

# LAMP Stack with VirtualHosts On Centos 6.x

This article illustrates how to install the Apache MySQL PHP Stack on Centos 6.x.

Additionally, with this configuration, you can serve Multiple Domains using the Virtual Hosts Apache directive.

## Install Apache

Invoke yum for installation of Apache

```
yum install -y httpd mod_ssl httpd-devel
```

@!:{httpd-devel libraries were included in order to have module compile capabilities, as well as being able to install modules from source

Enable autostart of the Apache service

```
chkconfig httpd on
```

Start the Apache service

```
service httpd restart
```

## Install PHP

Install PHP, et al

```
yum install -y php php-mysql php-common php-mbstring php-mcrypt php-devel php-xml php-pecl-memcache php-pspell php-snmp php-xmlrpc php-gd
```

Restart the Apache service

```
service httpd restart
```

## Check DNS

Ensure there exists a DNS entry for the domain you want to use.

If this is a lab setup, or completely local, you can simply create a hosts entry for the domain, e.g.

```
vi /etc/hosts
```

[divider]

## Virtual Hosts

The **NameVirtualHost** directive allows us to host multiple websites on a single web server.

Example:

You want to host **mydomain1.com** on your web server

You also want to host **mydomain2.com** on your web server

In order to accomplish this, you'll need to:

- enable the NameVirtualHost directive
- create appropriate configuration files for the domains in question, e.g.:

```
/etc/httpd/conf.d/mydomain1.com.conf
```

```
/etc/httpd/conf.d/mydomain2.com.conf
```

For now, let's configure just one domain, *mydomain1.com*:

[divider]

## Create Vhosts Config Directories

```
Create a vhost config folder
```

```
mkdir -p /etc/httpd/vhost.d
```

## Configure NameVirtualHost Directive

Add an include directive to the apache config file:

```
vim /etc/httpd/conf/httpd.conf  
    Include vhost.d/*.conf
```

@!:{The above makes it so that any files ending in .conf under the folder vhost.d are included as part of the httpd.conf configuration

Notice that **vhost.d** is a relative path. The full path would be evaluated as ServerRoot/vhost.d, where ServerRoot is /etc/httpd (see the httpd.conf file for more information)

Comment out any Listen directives and add an include directive to a separate ports settings config file:

```
#Listen 12.34.56.78:80  
#Listen 80  
Include ports.conf
```

@!:{The above makes it so that the ports.conf file is included as part of the httpd.conf configuration

What this accomplishes is a separation of port specification from the main config file

Create a ports config file

```
vi /etc/httpd/ports.conf
```

With contents:

```
Listen $Port  
NameVirtualHost $IPPUBLIC:$Port  
NameVirtualHost $IPPRIVATE:$Port
```

```
NameVirtualHost *:$Port
```

Where **\$Port** is the numeric value of the port number through which you want Apache to listen for traffic

*#e.g.*

```
NameVirtualHost 192.168.250.188:80
```

```
NameVirtualHost 127.0.0.1:80
```

```
NameVirtualHost *:80
```

Restart Apache

```
service httpd restart
```

## Create The Config File for the Virtual Host/Domain

Create a config file for your domain

```
vim /etc/httpd/vhost.d/mydomain1.conf
```

```
<VirtualHost *:80>
```

```
    ServerName mydomain1.com
```

```
    ServerAlias www.mydomain1.com
```

```
    DocumentRoot /var/www/vhosts/mydomain1.com
```

```
    <Directory /var/www/vhosts/mydomain1.com>
```

```
        Options Indexes FollowSymLinks MultiViews
```

```
        AllowOverride All
```

```
    </Directory>
```

```
        CustomLog /var/log/httpd/mydomain1.com-access.log  
combined
```

```
        ErrorLog /var/log/httpd/mydomain1.com-error.log
```

```
# Possible values include: debug, info, notice, warn,
error, crit,
# alert, emerg.
LogLevel warn

</VirtualHost>
```

Make sure your document root exists!

```
mkdir /var/www/vhosts/mydomain1.com
#-OR Try this One-liner-#
ls /var/www/vhosts/mydomain1.com 2> /dev/null || echo does
not exist;echo creating folder;mkdir -p
/var/www/vhosts/mydomain1.com && echo created folder!
```

[divider]

## Modify Firewall

You'll need to poke a hole in the firewall to allow communication to the Apache listening port (by default port 80):

Edit iptables config

```
vi /etc/sysconfig/iptables
A INPUT -m state --state NEW -m tcp -p tcp --dport 80 -j ACCEPT
```

