Lab 4. Integration of Tomcat with the Apache Web Server

The Apache HTTP server is one of the most widely used frontend web servers across the IT industry. This project was open sourced in 1995 and is owned by The Apache Software Foundation.

This lab is very useful for the web administrator who works on enterprise-level web integration. It gives a very good idea about how integration is implemented in IT organizations. So, if you are thinking of enhancing your career in enterprise-level integrations of applications, then read this lab carefully.

In this lab, we will discuss the following topics:

- The Apache HTTP installation
- The various modules of Apache
- Integration of Apache with Tomcat 8
- How IT industry environments are set up

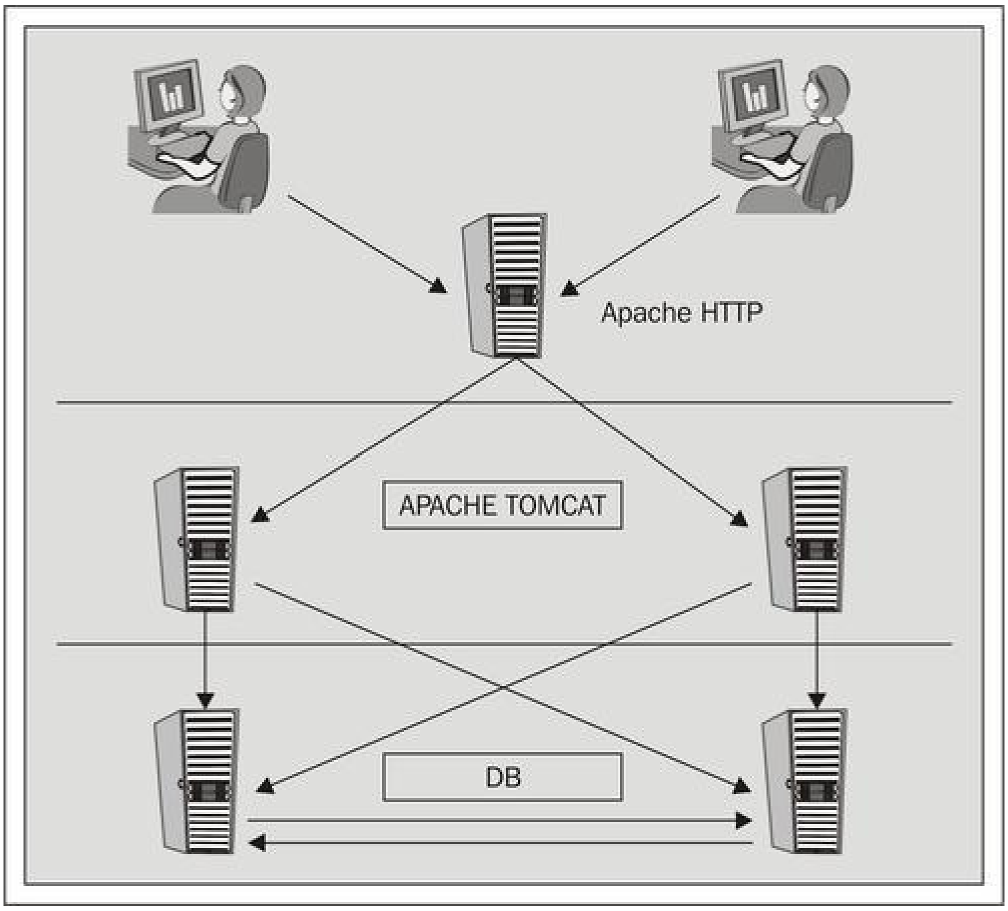
User request flow (web/application level)

Before we discuss the installation of Apache, let's discuss a high-level overview of how the request flows from the web and application server for an application in IT industries. The following figure shows the process flow for a user request, in a web application. The step-by-step involvement of each component is as follows:

1. The user hits the URL in the browser and the request goes to the HTTP server instead of Tomcat.

2. The HTTP server accepts the request and redirects it to Tomcat for business logic processing.

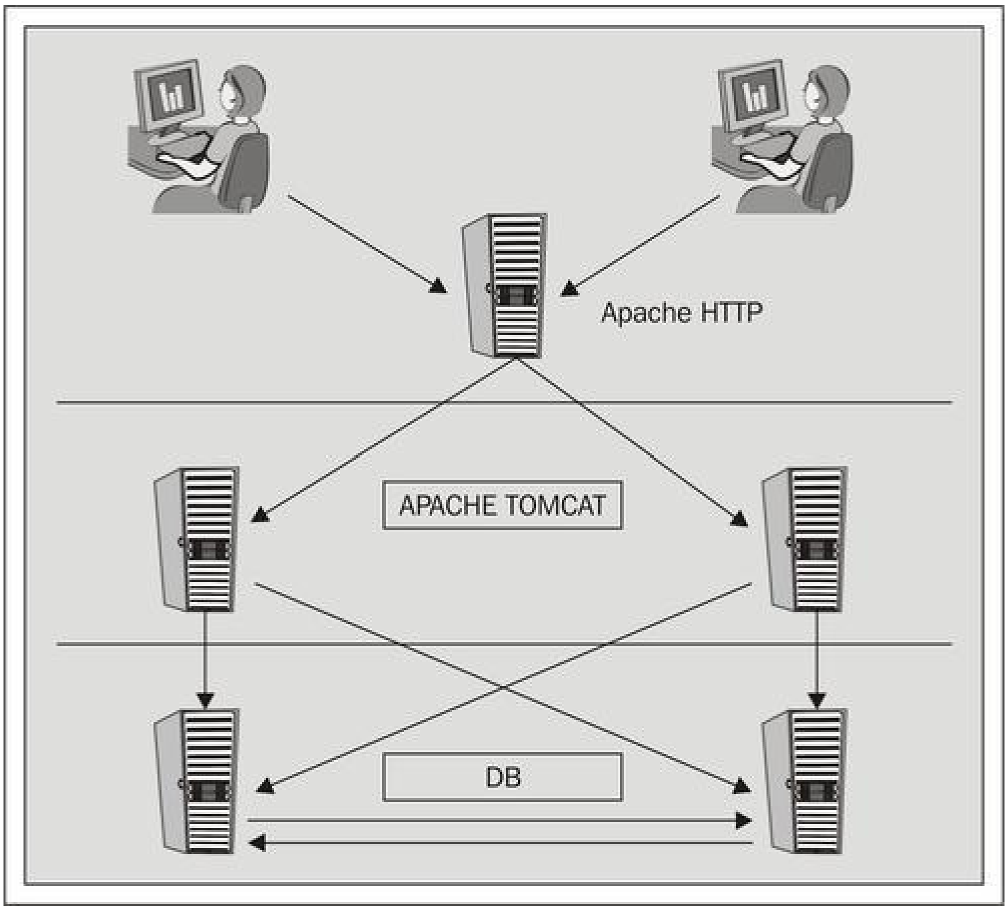
3. Tomcat internally contacts the database server to fetch the data, and sends the response back to the user through the same channel of request:



User request flow (web/application level)

Before we discuss the installation of Apache, let's discuss a high-level overview of how the request flows from the web and application server for an application in IT industries. The following figure shows the process flow for a user request, in a web application. The step-by-step involvement of each component is as follows:

1. The user hits the URL in the browser and the request goes to the HTTP server instead of Tomcat.
2. The HTTP server accepts the request and redirects it to Tomcat for business logic processing.
3. Tomcat internally contacts the database server to fetch the data, and sends the response back to the user through the same channel of request:



Why the Apache HTTP server

The Apache HTTP server is one of the most successful and common web servers used in IT industries. The reason being that it is supported by open source communities. In IT industries, the Apache HTTP server is heavily used as a frontend web server for the following reasons:

- Efficiently serves static content
- Increase the speed by 10 percent
- Clustering
- Security
- Multiple website hosting
- Modules
- Decorator

Note

We can create Redirect and Rewrites in application code also. These rules are in the form of servlet classes.

Installation of the Apache HTTP

The Apache installation can be done using various methods, based on the requirement of the infrastructure. For example, if you want to run multiple Apache instances on a single machine, then the Source installation will be used. There are mainly three types of installations done in various web environments:

- Source
- Binary
- RPM/exe

Installation of Apache HTTP on Linux (non-DOS environment)

In this installation, we will discuss the Source installation. Following are the steps involved in Source installation:

Note: Setup httpd-2.2.19.tar.gz has been downloaded and unzipped in /opt directory already.

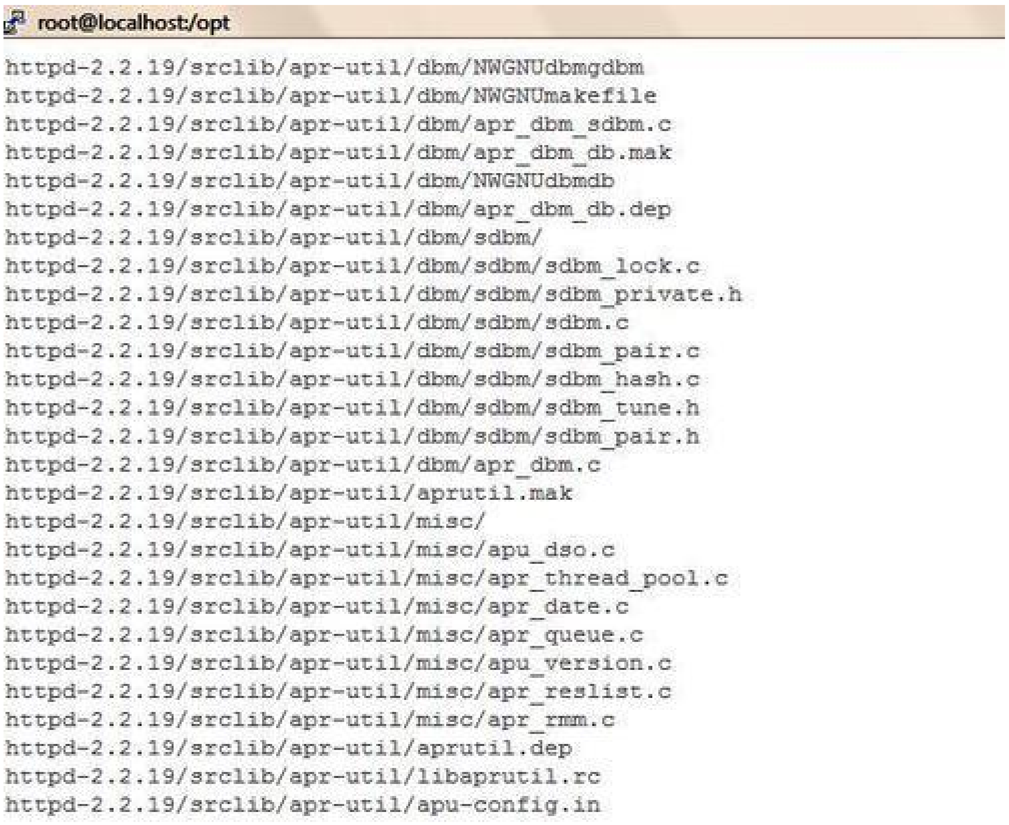
1. Download the Apache httpd server from the Apache official site. http://archive.apache.org/dist/httpd/httpd-2.2.19.tar.gz. The following screenshot shows the file to be downloaded (httpd-2.2.X.tar.gz where 2.2 is the major version and X is the minor version):

2. Once the download is complete, the source file is stored in the home directory of the user (in our case it's /root). The source file comes in the form of tar.gz. Run the following command to unzip the source. First create a folder httpd and then extract the content in the httpd directory.

