Why People Use Proxy Server and How to Use Proxy Server

I. Why People Use Proxy Server

A proxy server is a computer that acts as an intermediary between the user's computer and the Internet. It allows client computers to make indirect network connections to other network services. If use proxy server, client computers will first connect to the proxy server, requesting some resources like web pages, games, videos, mp3, e-books, any other resources which are available from various servers over Internet. As soon as getting such request, the proxy server will seek for the resources from the cache in its local hard disk. If the resources have been cached before, the proxy server will return them to the client computers. If not cached, it will connect to the relevant servers and request the resources on behalf of the client computers. Then it 'caches' resources from the remote servers, and returns subsequent requests for the same content directly.

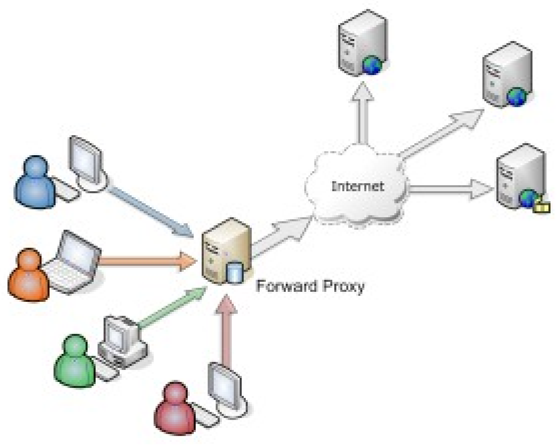
Nowadays, we use proxy server for various purpose like sharing Internet connections on a local area network, hide our IP address, implement Internet access control, access blocked websites and so on. Bellow are some benefits why people use proxy server:

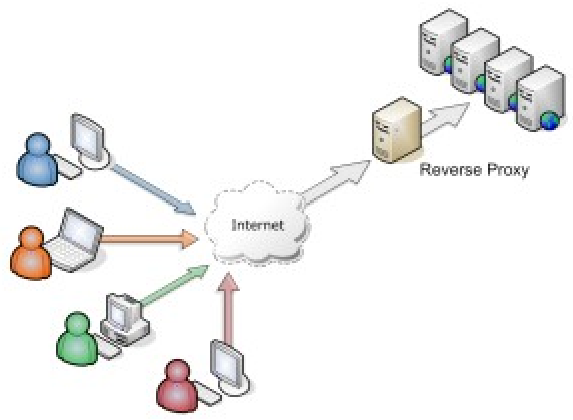
- To share Internet connection on a LAN. Some small businesses and families have multiple computers but with only one Internet connection, they can share Internet connection for other computers on the LAN with a proxy server.
- To speed up Internet surfing. If use proxy server, all requests from client computers will reach the proxy server at first, if the proxy server has cached the required resources in its local hard disk before with the web cache function, clients will get feedback directly from proxy server, it will be more quickly than direct accessing.
- To hide the IP address of the client computer so that it can surf anonymous, this is mostly for security reasons. A proxy server can act as an intermediary between the user's computer and the Internet to prevent from attack and unexpected access.
- To implement Internet access control like authentication for Internet connection, bandwidth control, online time control, Internet web filter and content filter etc.
- To bypass security restrictions and filters. For example, many work offices have blocked facebook and myspace however, you can use proxy server to bypass such restrictions and access blocked websites easily.
- To scan outbound content, e.g., for data leak protection.
- To circumvent regional restrictions. For example, a server using IP-based geolocation to restrict its service to a certain country can be accessed using a proxy located in that country to access the service.

What's the difference between a reverse proxy and forward proxy?

A **forward proxy** is a proxy configured to handle requests for a group of clients under the local Administrators control to an unknown or arbitrary group of resources that are outside of their control. Usually the word “forward” is dropped and it is referred to simply as a proxy, this is the case in Microsoft’s topology. A good example is a web proxy appliance which accepts web traffic requests from client machines in the local network and proxies them to servers on the internet. The purpose of a forward proxy is to manage traffic to the client systems

A **reverse proxy** is a proxy configured to handle requests from a group of remote or arbitrary clients to a group of known resources under the control of the local Administrator. An example of this is a load balancer (a.k.a. application delivery controller) that provides application high availability and optimization to workloads like as Microsoft Skype, Exchange and SharePoint. The purpose of a reverse proxy is to manage the server systems.





About Apache Tomcat

Apache Tomcat is a web server and servlet container that is used to serve Java applications. Tomcat is an open source implementation of the Java Servlet and JavaServer Pages technologies, released by the Apache Software Foundation.

This tutorial covers the basic installation and some configuration of Tomcat 7 on your Ubuntu 14.04 server.

