



## Table of contents

Table of contents.....	1
1. Preconditions.....	2
2. Quick Install Guide.....	3
3. Detailed description.....	4
3.1. Installation of Apache with the help of the package manager.....	4
3.2. First start of Apache.....	4
3.3. Install a new Apache Web.....	5
3.3.1. Installing a virtual host.....	5
3.3.2. Installing a new webroot.....	6
3.3.3. Test of the newly created web.....	7
3.4. Installation of Railo.....	7
3.4.1. Preconditions.....	7
3.4.2. Download and unpacking of the Railo-archive.....	7
3.4.3. Change the Rights for the Railo directory.....	8
3.4.4. First Start of Resin.....	8
3.4.5. Using the precompiled mod_caucho.so.....	9
3.4.6. Invoke Resin into Apache.....	9
3.5. Starting Resin at system start.....	10



## 1. Preconditions

The following document describes the mechanism of Railo on a Linux distribution. Since there is a multiplicity of distributions with different package managers, the commands are exemplary.

In this documentation the installation for the following constellation is described:

- Apache serves as Web server
- an individual Web `www.example.com` in Apache will call the Resin handler
- the Resin server runs under the same user, as Apache
- Railo and Resin are installed in the listing `/opt/railo/`
- The Web root of the example web is `/var/www/hosted/www.example.com/htdocs/`

Railo as an application within Resin is already preconfigured

### Hint

Please note, that the description of the foreign manufactured products Apache and Resin are not complete, but only a kind manual substitute. The complete documentations of Apache and Resin can be found here:

- Apache: http-Server: <http://httpd.apache.org/docs/2.0/>
- Resin: <http://www.caucho.com/resin-3.0/index.xtp>



## 2. Quick Install Guide

The following are the substantial steps, in order to implement the configuration

1. Installation of Apache
2. Setting up a virtual hosts in Apache
3. Unpack Railo and Resin
4. Implementation of mod\_caucho into Apache
5. Implementation of the Resin-handler
6. Start of Resin when the system boots



### 3. Detailed description

#### 3.1. Installation of Apache with the help of the package manager

First the Apache daemon is to be installed. You can compile Apache from the sources or use the precompiled version of the own distribution. In the second case the necessary commands depend on the installation tools and the package names on the used distribution.

The installation of packages demands root-rights. Most distribution lock the logon as root. In this case logon as a regular user and execute the command

```
su -
```

in order to work as root.

##### Special case Ubuntu

For administrative work Ubuntu uses the sudo mechanism:

```
user@ubuntu $ sudo apt-get install [package-name]
```

For sophisticated work as root you can open a shell with the help of the sudo-command:

```
user@ubuntu $ sudo bash
```

##### Mandriva

Mandriva uses rpm-packages, and has own tools next to the rpm- ones. These tools check for instance package dependencies and install missing packages:

```
root@mandrake $ urpmi [package-name]
```

#### 3.2. First start of Apache

The starting/stopping of Apache with the root user has to be done using the initscript. This script can be found in the folder /etc/init.d/.

Hint

Depending on the distribution the startscript is named

- httpd
- apache or
- apache2

This script is called with all necessary parameters start or stop, like:

```
$ /etc/init.d/apache2 start
```



To check whether Apache is running you can check the process list using

```
$ ps -ef | grep http
```

or by displaying netstat.

```
$ netstat -ap
```

In the column „Local Address“ after the colon you will find the port number 80 and in the right the process number and the name of the process, that binds this port.

In spite of the port number a name can be displayed as well. You can search for this name in /etc/services (ubuntu for eg. returns www as port).

```
grep „www“ /etc/services
```

### 3.3. Install a new Apache Web

#### 3.3.1. Installing a virtual host

Configure Apache according your desires.

##### Hint

You will find the configuration files most likely in one of the following directories:

- /user/local/apache/conf
- /etc/httpd/conf oder
- /etc/apache/conf

Install a new virtual host in the Apache configuration. These entries in most cases have to be done in the file httpd.conf.

Activate the usage of virtual hosts with the following notation

```
VirtualHosts *
```

Provide a directory section for the directory /var/www/hosted/ and allow access rights for incoming requests. Orient yourself at the directory-section of the already present default-webs.

```
<DI RECTORY /var/www/hosted/>
...
</DI RECTORY>
```

Provide a section virtualhost for the web (serveralias) www.example.com with the webroot /var/www/hosted/www.example.com/htdocs/.

```
<Vi rtual Host *>
  ServerAl ias www. exampl e. com
  DocumentRoot /var/www/hosted/www. exampl e. com/htdocs/
```



```

    DirectoryIndex index.cfm index.php index.htm index.html
    # further options ...
</VirtualHost>

```

### Special case virtual host in Ubuntu

The configuration of Apache in Ubuntu is located in `/etc/apache2/`. In here a lot of include-commands are splitting the `httpd.conf` file into many individual parts. There are directories containing the name „available“. Within them all shipped modules and configurations are placed thematically.