   ```
   tar -zxvf httpd-2.2.X.tar.gz
   ```

Note

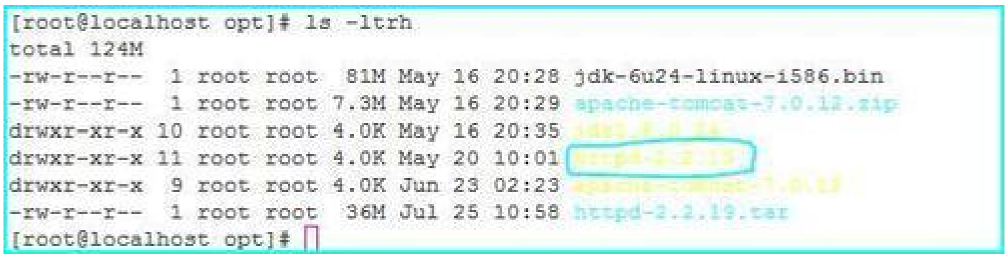
```
We have extracted the` httpd-2.2.19.tar.gz` in the /opt
directory.
```



```
httpd-2.2.19/srclib/apr-util/dbm/NWGNUdbmgdbm
httpd-2.2.19/srclib/apr-util/dbm/NWGNUmakefile
httpd-2.2.19/srclib/apr-util/dbm/apr_dbm_sdbm.c
httpd-2.2.19/srclib/apr-util/dbm/apr_dbm_db.mak
httpd-2.2.19/srclib/apr-util/dbm/NWGNUdbmdb
httpd-2.2.19/srclib/apr-util/dbm/apr_dbm_db.dep
httpd-2.2.19/srclib/apr-util/dbm/sdbm/
httpd-2.2.19/srclib/apr-util/dbm/sdbm/sdbm_lock.c
httpd-2.2.19/srclib/apr-util/dbm/sdbm/sdbm_private.h
httpd-2.2.19/srclib/apr-util/dbm/sdbm/sdbm.c
httpd-2.2.19/srclib/apr-util/dbm/sdbm/sdbm_pair.c
httpd-2.2.19/srclib/apr-util/dbm/sdbm/sdbm_hash.c
httpd-2.2.19/srclib/apr-util/dbm/sdbm/sdbm_tune.h
httpd-2.2.19/srclib/apr-util/dbm/sdbm/sdbm_pair.h
httpd-2.2.19/srclib/apr-util/dbm/apr_dbm.c
httpd-2.2.19/srclib/apr-util/aprutil.mak
httpd-2.2.19/srclib/apr-util/misc/
httpd-2.2.19/srclib/apr-util/misc/apu_dso.c
httpd-2.2.19/srclib/apr-util/misc/apr_thread_pool.c
httpd-2.2.19/srclib/apr-util/misc/apr_date.c
httpd-2.2.19/srclib/apr-util/misc/apr_queue.c
httpd-2.2.19/srclib/apr-util/misc/apu_version.c
httpd-2.2.19/srclib/apr-util/misc/apr_reslist.c
httpd-2.2.19/srclib/apr-util/misc/apr_rmm.c
httpd-2.2.19/srclib/apr-util/aprutil.dep
httpd-2.2.19/srclib/apr-util/libaprutil.rc
httpd-2.2.19/srclib/apr-util/apu-config.in
```

3. You can check the directory using the following command. The result displayed is similar to the following screenshot:

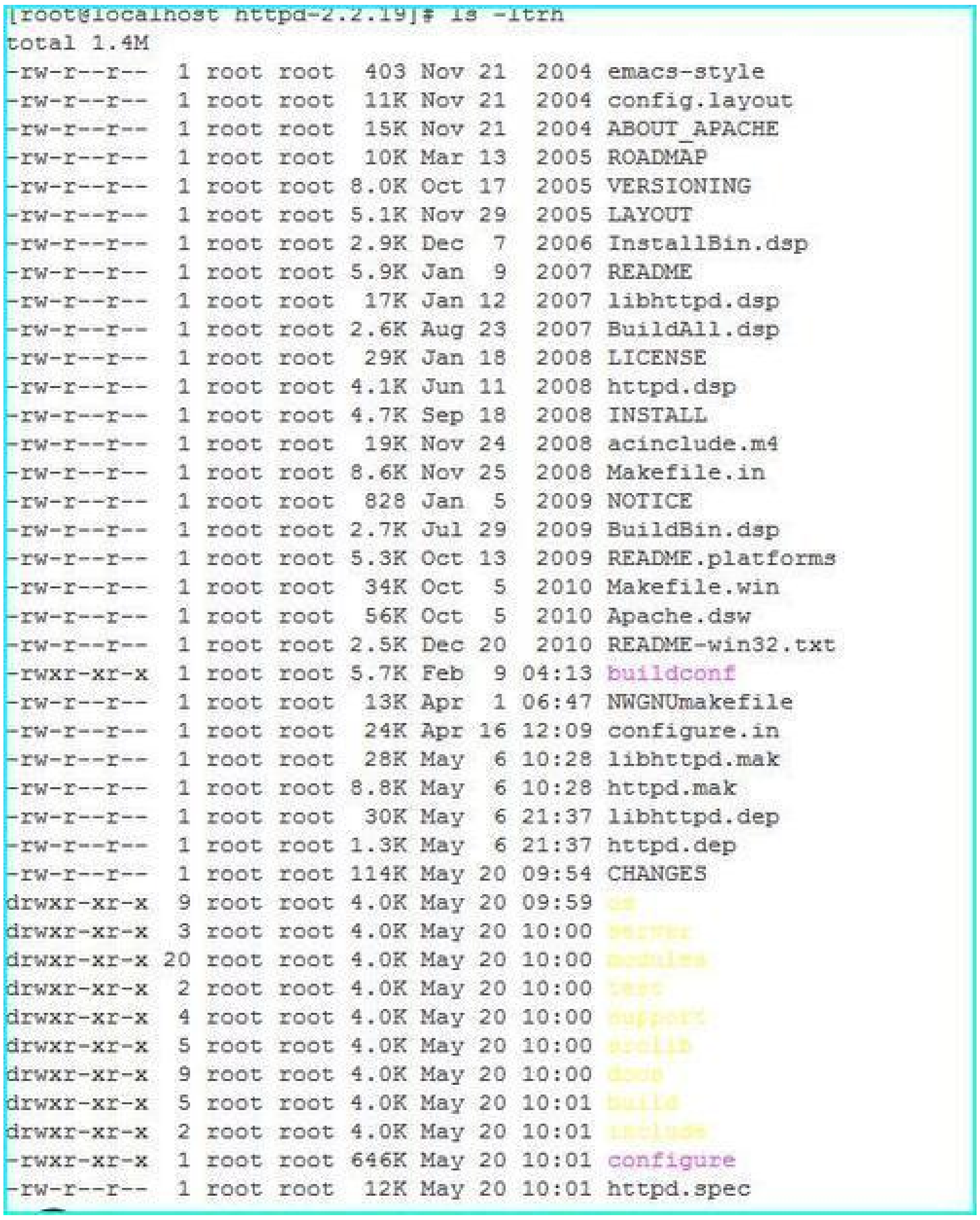
```
ls -ltrh
```



```
[root@localhost opt]# ls -ltrh
total 124M
-rw-r--r--  1 root root  81M May 16 20:28 jdk-6u24-linux-i586.bin
-rw-r--r--  1 root root 7.3M May 16 20:29 apache-tomcat-7.0.12.zip
drwxr-xr-x 10 root root 4.0K May 16 20:35 jdk1.6.0_24
drwxr-xr-x 11 root root 4.0K May 20 10:01 httpd-2.2.19
drwxr-xr-x  9 root root 4.0K Jun 23 02:23 apache-tomcat-7.0.19
-rw-r--r--  1 root root  36M Jul 25 10:58 httpd-2.2.19.tar
[root@localhost opt]# 
```

4. Then access the extracted directory using the following command. The result is similar to the following screenshot:

```
cd httpd-2.2.19
ls -ltrh
```



```
[root@localhost httpd-2.2.19]# ls -ltrh
total 1.4M
-rw-r--r--  1 root root  403 Nov 21  2004 emacs-style
-rw-r--r--  1 root root  11K Nov 21  2004 config.layout
-rw-r--r--  1 root root  15K Nov 21  2004 ABOUT_APACHE
-rw-r--r--  1 root root  10K Mar 13  2005 ROADMAP
-rw-r--r--  1 root root 8.0K Oct 17  2005 VERSIONING
-rw-r--r--  1 root root 5.1K Nov 29  2005 LAYOUT
-rw-r--r--  1 root root 2.9K Dec  7  2006 InstallBin.dsp
-rw-r--r--  1 root root 5.9K Jan  9  2007 README
-rw-r--r--  1 root root  17K Jan 12  2007 libhttpd.dsp
-rw-r--r--  1 root root 2.6K Aug 23  2007 BuildAll.dsp
-rw-r--r--  1 root root  29K Jan 18  2008 LICENSE
-rw-r--r--  1 root root 4.1K Jun 11  2008 httpd.dsp
-rw-r--r--  1 root root 4.7K Sep 18  2008 INSTALL
-rw-r--r--  1 root root  19K Nov 24  2008 acinclude.m4
-rw-r--r--  1 root root 8.6K Nov 25  2008 Makefile.in
-rw-r--r--  1 root root  828 Jan  5  2009 NOTICE
-rw-r--r--  1 root root 2.7K Jul 29  2009 BuildBin.dsp
-rw-r--r--  1 root root 5.3K Oct 13  2009 README.platforms
-rw-r--r--  1 root root  34K Oct  5  2010 Makefile.win
-rw-r--r--  1 root root  56K Oct  5  2010 Apache.dsw
-rw-r--r--  1 root root 2.5K Dec 20  2010 README-win32.txt
-rwxr-xr-x  1 root root 5.7K Feb  9 04:13 buildconf
-rw-r--r--  1 root root  13K Apr  1 06:47 NWGNUmakefile
-rw-r--r--  1 root root  24K Apr 16 12:09 configure.in
-rw-r--r--  1 root root  28K May  6 10:28 libhttpd.mak
-rw-r--r--  1 root root 8.8K May  6 10:28 httpd.mak
-rw-r--r--  1 root root  30K May  6 21:37 libhttpd.dep
-rw-r--r--  1 root root 1.3K May  6 21:37 httpd.dep
-rw-r--r--  1 root root 114K May 20 09:54 CHANGES
drwxr-xr-x  9 root root 4.0K May 20 09:59 os
drwxr-xr-x  3 root root 4.0K May 20 10:00 server
drwxr-xr-x 20 root root 4.0K May 20 10:00 modules
drwxr-xr-x  2 root root 4.0K May 20 10:00 test
drwxr-xr-x  4 root root 4.0K May 20 10:00 support
drwxr-xr-x  5 root root 4.0K May 20 10:00 srclib
drwxr-xr-x  9 root root 4.0K May 20 10:00 docs
drwxr-xr-x  5 root root 4.0K May 20 10:01 build
drwxr-xr-x  2 root root 4.0K May 20 10:01 include
-rwxr-xr-x  1 root root 646K May 20 10:01 configure
-rw-r--r--  1 root root  12K May 20 10:01 httpd.spec
```

5. After the verification of the directory, it's time to install the Apache HTTP server on Linux. By default, the execution permission is not set to true on the source folder. For that, we have to run the chown command to make it true.

   ```
   [root@localhost httpd-2.2.19]# chown 0755 configure
   ```

By default **Apache Portable Runtime** (**APR**) is not installed in the 2.2 version, we have to install it. Let's discuss APR and its utilities in detail.

Note

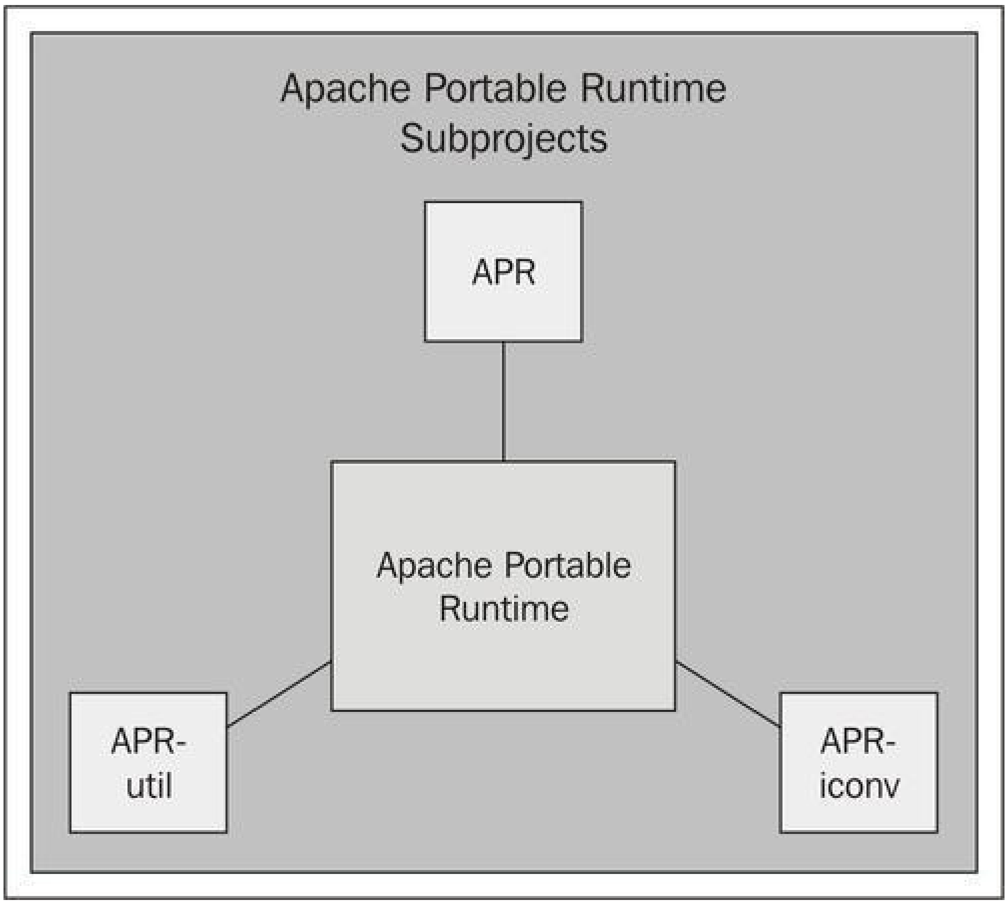
`/configure` with included APR is enabled from the version 2.2.3.

