**Virtual Host + Apache httpd server + Tomcat + mod\_jk connector**

n my last post ([Virtual Host in Tomcat](http://www.ramkitech.com/2012/02/understanding-virtual-host-concept-in.html)) we discussed about how setup the virtual host in Tomcat. Its cost effective technique because only one public IP is enough to host multiple domain. If we have big organization and each department want to host their website in locally in different machine. then how to achieve the virtual host concept?. In this post we will see the how we do this.

### Update :  I posted [Virtual Host + Nginx + Tomcat](http://www.ramkitech.com/2013/02/virtual-host-nginx-tomcat.html) Its easy to configure, compare to Apache httpd server

**Problem Scenario:**  
        In big organization they have multiple department, each department want to host their website in different machine. so these websites are accessed locally with different local IP address.

|  |
| --- |
| [http://1.bp.blogspot.com/-TLyyrW-Wk6Y/T2gOrugVt0I/AAAAAAAAB_k/7kyCrNCHl3s/s640/outline_structure.jpg](http://1.bp.blogspot.com/-TLyyrW-Wk6Y/T2gOrugVt0I/AAAAAAAAB_k/7kyCrNCHl3s/s1600/outline_structure.jpg) |
|  |

When we mapping to public address then we face the problem. We have two choice either purchase as many public address or Put one server front  and delegate these request.  
  
 We going to use 2nd option. we put Apache httpd web server in front of all department servers. so only one public IP is enough. All domain DNS entries are pointed to Apache httpd server. Then Apache server delegates these request to corresponding tomcat server. This process is completely transparent from users(browser) perspective.

|  |
| --- |
| [http://4.bp.blogspot.com/-wWtKl_9eNvA/T2gOtW0LCVI/AAAAAAAAB_0/weXELW0y8ag/s640/virtual_host_apache_and_tomcat_outline.jpg](http://4.bp.blogspot.com/-wWtKl_9eNvA/T2gOtW0LCVI/AAAAAAAAB_0/weXELW0y8ag/s1600/virtual_host_apache_and_tomcat_outline.jpg) |
| Outline Structure of Virtual Host Implementation |

**How Apache httpd web server communicate to Tomcat server**  
               Before we going to detail about how communication happen between httpd server and tomcat  
  
How many ports are bind when we start single tomcat?

* shutdown port
* http connector port
* https connector port (optional)
* ajp port

The port configuration are stored in $CATALINA\_HOME/conf/server.xml file. we can change the ports when its necessary.  
  
here AJP**(Apache JServ Protocol)**  is a binary protocol that can proxy inbound requests from a web server through to an application server that sits behind the web server.  
  
Apache httpd webserver communicate to Tomcat sever through **AJP** protocol.  
  
When we install Apache httpd server, It don't have inbuilt capability to support ajp protocol. so we need **mod\_jk** module. Its add the ajp support to Apache httpd server.

**Steps to Implement Virtual Host Concept in this Scenario:**

1. Install Apache httpd Web Server
2. Install mod\_jk connector
3. Configure JK Connector
4. Configure Apache httpd server apply virtual host concepts

**Prerequisite** : We already installed Tomcat in different departments and deployed the application and works fine.  
  
**Install Apache httpd web server**  
We can install Apache web server in two ways.  
                         - Binary module  
                         - From Source  
  
We can install Apache httpd server from distribution package manager (either apt-get or yum). Or we can download the source code and then compile and install.  
  
**Note : -**  
  
 we need to install apr, apr-util and pcre libraries because Apache httpd server depends on these libs.  
But these libraries are no need to install from source. because we not going to do any customization.  
so install these lib as binary using apt-get (debian/ubuntu) or yum command (redhat/fedora).  
  
in ubuntu u can search through  
  
sudo apt-cache search apr  
  
  
  
we use second option. First download the httpd server source code from [here](http://httpd.apache.org/) . then extract it and install  
  
**./configure --prefix=/usr/local/apache    --enable-rewrite=shared  --enable-proxy=shared**  
**make**  
**sudo make install**

here

**--prefix** option to mention where the location we going to install Apache httpd server.

**--enable-rewrite** and **--enable-proxy**options to enable these module in shared mode. These modules are not needed now. but we used in future for rewrite the URL before handover to next chain of servers and load-balancing support.