There are two basic ways to install Tomcat on Ubuntu:

- Install through apt-get. This is the simplest method.
- Download the binary distribution from the Apache Tomcat site. This guide does not cover this method; refer to Apache Tomcat Documentation for instructions.

Step One — Prerequisites

Before you begin with this guide, you should have a separate, non-root user account set up on your server. You can learn how to do this by completing steps 1-4 in the initial server setup for Ubuntu 14.04. We will be using the `demo` user created here for the rest of this tutorial.

Step Two - Install Tomcat

The first thing you will want to do is update your apt-get package lists:

```
sudo apt-get update
```

Now you are ready to install Tomcat. Run the following command to start the installation:

```
sudo apt-get install tomcat7
```

Answer `yes` at the prompt to install tomcat. This will install Tomcat and its dependencies, such as Java, and it will also create the `tomcat7` user. It also starts Tomcat with its default settings.

Let's make a quick change to the Java options that Tomcat uses when it starts. Open the Tomcat7 parameters file:

```
sudo nano /etc/default/tomcat7
```

Find the `JAVA_OPTS` line and replace it with the following. Feel free to change the `Xmx` and `MaxPermSize` values—these settings affect how much memory Tomcat will use:

/etc/default/tomcat7 — JAVA_OPTS

```
JAVA_OPTS="-Djava.security.egd=file:/dev/./urandom -Djava.awt.headless=true
-Xmx512m -XX:MaxPermSize=256m -XX:+UseConcMarkSweepGC"
```

Save and exit.

Now restart Tomcat with this command:

```
sudo service tomcat7 restart
```

Tomcat is not completely set up yet, but you can access the default splash page by going to your domain or IP address followed by `:8080` in a web browser:

```
Open in web browser:
http://server_IP_address:8080
```

You will see a splash page that says "It works!", in addition to other information. Now we will go deeper into the installation of Tomcat.

Step Three - Installing Additional Packages

Note: This section is not necessary if you are already familiar with Tomcat and you do not need to use the web management interface, documentation, or examples. If you are just getting into Tomcat for the first time, please continue.

With the following command, we will install the Tomcat online documentation, the web interface (manager webapp), and a few example webapps:

```
sudo apt-get install tomcat7-docs tomcat7-admin tomcat7-examples
```

Answer `yes` at the prompt to install these packages. We will get into the usage and configuration of these tools in a later section. Next, we will install the Java Development Kit.

Step Four - Install Java Development Kit (Optional)

If you are planning on developing apps on this server, you will want to be sure to install the software in this section.

(didnt consider this in this tutorial)

Step 5 - Configure Tomcat Web Management Interface

In order to use the manager webapp installed in Step 3, we must add a login to our Tomcat server. We will do this by editing the `tomcat-users.xml` file:

```
sudo nano /etc/tomcat7/tomcat-users.xml
```

This file is filled with comments which describe how to configure the file. You may want to delete all the comments between the following two lines, or you may leave them if you want to reference the examples:

tomcat-users.xml excerpt

```
<tomcat-users>
...
</tomcat-users>
```

You will want to add a user who can access the `manager-gui` and `admin-gui` (the management interface that we installed in Step Three). You can do so by defining a user similar to the example below. Be sure to change the username and password to something secure:

tomcat-users.xml — Admin User

```
<tomcat-users>
    <user username="admin" password="password" roles="manager-gui,admin-gui"/>
</tomcat-users>
```

Save and quit the tomcat-users.xml file. To put our changes into effect, restart the Tomcat service:

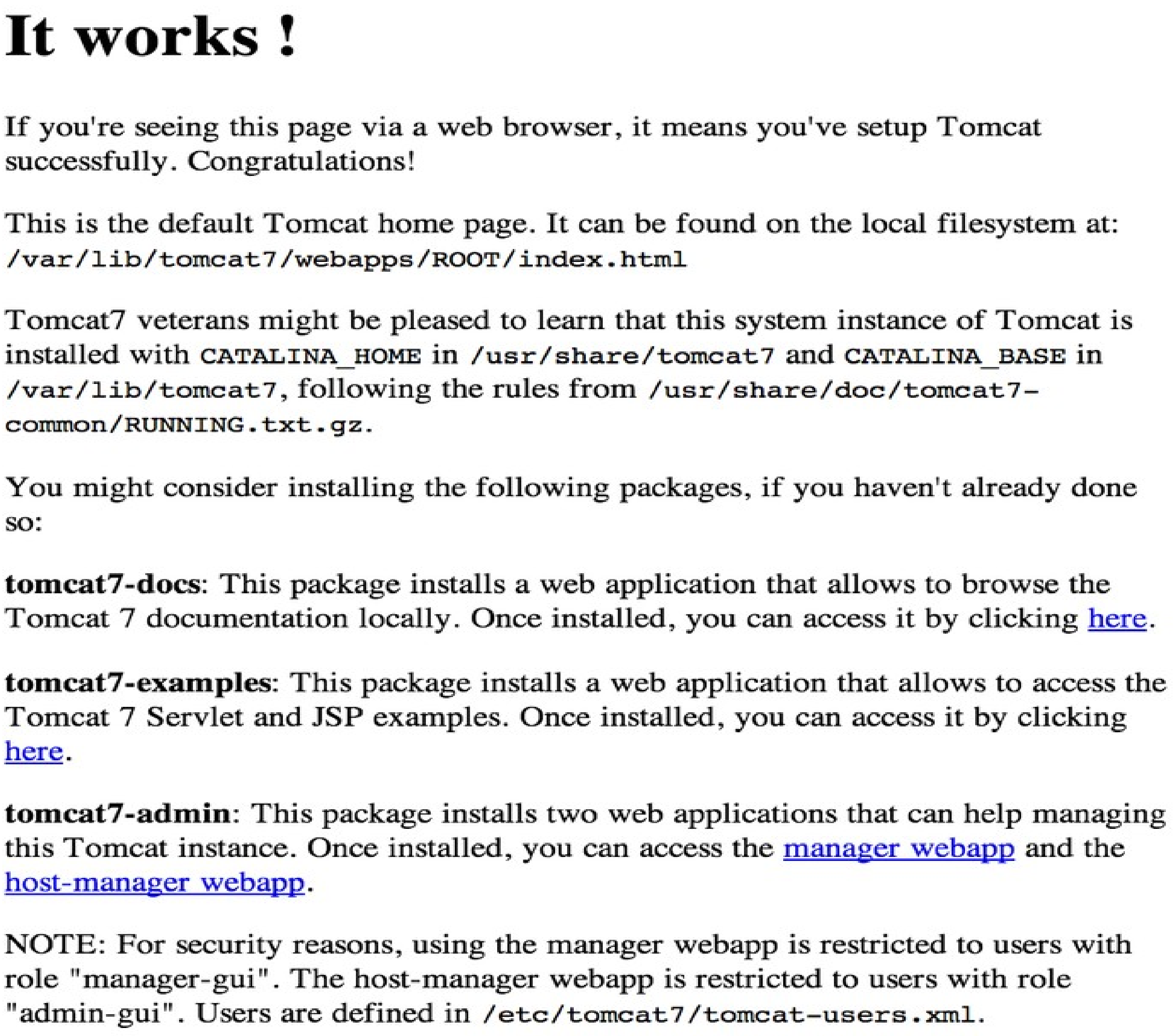
```
sudo service tomcat7 restart
```

Step 6 - Access the Web Interface

Now that we've configured an admin user, let's access the web management interface in a web browser:

```
Open in web browser:
http://server_IP_address:8080
```

You will see something like the following image:



It works !

If you're seeing this page via a web browser, it means you've setup Tomcat successfully. Congratulations!

This is the default Tomcat home page. It can be found on the local filesystem at: `/var/lib/tomcat7/webapps/ROOT/index.html`

Tomcat7 veterans might be pleased to learn that this system instance of Tomcat is installed with `CATALINA_HOME` in `/usr/share/tomcat7` and `CATALINA_BASE` in `/var/lib/tomcat7`, following the rules from `/usr/share/doc/tomcat7-common/RUNNING.txt.gz`.

You might consider installing the following packages, if you haven't already done so:

tomcat7-docs: This package installs a web application that allows to browse the Tomcat 7 documentation locally. Once installed, you can access it by clicking here.

tomcat7-examples: This package installs a web application that allows to access the Tomcat 7 Servlet and JSP examples. Once installed, you can access it by clicking here.

tomcat7-admin: This package installs two web applications that can help managing this Tomcat instance. Once installed, you can access the manager webapp and the host-manager webapp.

NOTE: For security reasons, using the manager webapp is restricted to users with role "manager-gui". The host-manager webapp is restricted to users with role "admin-gui". Users are defined in `/etc/tomcat7/tomcat-users.xml`.