Apache Portable Runtime

Apache Portable Runtime is an open source project, which is supported by the Apache Foundation software. The main goal of this project is to provide the developer with an API, through which they can code and predict the identical behavior, regardless of different platforms. It eliminates the requirement of additional code dependency for different operating systems. For more information on this project, please visit http://apr.apache.org/.

Tomcat 8 uses APR to provide the capability of scalability, performance, and best collaboration with native technologies.

The Apache Portable Runtime project is again divided into three subprojects, to enhance and simplify the capability of this project. The following figure shows the different subprojects for APR:



APR is a portable runtime library, through which Apache integrates with other native technologies. It is also helpful in resolving the problem of threads and processes. For more information on APR, please visit http://apr.apache.org/docs/apr/trunk/index.html.

Apache Portable Runtime Utility (**APR-util**) is a companion library for APR. To install this utility, the GCC++ package should be installed to the OS (http://apr.apache.org/docs/apr-util/trunk/).

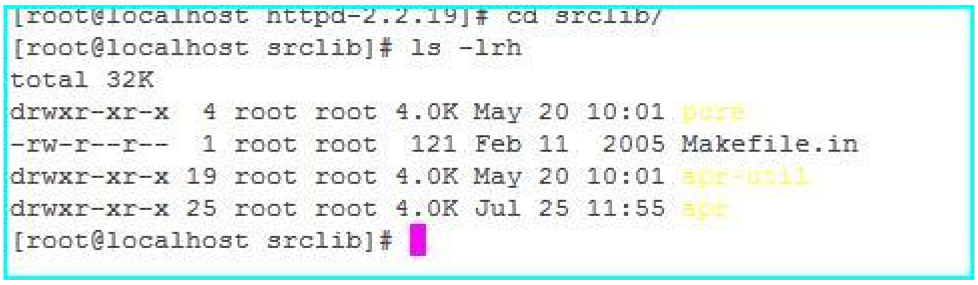
APR-iconv is a portable implementation of the `iconv()` library (http://apr.apache.org/docs/apr-iconv/trunk/).

Installation of APR/APR-util

APR/APR-util comes with the source of the Apache package, and they can be found in the following directories as shown in the following screenshot:

- APR: `Installdir/srclib/apr`
- APR-util: `Instaldir/srclib/apr-util`

As we have extracted the source in `/opt/httpd-2.2.19`, the source directory is also found in the same directory.



```
[root@localhost httpd-2.2.19]# cd srclib/
[root@localhost srclib]# ls -lrh
total 32K
drwxr-xr-x  4 root root 4.0K May 20 10:01 pcre
-rw-r--r--  1 root root  121 Feb 11  2005 Makefile.in
drwxr-xr-x 19 root root 4.0K May 20 10:01 apr-util
drwxr-xr-x 25 root root 4.0K Jul 25 11:55 apr
[root@localhost srclib]#
```

Let's start with the installation of APR, followed by the installation of APR-util.

The APR installation can be done in three steps using three commands. The steps are as follows:

1. Enter the source directory of apr and apr-util, then configure the code using the following commands:

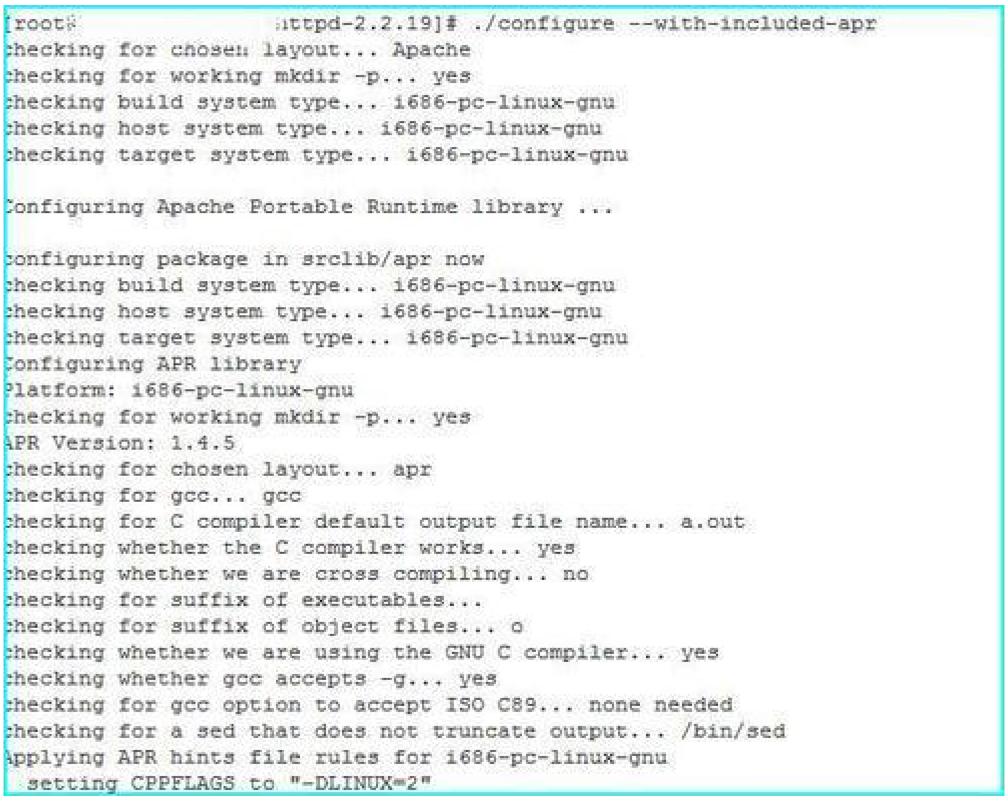
   ```
   [root@localhost srclib]# cd /opt/httpd-2.2.19/srclib/apr
   [root@localhost apr]# ./configure --prefix=/opt/httpd/apr-httpd/
   [root@localhost apr-util]# /configure --prefix=/usr/local/apr-util-httpd/ --
   with-apr=/usr/local/apr-httpd/
   Make
   Make install
   ```

Note

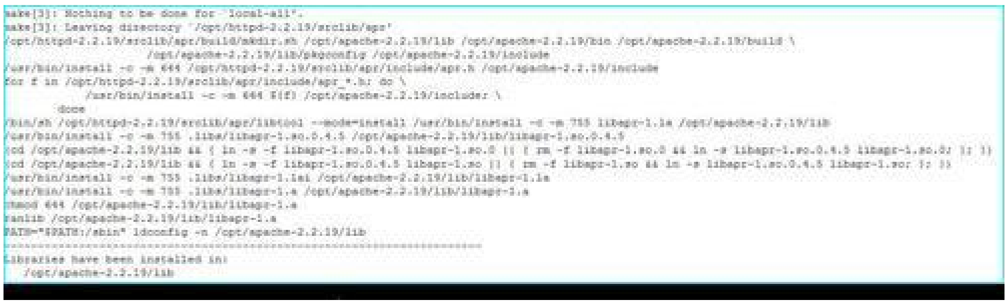
```
The APR/APR-util installation should be done first, if we are
compiling the source code of Apache manually. If we miss installing
APR/APR-util, then at the` make` command execution for
Apache, source will produce an error.
```

2. You can configure Apache using the following command. The following screenshot shows the output of the command when executed:

   ```
   [root@localhost httpd-2.2.19]#./configure --with-included-apr --
   prefix=/opt/apache-2.2.19
   ```



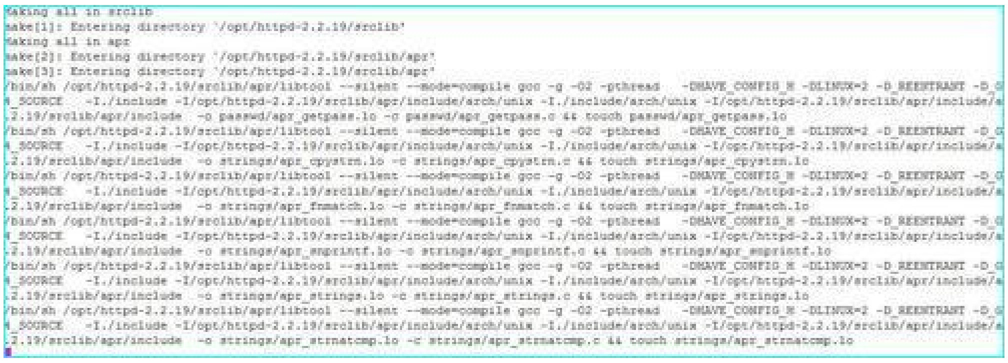
```
[root@                httpd-2.2.19]# ./configure --with-included-apr
checking for chosen layout... Apache
checking for working mkdir -p... yes
checking build system type... i686-pc-linux-gnu
checking host system type... i686-pc-linux-gnu
checking target system type... i686-pc-linux-gnu

Configuring Apache Portable Runtime library ...

configuring package in srclib/apr now
checking build system type... i686-pc-linux-gnu
checking host system type... i686-pc-linux-gnu
checking target system type... i686-pc-linux-gnu
Configuring APR library
Platform: i686-pc-linux-gnu
checking for working mkdir -p... yes
APR Version: 1.4.5
checking for chosen layout... apr
checking for gcc... gcc
checking for C compiler default output file name... a.out
checking whether the C compiler works... yes
checking whether we are cross compiling... no
checking for suffix of executables...
checking for suffix of object files... o
checking whether we are using the GNU C compiler... yes
checking whether gcc accepts -g... yes
checking for gcc option to accept ISO C89... none needed
checking for a sed that does not truncate output... /bin/sed
Applying APR hints file rules for i686-pc-linux-gnu
  setting CPPFLAGS to "-DLINUX=2"
```



```
make[3]: Nothing to be done for `local-all'.
make[3]: Leaving directory `/opt/httpd-2.2.19/srclib/apr'
/opt/httpd-2.2.19/srclib/apr/build/mkdir.sh /opt/apache-2.2.19/lib /opt/apache-2.2.19/bin /opt/apache-2.2.19/build \
                   /opt/apache-2.2.19/lib/pkgconfig /opt/apache-2.2.19/include
/usr/bin/install -c -m 644 /opt/httpd-2.2.19/srclib/apr/include/apr.h /opt/apache-2.2.19/include
for f in /opt/httpd-2.2.19/srclib/apr/include/apr_*.h; do \
            /usr/bin/install -c -m 644 ${f} /opt/apache-2.2.19/include; \
        done
/bin/sh /opt/httpd-2.2.19/srclib/apr/libtool --mode=install /usr/bin/install -c -m 755 libapr-1.la /opt/apache-2.2.19/lib
/usr/bin/install -c -m 755 .libs/libapr-1.so.0.4.5 /opt/apache-2.2.19/lib/libapr-1.so.0.4.5
(cd /opt/apache-2.2.19/lib && { ln -s -f libapr-1.so.0.4.5 libapr-1.so.0 || { rm -f libapr-1.so.0 && ln -s libapr-1.so.0.4.5 libapr-1.so.0; }; })
(cd /opt/apache-2.2.19/lib && { ln -s -f libapr-1.so.0.4.5 libapr-1.so || { rm -f libapr-1.so && ln -s libapr-1.so.0.4.5 libapr-1.so; }; })
/usr/bin/install -c -m 755 .libs/libapr-1.lai /opt/apache-2.2.19/lib/libapr-1.la
/usr/bin/install -c -m 755 .libs/libapr-1.a /opt/apache-2.2.19/lib/libapr-1.a
chmod 644 /opt/apache-2.2.19/lib/libapr-1.a
ranlib /opt/apache-2.2.19/lib/libapr-1.a
PATH="$PATH:/sbin" ldconfig -n /opt/apache-2.2.19/lib
----------------------------------------------------------------------
Libraries have been installed in:
   /opt/apache-2.2.19/lib
```