**Install mod\_jk connector**

 Now Apache httpd server is ready. we need to add ajp support to server.

download the mod\_jk connector module from [here](http://tomcat.apache.org/download-connectors.cgi). extract it and install it

**cd native**

**./configure    --with-apxs=/usr/local/apache/bin/apxs**

**make**

**sudo make install**

here **--with-apxs** option to specify where apxs module is located. so we need to give Apache httpd server location.

now mod\_jk.so files is created on modules directory in apache installed location (/usr/local/apache/modules)

**Configure mod\_jk connector**

This step have 2 sub step

* Create workers.properties file
* Load and configure the JK connector module in apache httpd.conf file

**Create workers.properties file**

mod\_jk connector is ready. but this connector is works based on configuration file. so we need to create configuration file called**workers.properties** file

this file syntax is key=value pair,

here we define the workers. i.e all department tomcat hosts IP address and ajp port for corressponding tomcat.

here entry format is look like

**worker.<name>.property=<value>**

for example

worker.<name>.type=ajp13

worker.<name>.port=<ajp port>

worker.<name>.host=<tomcat ip addr>

worker.list=<name>

here

worker.list key have all workers name separated by comma.

type = here type of the worker. we use ajp13 version

port= we specify the ajp port **(not http port )** of that server

host= IP address or host name of tomcat server

**workers.properties**

**worker.list=department1,department2,department3**

**worker.department1.type=ajp13**

**worker.department1.port=5000**

**worker.department1.host=192.168.5.10**

**worker.department2.type=ajp13**

**worker.department2.port=5000**

**worker.department2.host=192.168.6.10**

**worker.department3.type=ajp13**

**worker.department3.port=5000**

**worker.department3.host=192.168.7.10**

**Add Entry in httpd.conf**

Apache httpd server is installed. mod\_jk module is installed and workers.properties file is created. but these 3 are isolated. we put together,  
we need to configure the httpd server.

find the**conf/httpd.conf** file in Apache installed location and add these following entries to it

**LoadModule    jk\_module  modules/mod\_jk.so**

**JkWorkersFile conf/workers.properties**

**JkLogFile     logs/mod\_jk.log**

**JkLogLevel    emerg**

**JkLogStampFormat "[%a %b %d %H:%M:%S %Y] "**

**JkOptions     +ForwardKeySize +ForwardURICompat -ForwardDirectories**

**JkRequestLogFormat     "%w %V %T"**

here

LoadModule    - Load mod\_jk shared module to Apache httpd server (enable the mod\_jk module)

JkWorkersFile - Specify the workers.properties file location

all others are logging system of mod\_jk. Its boilerplate code just copy and paste.

**Delegate httpd to Tomcat**

now we inform to Apache httpd server how delegate the request to corresponding server.

**JkMount  /department1\* department1**

**JkMount  /department2\* department2**

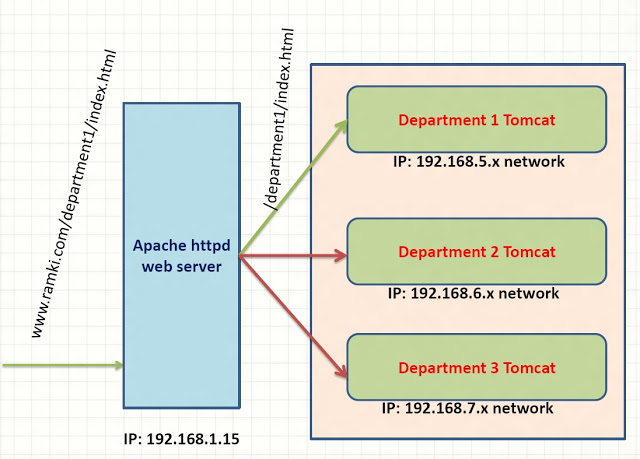
**JkMount  /department3\* department3**

here

JkMount - specify the if URL have /department1\* pattern then that request delegate to department1 worker. that worker IP address and port is specified in workers.properties file.

