

1. Installing Resin/Railo on CentOS

There are a lot of users who would like to test Railo on their Linux systems but who are confused about how to get started. In this short tutorial I'm going to show you how easy it is to get up and running with a simple app server on CentOS 5.2. More tutorials are going to follow over the next weeks.

1.1. Introduction

I set up a server for hostek.com which is offering Railo VPS hosting in the future as well. There might be some paths related to the system at hostek, but they are of course only arbitrary. Since some hosters have already agreed on offering Railo hosting, we will publicize a list of them shortly on our website. So let's continue with the installation of Railo/Resin on CentOS 5.2.

1.2. Preparing the System

For this tutorial we assume that you have an installed Linux system running CentOS 5.2 with the latest updates applied. There should be a basic installation of the Apache Web Server present (if it isn't don't worry, it will get installed automatically by some of the commands we're going to issue later on). In addition you will need the base development tools like **automake**, **autoconf**, **binutils** and **gcc**. Those can be found in the section „Development“, just use the standard settings there, no fancy stuff for X development needed.

Also, you will need a working SSH connection to your server as the superuser „root“.

To prepare the system for the configuration of Railo, some additional development libraries are needed. They are necessary because we need to compile an additional module (**mod_caucho.so**) for the Apache Web Server which essentially is the link between Apache and the Railo application server. Using CentOS, this is done with just one simple command on the shell (make sure you're logged in as the superuser „root“):

<code>



```
<code> yum install httpd-devel openssl-devel lynx
</code>
```

The installation of the above packages takes only a couple of seconds. If you receive an error chances are that the yum service is not available. Please have your sysadmin install it for you in order to proceed.

1.3. Download and installation of Railo and the JRE

After installation of the development packages you have to download the required Railo/Resin package from the Railo website.

The link is: <http://www.railo-technologies.com/download.cfm?item=/railo/remote/download/3.0.1.000/server/all/railo-3.0.1.000-resin-3.1.2-without-jre.tar.gz>

Since it is best practice to keep your directories clean, so we are going to create some additional directories to store the downloaded files first:

```
<code> cd ~
mkdir -p soft/tmp
cd soft/tmp
wget http://www.railo-technologies.com/download.cfm?item=/railo/remote/download/3.0.1.000/server/all/railo-3.0.1.000-resin-3.1.2-without-jre.tar.gz
</code>
```

As you may have noticed in the link name, the version downloaded comes without a JRE installed. You should therefore download a current JDK from Sun. Of course it might be a different one depending on the JDK version you are installing.

The link is: http://cds.sun.com/is-bin/INTERSHOP.enfinity/WFS/CDS-CDS_Developer-Site/en_US/-/USD/VerifyItem-Start/jdk-6u10-linux-i586.bin?BundledLineItemUUID=Zc1lBe.lm5oAAAE4fNzlni9&OrderID=Nb5lBe.lrBgAAAEdyPNzlni9&ProductID=7GBlBe.pq0AAAEb4aMQVbLE&FileName=/jdk-6u10-linux-i586.bin

IMPORTANT: When selecting the download, make sure to grab the JDK, not just the JRE! While the JDK contains everything needed to develop and therefore compile Java code and extensions, the JRE will only contain the code necessary to actually run Java apps. Since we are going to compile some stuff, the JRE won't suffice.

```
<code> wget -o jdk-6u10-linux-i586.bin "http://cds.sun.com/is-bin/INTERSHOP.enfinity/WFS/CDS-CDS_Developer-Site/en_US/-/USD/VerifyItem-Start/jdk-6u10-linux-i586.bin?BundledLineItemUUID=Zc1lBe.lm5oAAAE4fNzlni9&OrderID=Nb5lBe.lrBgAAAEdyPNzlni9&ProductID=7GBlBe.pq0AAAEb4aMQVbLE&FileName=/jdk-6u10-linux-i586.bin"
</code>
```

```
g0AAAAAEb4aMQVbLE&FileName=/jdk-6u10-linux-i586.bin"
</code>
```

To be able to use the file just downloaded, we have to make it executable. Also, again to keep the system clean and the paths consistent, we create another directory where we are going to install all our stuff:

```
<code>
  chmod 755 jdk-6u10-linux-i586.bin

  cd /opt
  mkdir soft
  cd soft
  /root/soft/tmp/jdk-6u10-linux-i586.bin
  ln -s jdk1.6.0_10 java
</code>
```