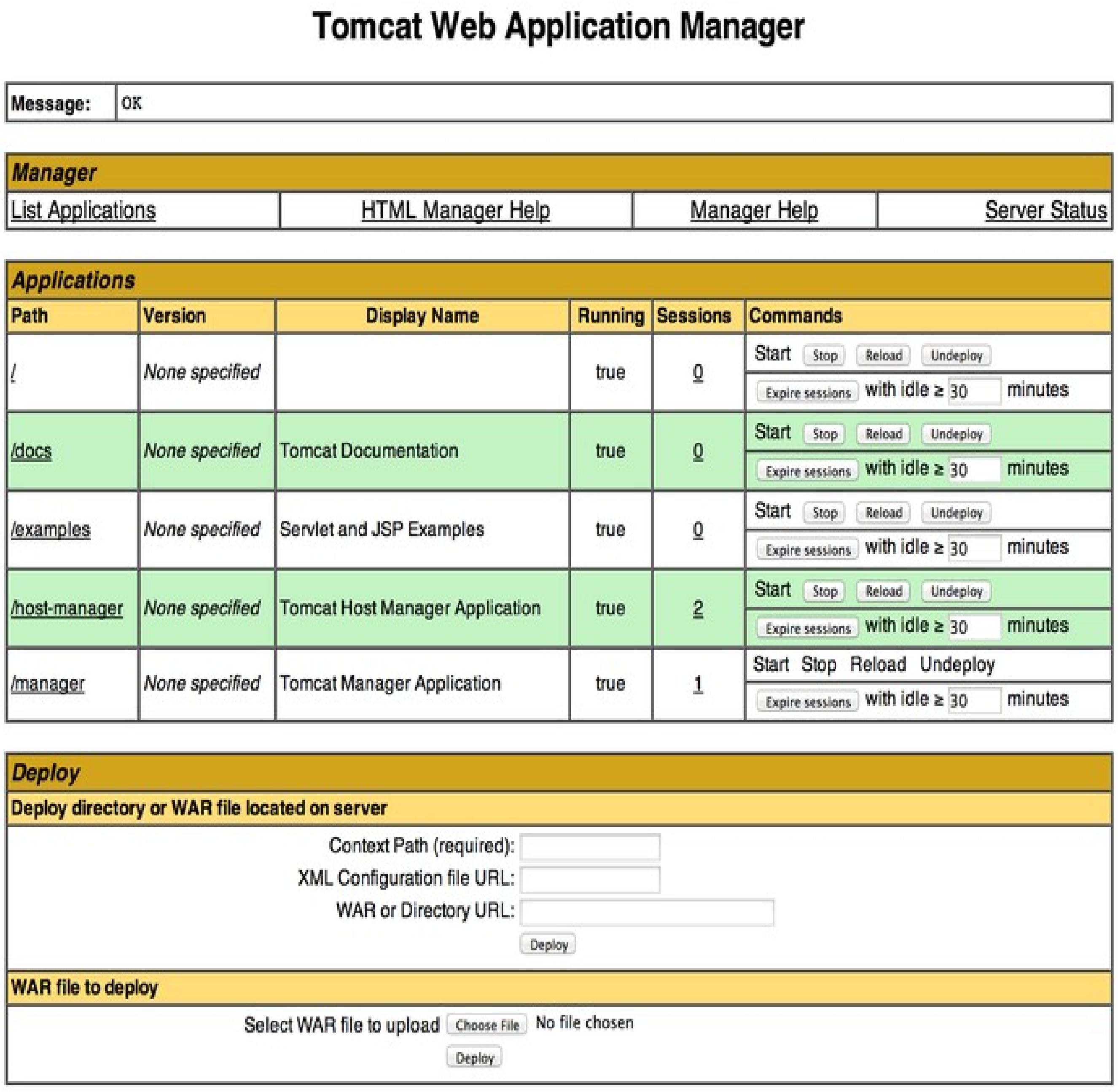
As you can see, there are four links to packages you installed in Step Three:

- tomcat7-docs: Online documentation for Tomcat. Accessible via `http://server_IP_address:8080/docs/`
- tomcat7-examples: Tomcat 7 Servlet and JSP examples. You can click through the example webapps to get a basic idea of how they work (and also look at the source code to see how they were implemented). Accessible via

`http://server_IP_address:8080/examples/`

- tomcat7-admin (manager-webapp): Tomcat Web Application Manager. This will allow you to manage and your Java applications.
- tomcat7-admin (host-manager): Tomcat Virtual Host Manager.

Let's take a look at the Web Application Manager, accessible via the link or `http://server_IP_address:8080/manager/html`:



Tomcat Web Application Manager

Message:	OK

Manager			
List Applications	HTML Manager Help	Manager Help	Server Status

Path	Version	Display Name	Running	Sessions	Commands
/	None specified		true	0	Start Stop Reload Undeploy Expire sessions with idle ≥ 30 minutes
/docs	None specified	Tomcat Documentation	true	0	Start Stop Reload Undeploy Expire sessions with idle ≥ 30 minutes
/examples	None specified	Servlet and JSP Examples	true	0	Start Stop Reload Undeploy Expire sessions with idle ≥ 30 minutes
/host-manager	None specified	Tomcat Host Manager Application	true	2	Start Stop Reload Undeploy Expire sessions with idle ≥ 30 minutes
/manager	None specified	Tomcat Manager Application	true	1	Start Stop Reload Undeploy Expire sessions with idle ≥ 30 minutes