3. The previous screenshot describes the progress of the configure command. Once the command is executed, it will get the return code 0 otherwise you will see an error on the screen. Then, run the make command on the server to compile the code. The following figure shows the output of the make command:

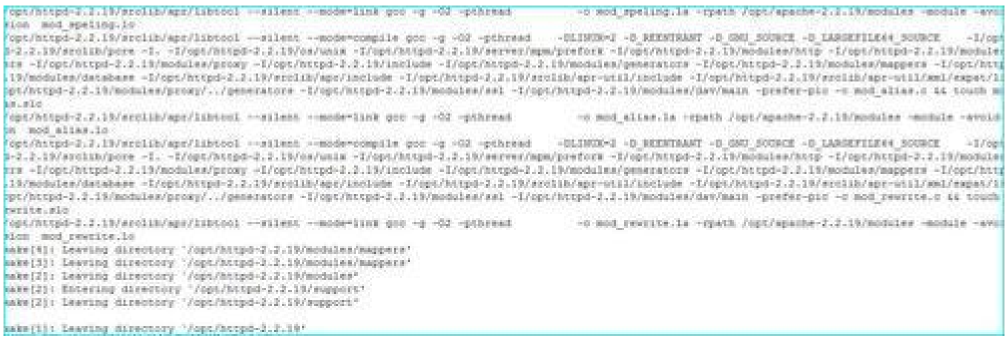
   ```
   [root@localhost httpd-2.2.X]#make
   ```



```
Making all in srclib
make[1]: Entering directory `/opt/httpd-2.2.19/srclib'
Making all in apr
make[2]: Entering directory `/opt/httpd-2.2.19/srclib/apr'
make[3]: Entering directory `/opt/httpd-2.2.19/srclib/apr'
/bin/sh /opt/httpd-2.2.19/srclib/apr/libtool --silent --mode=compile gcc -g -O2 -pthread   -DHAVE_CONFIG_H -DLINUX=2 -D_REENTRANT -D_G
N_SOURCE   -I./include -I/opt/httpd-2.2.19/srclib/apr/include/arch/unix -I./include/arch/unix -I/opt/httpd-2.2.19/srclib/apr/include/a
.2.19/srclib/apr/include  -o passwd/apr_getpass.lo -c passwd/apr_getpass.c && touch passwd/apr_getpass.lo
/bin/sh /opt/httpd-2.2.19/srclib/apr/libtool --silent --mode=compile gcc -g -O2 -pthread   -DHAVE_CONFIG_H -DLINUX=2 -D_REENTRANT -D_G
N_SOURCE   -I./include -I/opt/httpd-2.2.19/srclib/apr/include/arch/unix -I./include/arch/unix -I/opt/httpd-2.2.19/srclib/apr/include/a
.2.19/srclib/apr/include  -o strings/apr_cpystrn.lo -c strings/apr_cpystrn.c && touch strings/apr_cpystrn.lo
/bin/sh /opt/httpd-2.2.19/srclib/apr/libtool --silent --mode=compile gcc -g -O2 -pthread   -DHAVE_CONFIG_H -DLINUX=2 -D_REENTRANT -D_G
N_SOURCE   -I./include -I/opt/httpd-2.2.19/srclib/apr/include/arch/unix -I./include/arch/unix -I/opt/httpd-2.2.19/srclib/apr/include/a
.2.19/srclib/apr/include  -o strings/apr_fnmatch.lo -c strings/apr_fnmatch.c && touch strings/apr_fnmatch.lo
/bin/sh /opt/httpd-2.2.19/srclib/apr/libtool --silent --mode=compile gcc -g -O2 -pthread   -DHAVE_CONFIG_H -DLINUX=2 -D_REENTRANT -D_G
N_SOURCE   -I./include -I/opt/httpd-2.2.19/srclib/apr/include/arch/unix -I./include/arch/unix -I/opt/httpd-2.2.19/srclib/apr/include/a
.2.19/srclib/apr/include  -o strings/apr_snprintf.lo -c strings/apr_snprintf.c && touch strings/apr_snprintf.lo
/bin/sh /opt/httpd-2.2.19/srclib/apr/libtool --silent --mode=compile gcc -g -O2 -pthread   -DHAVE_CONFIG_H -DLINUX=2 -D_REENTRANT -D_G
N_SOURCE   -I./include -I/opt/httpd-2.2.19/srclib/apr/include/arch/unix -I./include/arch/unix -I/opt/httpd-2.2.19/srclib/apr/include/a
.2.19/srclib/apr/include  -o strings/apr_strings.lo -c strings/apr_strings.c && touch strings/apr_strings.lo
/bin/sh /opt/httpd-2.2.19/srclib/apr/libtool --silent --mode=compile gcc -g -O2 -pthread   -DHAVE_CONFIG_H -DLINUX=2 -D_REENTRANT -D_G
N_SOURCE   -I./include -I/opt/httpd-2.2.19/srclib/apr/include/arch/unix -I./include/arch/unix -I/opt/httpd-2.2.19/srclib/apr/include/a
.2.19/srclib/apr/include  -o strings/apr_strnatcmp.lo -c strings/apr_strnatcmp.c && touch strings/apr_strnatcmp.lo
```

Note

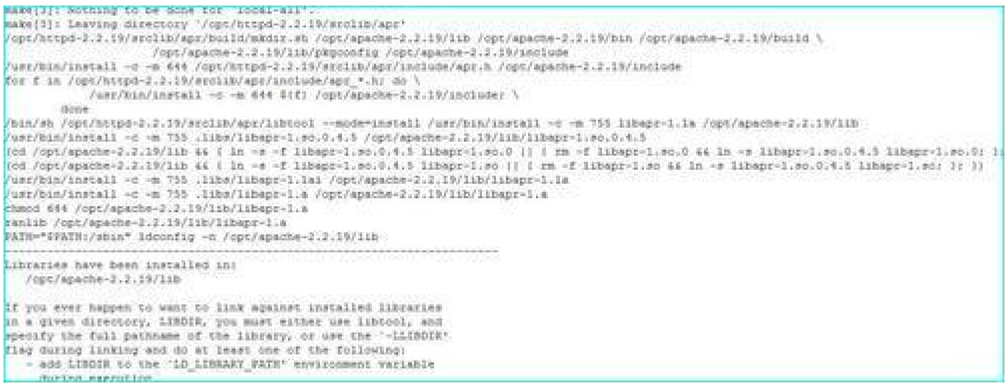
```
It is very important to check the output of the` make`
command, as it gives an error most of the times.
```



```
/opt/httpd-2.2.19/srclib/apr/libtool --silent --mode=link gcc -g -O2 -pthread         -o mod_speling.la -rpath /opt/apache-2.2.19/modules -module -avo
sion  mod_speling.lo
/opt/httpd-2.2.19/srclib/apr/libtool --silent --mode=compile gcc -g -O2 -pthread    -DLINUX=2 -D_REENTRANT -D_GNU_SOURCE -D_LARGEFILE64_SOURCE    -I/op
d-2.2.19/srclib/pcre -I. -I/opt/httpd-2.2.19/os/unix -I/opt/httpd-2.2.19/server/mpm/prefork -I/opt/httpd-2.2.19/modules/http -I/opt/httpd-2.2.19/module
ers -I/opt/httpd-2.2.19/modules/proxy -I/opt/httpd-2.2.19/include -I/opt/httpd-2.2.19/modules/generators -I/opt/httpd-2.2.19/modules/mappers -I/opt/http
.19/modules/database -I/opt/httpd-2.2.19/srclib/apr/include -I/opt/httpd-2.2.19/srclib/apr-util/include -I/opt/httpd-2.2.19/srclib/apr-util/xml/expat/li
pt/httpd-2.2.19/modules/proxy/../generators -I/opt/httpd-2.2.19/modules/ssl -I/opt/httpd-2.2.19/modules/dav/main -prefer-pic -c mod_alias.c && touch mo
as.slo
/opt/httpd-2.2.19/srclib/apr/libtool --silent --mode=link gcc -g -O2 -pthread         -o mod_alias.la -rpath /opt/apache-2.2.19/modules -module -avoid
on  mod_alias.lo
/opt/httpd-2.2.19/srclib/apr/libtool --silent --mode=compile gcc -g -O2 -pthread    -DLINUX=2 -D_REENTRANT -D_GNU_SOURCE -D_LARGEFILE64_SOURCE    -I/op
d-2.2.19/srclib/pcre -I. -I/opt/httpd-2.2.19/os/unix -I/opt/httpd-2.2.19/server/mpm/prefork -I/opt/httpd-2.2.19/modules/http -I/opt/httpd-2.2.19/module
ers -I/opt/httpd-2.2.19/modules/proxy -I/opt/httpd-2.2.19/include -I/opt/httpd-2.2.19/modules/generators -I/opt/httpd-2.2.19/modules/mappers -I/opt/http
.19/modules/database -I/opt/httpd-2.2.19/srclib/apr/include -I/opt/httpd-2.2.19/srclib/apr-util/include -I/opt/httpd-2.2.19/srclib/apr-util/xml/expat/li
pt/httpd-2.2.19/modules/proxy/../generators -I/opt/httpd-2.2.19/modules/ssl -I/opt/httpd-2.2.19/modules/dav/main -prefer-pic -c mod_rewrite.c && touch
rewrite.slo
/opt/httpd-2.2.19/srclib/apr/libtool --silent --mode=link gcc -g -O2 -pthread         -o mod_rewrite.la -rpath /opt/apache-2.2.19/modules -module -avo
sion  mod_rewrite.lo
make[4]: Leaving directory `/opt/httpd-2.2.19/modules/mappers'
make[3]: Leaving directory `/opt/httpd-2.2.19/modules/mappers'
make[2]: Leaving directory `/opt/httpd-2.2.19/modules'
make[2]: Entering directory `/opt/httpd-2.2.19/support'
make[2]: Leaving directory `/opt/httpd-2.2.19/support'

make[1]: Leaving directory `/opt/httpd-2.2.19'
```

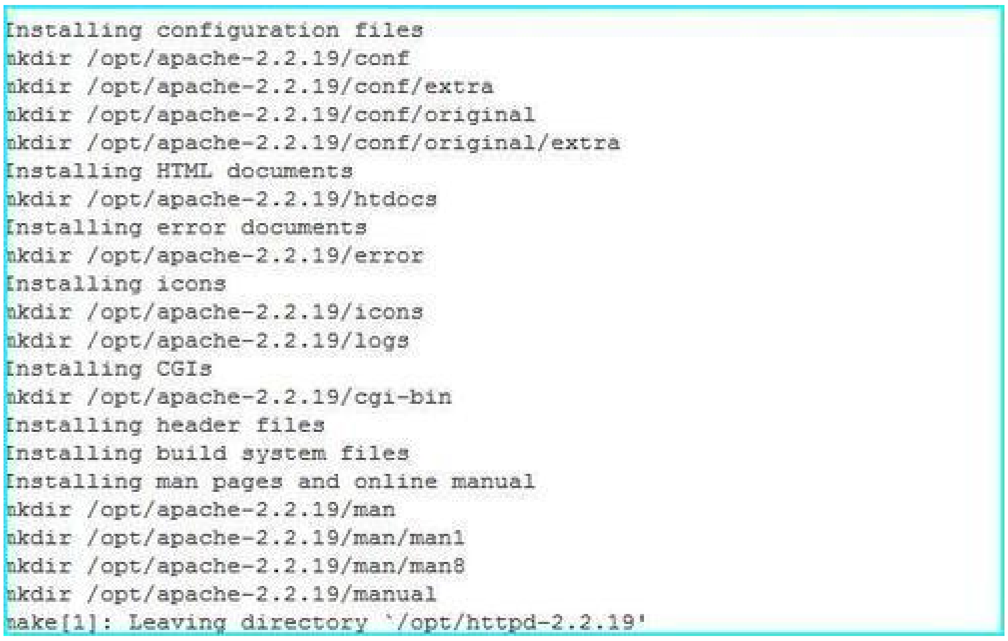
4. The previous and the following screenshots show the completion without any error. To proceed with the installation of make, we have to run the following command:

   ```
   [root@localhost httpd-2.2.X]#make install
   ```



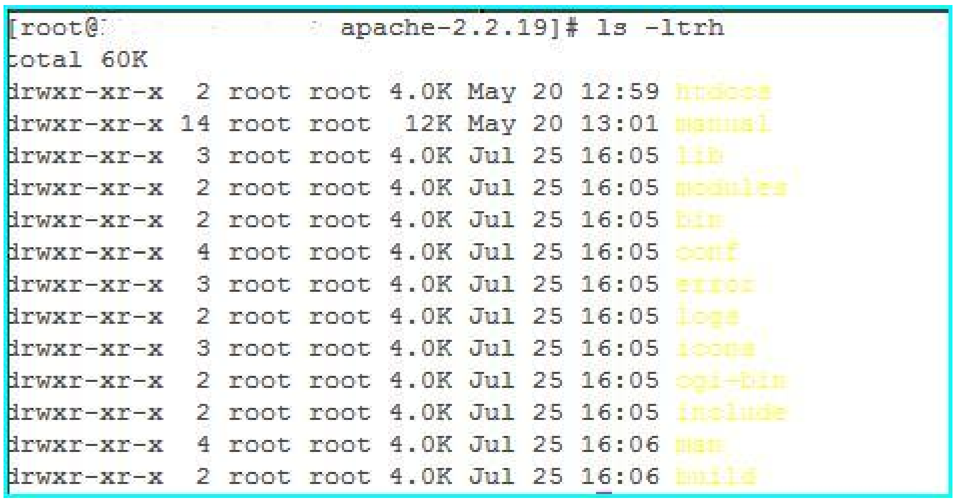
```
make[3]: Nothing to be done for `local-all'.
make[3]: Leaving directory `/opt/httpd-2.2.19/srclib/apr'
/opt/httpd-2.2.19/srclib/apr/build/mkdir.sh /opt/apache-2.2.19/lib /opt/apache-2.2.19/bin /opt/apache-2.2.19/build \
                     /opt/apache-2.2.19/lib/pkgconfig /opt/apache-2.2.19/include
/usr/bin/install -c -m 644 /opt/httpd-2.2.19/srclib/apr/include/apr.h /opt/apache-2.2.19/include
for f in /opt/httpd-2.2.19/srclib/apr/include/apr_*.h; do \
            /usr/bin/install -c -m 644 ${f} /opt/apache-2.2.19/includer \
        done
/bin/sh /opt/httpd-2.2.19/srclib/apr/libtool --mode=install /usr/bin/install -c -m 755 libapr-1.la /opt/apache-2.2.19/lib
/usr/bin/install -c -m 755 .libs/libapr-1.so.0.4.5 /opt/apache-2.2.19/lib/libapr-1.so.0.4.5
(cd /opt/apache-2.2.19/lib && { ln -s -f libapr-1.so.0.4.5 libapr-1.so.0 || { rm -f libapr-1.so.0 && ln -s libapr-1.so.0.4.5 libapr-1.so.0; };
(cd /opt/apache-2.2.19/lib && { ln -s -f libapr-1.so.0.4.5 libapr-1.so || { rm -f libapr-1.so && ln -s libapr-1.so.0.4.5 libapr-1.so; }; })
/usr/bin/install -c -m 755 .libs/libapr-1.lai /opt/apache-2.2.19/lib/libapr-1.la
/usr/bin/install -c -m 755 .libs/libapr-1.a /opt/apache-2.2.19/lib/libapr-1.a
chmod 644 /opt/apache-2.2.19/lib/libapr-1.a
ranlib /opt/apache-2.2.19/lib/libapr-1.a
PATH="$PATH:/sbin" ldconfig -n /opt/apache-2.2.19/lib
----------------------------------------------------------------------
Libraries have been installed in:
   /opt/apache-2.2.19/lib

If you ever happen to want to link against installed libraries
in a given directory, LIBDIR, you must either use libtool, and
specify the full pathname of the library, or use the `-LLIBDIR'
flag during linking and do at least one of the following:
   - add LIBDIR to the `LD_LIBRARY_PATH' environment variable
     during execution
```

5. The previous command installs the Apache HTTP on the server, as shown in the following screenshot. It shows the completion on the server. If you view the previous screenshot, you will find that it creates the directory structure, files, manpage and htdocs, as shown in the next screenshot:



```
Installing configuration files
nkdir /opt/apache-2.2.19/conf
nkdir /opt/apache-2.2.19/conf/extra
nkdir /opt/apache-2.2.19/conf/original
nkdir /opt/apache-2.2.19/conf/original/extra
Installing HTML documents
nkdir /opt/apache-2.2.19/htdocs
Installing error documents
nkdir /opt/apache-2.2.19/error
Installing icons
nkdir /opt/apache-2.2.19/icons
nkdir /opt/apache-2.2.19/logs
Installing CGIs
nkdir /opt/apache-2.2.19/cgi-bin
Installing header files
Installing build system files
Installing man pages and online manual
nkdir /opt/apache-2.2.19/man
nkdir /opt/apache-2.2.19/man/man1
nkdir /opt/apache-2.2.19/man/man8
nkdir /opt/apache-2.2.19/manual
nake[1]: Leaving directory `/opt/httpd-2.2.19'
```

6. After the make install is complete, the directory structure of the Apache HTTP server is created in the current path of the installation. Let's quickly see how the directory looks. The following screenshot shows the directory structure of the Apache HTTP server. In 90 percent of cases, Apache administrators work on the conf, modules, and htdocs directories for performing day-to-day operations.



```
[root@[redacted] apache-2.2.19]# ls -ltrh
cotal 60K
drwxr-xr-x  2 root root 4.0K May 20 12:59 htdocs
drwxr-xr-x 14 root root  12K May 20 13:01 manual
drwxr-xr-x  3 root root 4.0K Jul 25 16:05 lib
drwxr-xr-x  2 root root 4.0K Jul 25 16:05 modules
drwxr-xr-x  2 root root 4.0K Jul 25 16:05 bin
drwxr-xr-x  4 root root 4.0K Jul 25 16:05 conf
drwxr-xr-x  3 root root 4.0K Jul 25 16:05 error
drwxr-xr-x  2 root root 4.0K Jul 25 16:05 logs
drwxr-xr-x  3 root root 4.0K Jul 25 16:05 icons
drwxr-xr-x  2 root root 4.0K Jul 25 16:05 cgi-bin
drwxr-xr-x  2 root root 4.0K Jul 25 16:05 include
drwxr-xr-x  4 root root 4.0K Jul 25 16:06 man
drwxr-xr-x  2 root root 4.0K Jul 25 16:06 build
```

7. Before we end the installation, it is necessary to start the services of HTTP to verify the instance is properly installed. The best way to check the configuration is by running the configtest script. This script comes by default with Apache httpd, only in a non-DOS environment. The script can be found in APACHE_HOME/bin.

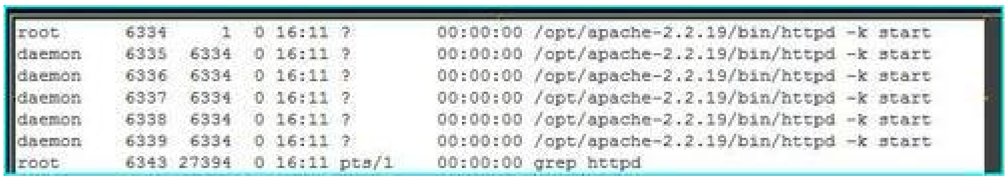
```
[root@localhost bin]# ./apachectl configtest
Syntax OK
```

- Then restart Apache using the following command:

```
[root@root@localhost bin]# ./apachectl start
```

- Once you start Apache, it's very important to verify the instance status. You can verify the system using the `ps` command:

```
ps -ef |grep httpd
```



```
root      6334     1  0 16:11 ?        00:00:00 /opt/apache-2.2.19/bin/httpd -k start
daemon    6335  6334  0 16:11 ?        00:00:00 /opt/apache-2.2.19/bin/httpd -k start
daemon    6336  6334  0 16:11 ?        00:00:00 /opt/apache-2.2.19/bin/httpd -k start
daemon    6337  6334  0 16:11 ?        00:00:00 /opt/apache-2.2.19/bin/httpd -k start
daemon    6338  6334  0 16:11 ?        00:00:00 /opt/apache-2.2.19/bin/httpd -k start
daemon    6339  6334  0 16:11 ?        00:00:00 /opt/apache-2.2.19/bin/httpd -k start
root      6343 27394  0 16:11 pts/1    00:00:00 grep httpd
```

The previous screenshot shows the status of the HTTP process, this means the HTTP server is running properly.

Note

We can directly install the Apache package in Debain Linux (Ubuntu), using the `apt-get` command. The following command shows the syntax for the installation:

`sudo apt-get install apache2`

Also, you can install the Apache using the `yum` utility. This utility is used mainly in CentOS using the command:

`yum -y install httpd`

Apache Jserv protocol

This protocol was mainly developed to transfer data over the network in binary format instead of plain text. It uses TCP and a packet-based protocol, hence, increasing the performance of the web servers. Another informational point is that decryption of requests is done on the web server end so that the application server doesn't have a high load.

Note

If you are using AJP, the network traffic is reduced, as the tariff passes over the TCP protocol.

`mod_jk` and `mod_proxy` are based on the AJP protocol. They are also helpful in transmitting a high content response over the browser.

Note

If we use the latest version of `mod_jk` for integration of Apache and Tomcat, then we can store the response header of 64k in the web browsers. This process is very useful in the case of SSO enabled applications or storing Java session values in the browser.

Installation and configuration of mod_jk

`mod_jk` is an AJP connector which is used to integrate web servers such as Apache or IIS to Tomcat 8. In case we don't install `mod_jk`, then we cannot use frontend web servers for Tomcat. This module is very helpful in order to hide Tomcat behind the frontend web server and also eliminates the port number while browsing the URL. It involves multiple steps starting from installation and configuration. Let's first discuss the installation of `mod_jk`.

Installation of mod_jk

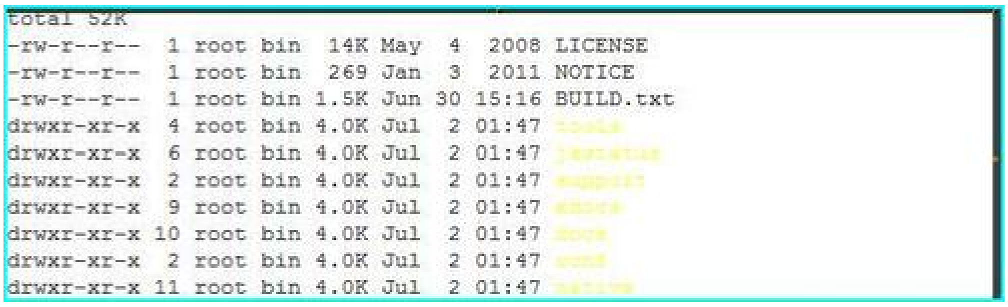
The `mod_jk` source can be downloaded from its official site, http://tomcat.apache.org/download-connectors.cgi. It is always recommended to download the latest stable version from the site for the implementation.

1. Once the source is downloaded we have to extract it in the server directory using the following command:

    ```
    [root@localhost opt]# tar -zxvf tomcat-connectors-1.2.x-src.tar
    ```

- where x is the minor version number.