The includes are text files that are located in directories containing „enabled“ in their name. In these enabled-directories mostly same named softlinks pointing to the corresponding file in the available directory are contained.

Read more in the included readme file: `/etc/apache2/README`.

Steps:

1. Activate virtualhosts in the file `available_hosts/default`
2. Create e file `www.example.com` below `available_hosts` and enter the entries for the section `virtualhost`.
3. In the directory `enabled_hosts` create a softlink pointing to the file `./available_hosts/ www.example.com`

### 3.3.2. Installing a new webroot

Create the directory `/var/www/hosted/www.example.com/htdocs/`.

```
$ mkdir /var/www/hosted/www.example.com/htdocs
```

Just for testing purposes create a start file named `index.html` containing only your webname.

```
echo www.example.com >/var/www/hosted/www.example.com/htdocs/index.html
```

Pass the webroot to the Apache-user. Check the process list in order to see with which user the Apache-daemon currently runs.

#### Hint

Typical user and group names are:

- `wwwrun` (Suse)
- `www-data` (Ubuntu) or
- `apache`.

Using `id [username]` you can determine the group of the user.

By using `chown -R user:group directory` you can change the owner rights:

```
$ chown -R apache:apache /var/www/hosted/
```



### 3.3.3. Test of the newly created web

In order for the new configuration to be loaded you have to restart the Apache daemon. Before this you should check the configuration for valid syntax:

```
$ apache2ctl -t
```

If a message „Syntax OK“ is displayed, restart the service:

```
$ /etc/init.d/apache2 restart
```

Check the web by calling the web in a browser.

```
http://www.example.com/index.html
```

## 3.4. Installation of Railo

### 3.4.1. Preconditions

Railo is an application within Resin. In order to run Resin you need:

- a Java-runtime environment (shipped)
- Perl (must be installed on the system, needed for the mod\_caucho)

For the integration into Apache you need the file mod\_caucho.so which has to be loaded as a module. You can compile this file by yourself too. But therefore on your system you additionally need:

- sourcecode of your Apache-version
- the Java Development Kit (SDK or JDK)
- Compiler-tools, gcc and make

### 3.4.2. Download and unpacking of the Railo-archive

Download the Railo-archive (including Resin and JRE) from the website

<http://www.railo.ch>

```
$ wget (...)railo-1.0.0.027-resin-3.0.14-with-jre-linux.tar.gz
```

and copy/move it as root to /opt, in order to unpack it there. Since the unpacking of the archive generates a long directory name, we create a softlink with the name „railo“, which will point tot the future productive version.

```
$ mv railo*.tar.gz /opt
$ tar -xzf railo*.tar.gz
$ ln -s railo-1.0.0.026-resin-3.0.14-with-jre-linux railo
```

Unnecessary files can be deleted:



```
$ rm railo*.tar.gz
$ cd railo
$ rm -rf httpd.exe install -service.bat remove-service.bat setup.exe win32
```

Configuring the webs in resin.conf

Edit the file /opt/railo/conf/resin.conf – Enter the mapping pointing to your host:

```
(...)
<host id="www.example.com">
  <root-directory>
    /var/www/hosted/www.example.com/htdocs/
  </root-directory>
  <web-app id="/" document-directory="webapps/ROOT"/>
</host>
(...)
```

### 3.4.3. Change the Rights for the Railo directory

The Resin daemon is executed with the same user as Apache. This simplifies the vereinfacht the rights management for the access of applications on the Apache-webroot.

```
$ chown -R apache:apache /opt/railo
```

### 3.4.4. First Start of Resin

Change into the context of the Apache-user und start Resin

```
$ su - apache
$ /opt/railo/bin/httpd.sh start
```

In the directory /opt/railo/logs you can check whether the start of Railo went without any problems. After you enter

```
$ ps -ef
```

you should find the Resin process in this process list:

```
perl ./wrapper.pl -chdir -name httpd -class com.caucho.server.resin.Resin start
```

In the output of netstat –ap should be listed 2 java-processes listening to the ports 8600 and 6802 started as apache:

```
$ netstat -ap
Proto Recv-Q Send-Q Local Address Foreign Address State
PID/Program name
tcp6 0 0 localhost:6802 *:* LISTEN
12134/java
tcp6 0 0 *:8600 *:* LISTEN
12134/java
(...)
```



### 3.4.5. Using the precompiled mod\_caucho.so

Copy the file mod\_caucho.so from /opt/railo/ into the module-directory of Apache. Check the httpd.conf to see where from the other modules are loaded from.

