Running Jenkins behind Apache

In situations where you have existing web sites on your server, you may find it useful to run Jenkins (or the servlet container that Jenkins runs in) behind Apache, so that you can bind Jenkins to the part of a bigger website that you may have. This document discusses some of the approaches for doing this.

Make sure that you change the Jenkins httpListenAddress from its default of 0.0.0.0 to 127.0.0.1 or any Apache-level restrictions can be easily bypassed by hitting the Jenkins port directly.

mod_proxy

mod_proxy works by making Apache perform "reverse proxy" — when a request arrives for certain URLs, Apache becomes a proxy and further forward that request to Jenkins, then it forwards the response back to the client.

The following Apache modules must be installed :

```
a2enmod proxy
a2enmod proxy_http
a2enmod headers
```

A typical set up for mod_proxy would look like this:

```
ProxyPass         /jenkins  http://localhost:8081/jenkins nocanon
ProxyPassReverse  /jenkins  http://localhost:8081/jenkins
ProxyRequests     Off
AllowEncodedSlashes NoDecode

# Local reverse proxy authorization override
# Most unix distribution deny proxy by default (ie /etc/apache2/mods-
enabled/proxy.conf in Ubuntu)
<Proxy http://localhost:8081/jenkins*>
  Order deny,allow
  Allow from all
</Proxy>
```

This assumes that you run Jenkins on port 8081.

For this set up to work, the context path of Jenkins **must be the same** between your Apache and Jenkins (that is, you can't run Jenkins on http://localhost:8081/ci and have it exposed at http://localhost:80/jenkins).
Set the context path in Windows by modifying the `jenkins.xml` configuration file and adding --prefix=/jenkins (or similar) to the <arguments> entry.
Set the context path when using the Ubuntu package by adding `--prefix=/jenkins` to `JENKINS_ARGS` in `/etc/default/jenkins` (or in `/etc/sysconfig/jenkins` for RHEL/CentOS package)

When running on a dedicated server and you are using / as context, make sure you add a slash at the end of all URLs in proxy params in apache. Otherwise you might run into proxy errors. So

```
ProxyPass / http://localhost:8080/ nocanon
```

instead of

```
ProxyPass / http://localhost:8080 nocanon     # wont work
```

Note that this does **not** apply to the `ProxyPassMatch` directive, which behaves differently than `ProxyPass`. Below is an example of `ProxyPassMatch` to proxy all URLs other than `/.well-known` (a URL required by letsencrypt):

```
ProxyPassMatch  ^/(?\!.well-known)  http://localhost:8080 nocanon
```

The *ProxyRequests Off* prevents Apache from functioning as a forward proxy server (except for *ProxyPass*), it is advised to include it unless the server should function as a proxy.

Both the `nocanon` option to `ProxyPass`, *and* `AllowEncodedSlashes NoDecode`, are required for certain Jenkins features to work.

If you are running Apache on a Security-Enhanced Linux (SE-Linux) machine it is essential to make SE-Linux do the right thing by issuing as root

```
setsebool -P httpd_can_network_connect true
```

If this is not issued Apache will not be allowed to forward proxy requests to Jenkins and only an error message will be displayed.

Because Jenkins already compress its output, you can not use the normal proxy-html filter to modify urls:

```
SetOutputFilter proxy-html
```

Instead you can use the following:

```
SetOutputFilter INFLATE;proxy-html;DEFLATE
ProxyHTMLURLMap http://your_server:8080/jenkins /jenkins
```

http://wiki.uniformserver.com/index.php/Reverse_Proxy_Server_2:_mod_proxy_html_2
But since Jenkins seems to be well behaved it's even better to just not use SetOutputFilter and ProxyHTMLURLMap.

If there are problems with Jenkins sometimes servicing random garbage pages, then the following may help:

```
SetEnv proxy-nokeepalive 1
```

Some plug-ins determine URLs from client requests from Host header, so if you experience some problems with wrong URLs, you can try to switch on `ProxyPreserveHost` directive, which is switched off by default:

```
ProxyPreserveHost On
```

mod_proxy with HTTPS

If you'd like to run Jenkins with reverse proxy in HTTPS, one user reported that HTTPS needs to be terminated at Jenkins, not at the front-end Apache. See this e-mail thread for more discussion.