2. Once the code is extracted, a directory is created in the current path named as tomcat-connectors-1.2.32. It's the home directory of the mod_jk source. The following screenshot shows the extracted code in the tomcat-connectors-1.2.32 directory, which is created after the execution of the previous command:



```
total 52K
-rw-r--r--  1 root bin  14K May  4  2008 LICENSE
-rw-r--r--  1 root bin  269 Jan  3  2011 NOTICE
-rw-r--r--  1 root bin 1.5K Jun 30 15:16 BUILD.txt
drwxr-xr-x  4 root bin 4.0K Jul  2 01:47 tools
drwxr-xr-x  6 root bin 4.0K Jul  2 01:47 jkstatus
drwxr-xr-x  2 root bin 4.0K Jul  2 01:47 support
drwxr-xr-x  9 root bin 4.0K Jul  2 01:47 xdocs
drwxr-xr-x 10 root bin 4.0K Jul  2 01:47 docs
drwxr-xr-x  2 root bin 4.0K Jul  2 01:47 conf
drwxr-xr-x 11 root bin 4.0K Jul  2 01:47 native
```

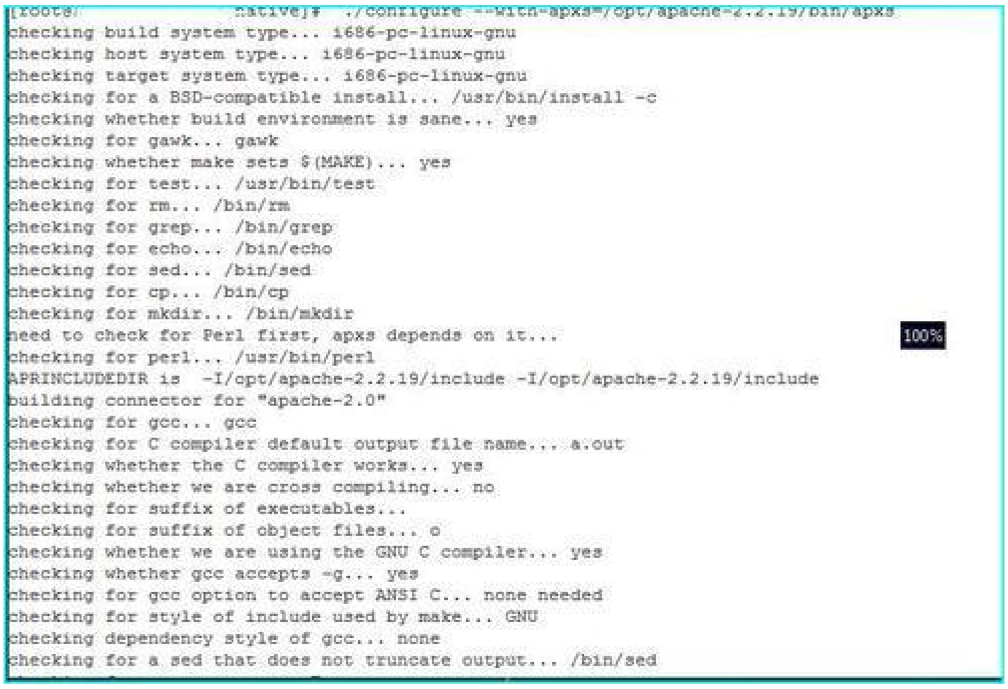
3. Go to the native directory of the mod_jk source using the following command and then run the configure command:

    ```
    [root@localhost opt]# cd /opt/tomcat-connectors-1.2.32-src/native ./configure -
    -with-apxs=/opt/apache-2.2.19/bin/apxs
    ```

Note

```
**Tip for configuration**

`mod_jk` is specific to the Apache version and the[
**Apache Extension Tool** (**APXS**) should be
used for the current version of Apache, which we will use in the
environment.

Once` mod_jk` is compiled on one server, there is no need
to generate for another Apache instance. It can be directly copied
to the other instance.

This trick is tested on Linux only.
```

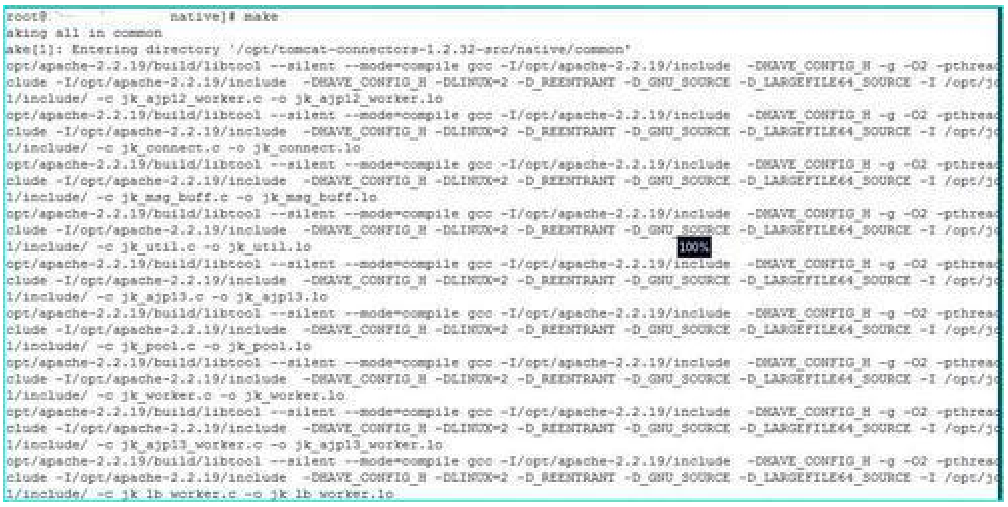
4. The following screenshot shows the installation process using the APXS module. Installation of mod_jk begins by running the configure command:



```
[root@            native]#  ./configure --with-apxs=/opt/apache-2.2.19/bin/apxs
checking build system type... i686-pc-linux-gnu
checking host system type... i686-pc-linux-gnu
checking target system type... i686-pc-linux-gnu
checking for a BSD-compatible install... /usr/bin/install -c
checking whether build environment is sane... yes
checking for gawk... gawk
checking whether make sets $(MAKE)... yes
checking for test... /usr/bin/test
checking for rm... /bin/rm
checking for grep... /bin/grep
checking for echo... /bin/echo
checking for sed... /bin/sed
checking for cp... /bin/cp
checking for mkdir... /bin/mkdir
need to check for Perl first, apxs depends on it...
checking for perl... /usr/bin/perl
APRINCLUDEDIR is  -I/opt/apache-2.2.19/include -I/opt/apache-2.2.19/include
building connector for "apache-2.0"
checking for gcc... gcc
checking for C compiler default output file name... a.out
checking whether the C compiler works... yes
checking whether we are cross compiling... no
checking for suffix of executables...
checking for suffix of object files... o
checking whether we are using the GNU C compiler... yes
checking whether gcc accepts -g... yes
checking for gcc option to accept ANSI C... none needed
checking for style of include used by make... GNU
checking dependency style of gcc... none
checking for a sed that does not truncate output... /bin/sed
```

5. Once the configuration is done, you need to run the make command, which compiles the source code, as shown in the following screenshot:

```
[root@localhost apache-2.0]# make
```

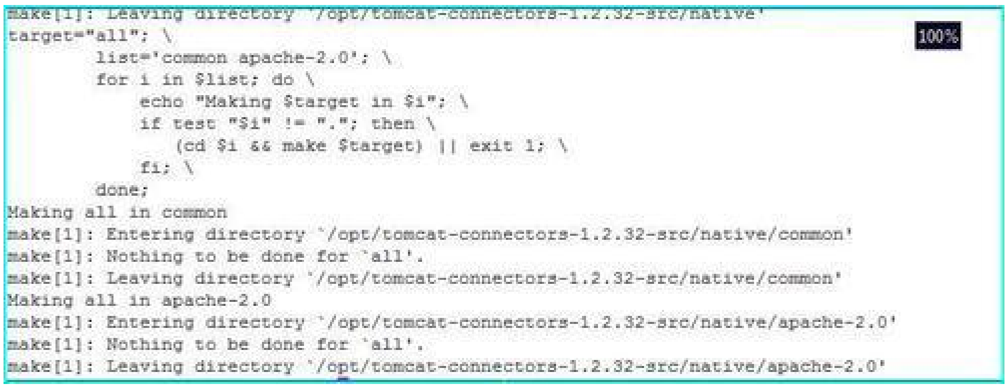


```
root@            native]# make
aking all in common
ake[1]: Entering directory '/opt/tomcat-connectors-1.2.32-src/native/common'
opt/apache-2.2.19/build/libtool --silent --mode=compile gcc -I/opt/apache-2.2.19/include  -DHAVE_CONFIG_H -g -O2 -pthrea
clude -I/opt/apache-2.2.19/include  -DHAVE_CONFIG_H -DLINUX=2 -D_REENTRANT -D_GNU_SOURCE -D_LARGEFILE64_SOURCE -I /opt/j
1/include/ -c jk_ajp12_worker.c -o jk_ajp12_worker.lo
opt/apache-2.2.19/build/libtool --silent --mode=compile gcc -I/opt/apache-2.2.19/include  -DHAVE_CONFIG_H -g -O2 -pthrea
clude -I/opt/apache-2.2.19/include  -DHAVE_CONFIG_H -DLINUX=2 -D_REENTRANT -D_GNU_SOURCE -D_LARGEFILE64_SOURCE -I /opt/j
1/include/ -c jk_connect.c -o jk_connect.lo
opt/apache-2.2.19/build/libtool --silent --mode=compile gcc -I/opt/apache-2.2.19/include  -DHAVE_CONFIG_H -g -O2 -pthrea
clude -I/opt/apache-2.2.19/include  -DHAVE_CONFIG_H -DLINUX=2 -D_REENTRANT -D_GNU_SOURCE -D_LARGEFILE64_SOURCE -I /opt/j
1/include/ -c jk_msg_buff.c -o jk_msg_buff.lo
opt/apache-2.2.19/build/libtool --silent --mode=compile gcc -I/opt/apache-2.2.19/include  -DHAVE_CONFIG_H -g -O2 -pthrea
clude -I/opt/apache-2.2.19/include  -DHAVE_CONFIG_H -DLINUX=2 -D_REENTRANT -D_GNU_SOURCE -D_LARGEFILE64_SOURCE -I /opt/j
1/include/ -c jk_util.c -o jk_util.lo
opt/apache-2.2.19/build/libtool --silent --mode=compile gcc -I/opt/apache-2.2.19/include  -DHAVE_CONFIG_H -g -O2 -pthrea
clude -I/opt/apache-2.2.19/include  -DHAVE_CONFIG_H -DLINUX=2 -D_REENTRANT -D_GNU_SOURCE -D_LARGEFILE64_SOURCE -I /opt/j
1/include/ -c jk_ajp13.c -o jk_ajp13.lo
opt/apache-2.2.19/build/libtool --silent --mode=compile gcc -I/opt/apache-2.2.19/include  -DHAVE_CONFIG_H -g -O2 -pthrea
clude -I/opt/apache-2.2.19/include  -DHAVE_CONFIG_H -DLINUX=2 -D_REENTRANT -D_GNU_SOURCE -D_LARGEFILE64_SOURCE -I /opt/j
1/include/ -c jk_pool.c -o jk_pool.lo
opt/apache-2.2.19/build/libtool --silent --mode=compile gcc -I/opt/apache-2.2.19/include  -DHAVE_CONFIG_H -g -O2 -pthrea
clude -I/opt/apache-2.2.19/include  -DHAVE_CONFIG_H -DLINUX=2 -D_REENTRANT -D_GNU_SOURCE -D_LARGEFILE64_SOURCE -I /opt/j
1/include/ -c jk_worker.c -o jk_worker.lo
opt/apache-2.2.19/build/libtool --silent --mode=compile gcc -I/opt/apache-2.2.19/include  -DHAVE_CONFIG_H -g -O2 -pthrea
clude -I/opt/apache-2.2.19/include  -DHAVE_CONFIG_H -DLINUX=2 -D_REENTRANT -D_GNU_SOURCE -D_LARGEFILE64_SOURCE -I /opt/j
1/include/ -c jk_ajp13_worker.c -o jk_ajp13_worker.lo
opt/apache-2.2.19/build/libtool --silent --mode=compile gcc -I/opt/apache-2.2.19/include  -DHAVE_CONFIG_H -g -O2 -pthrea
clude -I/opt/apache-2.2.19/include  -DHAVE_CONFIG_H -DLINUX=2 -D_REENTRANT -D_GNU_SOURCE -D_LARGEFILE64_SOURCE -I /opt/j
1/include/ -c jk_lb_worker.c -o jk_lb_worker.lo
```