Deploy

Deploy directory or WAR file located on server

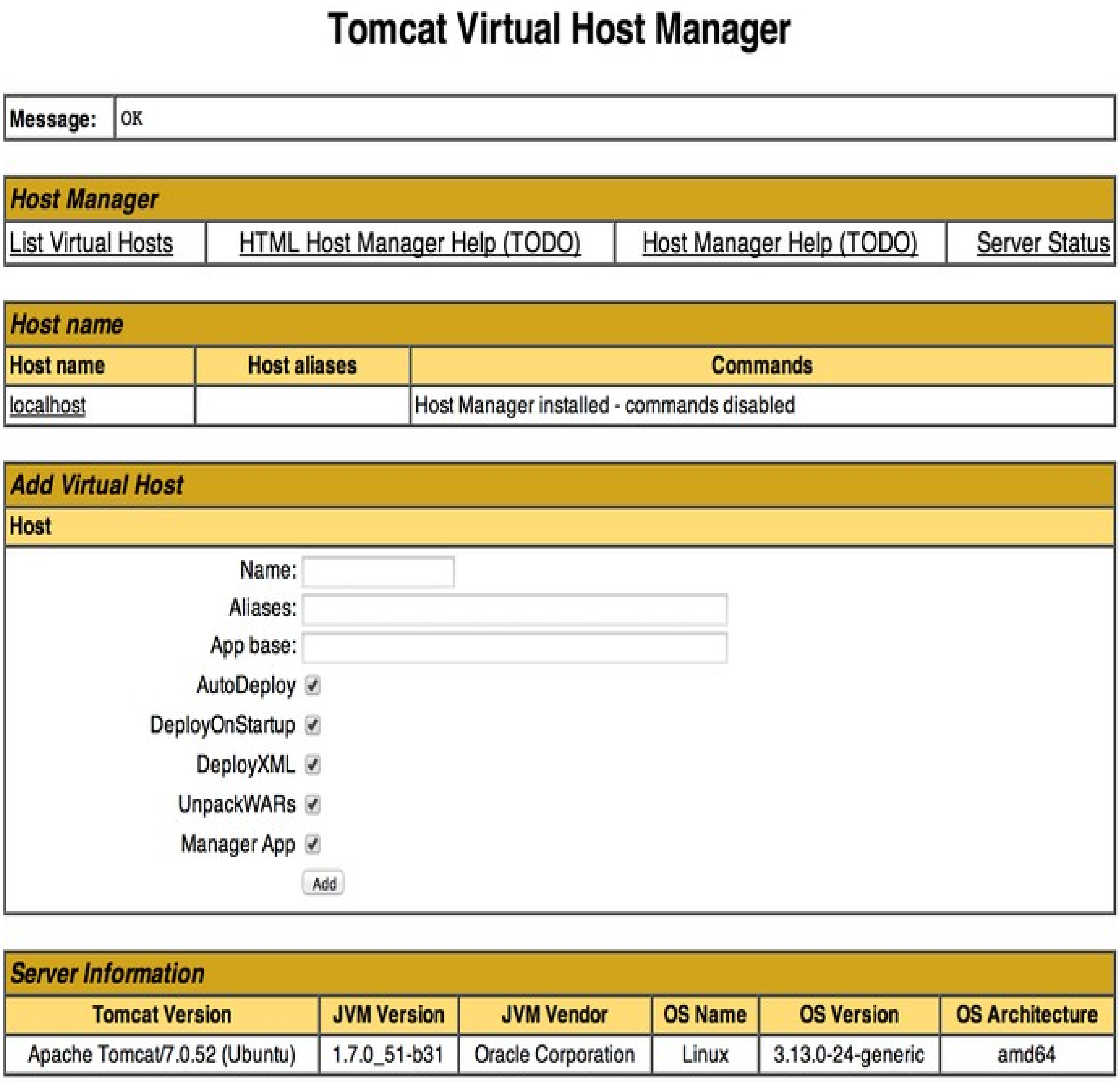
Context Path (required):
XML Configuration file URL:
WAR or Directory URL:
Deploy

WAR file to deploy

Select WAR file to upload Choose File No file chosen
Deploy

The Web Application Manager is used to manage your Java applications. You can Start, Stop, Reload, Deploy, and Undeploy here. You can also run some diagnostics on your apps (i.e. find memory leaks). Lastly, information about your server is available at the very bottom of this page.