Note that you also may need to have access to your host via regular http, else the admin interface test will report a broken proxy setup.

Alternatively, you can add an additional `ProxyPassReverse` directive to redirect non-SSL URLs generated by Jenkins to the SSL side. Assuming that your webserver is `your.host.com`, placing the following within the SSL virtual host definition will do the trick:

```
ProxyRequests     Off
ProxyPreserveHost On
AllowEncodedSlashes NoDecode

<Proxy http://localhost:8081/jenkins*>
  Order deny,allow
  Allow from all
</Proxy>

ProxyPass         /jenkins  http://localhost:8081/jenkins nocanon
ProxyPassReverse  /jenkins  http://localhost:8081/jenkins
ProxyPassReverse  /jenkins  http://your.host.com/jenkins
```

Yet another option is to rewrite the Location headers that contain non-ssl URL's generated by Jenkins. If you want to access Jenkins from https://www.example.com/jenkins, placing the following within the SSL virtual host definition also works:

```
ProxyRequests     Off
ProxyPreserveHost On
ProxyPass /jenkins/ http://localhost:8081/jenkins/ nocanon
AllowEncodedSlashes NoDecode

<Location /jenkins/>
  ProxyPassReverse /
  Order deny,allow
  Allow from all
</Location>

Header edit Location ^http://www.example.com/jenkins/ https://www.
example.com/jenkins/
```

But it may also work fine to just use simple forwarding as above (the first HTTPS snippet), and add

```
RequestHeader set X-Forwarded-Proto "https"
RequestHeader set X-Forwarded-Port "443"
```

in the HTTPS site configuration, as the Docker demo (below) does. (`X-Forwarded-Port` is not interpreted by Jenkins prior to JENKINS-23294 so it may also be desirable to configure the servlet container to specify the originating port.)

The collection of snippets above simply don't work out of the box (July 2014), here is a full Apache-oriented "sites-enabled" file (ex: "sites-enabled/example") for a dedicated Jenkins host, combining the ideas from snippets #1 and #3. This was formulated on the TurnKeyLinux Jenkins appliance (v 13.0), after having updated Jenkins to "1.572". TODO (if anyone understands how to do so): Define a more selective path for the <Proxy *> tag, instead of *; I currently have the impression that the <Proxy> section is not even needed.

```
NameVirtualHost *:80
NameVirtualHost *:443

<VirtualHost *:80>
    ServerAdmin  webmaster@localhost
    Redirect permanent / https://www.example.com/
</VirtualHost>

<VirtualHost *:443>
    SSLEngine on
    SSLCertificateFile /etc/ssl/certs/cert.pem
    ServerAdmin  webmaster@localhost
    ProxyRequests     Off
    ProxyPreserveHost On
    AllowEncodedSlashes NoDecode
    <Proxy *>
        Order deny,allow
        Allow from all
    </Proxy>
    ProxyPass         /  http://localhost:8080/ nocanon
    ProxyPassReverse  /  http://localhost:8080/
    ProxyPassReverse  /  http://www.example.com/
    RequestHeader set X-Forwarded-Proto "https"
    RequestHeader set X-Forwarded-Port "443"
</VirtualHost>
```

mod_ajp/mod_proxy_ajp

More info welcome. Probably we should move the contents from here

I wanted to have Jenkins running in a different workspace than my normal Tomcat server, but both available via the Apache web server. So, first up, modify Jenkins to use a different web and ajp port than Tomcat:

```
HTTP_PORT=9080
AJP_PORT=9009
...
nohup java -jar "$WAR" --httpPort=$HTTP_PORT --ajp13Port=$AJP_PORT --
prefix=/jenkins >> "$LOG" 2>&1 &
```

Then setup Apache so that it knows that the prefix `/jenkins` is being served by AJP in the httpd.conf file:

```
LoadModule jk_module          libexec/httpd/mod_jk.so

AddModule     mod_jk.c

#== AJP hooks ==
JkWorkersFile /etc/httpd/workers.properties
JkLogFile     /private/var/log/httpd/mod_jk.log
JkLogLevel    info
JkLogStampFormat "[%a %b %d %H:%M:%S %Y] "
JkOptions     +ForwardKeySize +ForwardURICompat -ForwardDirectories
JkRequestLogFormat     "%w %V %T"
# Here are 3 sample applications - 2 that are being served by Tomcat,
and Jenkins
JkMount  /friki/* worker1
JkMount  /pebble/* worker1
JkMount  /jenkins/* worker2
```

