
  PV         VG   Fmt  Attr PSize   PFree
  /dev/md3   vg00 lvm2 a--  925.51g 910.51g
```

Extend the `var` logical volume by a further 400G:

```
$ sudo lvextend -L +400G /dev/mapper/vg00-var
  Size of logical volume vg00/var changed from 5.00 GiB (1280 extents) to 405.00 GiB (103680 extents).
  Logical volume var successfully resized.
```

Also extend the `home` logical volume by 50G:

```
$ sudo lvextend -L +50G /dev/mapper/vg00-home
  Size of logical volume vg00/home changed from 5.00 GiB (1280 extents) to 55.00 GiB (14080 extents).
  Logical volume home successfully resized.
```

A 'logical volume show' `lvs` command shows that we have been successful:

```
$ sudo lvs
  LV   VG   Attr       LSize   Pool Origin Data%  Meta%  Move Log Cpy%Sync Convert
  home vg00 -wi-ao----  55.00g
  usr  vg00 -wi-ao----   5.00g
  var  vg00 -wi-ao---- 405.00g
```

The physical disk consumption needs to be aligned with the new logical volume sizings. We can see how these volumes are actually mounted in the operating system:

```
$ mount | grep var
/dev/mapper/vg00-var on /var type ext4 (rw,noatime,quota,usrquota,data=ordered)
$ mount | grep home
/dev/mapper/vg00-home on /home type ext4 (rw,noatime,quota,usrquota,data=ordered)
```

Both volumes use the EXT4 file system. For this file system we use the `resize2fs<//code> to extend the physical volume:`

```
$ sudo resize2fs /dev/mapper/vg00-var
resize2fs 1.42.9 (28-Dec-2013)
Filesystem at /dev/mapper/vg00-var is mounted on /var; on-line resizing required
old_desc_blocks = 1, new_desc_blocks = 26
The filesystem on /dev/mapper/vg00-var is now 106168320 blocks long.

$ sudo resize2fs /dev/mapper/vg00-home
resize2fs 1.42.9 (28-Dec-2013)
Filesystem at /dev/mapper/vg00-home is mounted on /home; on-line resizing required
old_desc_blocks = 1, new_desc_blocks = 4
The filesystem on /dev/mapper/vg00-home is now 14417920 blocks long.
```

A quick check shows that we have also been successful in extending the physical volume:

```
$ df -h
Filesystem            Size  Used Avail Use% Mounted on
/dev/md1              4.0G  225M  3.7G   6% /
devtmpfs              7.8G     0  7.8G   0% /dev
tmpfs                 7.8G     0  7.8G   0% /dev/shm
tmpfs                 7.8G   33M  7.8G   1% /run
tmpfs                 7.8G     0  7.8G   0% /sys/fs/cgroup
/dev/mapper/vg00-usr  4.8G  2.4G  2.2G  53% /usr
none                  7.8G     0  7.8G   0% /tmp
/dev/mapper/vg00-var  399G  3.6G  379G   1% /var
/dev/mapper/vg00-home  55G  3.8G   49G   8% /home
tmpfs                 1.6G     0  1.6G   0% /run/user/10002
```

File Sharing between Servers

TODO: Review what directories need to be shared

File Sharing over the Network: Within the local server network, we use the Network File System (NFS) protocol to share directries.

By convention, file systems on a server that are exported to other servers are hosted off a root-directory called ‘/exports’ and file systems from a remote server are mounted locally on a root-directory called ‘/mnt’.

Preparing for Content Network Delivery: A root-directory will hold all static data so that it is easily identifiable for future mapping to content delivery networks or other storage array systems. It is named ‘/exports/cdn/[build]’, where [build] is the ISO-date of the production release preceded with a prefix ‘build_’, and is created as in the following example, where the build date is 1st of June 2013: build_20130601. The directory is prepared with the correct ownership and access rights:

```
madman@s16972616 mweb1 ~ $ sudo mkdir -p /exports/cdn/build_20130601
madman@s16972616 mweb1 ~ $ sudo chmod -R g+w /exports/cdn/build_20130601
madman@s16972616 mweb1 ~ $ ls -al /exports/cdn
total 12
drwxr-xr-x  3 root     root     4096 Jun 10 11:36 .
dr-xr-xr-x 23 root     root     4096 Jun 10 11:36 ..
```

Export configurations to the other servers: By sharing the relevant configuration files for read-only use by other servers allows any changes in a configuration to be instantly visible across the system. For file sharing within a secured environment, such as the PROD environment in a data center, it is sufficient to share files unencrypted over NFS (Network File System) – a network protocol similar to Windows CIFS. The Content Management System’s configuration file, configuration,php, needs to be shared from the webserver. This file resides in a directory /var/www/html/[build-instance], where [build-instance] is the name of the build. As new builds are deployed, this instance name will change, and so will the actual directory name. The solution is to create a new sharable directory and to manually create a file-link from the [build-instance] directory to there.

Install NFS support

```
madman@s16972616 mweb1 ~ $ sudo yum install nfs-utils nfs-utils-lib
. . .
Complete!
```

Configure nfs, nfslock and rpcbind to run as daemons and start then up:

```
madman@s16972616 mweb1 ~ $ sudo chkconfig --level 35 nfs on
madman@s16972616 mweb1 ~ $ sudo chkconfig --level 35 nfslock on
madman@s16972616 mweb1 ~ $ sudo chkconfig --level 35 rpcbind on
madman@s16972616 mweb1 ~ $ sudo service rpcbind start
madman@s16972616 mweb1 ~ $ sudo service nfslock start
madman@s16972616 mweb1 ~ $ sudo service nfs start
```

Create a sharable directory on the webserver where the configuration files are shared to:

```
madman@s16972616 mweb1 ~ $ sudo mkdir /var/www/html/config
[sudo] password for madman:
```

Make this directory exportable by adding it using an editor to the /etc/exports file:

```
madman@s16972616 mweb1 ~ $ sudo nano /etc/exports
```

Add this to the file:

```
/var/www/html/config            mapp1(ro,sync,no_root_squash,no_subtree_check)
```

Where:

- ro: The client can only read within the shared directory
- sync: Sync confirms requests to the shared directory only once the changes have been committed.
- no_subtree_check: This option prevents the subtree checking. When a shared directory is the subdirectory of a larger filesystem, nfs performs scans of every directory above it, in order to verify its permissions and details. Disabling the subtree check may increase the reliability of NFS, but reduce security.
- no_root_squash: This phrase allows root to connect to the designated directory

Activate the latest export settings:

```
madman@s16972616 mweb1 ~ $ sudo exportfs –a
```

Make a symbolic link for the web application's build instance's configuration file (configuration.php). Since the web server may potentially be hosting multiple web applications, it is useful the name the symbolic linked file after the instance build name that it was exported from. We export the configuration file as file named [build-instance].conf, Assuming we have a build instance called 'demo', the configuration file in the exported /var/www/html/config directory is named demo.conf.

```
madman@s16972616 mweb01 ~ $ cd /var/www/html/demo
madman@s16972616 mweb01 /var/www/html/demo $ ls configuration.php
configuration.php
madman@s16972616 mweb01 /var/www/html/demo $ cd ../config
madman@s16972616 mweb01 /var/www/html/config $ sudo ln -s ../demo/configuration.php demo.conf
madman@s16972616 mweb01 /var/www/html/config $ ls -al
total 8
drwxr-xr-x 2 root root 4096 Jun  3 15:00 .
drwxr-xr-x 3 root root 4096 Jun  3 14:59 ..
lrwxrwxrwx 1 root root   36 Jun  3 15:00 demo.conf -> ../demo/configuration.php
```

Set up the NFS client:

```
madman@s16972617 mapp01 ~ $ sudo yum install nfs-utils nfs-utils-lib
. . .
Complete!
```

If the NFS-client installation was successful, one should be able to query the NFS server (mweb1) what has been exported to it:

```
madman@s16972617 mapp01 ~ $ sudo showmount -e mweb01
Export list for mweb01:
/var/www/html/config mapp01
```

Make mount point on the client where the remote share is to be mounted, then then mount it. Test that the mount contains what you expected – in this case we expect the demo.conf to appear in there. It is also a symbolic link to a different file on the remote server, so we need to make sure that we can read the content:

```
madman@s16972617 mapp01 ~ $ sudo mkdir /mnt/config
Mount the exported directory from server mweb01 here:
madman@s16972617 mapp01 ~ $ sudo mount mweb01:/var/www/html/config  /mnt/config
madman@s16972617 mapp01 ~ $ ls -al /mnt/config
```

Finalizing the Server Installations

Some final installation steps are required before the server can be considered ready for production use, to ensure basic recovery from failure and security.

Rebooting the server

Not only should we be able to demonstrate a successfully reboot of each server (should this ever be required), but it is recommended that the server memory be flushed after all the configuration changes that have been applied to it in the course of the installation. A server can either be rebooted from the PLESK control panel, or from the command line as shown here:

Important:

Remember to use the *-r* option (which means *restart*) so that the server comes back up again, rather than just *halt*-ing the server.

```
madman@s16972617 mapp1 ~ $ sudo shutdown -r now

Broadcast message from madman@s16972617.onlinehome-server.info
        (/dev/pts/0) at 13:12 ...

The system is going down for reboot NOW!
madman@s16972617 mapp1 ~ $ Connection to s470071337.websitehome.co.uk closed by remote host.
Connection to s470071337.websitehome.co.uk closed.
```

Attempt connecting to the server after 10 seconds or so to give the server some time to come back up. From your local terminal, open an SSH session again:

```
bignosethethird@terminator $ ssh madman@mapp01
#######################################################
# AUTHORIZED ACCESS ONLY.                             #
# You are entering into a secured area. Your IP, Login #
# Time, and Username has been noted and sent to the   #
# server administrator. This service is restricted to  #
# authorized users only. All activities on this system #
# are logged. Unauthorized access will be fully       #
# investigated and reported to the relevant law       #
# enforcement agencies. Disconnect IMMEDIATELY if you  #
# are not an authorized user.                         #
#######################################################

Last login: Mon Jun 24 13:06:22 1066 from host86-154-124-192.range86-154.btcentralplus.com
#######################################################
#       Welcome to another happy server of            #
#  __  __   _ _____ ____ _   _ _   _  _____          #
# |  \/  |  / \|_   _/ ___| | | (_) | |__ (_)____     #
# | |\/| | / _ \ | || |   | |_| | | | '_ \| |_  /     #
# | |  | |/ ___ \| || |___|  _  | |_| |_) | |/ /____  #
# |_|  |_/_/   \_\_| \____|_| |_|_(_)_.__/|_/_______|  #
#                                                     #
#######################################################
```

Success!

The server, which is very far away deep in a data centre, has rebooted with no problems.

Or:

Oh Dear!

It is possible to remedy any booting problems by contacting the hosting providers with the contract and server details via email on support@1and1.co.uk, or by telephone: +44 333 336 5691 / +44 871 641 2121

System Administration Tasks

It is occasionally required to perform some system-level tasks such as clearing storage space up or to migrate deployments between environments, e.g. from a testing environment to a production environment. These processes are described below. Most of these processes can be scripted to either manually execute or executed on a batch basis (via a *cron* batch job).

Setting up the System Administration Work Area

The system administration tasks are normally executed by user 'madman'. All system administration scripts are held in the /usr/local/bin directory. This directory is already on the user's path and you can test this by displaying the PATH environment variable:

```
$  echo $PATH
/usr/local/bin:/bin:/usr/bin:/usr/local/sbin:/usr/sbin:/sbin
```

Check who else has attempted to log into each server

All failed SSH-logins are noted in the file /var/log/messages. Each week a new file is created and the previous week's file is saved as a date-stamped file name. Message files over 30 days old are automatically deleted.