Now let's take a look at the Virtual Host Manager, accessible via the link or `http://server_IP_address:8080/host-manager/html/`:


Tomcat Virtual Host Manager

Message:	OK

Host Manager

List Virtual Hosts	HTML Host Manager Help (TODO)	Host Manager Help (TODO)	Server Status

Host name

Host name	Host aliases	Commands
localhost		Host Manager installed - commands disabled

Add Virtual Host

Host

Name:	
Aliases:	
App base:	
AutoDeploy	✓
DeployOnStartup	✓
DeployXML	✓
UnpackWARs	✓
Manager App	✓

Server Information

Tomcat Version	JVM Version	JVM Vendor	OS Name	OS Version	OS Architecture
Apache Tomcat/7.0.52 (Ubuntu)	1.7.0_51-b31	Oracle Corporation	Linux	3.13.0-24-generic	amd64

From the Virtual Host Manager page, you can add virtual hosts to serve your applications in.

Setting Up Reverse Proxy

By default tomcat uses port 8080 to serve http traffic and 8443 to serve https traffic. You might want to use the default http and https port. You can change the Tomcat port to use port 80 and 443 or you can install a web server like Apache httpd or Nginx. This webserver will listen on port 80 and 443 and proxy the traffic to Apache Tomcat.

Installing Apache httpd

You can use `apt-get` to install Apache httpd. To install Apache httpd you can use the command below :

```
$ sudo apt-get install apache2
```

Configuring Apache httpd

Apache httpd has a proxy module, but it's not enabled by default. We need to enable this module first.

```
$ sudo a2enmod proxy $ sudo a2enmod proxy_http
```

Restart Apache httpd to reload configuration

```
$ sudo service apache2 restart
```

Open the Apache httpd default sites configuration on `/etc/apache2/sites-enabled/000-default.conf`. If you have a virtual host configuration, you can open the virtual host configuration that you want to use. You need to add configuration below between

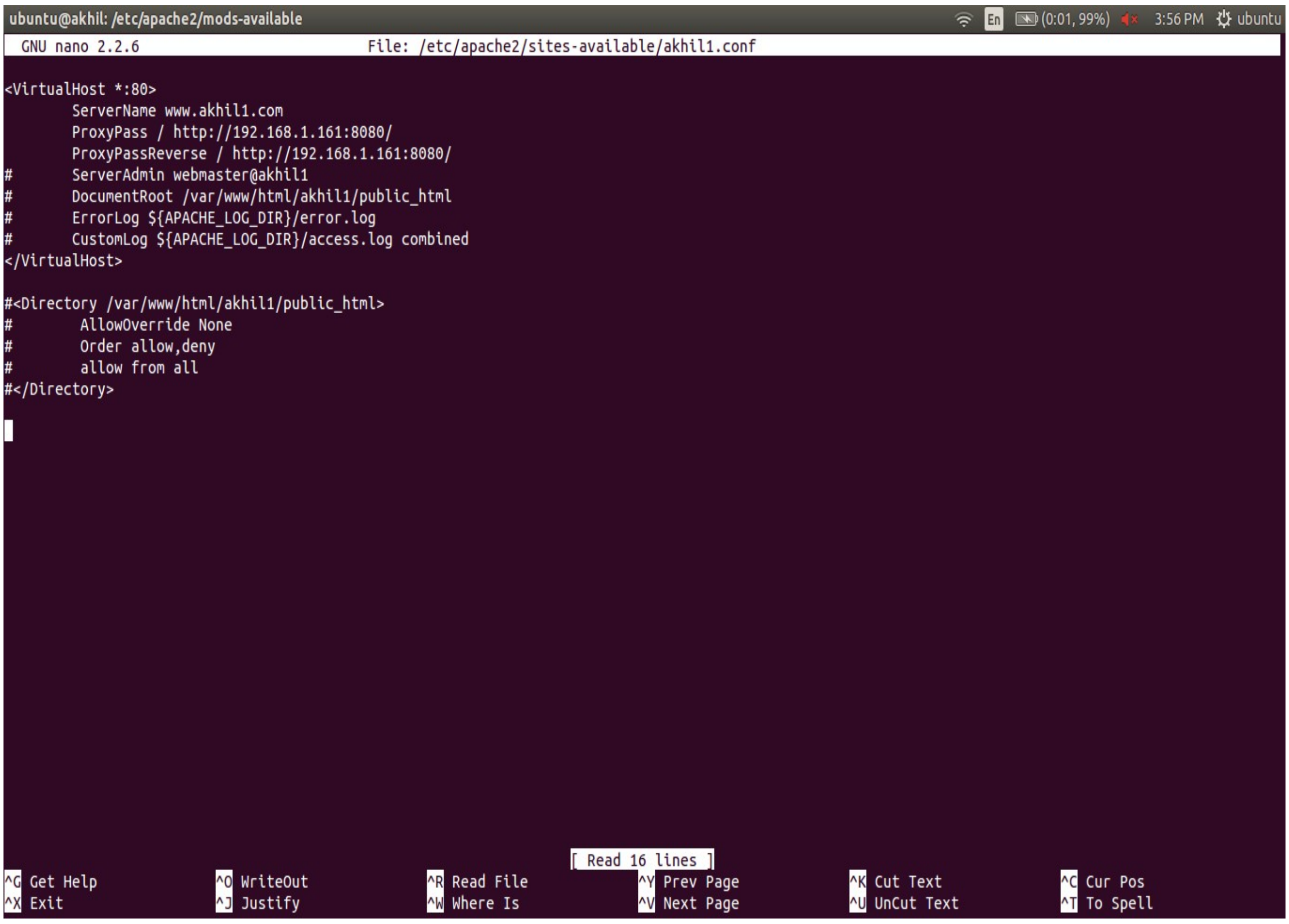
`<VirtualHost*:80>` and `</VirtualHost>`

```
ProxyPass / http://127.0.0.1:8080/

ProxyPassReverse / http://127.0.0.1:8080/
```