-   -

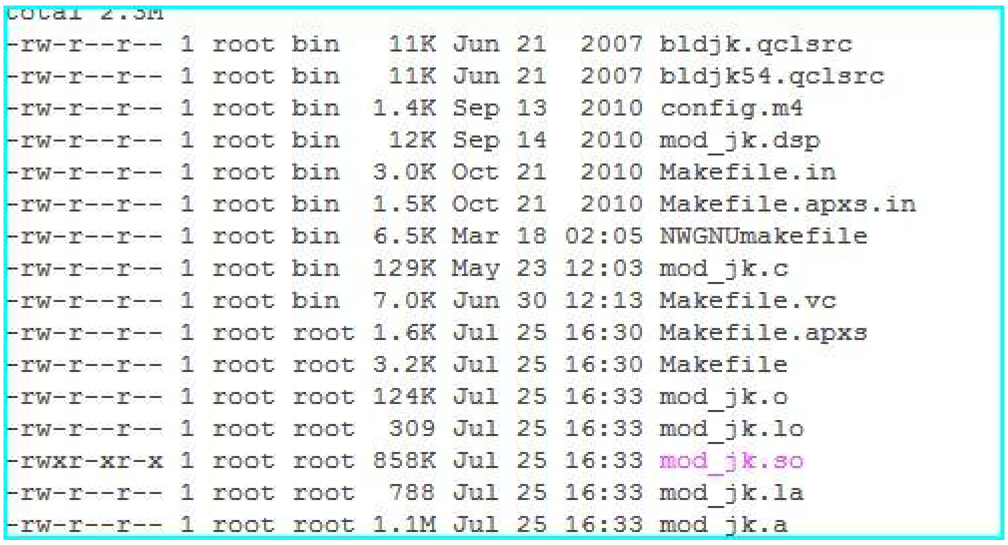
6. After the code is compiled using the make command then installation of the code is done using the command make install:

```
root@localhost apache-2.0]# make install
```



```
make[1]: Leaving directory `/opt/tomcat-connectors-1.2.32-src/native'
target="all"; \
        list='common apache-2.0'; \
        for i in $list; do \
            echo "Making $target in $i"; \
            if test "$i" != "."; then \
               (cd $i && make $target) || exit 1; \
            fi; \
        done;
Making all in common
make[1]: Entering directory `/opt/tomcat-connectors-1.2.32-src/native/common'
make[1]: Nothing to be done for `all'.
make[1]: Leaving directory `/opt/tomcat-connectors-1.2.32-src/native/common'
Making all in apache-2.0
make[1]: Entering directory `/opt/tomcat-connectors-1.2.32-src/native/apache-2.0'
make[1]: Nothing to be done for `all'.
make[1]: Leaving directory `/opt/tomcat-connectors-1.2.32-src/native/apache-2.0'
```

7. Once the execution is complete, it will create the module in the apache-2.0 directory of the source, as shown in the following screenshot:



```
total 2.3M
-rw-r--r-- 1 root bin   11K Jun 21  2007 bldjk.qclsrc
-rw-r--r-- 1 root bin   11K Jun 21  2007 bldjk54.qclsrc
-rw-r--r-- 1 root bin  1.4K Sep 13  2010 config.m4
-rw-r--r-- 1 root bin   12K Sep 14  2010 mod_jk.dsp
-rw-r--r-- 1 root bin  3.0K Oct 21  2010 Makefile.in
-rw-r--r-- 1 root bin  1.5K Oct 21  2010 Makefile.apxs.in
-rw-r--r-- 1 root bin  6.5K Mar 18 02:05 NWGNUmakefile
-rw-r--r-- 1 root bin  129K May 23 12:03 mod_jk.c
-rw-r--r-- 1 root bin  7.0K Jun 30 12:13 Makefile.vc
-rw-r--r-- 1 root root 1.6K Jul 25 16:30 Makefile.apxs
-rw-r--r-- 1 root root 3.2K Jul 25 16:30 Makefile
-rw-r--r-- 1 root root 124K Jul 25 16:33 mod_jk.o
-rw-r--r-- 1 root root  309 Jul 25 16:33 mod_jk.lo
-rwxr-xr-x 1 root root 858K Jul 25 16:33 mod_jk.so
-rw-r--r-- 1 root root  788 Jul 25 16:33 mod_jk.la
-rw-r--r-- 1 root root 1.1M Jul 25 16:33 mod_jk.a
```

Configuration of mod_jk in Apache

Configuration of `mod_jk` is a little complicated in Apache. There are various ways of performing the configuration, but the most commonly used option is the concept of creation of `workers.properties` and `mod_jk.conf`. The steps to be performed are mentioned below:

1. Copy the mod_jk.so from the apache 2.0 directory of the connector source to the modules directory of the Apache httpd server by using the following command:

   ```
   [root@localhost apache-2.0]# cp mod_jk.so /opt/apache-2.2.19/modules/
   chmod 755 mod_jk.so
   ```

- The previous command sets the execution permission.

```
    ```
    chown root:root mod_jk.so
    ```
```

- The previous command sets the the ownership to root.

2. To edit the configuration of the httpd server, you have to create the new file called as mod_jk.conf in the conf directory of $APACHE_HOME/conf as follows:

   ```
   [root@localhost apache-2.0]# cd /opt/apache-2.2.19/conf
   vi mod-jk.conf
   LoadModule jk_module modules/mod_jk.so
   JkWorkersFile conf/workers.properties
   JkLogFile logs/mod_jk.log
   JkLogLevel info
   JkMount /sample/* node1
   ```

```
The `mod_jk.conf` file contains the following details:
```

- **Module path:** It defines the location of the

  ```
  module from where Apache loads the module during the startup
  process, for
  example,` LoadModule jk_module modules/mod_jk.so`.
  ```

- **Worker file path:** It defines the location of the

  ```
  worker file, this file contains the information of the Tomcat
  instance details such as the IP, port and load balancing methods
  such as` JkWorkersFile conf/workers.properties.`
  ```

- **Log file:** It records the activity for Apache

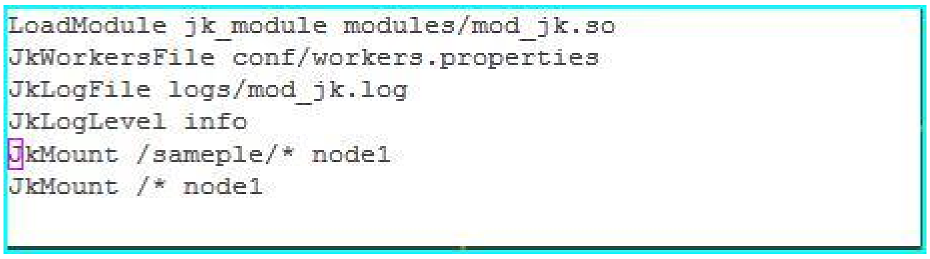
  ```
  Tomcat integration, it also records the connectivity health
  check run between Apache/Tomcat
  (`JkLogFile logs/mod_jk.log`).
  ```

- **URL mapping:** It defines the context path for

  ```
  Apache and also sets the rules such as redirecting the request
  if you get any request with the defined URL, for
  example,` JkMount /sample/* node1`. This means
  whenever the user hit the
  URL` http://localhost/sample`, the request will
  redirect to the Tomcat node1.
  ```

- **Log level:** This parameter captures the different

  ```
  events performed by` mod_jk` in the logs
  (`JkLogLevel info`).
  ```



```
LoadModule jk_module modules/mod_jk.so
JkWorkersFile conf/workers.properties
JkLogFile logs/mod_jk.log
JkLogLevel info
JkMount /sameple/* node1
JkMount /* node1
```

3. Create a new file named as workers. properties in the conf using the following command:

   ```
   [root@localhost conf]# vi workers.properties
   worker.list=node1
   worker.node1.port=8009
   worker.node1.host=10.130.240.51
   worker.node1.type=ajp13
   worker.node1.lbfactor=1
   ```

```
`workers.properties` contain the following details:
```

- Node name (common name for the host)
- AJP port details for Tomcat (the port on which Tomcat accepts

  ```
    the request for AJP)
  ```

- Host IP for Tomcat (the IP address where the Tomcat instance is

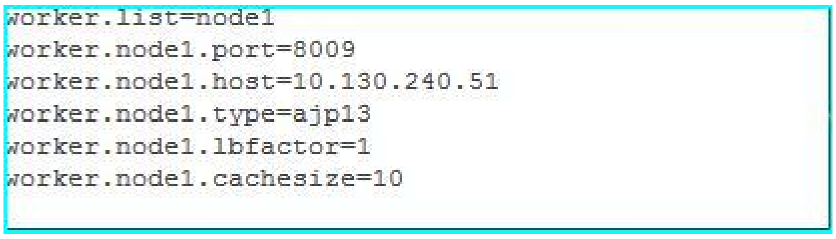
  ```
    running)
  ```

- Protocol used (the protocol used for communication by default is

  ```
    AJP)
  ```

- Load balancing method (Round robin, persistence, and so

  ```
    on)
  ```



```
worker.list=node1
worker.node1.port=8009
worker.node1.host=10.130.240.51
worker.node1.type=ajp13
worker.node1.lbfactor=1
worker.node1.cachesize=10
```

4. The last step is to include the mod_jk.conf in the main configuration file of the httpd, that is, httpd.conf.

   ```
   [root@localhost conf]# vi httpd.conf
   ```

Note

```
Include` conf/mod_jk.conf` should be added at the end
of` httpd.conf`.
```

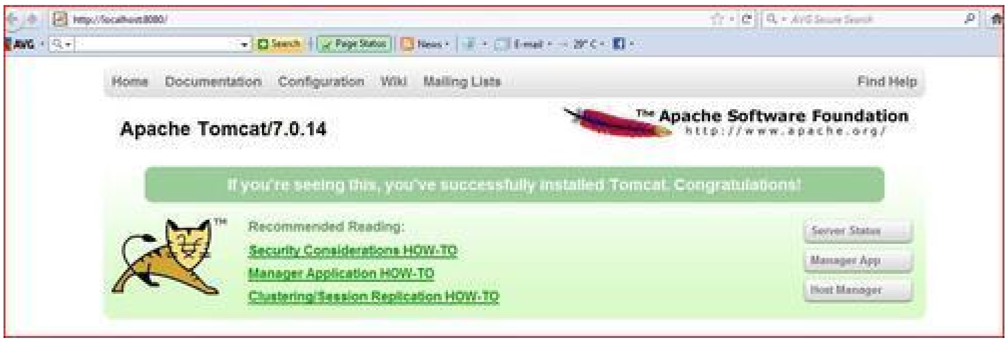
Now we are done with configuration of `mod_jk` in the Apache HTTP configuration file (`httpd.conf`). But `mod_jk` will not work until we recycle the Apache httpd services. So why wait? Let's recycle by running the following command:

```
[root@localhost bin]# ./apachectl stop
[root@root@localhost bin]# ./apachectl start
```

Note

In case the Apache services are not displayed after the configuration, then we will run the `configtest.sh` placed in the `bin` directory that shows the issues with configuration.

Once we are done with the Apache web server configurations, followed by the web server service restart, it's now time to test the application. In Lab 1, we had tested the application by using the host and port number on which Tomcat services were running `http://localhost:8080/applicationname` , as shown in the following screenshot:



http://localhost:8080/

Home Documentation Configuration Wiki Mailing Lists Find Help

Apache Tomcat/7.0.14

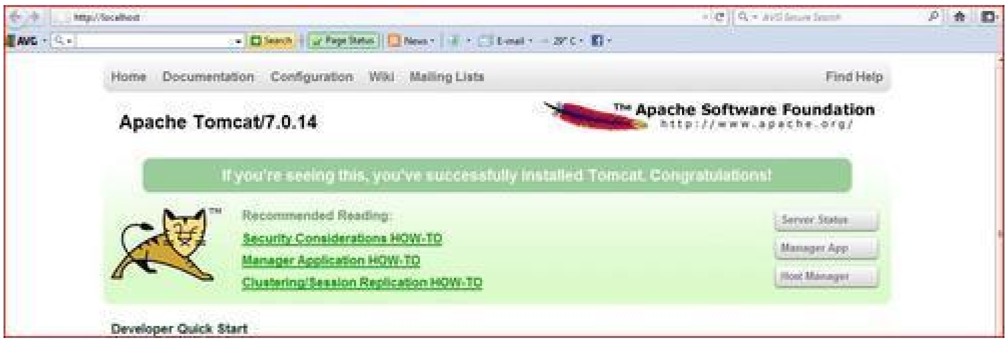
The Apache Software Foundation http://www.apache.org/

If you're seeing this, you've successfully installed Tomcat. Congratulations!