Then finally the workers.conf file specified above, that just tells AJP which port to use for which web application:

```
# Define 2 real workers using ajp13
worker.list=worker1,worker2
# Set properties for worker1 (ajp13)
worker.worker1.type=ajp13
worker.worker1.host=localhost
worker.worker1.port=8009
worker.worker1.lbfactor=50
worker.worker1.cachesize=10
worker.worker1.cache_timeout=600
worker.worker1.socket_keepalive=1
# Set properties for worker2 (ajp13)
worker.worker2.type=ajp13
worker.worker2.host=localhost
worker.worker2.port=9009
worker.worker2.lbfactor=50
worker.worker2.cachesize=10
worker.worker2.cache_timeout=600
worker.worker2.socket_keepalive=1
worker.worker2.recycle_timeout=300
```

mod_proxy_ajp+SSL

AJP is an arguably cleaner alternative for an SSL-enabled reverse proxy, since Jenkins will get all pertinent HTTP headers untouched. Configuration is a snap too, in three simple steps:

1. Configure an AJP port for Jenkins (as mentioned above)

```
HTTP_PORT=-1
AJP_PORT=9009
...
nohup java -jar "$WAR" --httpPort=$HTTP_PORT --ajp13Port=$AJP_PORT --
prefix=/jenkins >> "$LOG" 2>&1 &
```

2. Enable mod_proxy_ajp in Apache:

```
# a2enmod proxy_ajp
```

3. Include the following snippet in your SSL-enabled VirtualHost:

```
<VirtualHost *:443>
...
        SSLEngine on
...
        AllowEncodedSlashes NoDecode
        ProxyRequests Off
        ProxyPass /jenkins ajp://localhost:9009/jenkins nocanon
</VirtualHost>
```

Note the use of `AllowEncodedSlashes` and `ProxyPass...nocanon` to persuade Apache to leave PATH_INFO alone.

mod_rewrite

Some people attempted to use mod_rewrite to do this, but this will never work if you do not add a *ProxyPassReverse*.
See the thread if you'd like to know why.

The following Apache modules must be installed :

```
a2enmod rewrite
a2enmod proxy
a2enmod proxy_http
```

A typical set up for mod_rewrite would look like this:

```
# Use last flag because no more rewrite can be applied after proxy pass
RewriteRule       ^/jenkins(.*)$  http://localhost:8081/jenkins$1 [P,L]
ProxyPassReverse  /jenkins        http://localhost:8081/jenkins
ProxyRequests     Off

# Local reverse proxy authorization override
# Most unix distribution deny proxy by default (ie /etc/apache2/mods-
enabled/proxy.conf in Ubuntu)
<Proxy http://localhost:8081/jenkins*>
  Order deny,allow
  Allow from all
</Proxy>
```

This assumes that you run Jenkins on port 8081. For this set up to work, the context path of Jenkins must be the same between your Apache and Jenkins (that is, you can't run Jenkins on http://localhost:8081/ci and have it exposed at http://localhost:80/jenkins)

The *ProxyRequests Off* prevents Apache from functioning as a forward proxy server (except for *ProxyPass*), it is advised to include it unless the server should function as a proxy.

Testing compatibility from plugins

Try https://index.docker.io/u/jglick/jenkins-demo-reverse-proxy/ to see if your plugin works behind an Apache reverse proxy.

Proxying CLI commands using the HTTP(S) transport with Jenkins >= 2.54

Using the plain CLI protocol with the HTTP(S) transport to access Jenkins >= 2.54 through an Apache reverse proxy does not work. (See

JENKINS-47279 - Full-duplex HTTP(S) transport with plain CLI protocol does not work with Apache reverse proxy in Jenkins >= 2.54 OPEN

, and update it if you have settings that do work!) As a workaround, you can use the CLI over SSH.