```
madman@s16972617 mapp1 ~ $ ls -al /var/log/mess*
-rw------- 1 root root    869 Jun  3 02:59 /var/log/messages
-rw------- 1 root root 268573 May 19 02:48 /var/log/messages-20130519
-rw------- 1 root root  20711 May 26 02:41 /var/log/messages-20130526
-rw------- 1 root root 224355 Jun  2 03:52 /var/log/messages-20130602
```

Scan the message files for lines containing text similar to:

```
Jun  2 18:41:01 s16972617 named[5549]: client 91.212.124.83#8900: query (cache) 'www.ru/A/IN' denied
```

Since the pertinent term is "denied", it is sufficient to *grep* for this term alone and to extract the IP addresses of the offenders:

```
madman@s16972617 mapp1 ~ $ sudo grep denied /var/log/messages* | cut -f 5 -d ":"  | sed -e 's/\s*client\s*//' | sed -e 's/#.*//' | sort –u
109.163.233.53
173.232.104.215
173.242.114.26
173.242.121.199
178.18.19.140
180.76.5.164
... etc.
```

For added geo-political amusement, you can check where these attacks originate from by looking up the IP addresses:

```
madman@s16972617 mapp1 ~ $ whois 180.76.5.164 | grep address
address:        Baidu Plaza, No.10, Shangdi 10th street,Haidian District Beijing,100080
address:        10th Floor No.6 2nd North Street Haidian District Beijing,100080
```

You can also look up all the attacks in one command:

```
madman@s16972617 mapp1 ~ $ sudo grep denied messages* | cut -f 5 -d ":" | cut -f 1 -d "#" | sed -e 's/client//' | sort -u | xargs -I {} whois {} | grep address
```

CRON System Scheduler

Overview

All batch processes are driven from the CRON system scheduler. There is a working CRON scheduler on each server which perform basic server adminsitration and security management tasks (see rkhunter). Large processing jobs are from the application server so that production performacen is not impacted. It is unavoidable that a few batch jobs need to run on the production web servers too but these are kept to a minimum. As rule, batch jobs are scheduled to run at around 4am UTC when minimum impact on the production infrastructure will be experienced by users.

Batch jobs can be scheduled to run hourly, daily, weekly or any other recurring time pattern within limitations. A process can be scheduled by dropping its script (or symbolic link) in the relevant directory, or by adding a line to what is know as a <code>crontab file.

Any output that a batch job creates to either STDOUT or STDERR is collected and on termination of a batch job is emailed to the sysadmin. It is good practise to only explicitly output text to STDOUT from scripts on error conditions that justify a sysadmin being notified by email. Error messages are written to a log file, as is all other output from the script such as debug, trace, info and warning messages.

There are 5 approaches of how batch jobs can be invoked:

Approach 1: Local, non-root user level

Batch jobs can be set up by running `crontab -e` from the local account. The batch job will be run under the this user's account. There is currently no need to run any jobs at this level since all batch run under user `root`.

Approach 2: Root user level

Batch jobs can be set up by running `crontab -e` when logged in as user `root` and the batch job will be run with the priviledges of the `root` account. Each uncommented line is new scheduled job and is in the format:

```
.---------------- minute (0 - 59)
|  .------------- hour (0 - 23)
|  |  .---------- day of month (1 - 31)
|  |  |  .------- month (1 - 12) OR jan,feb,mar,apr ...
|  |  |  |  .---- day of week (0 - 6) (Sunday=0 or 7) OR sun,mon,tue,wed,thu,fri,sat
|  |  |  |  |
*  *  *  *  * command-to-be-executed
```

For the sake of simplicity we avoid this approach too.

Approach 3: Any user crontab file

Batch jobs can be set up by editing the file `/etc/crontab` using any editor as a sudo user. Batch jobs need to specify the user under which the job is run, which is why `user-name` has to be specified:

```
.---------------- minute (0 - 59)
|  .------------- hour (0 - 23)
|  |  .---------- day of month (1 - 31)
|  |  |  .------- month (1 - 12) OR jan,feb,mar,apr ...
|  |  |  |  .---- day of week (0 - 6) (Sunday=0 or 7) OR sun,mon,tue,wed,thu,fri,sat
|  |  |  |  |
*  *  *  *  * user-name command-to-be-executed
```

Approach 4: Any user crontab directory

An alternative is to drop the crontab file into the `/etc/cron.d` directory, in which the file can be given any name. Third-party vendors use this approach to deploy a scheduled job for their product. The crontab file format is the same as that for the `/etc/crontab` file, since the execution user needs to be specified. This is an example of the batch job configured in `/etc/cron.d/maldet_pub` to run the `maldet` malware checking process every 10 minutes as user `root`:

```
*/10 * * * * root /usr/local/maldetect/maldet --mkpubpaths >> /dev/null 2>&1
```

Approach 5: Drop batch scripts into a cron directory

Instead of having to write a crontab file with error-prone recurrance specifiers, it is possible to drop the batch scripts (or better: make symbolic links to them) into the either one of the following directories:

- `/etc/cron.hourly`: All scripts in this directory will be executed once per hour, on the hour, in the natural listing order.
- `/etc/cron.daily`: All scripts in this directory will be executed once per day at 4am, executed in the order of listing.
- `/etc/cron.weekly`: All scripts in this directory will be executed once per week on Monday at 4am, executed in the order of listing.

It is customary to prefix the filenames with a 2-digit value to indicate the order of execution, e.g.

```
$ ls -al /etc/cron.hourly/
total 44
drwxr-xr-x  2 root root  4096 Apr 22 16:10 .
drwxr-xr-x 95 root root 12288 Apr 15 11:09 ..
lrwxrwxrwx  1 root root    62 Oct 29  2014 01restart_fsm_daemon_ORG.sh -> /usr/local/etc/cron.hourly/01restart_fsm_daemon_ENVIRONMENT.sh
lrwxrwxrwx  1 root root    62 Sep 27  2014 01restart_fsm_daemon_PROD.sh -> /usr/local/etc/cron.hourly/01restart_fsm_daemon_ENVIRONMENT.sh
lrwxrwxrwx  1 root root    55 Apr 16 17:48 08update_data_ORG.sh -> /usr/local/etc/cron.hourly/08update_data_ENVIRONMENT.sh
lrwxrwxrwx  1 root root    55 Apr 16 17:48 08update_data_PROD.sh -> /usr/local/etc/cron.hourly/08update_data_ENVIRONMENT.sh
lrwxrwxrwx  1 root root    59 Apr 16 17:43 10update_keywords_ORG.sh -> /usr/local/etc/cron.hourly/10update_keywords_ENVIRONMENT.sh
lrwxrwxrwx  1 root root    59 Apr 16 17:43 10update_keywords_PROD.sh -> /usr/local/etc/cron.hourly/10update_keywords_ENVIRONMENT.sh
lrwxrwxrwx  1 root root    60 Nov 29 21:41 12update_ipdetails_PROD.sh -> /usr/local/etc/cron.hourly/12update_ipdetails_ENVIRONMENT.sh
lrwxrwxrwx  1 root root    62 Apr 21 15:53 66delete_bad_members_PROD.sh -> /usr/local/etc/cron.hourly/66delete_bad_members_ENVIRONMENT.sh
```

These batch jobs are executed every hour on the hour and this way there is no doubt what the order is that they will be executed in. Failures from multiple batch jobs are consolidated into one email to the sysadmin, rather than a separate email for each batch, as would be the case for the other approached to CRON jobs. This is the preferred way to control the execution of batch jobs, as there can potentially be many batch jobs in a complex system.

Note
We use symbolic links to the actual files that are deployed on the `/usr/local/etc/cron.hourly` directory sub-tree. This makes deployment up updated scripts easier, and also provides an element of protection from accidental deployment of scripts out of the Subversion code repository. Also note that the name of the symbolic link is different to the name of the real file being linked to. This is a trick used for running scripts for multiple environments on one server and some extra coding inside the script itself makes this process work.

Backing up and Restoring of Servers

Backup File Conventions

The convention is to back up and package the relevant content on each server as separate packages. The packages are implicitly grouped together by the name of the backup files, which is constructed from the build name and date that the backup was performed on, where:

- The build is named based on the release date of the build, in the form: build_YYYYMMDD
- The short-name source server alias is be added, e.g. mweb1
- The backup date is added to the backup file in the form YYYYMMDD
- The term _backup is added to the file name to remove any ambiguity of its origins and purpose
- The file extension is added depending on the type of backup:
  - Application code: tar.gz – which means lots of files archived together and compacted using GZIP compression
  - Data: sql.gz – One SQL file which will recreate the entire database, that is compacted using GZIP compression
  - Binary Dump: dump.gz – One binary file that is created from an application-specific backup tool, for example Subversion uses the dump command.
  - Images: tar – which means lots of files archived together but not compressed as the images themselves are already compressed (although providing persistence will be taken care of in future by the Cloud Flare service)

Example of a Joomla backup is: build_20130716_mweb1_20130720_backup.tar.gz

Backup File Creation

The process of backup file creation is run from the APP server to minimize impact on the DB and WEB servers. Data is extracted from the source servers via SSH or NFS file share - in the case of WEB server data, or a MySQL data link - in the case of database data. The backup files are created on the APP server. The backup process scripts are run in succession once a day shortly after the server's local midnight and is scheduled with a cron job. The scripts are:

- backup_remote_database.sh – makes a backup of the latest database build on the remote DB server
- backup_remote_application.sh - makes a backup of the latest website build on the remote WEB server

The scripts require no parameters and reside in the /home/madman/bin directory. Output of all the scripts is logged to the log file /var/log/matchi. When either of these scripts are run in batched mode in cron, an email is sent to the System Administrator if there has been a failure.

Backup File Location

Backup Files are located in the directory /backups on the APP server. Only user *root* can write to this directory, but all users (well, there is currently only one other user, 'madman'), can read it.

Off-Site Backup File Vaulting

The latest back-up files are also copied to Google Drive. Backup files can be manually retrieved from there in case of catastrophe. The account details for Google Drive are:

- User: matchi.biz@gmail.com
- Password: [*****]
- URL: https://drive.google.com

TODO: Vaulting scripts still needs to be added to the backup process. Due to limited available space, rigorous housekeeping is required.

Setting up the batch Backup Job

The standard scheduler is *cron*. In Red Hat, a dedicated *cron*-job exists that executes all scripts in the directory /etc/cron.daily, starting at 04:02 local server time. The scripts are executed in the order of their file name. Since it is desirable to execute backup scripts before any other daily processes, the batch scripts for backups are named 01… and 02… so that they will be processed first. Create the script /etc/cron.daily/01backup_remote_database.sh:

madman@s16972617 mapp1 ~ $ sudo nano /etc/cron.daily/01backup_remote_database.sh Password:

Enter this:

```
#!/bin/bash
if [[ -x /home/madman/bin/backup_remote_database.sh ]]
then
  /home/madman/bin/backup_remote_database.sh
  exit $?
else
  exit 99
fi
```

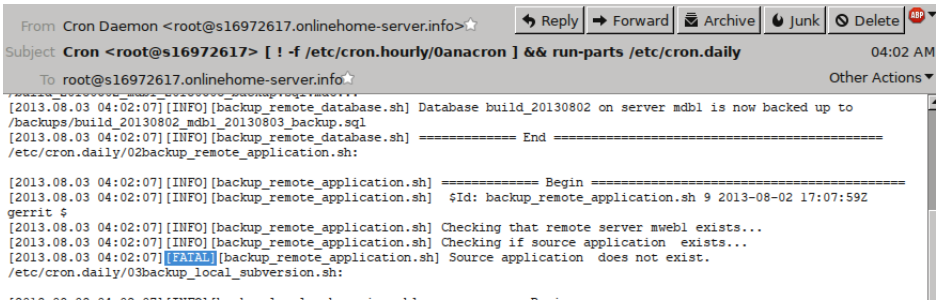
Save and exit. Make the script executable:

```
madman@s16972617 mapp1 ~ $ sudo chmod +x /etc/cron.daily/01backup_remote_database.sh
```

Apply a similar approach for all other required batch jobs and set the order in which they need to be executed using the file naming technique, e.g. create a script /etc/cron.daily/02backup_remote_application.sh, etc…

Execution of Batch Jobs

When any one of the batch jobs fail to complete successfully, then a notifications email is sent to user 'root', which is redirected to the assigned system administrator. The email contains all the STDOUT console output that generated during the execution of the batch jobs



If there no problems were encountered during the execution of the batch jobs, then no notification is sent. By convention, all logged output of MATCHI-specific processes is held in the file /var/log/matchi. The file can retrospectively be searched, using tools such as grep, for events of interest.

Validating of backup files To reduce the risk of backup file getting corrupted for whatever reason as they are copied between environments, a small check file is created when the backup file is created that should accompany the backup file at all times. The validity of the backup can be checked for at any point using the "md5sum –c" command against the actual checksum file:

madman@s16972617 mapp1 ~ $ ls -alrt /backups/ . . -rw-r--r-- 1 root root 118566221 Aug 3 04:02 build_20130802_mdb1_20130803_backup.sql -rw-r--r-- 1 root root 83 Aug 3 04:02 build_20130802_mdb1_20130803_backup.sql.md5 . . madman@s16972617 mapp1 ~ $ md5sum -c /backups/build_20130802_mdb1_20130803_backup.sql.md5 /backups/build_20130802_mdb1_20130803_backup.sql: OK

How to install an Application from backup on a new Server

Copy the backup file to the target server using SCP:

madman@s16972616 xxxxx ~ $ scp build_YYYYMMDD_YYYYMMDD_mweb1_backup.tar.bz2 madman@targetserver:~/.

If you have correctly set up the public keys between servers, then you should not be prompted for a password and the file should land in the home directory of the "madman" account. On the target server, unpack the backup file:

madman@targetserver ~ $ cd /var/www/html madman@targetserver /var/www/html $ sudo tar -xjvf ~/build_YYYYMMDD_YYYYMMDD_mweb1_backup.tar.bz2 Password:

This will create a new directory /var/www/html/build_YYYYMMDD that holds the Joomla application files. Make the new installation accessible to the LIGHTTP-daemon, by recursively (see the –R option) changing the owner and group to lighttpd: madman@s16972616 taergetserver /var/www/html $ sudo chown -R lighttpd:lighttpd matchi

How to Restore a Database Backup to a new database

The database backup was created in the backup script with a command like this:

```
$ mysqldump -h[servername] -u[dbusernamе] -p[password] --routines [build_YYYYMMDD] >  build_YYYYMMDD_$(date +%Y%m%d)_mdb1_backup.sql
```

Restoring is the reverse of this process: On the target database server, restore the database by running the database backup into a newly-create database. First create the target database. Then run the SQL backup file into the new database using the database's authentication credentials, as per the code below. Note the *-c* option when importing the data as this prevents the comments in the stored procedures from getting stripped out.

```
$ mysql -h[server] -uroot -p[pwd] -e 'create database build_YYYYMMDD'
$ mysql -c -h[server] -uroot -p[pwd] build_YYYYMMDD < build_YYYYMMDD_YYYYMMDD_mdb1_backup.sql
```

Test that the restore was successful by listing the tables:

```
$ mysql -h[server] -uroot -p[pwd] build_YYYYMMDD  -e 'show tables'
+-------------------------------------------+
| Tables_in_build_YYYYMMDD                  |
+-------------------------------------------+
| ix02_assets                               |
| ix02_associations                         |
| ix02_banner_clients                       |
etc ...
```

Web Server Build

Enabling Apache

If the Apache service does not start on server boot, start it like this and set it to start on boot-up:

```
$ sudo systemctl start httpd
$ sudo systemctl enable httpd
...
```

Install the MALDET malware Checker

This utility is required to ensure that uploaded documents that contain malware are cleansed, before being made available for user consumption.

The installation is as follows:

```
$ wget http://www.rfxn.com/downloads/maldetect-current.tar.gz
$ tar -xzf maldetect-current.tar.gz
$ cd maldetect*
$ sudo ./install.sh
```

Edit the configuration file:

```
$ sudo nano  /usr/local/maldetect/conf.maldet
```

Make the following changes:

```
email_alert="1"
email_addr="sysadmin@matchi.biz"
email_subj="[Security] MALDET Malware Alert from $(hostname)"
...
quarantine_hits="1"
quarantine_clean="1"
```

The subsequently-installed batch jobs will continuously monitor the relevant directories for files that contain malware.

Install ImageMagick

ImageMagick is used for validating, fixing and resizing any images on the web server. Install it (note the text-case of the package name):

```
$ sudo yum install ImageMagick
```

Install Libre Office

We need to install the headless version of Libre Office (similar to Open Office) so that icons can be made of attachment files, which are typically, PDF, Powerpoint, Excel and Word documents. The actual process that creates the icons is an inode-watcher that is driven and restarted by one of the batch jobs for the time being. Install Libre Office like this:

```
$ sudo yum install libreoffice-headless
```

Installing and Configuring PHP

Standard PHP Version across all servers

The minimum version of PHP required to run the Joomla Framework is version 5.3.10. The current version of PHP for all Matchi servers is 5.4.45. Check the PHP version. If it is installed,it should return the following:

```
$ php --version
PHP 5.4.45 (cli) (built: Feb 16 2016 17:32:49)
Copyright (c) 1997-2014 The PHP Group
Zend Engine v2.4.0, Copyright (c) 1998-2014 Zend Technologies
    with the ionCube PHP Loader v4.2.2, Copyright (c) 2002-2012, by ionCube Ltd.
```

Installation steps

By default, CentOS comes with a very old (weak, slow, vulnerability-prone) version of PHP. Set up the configuration for the REMI repository to install the desired version of PHP by adding the following:

```
$ sudo nano /etc/yum.repos.d/remi.repo
..
[remi-php56]
name=Les RPM de remi de PHP 5.6 pour Enterprise Linux 6 - $basearch
#baseurl=http://rpms.famillecollet.com/enterprise/6/php56/$basearch/
mirrorlist=http://rpms.famillecollet.com/enterprise/6/php56/mirror
# WARNING: If you enable this repository, you must also enable "remi"
enabled=1
gpgcheck=1
gpgkey=file:///etc/pki/rpm-gpg/RPM-GPG-KEY-remi
```

With the new PHP version enabled in the REMI repository, updatge the repository itselt:

```
$ sudo yum -y update php*
```

PHP should be installed by default on the server. If not, install it like this:

```
$ sudo yum -y install php-mysql php-devel php-gd php-pecl-memcache php-pspell php-snmp php-xmlrpc php-xml
```

It does not have the timezone setting set and some PHP modules refuse to work correctly without that setting. In the file /etc/php.ini, find the section marked 'Module Settings' and set the following:

```
date.timezone = "Europe/London"
```

You need to restart the Apache server to effect the timezone setting and the new version of PHP:

```
$ sudo /etc/init.d/httpd restart
Stopping httpd:                                            [  OK  ]
Starting httpd:
```

MySQL Database Server Build

The MySQL Database Server is based on the Standard Matchi Linux Server build, and has the following

- MySQL/MariaDB
- Support tools for MySQL/MariaDB

MySQL vs. MariaDB

MySQL and MariaDB are mostly code-equivalent databases (and thus functionally the same), however their licensing terms differ and MariaDB is a *free-er* database than MySQL.

By default, Red Hat and CentOS 6.x distributions come bundled with MySQL, whereas CentOS 7.x comes bundled with MariaDB, i.e.:

- Virtual Machines in the data centre run CentOS 6.5, therefore they use MySQL. MySQL is pre-installed.
- Physical server are built with CentOS 7.x, therefore they use MariaDB. MariaDB may need to be installed as an additional step. See the next section how to install it.

All the usual `mysql`-commands remain the same between MySQL and MariaDB and the only change is the actual name of the service-daemon: it is either `mysqld` or `mariadb`.

Installing the MySQL database on CentOS 6.x

To install the MySQL database service and client:

```
$ yum -y install mysql-server mysql
```

Start the MySQL service and set it up as a service so that it automatically restarts on server boot:

```
$ sudo /etc/init.d/mysqld start
..
$ sudo chkconfig --levels 2345 mysqld on
```

Note that CentOS 6.x still uses the *old* method of service controls through scripts in the `/etc/init.d` directory.

Enable consistent Stored Procedures coding

Run the following script to align the coding of stored procedures on the new database with the current Matchi standard:

```
$ mysql_upgrade -u root -p
Enter password:
```

Note
You should run this script after every upgrade of MySQL.

Installing the MariaDB database on RedHat 7.x

If it is not already installed on your server (not likely to be the case for hosted servers), then install the MariaDB database service and client:

```
$ yum -y install mariadb-server mariadb
```

Start the MariaDB service and set it up as a service so that it automatically restarts on server boot:

```
$ sudo systemctl start mariadb.service
...
$ sudo systemctl enable mariadb.service
```

CentOS 7.x uses the *systemctl* method of service controls through the *systemd* control mechanism, instead of the legacy *init.d* method of older version of Linux.

Configuration

The configuration of the MySQL database service is mostly managed through the file `/etc/my.cnf`.

Make a backup of the configuration file before making any changes. After having made the configuration changes, restart the database service when complete:

Note
Remember to test that you can still connect to the database after a service restart.

New Configuration on MySQL on Centos 6.5

```
$ sudo /etc/init.t/mysqld restart
Stopping mysqld:                                           [  OK  ]
Starting mysqld:                                           [  OK  ]
```

New Configuration on MariaDB on Centos 7.0

```
$ sudo systemctl restart mariadb
Stopping mariadb:                                          [  OK  ]
Starting mariadb:                                          [  OK  ]
```

Default Character Set and Collation

Our character set must always be set to UTF8 and our collation must always be set to UTF8-General-Case-Insensitive (`utf8_general_ci`). By default this is set to Swedish, which on a bad day can completely mess with an otherwise fine database.

Avoid collation disasters and make it so that the creation of a new database is always of the correct character set and collation, by adding these three sections of the file `/etc/my.cnf`:

```
[client]
default-character-set=utf8

[mysql]
default-character-set=utf8

[mysqld]
init_connect='SET collation_connection = utf8_general_ci'
character-set-server = utf8
collation-server = utf8_general_ci
```

Note
When we support multiple spoken-languages one day, we may need to consider changing the collation to UTF8-UNICODE-Case-Insensitive, which supports extended characters in collations more elegantly but is not as efficient.

Setting up Logging

The default installation on CentOS has very limited logging enabled. Since a number of log file types will need to be stored, it is best to create a dedicated directory in the `/var/log` directory. Create the directory if it does not already exist:

In Centos 6.0, you need to create these directories.