Recommended Reading:
Security Considerations HOW-TO
Manager Application HOW-TO
Clustering/Session Replication HOW-TO

Server Status
Manager App
Host Manager

After enabling the `mod_jk` configuration, you can check the URL without using the port number (`http://localhost/applicationname`). The following screenshot shows the application with the application's URL:



http://localhost

Home Documentation Configuration Wiki Mailing Lists Find Help

Apache Tomcat/7.0.14

The Apache Software Foundation http://www.apache.org/

If you're seeing this, you've successfully installed Tomcat. Congratulations!

Recommended Reading:
Security Considerations HOW-TO
Manager Application HOW-TO
Clustering/Session Replication HOW-TO

Server Status
Manager App
Host Manager

Developer Quick Start

mod_proxy configuration

`mod_proxy` configuration is very simple as compared to `mod_jk` configuration. Here, we need to add the module and redirect the URL to a virtual host.

Open the `httpd.conf` and place the following entry:

1. Place the following lines of code after the other LoadModule directives:

```
LoadModule proxy_module modules/mod_proxy.so
LoadModule proxy_http_module modules/mod_proxy_http.so
```

2. Place the following lines of code with your other VirtualHost, or at the bottom of the file:

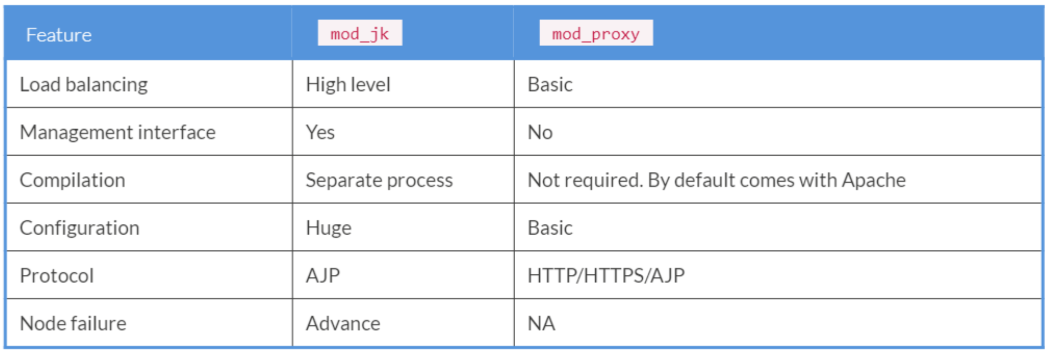
```
NameVirtualHost *
<VirtualHost *>
ServerName abc.com
ProxyRequests Off
<Proxy *>
Order deny,allow
Allow from all
</Proxy>
ProxyPass / http://localhost:8080/
ProxyPassReverse / http://localhost:8080/
<Location />
Order allow,deny
Allow from all
</Location>
</VirtualHost>
```

Save the configuration file. Based on the Rule of Thumb, every configuration change is reflected only after a recycle.

```
[root@localhost bin]# ./apachectl stop
[root@root@localhost bin]# ./apachectl start
```

Comparison between mod_jk and mod_proxy

We have discussed `mod_jk` and `mod_proxy` but we still don't know when to use which module to increase the speed of the web server. Let's compare both modules and find out which can be used in a real-time environment:



Feature	`mod_jk`	`mod_proxy`
Load balancing	High level	Basic
Management interface	Yes	No
Compilation	Separate process	Not required. By default comes with Apache
Configuration	Huge	Basic
Protocol	AJP	HTTP/HTTPS/AJP
Node failure	Advance	NA

The previous table shows the comparison of `mod_jk` and `mod_proxy`. Based on the features, the web administrator can decide which module should be used.

Note

In 90 percent of cases, `mod_jk is used with Apache Tomcat`

Common issues and troubleshooting for integration

There are many issues which may arise during the integration of Apache Tomcat or IIS Tomcat integration. Some of them are mentioned in the following section and we will find out the reason for these issues and their solutions These issues are very common with the integration of Tomcat.

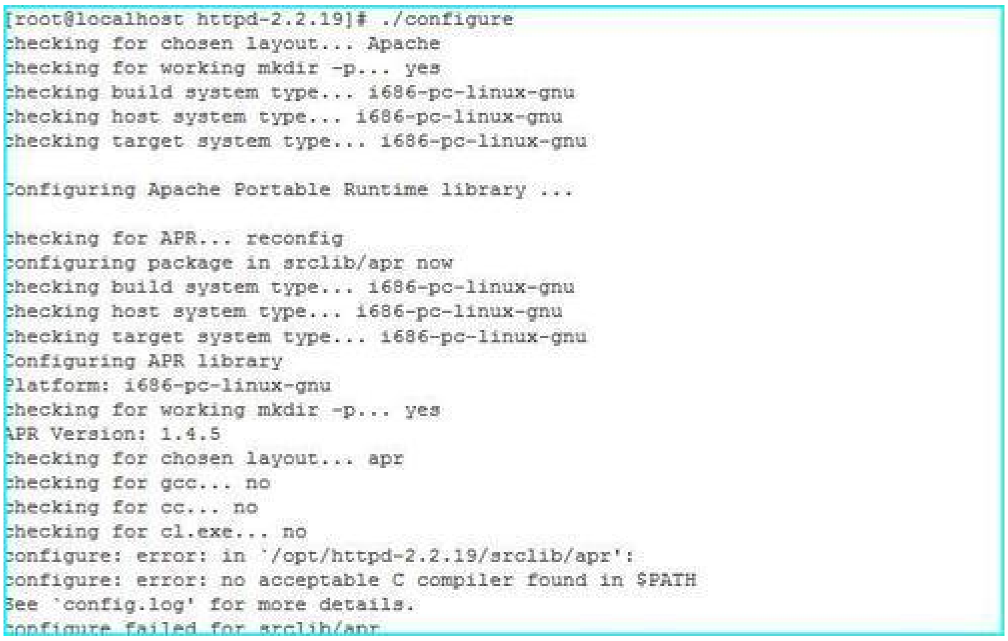
Scenario 1: The httpd server is not able to compile, and this results in the exit from the compilation mode.

Error:

```
configure: error: in `/opt/httpd-2.2.19/srclib/apr':
configure: error: no acceptable C compiler found in $PATH
See `config.log' for more details.
configure failed for srclib/apr
```

Reason: C compilers are missing, such as the GCC and GCC+.

Solution: Download the GCC compiler from the Internet and compile it, as per the instructions given there:



```
[root@localhost httpd-2.2.19]# ./configure
checking for chosen layout... Apache
checking for working mkdir -p... yes
checking build system type... i686-pc-linux-gnu
checking host system type... i686-pc-linux-gnu
checking target system type... i686-pc-linux-gnu

Configuring Apache Portable Runtime library ...

checking for APR... reconfig
configuring package in srclib/apr now
checking build system type... i686-pc-linux-gnu
checking host system type... i686-pc-linux-gnu
checking target system type... i686-pc-linux-gnu
Configuring APR library
Platform: i686-pc-linux-gnu
checking for working mkdir -p... yes
APR Version: 1.4.5
checking for chosen layout... apr
checking for gcc... no
checking for cc... no
checking for cl.exe... no
configure: error: in `/opt/httpd-2.2.19/srclib/apr':
configure: error: no acceptable C compiler found in $PATH
See `config.log' for more details.
configure failed for srclib/apr
```

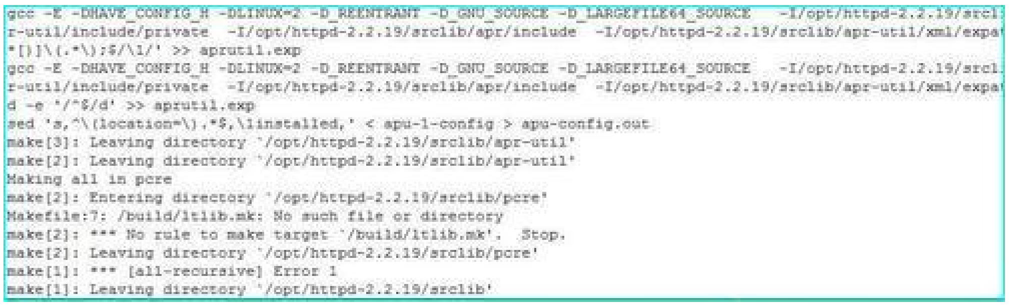
Scenario 2: Apache is not able to compile the `make` command, displaying an error and exiting the process.

Error: `make` is not able to compile the code.

Reason: `make` is not able to execute its functions.

Solution: Run the following command:

```
make clean
#Then
make
make install
```



```
gcc -E -DHAVE_CONFIG_H -DLINUX=2 -D_REENTRANT -D_GNU_SOURCE -D_LARGEFILE64_SOURCE   -I/opt/httpd-2.2.19/srcl
r-util/include/private  -I/opt/httpd-2.2.19/srclib/apr/include  -I/opt/httpd-2.2.19/srclib/apr-util/xml/expa
*[)]\(.*\);$/\1/' >> aprutil.exp
gcc -E -DHAVE_CONFIG_H -DLINUX=2 -D_REENTRANT -D_GNU_SOURCE -D_LARGEFILE64_SOURCE   -I/opt/httpd-2.2.19/srcl
r-util/include/private  -I/opt/httpd-2.2.19/srclib/apr/include  -I/opt/httpd-2.2.19/srclib/apr-util/xml/expa
d -e '/^$/d' >> aprutil.exp
sed 's,^\(location=\).*$,\1installed,' < apu-1-config > apu-config.out
make[3]: Leaving directory `/opt/httpd-2.2.19/srclib/apr-util'
make[2]: Leaving directory `/opt/httpd-2.2.19/srclib/apr-util'
Making all in pcre
make[2]: Entering directory `/opt/httpd-2.2.19/srclib/pcre'
Makefile:7: /build/ltlib.mk: No such file or directory
make[2]: *** No rule to make target `/build/ltlib.mk'.  Stop.
make[2]: Leaving directory `/opt/httpd-2.2.19/srclib/pcre'
make[1]: *** [all-recursive] Error 1
make[1]: Leaving directory `/opt/httpd-2.2.19/srclib'
```

Scenario 3: The Apache HTTP server is unable to connect to Tomcat 8.

Error: Unable to connect through AJP.

Reason: Port might get blocked or the AJP configuration is not correct.

Solution: Check the logs for more errors using the following command:

```
[root@localhost logs]# cat error_log
[Mon Jul 25 16:11:00 2011] [notice] Apache/2.2.19 (Unix) DAV/2 configured -- resuming
normal operations
[Mon Jul 25 16:52:16 2011] [notice] caught SIGTERM, shutting down
[Mon Jul 25 16:52:23 2011] [warn] No JkShmFile defined in httpd.conf. Using default
/opt/apache-2.2.19/logs/jk-runtime-status
[Mon Jul 25 16:52:23 2011] [warn] No JkShmFile defined in httpd.conf. Using default
/opt/apache-2.2.19/logs/jk-runtime-status
[Mon Jul 25 16:52:23 2011] [notice] Apache/2.2.19 (Unix) DAV/2 mod_jk/1.2.32
configured -- resuming normal operations
[root@localhost logs]# cat mod_jk.log
[Mon Jul 25 16:52:23.555 2011] [13355:3086857920] [warn]
jk_map_validate_property::jk_map.c (411): The attribute 'worker.node1.cachesize' is
deprecated - please check the documentation for the correct replacement.
[Mon Jul 25 16:52:23.555 2011] [13355:3086857920] [info] init_jk::mod_jk.c (3252):
mod_jk/1.2.32 () initialized
[Mon Jul 25 16:52:23.564 2011] [13356:3086857920] [warn]
jk_map_validate_property::jk_map.c (411): The attribute 'worker.node1.cachesize' is
deprecated - please check the documentation for the correct replacement.
[Mon Jul 25 16:52:23.564 2011] [13356:3086857920] [info] init_jk::mod_jk.c (3252):
mod_jk/1.2.32 () initialized
```

Then, run the `configtest` command on the server to verify the configuration using the following command:

```
[root@root@localhost bin]# ./apachectl configtest
Syntax OK
```

Summary

In this lab, we have discussed the integration of Apache with Tomcat 8 and their various component integrations.

In the next lab, we will discuss the security enhancement of Tomcat and their features such as application with their own security setting, server security SSL, and so on.