### 3.4.6. Invoke Resin into Apache

In order to do this 3 steps are necessary:

1. Let the Resin-module be loaded on Apache start
2. Add a virtual host for the desired Web to the call of Resin
3. In the Apache configuration add a line in order to load the module: `LoadModule caucho_module /usr/lib/apache/mod_caucho.so`

#### Special case Ubuntu

Create a file [name].load in the directory /etc/apache2/mods-available/ containing the LoadModule command.

After this create a softlink pointing to the recently created file in the directory /etc/apache2/mods-enabled.

In the virtual hosts section add the Resin handler and the Resin-Server:

```
<VirtualHost *>
  DocumentRoot /var/www/hosted/www.example.com/htdocs/
  ServerAlias www.example.com
  DirectoryIndex index.cfm index.php index.htm index.html
  # ----- resin START
  <IfModule mod_caucho.c>
    ResinConfigServer localhost 6802
    SetHandler caucho-request
    CauchoStatus yes
    <Location /caucho-status>
      SetHandler caucho-status
    </Location>
  </IfModule>
  # ----- resin END
</VirtualHost>
```

The inserting of the Caucho status is optional and should be deleted for a productive Web. The line „SetHandler caucho-request“ is important since Apache then displays a 503 error in case of the Resin daemon not running anymore.

Without this line the plain source code would be sent to the client!

Check afterwards if the Apache configuration is OK and restart Apache:

```
$ apache2ctl -t
$ /etc/init.d/apache2 restart
```

In a browser, call the web. The resulted page must display a link to the Railo admin pages and some debug output of.



### 3.5. Starting Resin at system start

In order for a service to be started and stopped with the starting and stopping of the system, a startscript that can handle the parameters start and stop is necessary.

As a basis for such a script you can use the following one:

```
$ cat /etc/init.d/resin
#!/bin/sh
#
# start services for Resin
#
# Version: @(#) /etc/rc.d/init.d/resin 1.0
#
# description: Starts and stops services for Resin
#
# history:
# 2006-10-18 1.0 rlo init with start and stop
# rlo: Railo
# -----
# CONFIG
# -----
#. /etc/rc.d/init.d/functions
sAppName="Resin"
sUser=apache
RESIN_HOME=/opt/railo
JAVA_HOME=$RESIN_HOME/jre
PATH=$PATH: $JAVA_HOME/bin
export PATH JAVA_HOME RESIN_HOME
# -----
# MAIN
# -----
case "$1" in
start)
# gprintf "Starting ${sAppName} as user $sUser: "
echo "Starting ${sAppName} as user $sUser ..."
# su ${sUser} -c "$RESIN_HOME/bin/httpd.sh -java_home \"${*\} \"
su ${sUser} -c "$RESIN_HOME/bin/httpd.sh start"
;;
stop)
# gprintf "Stopping ${sAppName}: "
echo "Stopping ${sAppName} ..."
# su ${sUser} -c "$RESIN_HOME/bin/httpd.sh -java_home \"${*\} \"
su ${sUser} -c "$RESIN_HOME/bin/httpd.sh stop"
echo
;;
restart)
gprintf "not implemented yet"
;;
status)
ps -ef | grep ${sUser}
;;
*)
echo "*** Usage: $0 {start|stop|restart|status}"
exit 1
esac
exit 0
# -----
# EOF
# -----
```

Insert this script in the directories named /etc/rc[No. of the run level].d/. Inside these directories there exist files starting with an S (for start) and K (for kill)



followed by the number of the priority and the name of the service. For the priority to start and stop Resin just out of simplicity use the values set for the Apache daemon:

```
$ cd /etc/  
$ find . -name "*apache*"
./init.d/apache2
./rc0.d/K91apache2
./rc1.d/K91apache2
./rc2.d/S91apache2
./rc3.d/S91apache2
./rc4.d/S91apache2
./rc5.d/S91apache2
./rc6.d/K91apache2
```

Create the links for Resin with the same priority in all run levels:

```
$ cd rc0.d
$ ln -s K91resin ../init.d/resin
$ cd ../rc1.d
$ ln -s K91resin ../init.d/resin
(...)
```