This installs the JRE 6.0 updater 10 together with the JDK. The JDK contains the JNI libraries that speed up the performance of Resin. To be able to easily switch to a newer version of the JDK once that becomes available, we created a symlink named „java“ in the last step. This gives us the additional benefit of being able to use a shorter name that's also easier to remember later on. Finally, we unpack the contents of the Railo package into the newly created folder /opt/soft and create a symbolic link to it as well.

```
<code>

#while still in /opt/soft
tar xzf /root/soft/tmp/railo-3.0.1.000-resin-3.1.2-with-jre.tar.gz
# shortcut to railo
ln -s railo-3.0.1.000-resin-3.1.2-with-jre railo
</code>
```

1.4. Compiling the connector module

First we have to configure the compilation. We have to point to the JDK home directory and to the path of the „apxs“ utility that has been installed with the httpd-devel package at the beginning. So in order to configure the compilation we call:

```
<code>
# Change into the Railo directory
cd railo
# configure the compile process
./configure --with-java-home=/opt/soft/java --with-apxs=/usr/sbin/apxs
</code>
```

The important thing to look after is a line called "checking for JNI in /opt/soft/java/include/linux ... found" in the output of the configuration utility execution. After the configuration we can build the module and install it:

```
<code>
```

```
# build the module
make
# install the module
make install
</code>
```

This step creates and installs the JNI libraries for Resin for a better performance of the application server. Also, the `mod_caucho` module for Apache is created and installed in the correct path. Lastly, some configuration changes are made to the `httpd.conf` of the Apache Web Server. To see what has happened in Apache you can edit the corresponding configuration file located here: `/etc/httpd/conf/httpd.conf`. At the end of the file you can see the following:

```
<code>
LoadModule caucho_module /usr/lib/httpd/modules/mod_caucho.so
ResinConfigServer localhost 6800
CauchoConfigCacheDirectory /tmp
CauchoStatus yes
</code>
```

There you can see that the Resin connector has been loaded and that it listens on port 6800. In addition some other configuration parameters have been inserted.

In order to start the application server you need to call the following script.
`/opt/soft/railo/bin/httpd.sh start`

But since it refers to a JRE that is only globally available on the system (without a path) we need to change the `java` setting to the correct path. It should be changed from `"java=java"` to `"java=/opt/soft/java/bin/java"`. Then we can start the service.

```
<code>
/opt/soft/railo/bin/httpd.sh start
</code>
```

In order to check whether everything is all right with Resin and Railo just check the log files located under `/opt/soft/railo/log`. There you should see any occurring errors. Apache is configured and the module should be loaded. In order to check the Apache status just do the following:

```
<code>
<!-- change into the apache directory -->
cd /etc/httpd
apachectl -t
<!-- this should display something like: Syntax OK -->
<!-- then we can restart apache -->
/etc/init.d/httpd restart
</code>
```

1.5. Testing the functionality

Now everything is up and running. In order to test Railo and Resin you can place a file called `caucho.conf` into the `conf.d` directory containing the following entries:

```
<code>
  <Location    /caucho-status>
    SetHandler  caucho-status
  </Location>
</code>
```

Now call the url: <http://ipaddress/caucho-status>. This should display the current status of Resin in your browser. In order to call Railo, just call the Railo administrator located here:

<http://ipaddress/railo-context/admin.cfm>

Then inside the server administrator you should set a server admin password and define a default password for all new web contexts. In order to configure Resin for multiple hosts, you can edit the `resin.conf` file and add host entries like for instance follows:

```
<code>
<host id="railo.hostek.com" root-directory="/opt/htdocs/wwwroot">
  <web-app id="/" root-directory="railo.hostek.com">
</host>
</code>
```

or if you want to create web contexts on a regexp basis on the fly, you can use the following:

```
<code>
  <host regexp="(.)">
    <host-name>${host.regexp[1]}</host-name>
    <root-directory>/opt/htdocs/wwwroot/${host.regexp[1]}</root-directory>
    <web-app id="/" document-directory="."/>
  </host>
</code>
```

This will implicitly create new web contexts inside the folder `/opt/htdocs/wwwroot` depending on the host name.

I hope this works for all of you. If you have any comments or questions, let me know.