```
$ sudo mkdir /var/log/mysql
$ sudo chown mysql:mysql /var/log/mysql
```

In Centos 7.x, the logging directories have already been established and are `/var/log/mariadb`.

Add the configurations to the configuration file `/etc/my.cnf` where necessary:

```
# Section:
[mysqld]
...
# MySQL error log - included aborted connections
# For Centos 6.x:
log_error                       = /var/log/mysql/error.log
# For Centos 7.x
log_error                       = /var/log/mariadb/mariadb.log

log_warnings                    = 2

# Slow Query Log - disabled by default
# For Centos 6.x:
slow_query_log_file             = /var/log/mysql/slow.log
# For Centos 7.x
slow_query_log_file             = /var/log/mariadb/slow.log
slow_query_log                  = 0
log_queries_not_using_indexes   = 1
long_query_time                 = 0.5
min_examined_row_limit          = 100

# General Query Log - disabled by default
# For Centos 6.x:
general_log_file                = /var/log/mysql/general.log
# For Centos 7.x
general_log_file                = /var/log/mariadb/general.log
general_log                     = 0
```

```
# Section:
[mysqld_safe]
...

# For Centos 6.x:
log_error                       = /var/log/mysql/mysqld.log
# For Centos 7.x
log_error                       = /var/log/mariadb/mysqld.log
```

The 'General' and 'Slow' logs are very resource intensive and can be manually enabled to trace performance errors, but should be turned off again when done:

```
$ mysql

mysql> set global general_log=1;
..
mysql> set global general_log=0;
```

```
mysql> set global slow_query_log=1;
..
mysql> set global slow_query_log=0;
```

Database Authentication

The default account for access on MySQL is `root`. This should not be confused with the Linux super user, `root`. The password for database user root is set up using the mysqladmin utility:

```
$ sudo mysqladmin -u root password
New password:
Confirm new password:
```

Local authentication

You can set up the login for local maintenance directly on the server. This way you don't need to enter the user name and password every time. Create and edit the file in your home directory, `.my.cnf`:

```
$ nano ~/.my.cnf
```

Add this text – and replace [password] with the password for database user `root`:

```
[client]
user = root
password = XXXXXXXXX
```

Save `Ctrl` `O` & Close `Ctrl` `X`.

Restricting Remote Database Access

```
$ sudo /usr/bin/mysql_secure_installation
```

The Database server only needs to be remotely accessed from the web server (currently `mweb01`, IP address: `87.106.201.248`) and from the application server (currently `mapp01`, IP address: `87.106.206.136`).

TODO:

Restricting local Data Access: It is good practice to set data-access restrictions to the 'root' account in MySQL, which will be done on a future system design iteration. For now, database user account root has full access to the MySQL database.

Log in to the MySQL database. Note that there is no need to specify a user and password when using the mysql-shell, since we have set up the ~/.my.cnf file, above. By default, the database can only be accessed from the local server, mdb1, but it also needs to be accessed from the web server and the application server, for which we grant access as follows by specifying the IP addresses:

```
$ mysql
mysql> use mysql;
```

Reading table information for completion of table and column names You can turn off this feature to get a quicker startup with -A

```
Database changed
mysql> GRANT ALL PRIVILEGES ON *.* TO  'root'@'87.106.201.248'  IDENTIFIED  BY  'XXXXXXXXXX';
Query OK, 0 rows affected (0.00 sec)
mysql> GRANT ALL PRIVILEGES ON *.* TO  'root'@'87.106.206.136'  IDENTIFIED  BY  'XXXXXXXXXX';
Query OK, 0 rows affected (0.00 sec)
mysql> GRANT ALL PRIVILEGES ON *.* TO  'root'@'localhost'  IDENTIFIED  BY  'XXXXXXXXXX';
Query OK, 0 rows affected (0.00 sec)
mysql> flush privileges;
Query OK, 0 rows affected (0.00 sec)
```

Install Database Management Tools

Note
Not proven yet! In the mean time, we use custom backup scripts

A few specialized data management tools are require for effective backing up, restoring and data maintenance.

Percona Database Backup and Restore Tool:

Select the latest version

Go to http://www.percona.com/downloads/XtraBackup/LATEST/ for the latest available version. Select the version corresponding to the version of Red Hat/CentOS and copy the URL - no need to download it, see the next step.

First fetch the install RPM file

Fetch the file RPM directly onto the server:

```
$ wget http://www.percona.com/redir/downloads/XtraBackup/LATEST/RPM/rhel6/x86_64/percona-xtrabackup-2.1.3-608.rhel6.x86_64.rpm
```

Install the downloaded RPM file:

```
$ sudo rpm -ivh percona-xtrabackup-2.1.3-608.rhel6.x86_64.rpm
```

More information about this utility is available at http://www.percona.com.

Editing SQL using Nano

You can add keyword-sensitive colour coding for the nano editor, by creating this configuration script on either the Database Server for remote editing on the server itself, or on your local machine.

Create a SQL-Language file

```
$ sudo nano /usr/share/nano/sql.nanorc
[sudo] password for madman:
```

Add this (copy and paste from this doc!):

```
syntax "sql" "\.sql$" "sqliterc$"

color brightred "\<[A-Z_][0-9A-Z_]+\>"
color green "\<(VARCHAR|TINYINT|TEXT|DATE|SMALLINT|MEDIUMINT|INT|INTEGER|BIGINT|FLOAT|DOUBLE|DECIMAL|DATETIME|TIMESTAMP|TIME|YEAR|UNSIGNED|CHAR|TINYBLOB|TINYTEXT|BLOB|MEDIUMBLOB|MEDIU
color brightyellow "\<(ALL|ASC|AS|ALTER|AND|ADD|AUTO_INCREMENT|BETWEEN|BINARY|BOTH|BY|BOOLEAN|CHANGE|CHECK|COLUMNS|COLUMN|CROSS|CREATE|DATABASES|DATABASE|DATA|DELAYED|DESCRIBE|DESC|DI
color brightyellow "\<(FIELDS|FIELD|FLUSH|FOR|FOREIGN|FUNCTION|FROM|GROUP|GRANT|HAVING|IGNORE|INDEX|INFILE|INSERT|INNER|INTO|IDENTIFIED|IN|IS|IF|JOIN|KEYS|KILL|KEY|LEADING|LIKE|LIMIT|
color brightyellow "\<(OPTIMIZE|OPTION|OPTIONALLY|ORDER|OUTFILE|OR|OUTER|ON|PROCEDURE|PROCEDURAL|PRIMARY|READ|REFERENCES|REGEXP|RENAME|REPLACE|RETURN|REVOKE|RLIKE|RIGHT|SHOW|SONAME|ST
color brightyellow "\<(UNIQUE|UNLOCK|USE|USING|UPDATE|VALUES|VARIABLES|VIEW|WITH|WRITE|WHERE|ZEROFILL|TYPE|XOR)\>"


## String highlighting.  You will in general want your comments and
## strings to come last, because syntax highlighting rules will be
## applied in the order they are read in.
color brightyellow "<[^=        ]*>" ""(\\.|[^"])*""
##
## This string is VERY resource intensive!
color brightyellow start=""(\\.|[^"])*\\[[:space:]]*$" end="^(\\.|[^"])*""

## Comment highlighting
color brightblue "\-\-.*$"
color brightblue start="/\*" end="\*/"

## Trailing whitespace
color ,green "[[:space:]]+$"
```

Enabling the Language file

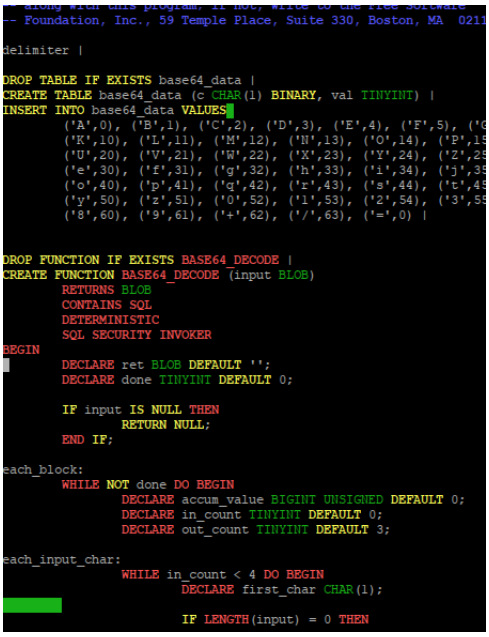
Enable this configuration file by adding the following to the main configuration file:

```
madman@s16972619 mdb1 ~ $ sudo nano /etc/nanorc
[sudo] password for madman:
```

Add this:

```
# SQL Code
include "/usr/share/nano/sql.nanorc"
```

Save and close. From now on, when you edit a SQL script in nano, SQL keywords it will look like this:



Set up the system Time Zone Data in MariaDB/MySQL

Explanation

MySQL can host a copy of the system's time zone definitions. These definitions differ from system to system, which is why this data is manually installed after the basic system and been built and the MySQL/MariaDB has been set up. The operating system's time zone database is defined by the files in the `/usr/share/zoneinfo` directory. The utility `mysql_tzinfo_to_sql` reads these files and populates the relevant tables in MySQL/MariaDB's `mysql` system database. All built-in time zone-based functions, such as `convert_tz`, refer to the content the time zone-tables in there. If data in not populated, then these functions return a NULL.

In the case of Matchi, we need to consider customer time zones (see Challenges), so the time zone information is therefore essential. The tables in the <cide>mysql</code> database that must be populated are:

- time_zone
- time_zone_leap_second
- time_zone_name
- time_zone_transition
- time_zone_transition_type

Installation

This is how to load MySQL's dictionary database from the operating system's time zone database:

```
$ mysql_tzinfo_to_sql /usr/share/zoneinfo | mysql mysql
```

Since local authentication has now been enabled in the `.my.cnf</conf< file, no further parameters are required for the <code>mysql` command. If this is yet set up, then also provide the user name (`-uroot` and password (`-pXXXXXX`) who also has rights to write to the `mysql` system database.

Verification

The table `time_zone` should contain some data now. Check it like this:

```
$ mysql mysql -e 'select * from time_zone limit 5'
+--------------+------------------+
| Time_zone_id | Use_leap_seconds |
+--------------+------------------+
|            1 | N                |
|            2 | N                |
|            3 | N                |
|            4 | N                |
|            5 | N                |
+--------------+------------------+
```

Connecting to the Database

Local connection to a database

You connect manually to a database while in an SSH session on the actual database server.

If you are logged in as user *madman* on the Database Server, and `/etc/my.cnf` is correctly set, then you can connect to the database without having to supply your username or password. You can list the databases, select a database, and so on as follows:

```
$ mysql
Welcome to the MySQL monitor.  Commands end with ; or \g.
..
Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

mysql> show databases;
+------------------------+
| Database               |
+------------------------+
| information_schema     |
| 20130611               |
| apsc                   |
| backup_test            |
..
```

Remote connection to a database from with the Matchi Datacenter

Specific access has been granted for the other Matchi servers in the datacenter to directly connect to the database on the Database Server through IP address restriction. It is not possible for any other devices, other than the assigned devices, to connect. (See the next section on how to connect from an arbitrary IP address)

From, say mapp01, connect to the database *test* on mdb01 as follows, using password authentication:

```
mapp01 ~ $ mysql -uroot -p******  -hmdb01 test
```

where:

- -u = Database user
- -p = Database user's Password
- -h = Server that the database is on

Remote connection to a database from outside the Matchi Datacenter

It is only possible to connect remotely to a database on the database server through a previously-established SSH-tunnel beteen yourself on the remote computer and one of the servers. This is a very secure method to connect to the servers and do data manipulations from a remote terminal.

This secure SSH tunnel can only be established if you have previously set up the public / private key-pair on your local machine and have your public key on the Database Server's key chain. When the SSH-tunnel is set up one can connect a local MySQL development program or terminal session to the remote database by actually connecting to the local machine and letting it act as a proxy to the remote server via the SH tunnel.

On your local machine, set up the SSH-tunnel in a separate terminal window so that this side of your local connection is on port 3307 and is presented on port 3306 on the remote side of the tunnel:

```
bignosethethird@terminator ~ $ ssh -L 3307:localhost:3306  madman@mdb01
```

Keep this tunneling session open and open another terminal to connect to the Database *test* through the tunnel. You are specifically indicating that you want to connect to the previoulsy-established tunnel by specifying the port number of the tunnel 3307 that the tunneling session is listening on:

```
bignosethethird@terminator ~ $ mysql -uroot -p***** -hlocalhost -P3307 test
```

where:

- -u = Database user
- -p = Database user's Password
- -h = Server that the database is on
- -P = Port number to connect to on your local side of the tunnel
- *test* = the name of the database on the server to connect to

Install phpMyAdmin

phpMyAdmin is a web-based, database development and administration tool. It can be installed on any server and made to access a database on any authorised server.

Installation

The EPEL Repository needs to be installed and enabled: Do this if this is not the case:

```
$ wget http://download.fedoraproject.org/pub/epel/6/x86_64/epel-release-6-8.noarch.rpm
# sudo rpm -ivh epel-release-6-8.noarch.rpm
```

Install phpMyAdmin as follows:

```
$ sudo yum install phpmyadmin
```

Configuration

By default, phpMyAdmin is configured to only access databases on the server that it is installed. It is also installed to be accessed from the same IP address (i.e. same server) that the phpMyAdmin service is installed on. Make the following changes to the Apache configuration file for phpMyAdmin to allow access from external addresses:

```
$ sudo nano /etc/httpd/conf.d/phpMyAdmin.conf
```

Change the 2 lines that contain `Allow from local` to `Allow from [ipaddress] [ipaddress] [domain] ...`. The IP addresses or domains of points where this service is likely to be accessed from need to added in here as a space-delimited list.

If access to this service is required via a domestic network provider, which mostly means that a new IP address is occasionally assigned to the subscriber, you need to specify a range of IP addresses. However, if the domestic router has a dynamic domain resolution service configured, you can set the domain instead of the IP address.

A range of IP addresses can be specified by only providing a partial IP address, e.g. to specify the range of IP addresses 105.226.0.0 - 105.227.255.255 (South Africa's Telkom subscribers in the Pretoria region), specify: `105.226 105.227`

Warning
It is important to not allow access from all IP addresses, as this represents a significant security risk. It is therefore far more preferable to allow access from domestic internet providers via dynamic domain names instead of a wide range of IP addresses: The specification of `105.226 105.227` theoretically allows access from approximately 120,000 different IP addresses (roughly 95% of 255x255x2)

Your Apache configuration should end containing 2 lines that read something like this:

```
Allow from 105.227.192.238 matchi.001.dyndns.com  matchi.002.dyndns.com
```

Determining an IP address for a domestic Internet service

If you need to set up access for a domestic, dynamic-allocated IP address service on a router that does not support dynamic domain name resolution, you can specify your current IP address by looking up the IP address of your own ADSL connection on the status report page on your router. Of visit http://www.whatismyip.com/.

Hiding the phpMyAdmin service

By default, the service name for the phpMyAdmin service is phpmyadmin and it can be accessed from the public network by browsing to this address: `http://mdb01/phpmyadmin`. An adversary may attempt to discover if such a service is available on a server by specifically targeting the term 'phpmyadmin'. A possible solution is to give the name of the publicly-visible service a cryptographic nonce. An 8 character-long nonce that is a valid URL can be generated like this:

```
$ openssl rand -base64 8 | tr -cd '[[:alnum:]]._-' | sed -e 's/\(.\{8\}\).*/\1\n/'
tE5ZyNCkS
```

Replace the line `Alias /phpMyAdmin /usr/share/phpMyAdmin` with `Alias /[nonce] /usr/share/phpMyAdmin`, where [nonce] is the nonce that was generated.

Restart the Apache Server

To effect new configuration changes, the Apache server needs to be restarted:

```
$ sudo service httpd restart
```

You should now be able to access the phpMyAdmin service by pointing your browser this URL: `http://mdb01/[nonce]`.

More information

- Apache configuration: http://httpd.apache.org/docs/2.2/mod/directives.html
- General phpMyAdmin Configuration: http://docs.phpmyadmin.net/en/latest/config.html

Prettify your command-line inteface!

Run this alias declaration and also add this alias to your `.bash_profile` file. The next time you need to access the database server using the command prompt, use `colormysql` command instead of `mysql` command.

```
alias colormysql=$(echo -e 'mysql --prompt="\x1B[31m\\u\x1B[34m@\x1B[32m\\h\x1B[0m:\x1B[36m\\d>\x1B[0m "')
```

It will look like this:

```
madman@mdb02 ~ $ alias colormysql=$(echo -e 'mysql --prompt="\x1B[31m\\u\x1B[34m@\x1B[32m\\h\x1B[0m:\x1B[36m\\d>\x1B[0m "')
madman@mdb02 ~ $ colormysql -usysdba -p
Enter password:
Welcome to the MariaDB monitor.  Commands end with ; or \g.
Your MariaDB connection id is 8775
Server version: 5.5.50-MariaDB MariaDB Server

Copyright (c) 2000, 2016, Oracle, MariaDB Corporation Ab and others.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

sysdba@localhost:(none)> show databases;
+--------------------+
| Database           |
+--------------------+
| information_schema |
| test_20161029      |
+--------------------+
2 rows in set (0.00 sec)
```

There exists a convention to prefix 'color' ('merkin spelling - stick to it, it's the UNIX way!) to standard commands to achieve just that, for example `colordiff` for the very colourful (sic) version of `diff`.

Application Server Build

This server's designated purpose is to perform as many as possible back-end operations that can be outsourced to it, in order to allow the web server and the database server to provide the serving of web content. Candidates for such asynchronous back-end processes are:

- State Machine Management
- Image Manipulations
- Data Manipulations
- Server Maintenance tasks
- Data Maintenance tasks
- Reporting functions
- Notifications Management
- Mass Mailing Management

Install Additional Packages

In addition to the packages installed on a standard Matchi server, we need install a few developer-oriented software packages due to the specialized nature of some of the tasks that this server needs to install

the tasks that this server needs to install:

Development packages

This installs gcc (a C/C++ compiler), git, swig, make, automake, doxygen, fortran (not required!), patch, flex, yacc / bison, and a few others in one go:

```
$ sudo yum groupinstall 'Development Tools'
Complete
```

Maths Libraries

Install a richer mathematical environment: Some heavy-duty mathematical functions are performed for the calculation of heuristics to match counter-parties up. For this, the GNU Arbitrary Precision Library has to be installed:

```
$ sudo yum install gmp-devel
```

Perl environment

Install a richer Perl environment: The servers come with a Perl environment, but some additional Perl-packages need to be installed. The best tool for installing Perl-packages is the CPAN tool, which is not yet installed. The first task therefore is to install the CPAN package manager for Perl as follows:

```
$ sudo yum install perl-CPANPLUS.noarch
```

Then run the CPAN package manager. The first time that it runs, it will prompt a number of times, to which you should just accept the default. After that, it should come up like this:

```
$ sudo cpan
Terminal does not support AddHistory.

cpan shell -- CPAN exploration and modules installation (v2.00)
Enter 'h' for help.

cpan[1]>
```

Use CPAN to install the necessary Perl packages and to upgrade the latest packages, by typing the following commands:

```
cpan> install CPAN
cpan> reload CPAN
cpan> install DBI
cpan> install DBD::mysql
cpan> install Pod::Perldoc
cpan> install Getopt::Long
cpan> install LWP
cpan> install Log::Log4perl
cpan> install DateTime
cpan> install DateTime::Format::MySQL
cpan> install DateTime::Format::DateParse
cpan> install Data::Dumper
cpan> install URI
cpan> install List::Util
cpan> install JSON::Parse
cpan> install PHP::Serialization
cpan> install MIME::Base64
cpan> install Error
cpan> install Math::BigInt
cpan> install Math::BigInt::GMP
cpan> install Math::Base36
cpan> install Digest::SHA2
cpan> install PadWalker
```

ImageMagick Utility

Install this utility on the Application Server if it is not already installed. ImageMagick is used for resizing and optimizing images that have been upload by users. Install it (note the character-case of the package name):

```
$ sudo yum install ImageMagick
```

GeoIP Utility

This utility is used for providing geographical details of IP addresses. Install it like this:

```
$ sudo yum install geoip
```

The geographical data needs to be updated for the first time and can then occasionally be updated again by running this command:

```
$ sudo GeoIP-update
```

Optional step: Manually replace the updates with *even more* accurate files:

```
$ cd /usr/share/GeoIP
$ sudo wget http://geolite.maxmind.com/download/geoip/database/GeoLiteCity.dat.gz
$ sudo wget http://geolite.maxmind.com/download/geoip/database/GeoLiteCountry/GeoIP.dat.gz
$ sudo wget http://download.maxmind.com/download/geoip/database/asnum/GeoIPASNum.dat.gz
$ sudo gunzip GeoIPASNum.dat.gz GeoIP.dat.gz GeoLiteCity.dat.gz
```

Usage:

- Country-level details:

```
$ geoiplookup   173.194.113.34
GeoIP Country Edition: US, United States
GeoIP ASNum Edition: AS15169 Google Inc.
```

- City-level details (will only work if you have run geoipupdate at least once):

```
$ geoiplookup -f /usr/share/GeoIP/GeoLiteCity.dat  173.194.113.34
GeoIP City Edition, Rev 1: US, CA, California, Mountain View, 94043, 37.419201, -122.057404, 807, 650
```

Note: The file /usr/share/GeoIP/GeoLiteCity.dat that is installed when geoipupdate is run does not always contain accurate information. It is possible to get a more accurate and complete data file by either manually installing the updates as described above, or by first obtaining a licence (at some considerable cost) from https://www.maxmind.com/en/my_license_key, and applying the license key in the file /etc/GeoIP.conf.

Install MATCHi-specific Application Programs

MATCHi-specific / home-grown Application Programs and scripts have been written in a number of programming languages and scripts:

- Perl

- Python
- PHP
- BASH

Some are batch-run programs and others are utility programs that are manually executed as and when required.

Deployment Directories

All these scripts reside in the directory /usr/local directory tree (as per standard Unix practise), which looks as follows:

```
$ tree -d -L 2 /usr/local
/usr/local
    ├── bin
    ├── etc
    │   └── cron
    ├── sbin
    └── share
        └── templates
```

Each directory contains:

- /usr/local/bin: Utility scripts, report genators, batch processes
- /usr/local/etc/cron: Batched calling scripts. These are called either hourly, daily, weekly or monthly by setting symbolic links between them and `/etc/cron.hourly`, `/etc/cron.daily`, `/etc/cron.weekly` respectively.
- /usr/local/sbin: Background daemons
- /usr/local/share/templates: Templates used for automated generation of document, reports and notification.

The scripts and application are owned by root and have their executable flag set (see the x's in the left hand columns):

```
$ ls -al /usr/local/bin
-rwxr-xr-x  1 root root    2228 Sep 15 00:18 adjoinall.sh
-rwxr-xr-x  1 root root    7555 Sep 22 18:34 backup_local_database.sh
-rwxr-xr-x  1 root root    5453 Sep 22 18:34 backup_local_subversion.sh
-rwxr-xr-x  1 root root    8850 Nov  2 11:54 backup_remote_application.sh
-rwxr-xr-x  1 root root    8787 Nov  2 11:54 backup_remote_database.sh
etc...
```

If the executable flag is not set, then you can set it like this:

```
$ sudo chmod +x /usr/local/bin/*
```

By default, all Unix users have the /usr/local/bin on their path. You can check this:

```
$ echo $PATH
/usr/local/bin:/bin:/usr/bin:/usr/local/sbin:/usr/sbin:/sbin:/home/madman/bin
```

Application Program Logging

All Application programs log to a common file called /var/log/matchi. The logging-libraries Log4Perl, Log4PHP, Log4Python etc. are used to perform the logging. The content of each log entry is sufficiently detailed to distinguish the application that wrote to it, contains the time stamp and the line of code. Logging occurs on 6 levels:

- TRACE: Only possible if turned on as an option when the application is run. Produces very verbose output. Used for tracing the value of variables of logic flow at a very detailed level.
- DEBUG: Only possible if turned on as an option when the application is run. Produces very verbose output. Used for tracing the value of variables of logic flow.
- INFO: Produces tracking information and metrics, such as when the program was run and when how records it processed.
- WARN: Notifies if an expected error condition was encountered that was dealt with in the program
- ERROR: An error in a transaction occurred and program either reattempts the transaction or ignores it.
- FATAL: A catastrophic system-based error occurred, or the program was abandoned because too many processing errors were encountered.

Preparing for Logging

The applications, if run as user `root`, will create the log file if it does not exist. In order to run the application as any other user, the user needs to have write access to the log file. This is done by making the user in question the log file owner:

```
$ sudo touch /var/log/matchi
$ sudo chown madman:adm /var/log/matchi
$ ls -al /var/log/matchi
-rw-r--r-- 1 madman adm 0 Jun 25 11:34 /var/log/matchi
```

Log File Rotation

A log file rotation process exists that places the previous day's log entries in a date-stamped file in the form `/var/log/matchi-YYYYMMDD`. An option exists to also compress the file using GZIP to save space, which is not used here. Log files older than 3 months (90 days) are automatically deleted. The log file rotation is set up by adding a configuration file to the `/etc/logrotate.d` directory. These configurations are read each day by the logrotate cron job and responded to accordingly:

```
$ sudo nano /etc/logrotate.d/matchi
```

Enter the following configuration:

```
/var/log/matchi {
  missingok
  notifempty
  create 0640 madman adm
  copytruncate
  rotate 90
}
```

Save and close `Ctrl-O` and `Ctrl-X`.

Viewing and searching log files

Various sophisticated tools exist to scan and view the content of log files. In the examples here we rely on simple yet powerful console commands.

To search for INFO events:

```
$ grep INFO /var/log/matchi
[2013/06/19 20:09:57][INFO][AlignNewCDN.pl-main::AlignIdea-243] Duration:  2466 seconds
...
```

To search for events that relate to a particular program:

```
$ grep IdeaFSM /var/log/matchi
[2013/06/19 19:23:26][DEBUG][IdeaFSM.pl-main::-141] Connecting to Database build_20130611
...
```

To search for events that occurred around a particular date range, e.g. 19 June to 20 June:

```
$ egrep '06/(19|20)' /var/log/match
```

To search for events that occurred around a particular time range, e.g. between 10:30pm to 11:30pm on 19 June:

```
$ egrep '06/19 (22:[3-5]|23:[0-2])' /var/log/matchi
```

To view a continuous scrolling of the log file, use the `tail -f` command, which only stops once you hit `Ctrl-C`.

```
$ tail -f /var/log/matchi
..
[2013/06/19 23:05:30][INFO][AlignIdea-239] Duration: 17 seconds
^C
```

For a colour-coded display of the running log, run `logwatch`. The code for `logwatch` is this one-liner:

```
tail -f -n 50 /var/log/matchi | sed 's#\[WARN\s*\]#\x1b[33m&\x1b[0m#; s#\[ERROR\]#\x1b[31m&\x1b[0m#;  s#\[FATAL\]#\x1b[31m\x1b[5m&\x1b[0m#; s#\[INFO\s*\]#\x1b[32m&\x1b[0m#; s#\[DEBUG\
```

It gives you this:

```
[2015/02/19 08:02:19][INFO] [09update_orglogos_PROD.sh][PROD] 0 Update MD5 signatures for each member's avatar and create t
[2015/02/19 08:02:19][DEBUG][09update_orglogos_PROD.sh][PROD] [354] select user_id,avatar from mtchi_comprofiler where avat
(modifytime,'0 01:01:01') > now()
[2015/02/19 08:02:19][INFO] [09update_orglogos_PROD.sh][PROD] [397] Create remaining 60x60 square organization logos in ~/
[2015/02/19 08:02:19][INFO] [09update_orglogos_PROD.sh][PROD] 0 For all new org records with no avatar create 60x60-sized i
[2015/02/19 08:02:19][TRACE][09update_orglogos_PROD.sh][PROD] [128] === END [PID 3776] on signal EXIT. Cleaning up ===
[2015/02/19 08:02:19][INFO] [10update_keywords_ORG.sh][ORG] 1 === BEGIN 1 $Id: 10update_keywords_ENVIRONMENT.sh 1744 2014-
=
[2015/02/19 08:02:19][INFO] [10update_keywords_ORG.sh][ORG] 1 Checking that we are running as user root...
[2015/02/19 08:02:19][INFO] [10update_keywords_ORG.sh][ORG] 1 Data updating on connection /usr/bin/mysql -hmdb01 -uroot -p
will start in 1 seconds...
[2015/02/19 08:02:20][INFO] [10update_keywords_ORG.sh][ORG] 1 Update keyword uses counts
[2015/02/19 08:02:20][DEBUG][10update_keywords_ORG.sh][ORG] [231] update mtchi_inno_tags t set t.uses = ( select count(iit
no_tags iit inner join mtchi_inno i on i.id = iit.inno_id where t.id = iit.tag_id and i.state in ('SUBMITTED','SPONSORPREV
VISIBLE') ); select row_count();
[2015/02/19 08:02:20][INFO] [10update_keywords_ORG.sh][ORG] 1 === END 1 Cleaning up ===
[2015/02/19 08:02:20][INFO] [10update_keywords_PROD.sh][PROD] 1 === BEGIN [3989] $Id: 10update_keywords_ENVIRONMENT.sh 1744
it $ ===
[2015/02/19 08:02:20][INFO] [10update_keywords_PROD.sh][PROD] 1 Checking that we are running as user root...
[2015/02/19 08:02:20][INFO] [10update_keywords_PROD.sh][PROD] 1 Data updating on connection /usr/bin/mysql -hmdb01 -uroot -
will start in 7 seconds...
[2015/02/19 08:02:27][INFO] [10update_keywords_PROD.sh][PROD] 1 Update keyword uses counts
[2015/02/19 08:02:27][DEBUG][10update_keywords_PROD.sh][PROD] [231] update mtchi_inno_tags t set t.uses = ( select count(i
inno_tags iit inner join mtchi_inno i on i.id = iit.inno_id where t.id = iit.tag_id and i.state in ('SUBMITTED','SPONSORPR
ERVISIBLE') ); select row_count();
[2015/02/19 08:02:27][INFO] [10update_keywords_PROD.sh][PROD] 1 === END [3989] Cleaning up ===
[2015/02/19 08:02:27][HEART][11housekeep_tmp_DEV.sh][DEV] Housekeeping is taking it easy... no files to clean up in /var/w
ome-server.info/dev.matchi.biz/tmp
```

Node.js

Requirement

Used for running the RabbitMQ Simulator

Installation

On the development server:

Note
If you have not done so yet, you should install the standard group of development tools:

```
$ sudo yum groupinstall 'Development Tools'
```

Now install Node.js:

```
$ sudo yum install nodejs
```

Followed by installing the Node Page Manager (npm):

```
$ sudo yum install npm.noarch
```

Now install Node's Express package and others:

```
$ sudo npm install
```

Usage

Download the Node.js application, e.g. RabbitMQSimulator:

```
$ cd ~/git
$ git clone https://github.com/RabbitMQSimulator
$ cd RabbitMQSimulator
```

Run the Node.js application, typically called app.js:

```
$ node app.js
Listening on port 3000
```

Point a browser to the server on port 3000 and proceed.

Hit `Ctrl-C` to stop the application.

{{Note|If you want the application run to persist Run the Node.js application, typically called app.js:

```
$ nohup node app.js &
nohup: ignoring input and appending output to `nohup.out'
```

You can close the server session and come back later and kill the Node.ja application if no longer required:

```
$ ps -ef | grep node
madman   13142     1  0 14:05 ?        00:00:00 node app.js
$ kill -9 13142
```

RabbitMQ Installation

First enable the EPEL package library for additional packages:

```
$ sudo sed -i 's/^enabled=0/enabled=1/' /etc/yum.repos.d/epel.repo
```

Then install the Erlang language:

```
$ sudo yum install erlang
```

Get the latest version of RabbitMQ Server:

```
$ cd installs
~/installs $ wget http://www.rabbitmq.com/releases/rabbitmq-server/v3.4.2/rabbitmq-server-3.4.2-1.noarch.rpm
--2015-01-04 01:03:15--  http://www.rabbitmq.com/releases/rabbitmq-server/v3.4.2/rabbitmq-server-3.4.2-1.noarch.rpm
Resolving www.rabbitmq.com... 192.240.153.117
Connecting to www.rabbitmq.com|192.240.153.117|:80... connected.
HTTP request sent, awaiting response... 200 OK
Length: 4100214 (3.9M) [application/x-redhat-package-manager]
Saving to: rabbitmq-server-3.4.2-1.noarch.rpm

100%[===================================================================>] 4,100,214    971K/s   in 4.9s

2015-01-04 01:03:22 (814 KB/s) - rabbitmq-server-3.4.2-1.noarch.rpm saved [4100214/4100214]
```

Install RabbitMQ Server:

```
~/installs $ sudo rpm -ivh rabbitmq-server-3.4.2-1.noarch.rpm
warning: rabbitmq-server-3.4.2-1.noarch.rpm: Header V4 DSA/SHA1 Signature, key ID 056e8e56: NOKEY
Preparing...                ########################################### [100%]
   1:rabbitmq-server        ########################################### [100%]
```

Remember to disable the EPEL package library: First enable the EPEL package library for additional packages:

```
$ sudo sed -i 's/^enabled=1/enabled=0/' /etc/yum.repos.d/epel.repo
```

Start the RabbitMQ Service

```
$ sudo /etc/init.d/rabbitmq-server start
Starting rabbitmq-server: SUCCESS
```

You can also check the status of the RabbitMQ service:

```
$ sudo rabbitmqctl status
Status of node rabbit@s16972617 ...
[{pid,6678},
 {running_applications,[{rabbit,"RabbitMQ","3.4.2"},
                        {mnesia,"MNESIA  CXC 138 12","4.5"},
                        {os_mon,"CPO  CXC 138 46","2.2.7"},
                        {xmerl,"XML parser","1.2.10"},
                        {sasl,"SASL  CXC 138 11","2.1.10"},
                        {stdlib,"ERTS  CXC 138 10","1.17.5"},
                        {kernel,"ERTS  CXC 138 10","2.14.5"}]},
 {os,{unix,linux}},
 ...
```

Auto-start the RabbitMQ Service on server reboot

```
$ chkconfig rabbitmq-server on
```

Install RabbitMQ libraries

On the application server, the message producers and consumers will mostly be written in Perl, so we install Net::RabbitFoot and its dependecies:

```
$ sudo yum install libxml2-devel.x86_64
```

hen we shell into root and install the Perl package, so that it is installed to the Vendor Perl installation instead of the local user's environment, as the Vendor environment is the Perl environment that will be called by RabbitMQ.

```
$ sudo su -
# cpan
cpan> install Net::RabbitFoot
```

We are now ready to write Perl code for RabbitMQ.

Subversion Code Management System

Subversion is our chosen source code management system, and is a primary tool used for creating deployments. See article http://www.if-not-true-then-false.com/2010/install-svn-subversion-server-on-fedora-centos-red-hat-rhel/

Installing Subversion Client

Install the Subversion Client if it is not already installed:

```
$ sudo yum -y install subversion
```

Installing Subversion Server

The Subversion Server's web access component is installed as follows:

```
$ sudo yum -y install mod_dav_svn
```

The web interface only allows you to browse the repository but not perform any other version-control / configuration management functions. To set up the web access component to server the pages via the Apache web server, create the file /etc/https/conf.d/subversion.conf:

```
# nano /etc/httpd/conf.d/subversion.conf
```

Add these configuration lines to the file:

```
LoadModule dav_svn_module     modules/mod_dav_svn.so
LoadModule authz_svn_module   modules/mod_authz_svn.so

<Location /svn>
   DAV svn
   SVNParentPath /var/www/svn
   AuthType Basic
   AuthName "Subversion repositories"
   AuthUserFile /etc/svn-auth-users
   Require valid-user
</Location>
```

Create subversion admin user

```
$ sudo htpasswd -cm /etc/svn-auth-users madman
New password:
Re-type new password:
Adding password for user madman
```

Create subsequent users

Note the use of –m, not –cm:

```
$ sudo htpasswd -m /etc/svn-auth-users user2
New password:
Re-type new password:
Adding password for user user2
```

Create Subversion Repository

Create and configure the Subversion repository and set ownership of the files:

```
$ sudo mkdir /var/www/svn
$ sudo chown –R apache:apache /var/www/svn/matchi
$ cd /var/www/svn
$ sudo svnadmin create matchi
$ sudo chown -R apache.apache matchi
```

Restrict access to known users only

Configure the access to the repository to disable anonymous access and to use access control based on the users that were previously created:

```
$ sudo nano /var/www/svn/matchi/conf/svnserve.conf
```

Make the following changes:

```
anon-access = none
authz-db = authz
```

Restart the Apache Service

To activate recent changes in the Apache configuration, restart the Apache service:

```
$ sudo /etc/init.d/httpd restart
```

Browse to Subversion on http://mapp1/svn/subversion and log in using one of the user accounts that were created to test that it works.

Create the Repository Framework

Create an empty dummy-structure as follows:

```
$ mkdir -p /tmp/svn/{trunk,branches,tags}
```

This creates the template directories that serve a particular function:

- trunk: Main source code repository
- tags: Snapshots of the trunk repository
- branches: Code forks from the trunk repository

This is the standard structure of a subversion repository. Although any other structure is possible, it needs to be in this form so that the JIRA Subversion component (and any other tools) can integrate with this repository.

Import (don't just copy) the empty template directory structure to the project repository:

```
$ svn import -m 'Initial import' /tmp/svn / http://mapp01/svn/matchi/
Adding         /tmp/svn/trunk
Adding         /tmp/svn/branches
Adding         /tmp/svn/tags
Committed revision 1.
```

Backing up Subversion

Backups of the Subversion repository are made evey night, and here is the process if an ad-hoc backup is required to other media (where YYYYMMDD is the current date stamp):

```
root@s16972617 mapp01 ~ # svnadmin dump /var/www/svn/matchi | gzip -9 > /backups/subversion_YYYYMMDD.dump.gz
```

Restore a Subversion Repository Backup

This restores the entire repository, which includes all the historical changes. It is not the same as a code check-out. To restore a repository to a new location, create a new repository location:

```
$ sudo svnadmin create /var/www/svn/matchi2
$ sudo chown –R apache:apache /var/www/svn/matchi2
```

Unpack the backup file:

```
$ gunzip /backups/subversion_YYYYMMDD.dump.gz
```

Load it to the new repository:

```
$ svnadmin load 2/var/www/svn/matchi < /backups/subversion_YYYYMMDD.dump
```

Adding Event-Hooks to Subversion

In the directory `/var/www/svn/matchi/hooks` are a number of template files, each one for a particular type of event on Subversion. A subsequent action can be performed when for instance when one or more files has been committed, such a sending an event to a log file or sending an email. The steps to set up a hook are as follows:

```
$ cd /var/www/svn/matchi/hooks
$ ls post-commit*
post-commit.tmpl
```

This file tells you what parameters to use or to pass to the calling script. In this case, it is:

```
...
#   [1] REPOS-PATH   (the path to this repository)
#   [2] REV          (the number of the revision just committed)
...
```

The policy is to call a code-managed script stored in `/usr/local/bin/` from this hook script instead coding the actual script in there, i.e. in this case, we call the script `/usr/local/bin/svnlogcommit.pl`:

```
$ nano post-commit
#!/bin/bash

REPOS="$1"  # e.g. /var/www/svn/matchi
REV="$2"    # e.g. 3087

/usr/local/bin/svnlogcommit.pl $REPOS $REV
```

The file `post-commit` calls a Matchi-specific script in `/usr/local/bin`. Save and close this file, post-commit, and set it to executable and set the ownership:

```
$ sudo chmod a+x post-commit
$ sudo chown apache:apache post-commit
```

In a similar way, event-hooks can be created for the other events that Subversion exposes if ever required.

Dropbox

Requirement

We currently need Dropbox on the application server for these reasons:

- Dropbox is a tactical tool for sharing large files between desktop devices to the server for those who do not access to facilities such as `rsync`.
- Dropbox is one of few ways through which to securely publish data files, for the Zoho CRM to upload data.

Warning
The use of Dropbbox in a production sever scenario is a tactical solution for the above scenarios, and other uses may create vulnerabilities on the server.

Architecture

Dropbox files are held on the `/dev/mapper/vg00-var` logical volume, under `/var/data/dropbox`.

The directory `/var/data/dropbox` is bind-mounted to `/data/dropbox`.

The Dropbox account is hosted by the system `root` account. By default, Dropbox files are held in the `/root/Dropbox` directory.

Installation

Follow the instructions on https://www.digitalocean.com/community/tutorials/how-to-install-dropbox-client-as-a-service-on-centos-7

Test the Daemon

The Dropbox daemon should now automatically start on server reboot. You can check it by carefully rebooting the server and checking if the process has started:

```
$ sudo shutdown -r now
```

Attempt to log in 2 minutes later and check if the dropbox process is running:

```
bignosethethird@ziltoid ~ $ ssh madman@[server]
Last login: Sat Jan XX XX:XX:XX XXXX from XXX.XXX.XXX.XXX
$ ps -ef | grep drop
root      7996     1  0 12:53 ?        00:00:03 /opt/dropbox/dropbox-lnx.x86_64-14.4.19/dropbox
```

This means that the daemon worked. If no *dropbox* process is running, then there was was a problem with your Dropbox configuration.

Test the file synchronization

Create random file in the `/data/dropbox` directory and confirm that it appears on the Dropbox web page view:

```
$ sudo su -
# cd /data/dropbox
# touch a
```

If a file called `a` appears in the web view, then the synchronization work.

Remember to clean up after testing:

```
# rm -f a
```

Office Server Build

Only do this in a place where is no crime. South Africa is not a good place to do this, but if you must, then bolt the box down and surround it with barbed wire in a server cage. And electrify the cage. Get some fierce dobermans too.

Follow the steps to build a basic server, and add the following features and configurations

Setting the local time zone

Look up the time zone file for the nearest city that the server is located to in `/usr/share/zoneinfo/[country]/[city]`. Here, for example, we are based in Johannesburg, so our time zone file is `/usr/share/zoneinfo/Africa/Johannesburg`. Remove the old time zone setting and assign a new time zone in one command, and then check that it was successful with the `date`-command.

```
$ sudo rm /etc/localtime; sudo ln -s /usr/share/zoneinfo/Africa/Johannesburg /etc/localtime
$ date
Mon 16 Feb 19:07:20 SAST 20xx
```

Sensors Support

Since it is a physical server, we need to monitor the actual temperature and other physical characteristics, as there is no VM-hosting operating system to do this for you. We install the `lm_sensors`-package:

```
$ sudo yum install lm_sensors
```

All the various sensors on the server need to be initially detected:

```
$ sudo sensors-detect
```

To view the temperature of the CPUs:

```
$ sensors
coretemp-isa-0000
Adapter: ISA adapter
Core 0:       +46.0°C  (high = +82.0°C, crit = +100.0°C)
Core 1:       +45.0°C  (high = +82.0°C, crit = +100.0°C)
Core 2:       +39.0°C  (high = +82.0°C, crit = +100.0°C)
Core 3:       +43.0°C  (high = +82.0°C, crit = +100.0°C)
```

Wake-On-LAN (WOL)

Significant amounts of electrical power can be saved with running a physical server by turning it completely off when it is not in use. Assuming an anticipated 12-hour availability for a development server between, say, 8am and 8pm, the server can be powered off with an 8pm-scheduled `shutdown -h` command. If the server's BIOS supports *scheduled power-up*, set it for 8am. It is not a problem if the BIOS does not support *scheduled power-up* because the server can be *woken up* using the Wake-On-LAN (WOL) function. This feature allows you to remotely turn a server on at any time by sending it a special network packet, a.k.a. *magic packet* from your computer, with the restrictions that:

- You are one the same LAN segment as the server, i.e. you can't do this from outside the office / data centre over a WAN
- The circuit breaker / power switch on the server is not open.
- There is electrical power. This should not be taken for granted in some parts of the world.

Configuring the server for remote Wake-On-LAN

Permanently enable the Wake-On-LAN feature on the server for all the ethernet interfaces, but ignore the loop-back `...-lo` interface:

```
$ ls /etc/sysconfig/network-scripts/ifcfg-*
/etc/sysconfig/network-scripts/ifcfg-enp0s25
/etc/sysconfig/network-scripts/ifcfg-enp3s2
/etc/sysconfig/network-scripts/ifcfg-lo
```

For each ethernet interface:

```
$ sudo nano /etc/sysconfig/network-scripts/ifcfg-enp0s25
$ sudo nano /etc/sysconfig/network-scripts/ifcfg-enp3s2
```

Add this line to each file:

```
ETHTOOL_OPTS="wol g"
```

Installing the Wake-On-LAN client

If you have a Linux client, install this:

```
pignosethethird@ziltoid ~ $ sudo equo install wakeonlan
```

If you have a Windows client, install this: http://sourceforge.net/projects/aquilawol

Waking up the beast

You need the MAC address of the physical ethernet interface that belongs to the server that you want to wake up. You can get all the MAC-addresses on the server like this:

```
$ grep HWADDR /etc/sysconfig/network-scripts/ifcfg-* | cut -f2 -d=
00:15:17:8C:DF:3C
00:15:17:8C:DF:3A
```

Power the server off:

```
$ sudo shutdown -h now
```

Wait a few minutes and send the magic packets, one for each MAC address:

```
$ wakeonlan 00:15:17:8C:DF:3C
$ wakeonlan 00:15:17:8C:DF:3A
```

Wait a few minutes for the server to reboot. You can see when you it has come up by pinging it every few seconds:

```
$ sudo ping mdev01
PING mdev01 (10.0.0.18) 56(84) bytes of data.
From ziltoid.local (10.0.0.11): icmp_seq=1 Destination Host Unreachable
From ziltoid.local (10.0.0.11): icmp_seq=2 Destination Host Unreachable
From ziltoid.local (10.0.0.11): icmp_seq=3 Destination Host Unreachable
```

```
From ziltoid.local (10.0.0.11): icmp_seq=4 Destination Host Unreachable
From ziltoid.local (10.0.0.11): icmp_seq=5 Destination Host Unreachable
From ziltoid.local (10.0.0.11): icmp_seq=6 Destination Host Unreachable
From ziltoid.local (10.0.0.11): icmp_seq=7 Destination Host Unreachable
From ziltoid.local (10.0.0.11): icmp_seq=8 Destination Host Unreachable
From ziltoid.local (10.0.0.11): icmp_seq=9 Destination Host Unreachable
From ziltoid.local (10.0.0.11): icmp_seq=10 Destination Host Unreachable
From ziltoid.local (10.0.0.11): icmp_seq=11 Destination Host Unreachable
...
64 bytes from mdev01 (10.0.0.18): icmp_seq=1 ttl=64 time=134 ms
64 bytes from mdev01 (10.0.0.18): icmp_seq=2 ttl=64 time=3.25 ms
64 bytes from mdev01 (10.0.0.18): icmp_seq=3 ttl=64 time=7.71 ms
```

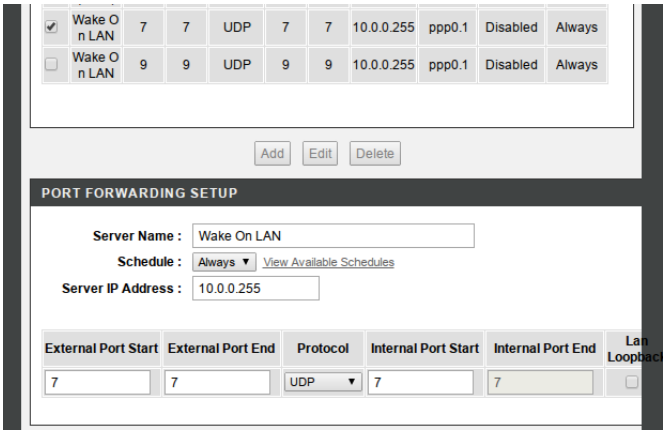
Yay! The server is now returning ICMP requests, which means that you should now be able to log in again.

Waking up the beast from a remote location

This is similar in principle to the local Wake-On-LAN process, except that you need to also specify the IP address or dynamic host name of the external router, as well as the port number over which the the magic packet will be sent.

Port-Forwarding on the Router

Wake-On-LAN normally sends its packets over ports 7 or 9. On the router, set up port-forwarding for ports 7 on UDP and 9 on UDP and have it broadcast to the entire subnet (in this case `10.0.0.255`). The part-forwarding configuration on the network edge would typically look like this:



Sending the remote magic packet

To wake up the server from a remote location, ensure you have `wakeonlan` or `wol` installed on there. You should also know the dynamic host name or the IP address for the subnet's router as seen from the public network. Issue this command with the host name and port to the MAC address on the subnet:

```
$ wol -h matchi002.dyndns.biz   -p 7 00:15:17:8C:DF:3A -v
Waking up 00:15:17:8C:DF:3A with matchi002.dyndns.biz:7...
$ wol -h matchi002.dyndns.biz   -p 7 00:15:17:8C:DF:3C -v
Waking up 00:15:17:8C:DF:3C with matchi002.dyndns.biz:7...
```

As if by magic, the server in the office should start up...

Putting the beast to sleep again

We will power the server down every evening at 20:00. It will be turned back on again whenever someone sends it a Wake-On-LAN signal, perhaps never again. Add a CRON-job to power the server down using the CRON-editor.

```
$ sudo crontab -e
```

The CRON-editor is actually the *vi*-editor, so the usual *vi*-editing keystrokes apply. Hit the `i`-key and insert the following text:

```
0 20 * * *  /usr/sbin/shutdown -h now >/dev/null 2>&1
```

When done, hit `Esc` and then the `:` `w` `q`-key sequence to save and quit.

Note that you should specify the full path for the command that is be executed in the CRON-job.

NTFS File Support

This is required so that portable hard drives that are unfortunately formatted using NTFS file system can be read and written to.

```
$ sudo yum install ntfs-3g
```

Mounting an NTFS-formatted external hard drive

Plug the drive into the USB port. Run the `dmesg`-command to see what UNIX-device is assigned to this new device

```
$ dmesg
...
[ 3019.406412] usbcore: registered new interface driver usb-storage
[ 3025.196501] scsi 6:0:0:0: Direct-Access     TOSHIBA  External USB 3.0 0101 PQ: 0 ANSI: 6
[ 3025.196887] sd 6:0:0:0: Attached scsi generic sg4 type 0
[ 3025.201988] sd 6:0:0:0: [sdd] 3907029164 512-byte logical blocks: (2.00 TB/1.81 TiB)
[ 3025.203545] sd 6:0:0:0: [sdd] Write Protect is off
[ 3025.203550] sd 6:0:0:0: [sdd] Mode Sense: 1f 00 00 00
[ 3025.204618] sd 6:0:0:0: [sdd] Write cache: disabled, read cache: enabled, doesnt support DPO or FUA
[ 3025.219988]  'sdd: sdd1'
[ 3025.234497] sd 6:0:0:0: [sdd] Attached SCSI disk
```

The new device in this case is `sdd` and the partition is `sdd1`. You can verify that it exists by looking in the `/dev`-directory:

```
$ ls /dev/sdd*
/dev/sdd  /dev/sdd1
```

Mount the external hard drive's partition to a mount point in any directory on the server. Traditionally, these mount points are in the `/mnt`-directory:

```
$ sudo mkdir /mnt/sdd1
```

```
$ sudo mkdir /mnt/sdd1
$ sudo mount /dev/sdd1 /mnt/sdd1
```

If you get no errors, then you have successfully mount the external hard drive and you can refer to it just like any other directory on the server:

```
$ ls /mnt/sdd1
installs   backups
....
```

Unmounting an external hard drive

It is unwise to simply remove an external hard drive from a computer by pulling it from the USB connection without un-mounting it first. By un-mounting it first, any pending data writes and file-system journal updates are allowed to be committed to the device so that it is in a known an stable state, so avoiding any potential loss of data. If any processed still have locks on files or directories the device, then the un-mounting operation will warn you about this and you will need to resolve these issues first before attempting to un-mount again.

To un-mount the `/dev/sdd1` device, use the `umount`-command:

```
$ sudo umount /dev/sdd1
```

You can also un-mount by referring to the actual mount-point:

```
$ sudo umount /mnt/sdd1
```

Development tools to install

Install the Database

Install the MariaDB database.

Install the MySQLi Adapter

This is important for Joomla!

```
$ sudo yum install php-mysqli php-pdo
```

Enable the Apache Service

If the Apache service does not start on server boot, start it like this and set it to start on boot-up:

```
$ sudo systemctl start httpd
$ sudo systemctl enable httpd
...
```

Install Apache's SSL Adaptor

Only do this if this development or test server that will serve its own certificate.

```
$ yum -y install mod_sll
```

Restart the Apache server:

```
$ systemctl restart httpsd
```

Install the Subversion Client

Install the client. Then do a check-out of the Matchi code repository into directory `~/svn`.

Lynx Text-based browser

Useful for some types of automated testing

```
$ sudo yum install lynx
```

Securing the Office Server

Setting up the Firewall

By default, the firewall-daemon `firewalld` is enabled and restricts traffic to only the SSH ports 21 and 22. Open port 80 for `http` traffic so that the development sites can be viewed remotely:

```
$  sudo firewall-cmd --zone=public --add-service=http --permanent
success
```

Reload the firewall-configuration:

```
$ sudo firewall-cmd --reload
success
```

This will throw an error if the firewalld-daemon is not running. Should this ever be the case, start the daemon like this:

```
$ sudo systemctl start firewalld
```

More info: http://wiki.centos.org/HowTos/Network/IPTables

Add development users to groups

The user `madman` is added to the group `apache`, so that web-source files can be modified by user `madman`.

```
$ sudo gpasswd -a madman apache
Adding user madman to group apache
```

Installing the Development Code

Prevent unauthorized access to the development environment by setting up a password file with the username *sysadmin* and the password *XXXXXXXXX*. The password file is created like this (see elsewhere how the content is actually generated):

```
$ sudo  echo 'sysadmin:$apr1$lPjrcpsb$IH1Ew2dzaItXlG/OaE2rZ1' > /etc/httpd/conf/htpasswd
```

Associate the password file to the development environment instance to the file `/etc/httpd/conf/httpd.conf`.

```
# DEV administrator
<Directory /var/www/html/dev_20150216/administrator>
 Allow from all
 AuthType Basic
 AuthName "IP address logged."
 AuthUserFile "/etc/httpd/conf/htpasswd"
 Require valid-user
</Directory>

# DEV
<Directory /var/www/html/dev_20150216>
 Allow from all
 AuthType Basic
 AuthName "Your IP address will be logged."
 AuthUserFile "/etc/httpd/conf/htpasswd"

 Require valid-user
</Directory>
```

Restart Apache after having made the changes. Here is another way to restart Apache:

```
$ sudo apachectl restart
```

Note
Remember to add a new section like this for future development instances.

Making Dynamic DNS Work

TODO

Install ddclient

TODO

Retrieved from "http://wiki.matchi.info/index.php?title=Server_Build&oldid=1594"

Categories: Pages with syntax highlighting errors | Server | Daemons | Database Connection

---

- This page was last modified on 4 December 2016, at 10:39.
-