This will proxy traffic for `http://<server-ipaddress>/` to Tomcat. Restart Apache httpd once again to reload config.

```
$ sudo service apache2 restart
```



```
ubuntu@akhil: /etc/apache2/mods-available                    (0:01, 99%)  3:56 PM  ubuntu
  GNU nano 2.2.6                    File: /etc/apache2/sites-available/akhil1.conf

<VirtualHost *:80>
        ServerName www.akhil1.com
        ProxyPass / http://192.168.1.161:8080/
        ProxyPassReverse / http://192.168.1.161:8080/
#       ServerAdmin webmaster@akhil1
#       DocumentRoot /var/www/html/akhil1/public_html
#       ErrorLog ${APACHE_LOG_DIR}/error.log
#       CustomLog ${APACHE_LOG_DIR}/access.log combined
</VirtualHost>

#<Directory /var/www/html/akhil1/public_html>
#        AllowOverride None
#        Order allow,deny
#        allow from all
#</Directory>

                                    [ Read 16 lines ]
^G Get Help      ^O WriteOut      ^R Read File     ^Y Prev Page     ^K Cut Text      ^C Cur Pos
^X Exit          ^J Justify       ^W Where Is      ^V Next Page     ^U UnCut Text    ^T To Spell
```

ProxyPass

Syntax - ProxyPass <path> <url> <path> <url>
Context -Server configuration, virtual host
Status - Base
Module - mod_proxy
Compatibility - ProxyPass is only available in Apache 1.1 and later.

This directive allows remote servers to map into the space of the local server; the local server does not act as a proxy in the conventional sense, but appears as a mirror of the remote server. <path> is the name of a local virtual path; <url> is a partial URL for the remote server.

Suppose the local server has address http://wibble.org/; then

```
ProxyPass /mirror/foo/ http://foo.com/
```

will cause a local request for the <http://wibble.org/mirror/foo/bar> to be internally converted into a proxy request to <http://foo.com/bar>.

ProxyPassReverse

Syntax - ProxyPassReverse <path> <url> <path> <url>
Context - Server configuration, virtual host
Status - Base
Module - mod_proxy
Compatibility - ProxyPassReverse is only available in Apache 1.3b6 and later.

This directive lets Apache adjust the URL in the Location header on HTTP redirect responses. This capability is essential when Apache is used as a reverse proxy, to avoid by-passing the reverse proxy because of HTTP redirects on the back end servers, which stay behind the reverse proxy.

<Path> is the name of a local virtual path.
<url> is a partial URL for the remote server - the same way they are used for the ProxyPass directive.

Example:
Suppose the local server has address http://wibble.org/; then

```
ProxyPass         /mirror/foo/ http://foo.com/
ProxyPassReverse  /mirror/foo/ http://foo.com/
```