Iwe changed /etc/hosts file like [**last post**](http://www.ramkitech.com/2012/02/understanding-virtual-host-concept-in.html) all web site domain pointed to apache httpd web server.

if we access **http://www.ramki.com/department1/index.html** how Apache httpd server process the request

[](http://1.bp.blogspot.com/-Mjl4sHW_eks/T2gOurUxLbI/AAAAAAAAB_8/hhAqLLS9P3o/s1600/virtual_host_apache_and_tomcat_processing.jpg)

Its perfectly called correct tomcat server and we got right response.

but we have 2 problem

**1.** i don't want the my URL like this   **http://www.ramki.com/department1/index.html**

              i want the my URL like this   **http://www.ramki.com/index.html**

remove the department1 from my URL. but department1 string is very important in URL because JkMount is works based on this matching string only.

2. **http://www.ramki.com/department1/index.html**  is for first tomcat and second department have domain   http://www.krishnan.com then

access second tomcat we use URL : **http://www.krishnan.com/department2/index.html**

but same time when we use URL **http://www.krishnan.com/department1/index.html**then we access first tomcat data

(i.e )  
**http://www.ramki.com/department1/index.html  == http://www.krishnan.com/department1/index.html**

because both these URL have department1 key word.. so JkMount is works based on these keyword. As the Result wrong interpretation. How to solve this Issue?.

**Virtual Host in Apache httpd Server**

we need add conditional JkMount. for example ramki.com domain asks the whereare paths like department1,department2 we need to search in that tomcat only. not other place. to add this conditioned we add virtual host entries.

Add virtual host entry in httpd.conf file

**Listen 80**

**NameVirtualHost \*:80**

**<VirtualHost \*:80>**

**ServerName www.ramki.com**

**JkMount  /department1\* department1**

**</VirtualHost>**

**<VirtualHost \*:80>**

**ServerName www.krishnan.com**

**JkMount  /department2\* department1**

**</VirtualHost>**

here

ServerName - domain name of the server

If  **http://www.krishnan.com/department1/index.html**URL is like this now. server matches the Server Name. here its matches 2nd virtual host entry. There are single JkMount entry is there in 2nd Virtual-Host. and there no matching department1 string.

**JkMount  /department2\* department1**

as the result 404 error page is responded. Its works good.

Now everythig works fine.

now my URL is  **http://www.ramki.com/department1/index.html**here i don't want department1 path in my URL.

i want simply     **http://www.ramki.com/index.html**then we use **mod\_rewrite**engine

**Listen 80**

**NameVirtualHost \*:80**

**<VirtualHost \*:80>**

**ServerName www.ramki.com**

**RewriteEngine on**

**RewriteLog logs/apache-mod\_rewrite**

**RewriteRule ^/(.\*)$ /department1/$1 [L,PT]**

**JkMount  /\* <dept\_name>**

**</VirtualHost>**

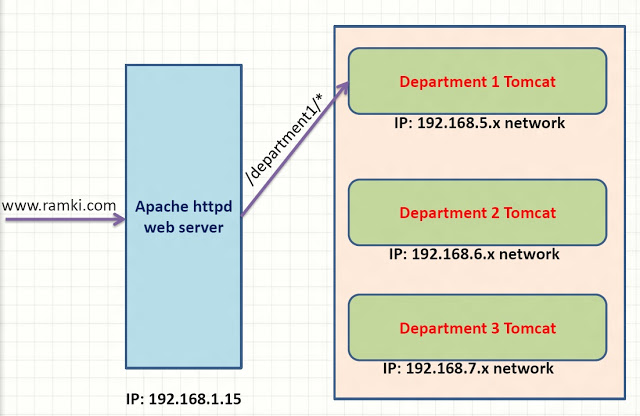
here

**RewriteEngine on** - Turn on the Rewrite module

**RewriteRule ^/(.\*)$ /department1/$1 [L,PT]**  - here **^/(.\*)$**its matches any string it capture the value to $1 and change the URL to /department1/$1

i.e

**http://www.ramki.com/index.html** here  **^/(.\*)$**its matches  index.html it capture to $1 and replaced to /department1/$1 ==> /department1/index.html

[](http://1.bp.blogspot.com/-NPbEsb2bFpQ/T2gOsqGcbMI/AAAAAAAAB_o/H-Dq1pQHrQs/s1600/virtual_host_apache_and_tomcat_final_processing.jpg)

Rewrite is happen before delegate the request to Tomcat. so we change the URL transparently to browser.

now client (browser) just send http://www.ramki.com is enough to access the Department 1 Tomcat.