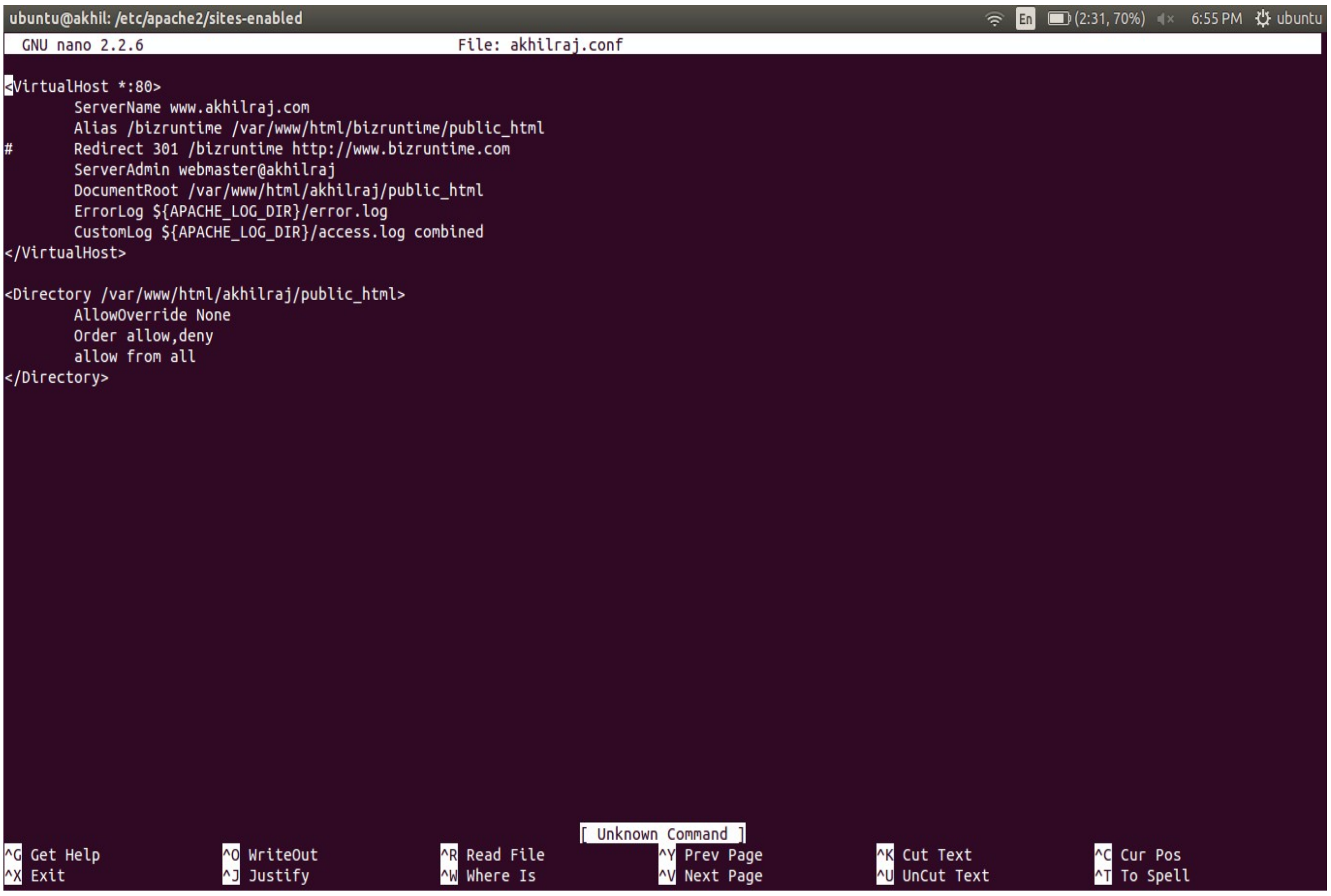
will not only cause a local request for the <http://wibble.org/mirror/foo/bar> to be internally converted into a proxy request to <http://foo.com/bar> (the functionality ProxyPass provides here).

It also takes care of redirects the server foo.com sends: when http://foo.com/bar is redirected by him to http://foo.com/quux Apache adjusts this to http://wibble.org/mirror/foo/quux before forwarding the HTTP redirect response to the client.

Note that this ProxyPassReverse directive can also be used in conjunction with the proxy pass-through feature ("RewriteRule ... [P]") from mod_rewrite because its doesn't depend on a corresponding ProxyPass directive.

Connecting fake name to real name using Alias

Alias <fake name> <true name>



```
ubuntu@akhil: /etc/apache2/sites-enabled                    (2:31, 70%)   6:55 PM   ubuntu
  GNU nano 2.2.6                              File: akhilraj.conf

<VirtualHost *:80>
        ServerName www.akhilraj.com
        Alias /bizruntime /var/www/html/bizruntime/public_html
#       Redirect 301 /bizruntime http://www.bizruntime.com
        ServerAdmin webmaster@akhilraj
        DocumentRoot /var/www/html/akhilraj/public_html
        ErrorLog ${APACHE_LOG_DIR}/error.log
        CustomLog ${APACHE_LOG_DIR}/access.log combined
</VirtualHost>

<Directory /var/www/html/akhilraj/public_html>
        AllowOverride None
        Order allow,deny
        allow from all
</Directory>

                                    [ Unknown Command ]
^G Get Help      ^O WriteOut      ^R Read File     ^Y Prev Page     ^K Cut Text      ^C Cur Pos
^X Exit          ^J Justify       ^W Where Is      ^V Next Page     ^U UnCut Text    ^T To Spell
```