Restart iptables

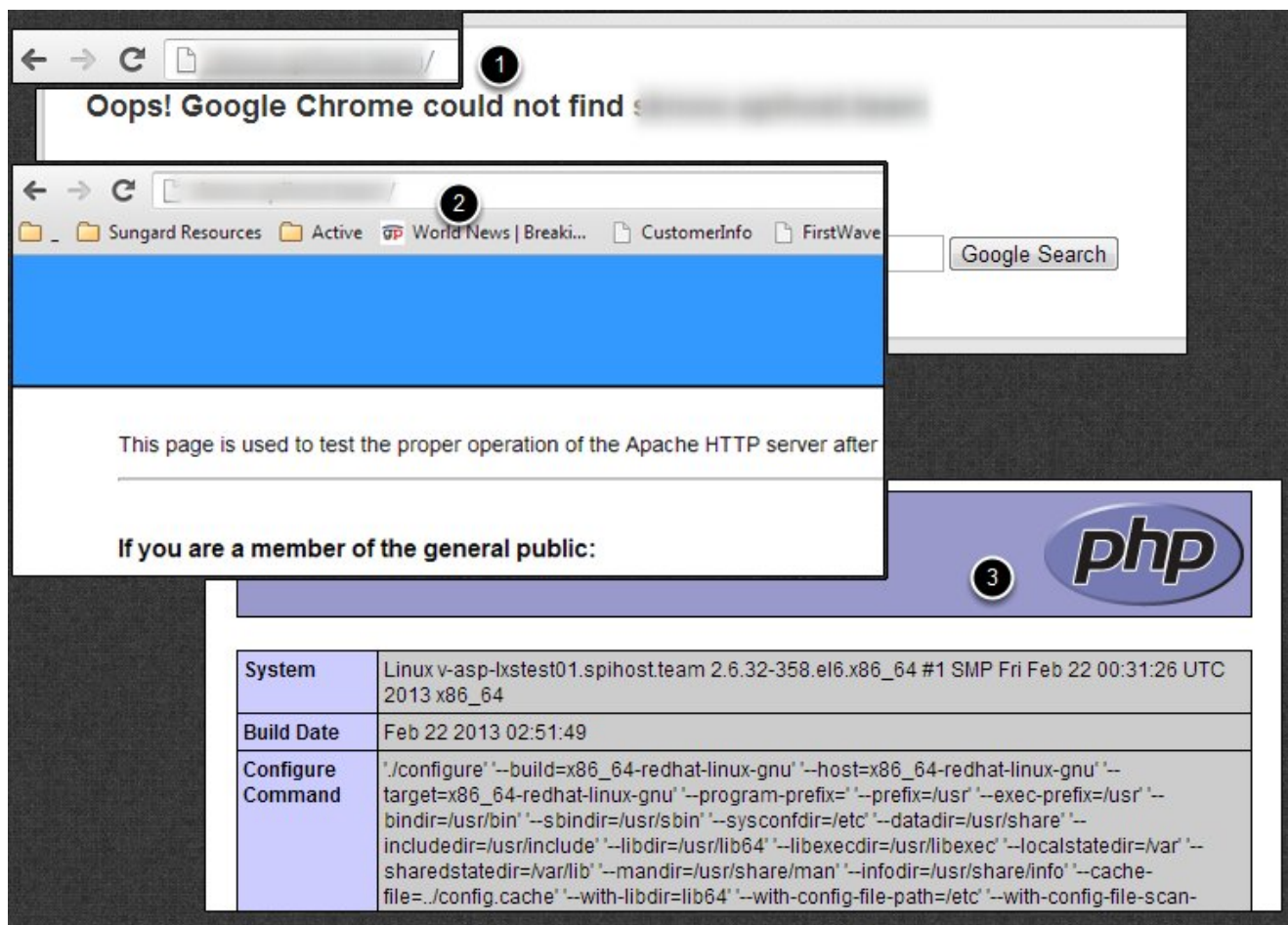
```
service iptables restart
```

[divider]

## Troubleshooting

[divider]

## Error – Could not find ...



1. Problem: When navigating to your domain via web browser, you receive an error similar to 'could not find'

Q:{Is DNS setup correctly?

Check:

```
nslookup mydomain1.com
```

if error then ensure DNS record exists on your DNS server

if Windows, try the `ipconfig /flushdns` command

Q:{Is Firewall to blame?

Check:

```
telnet $yourdomain $port
```

e.g.

```
telnet mydomain1.com 80
```

if error then ensure Firewall port is open:

```
vi /etc/sysconfig/iptables  
e.g. -A INPUT -m state --state NEW -m tcp -p tcp --dport 80 -j  
ACCEPT
```

Restart firewall:

```
service iptables restart
```

**2. Test website access again**  
Hopefully Success!

**3. Test PHP functionality:**

```
vi /var/www/vhosts/domain.com/index.php  
  
<?php  
phpinfo();  
?>  
:wq
```

Test website access again

```
http://mydomain1.com/index.php
```

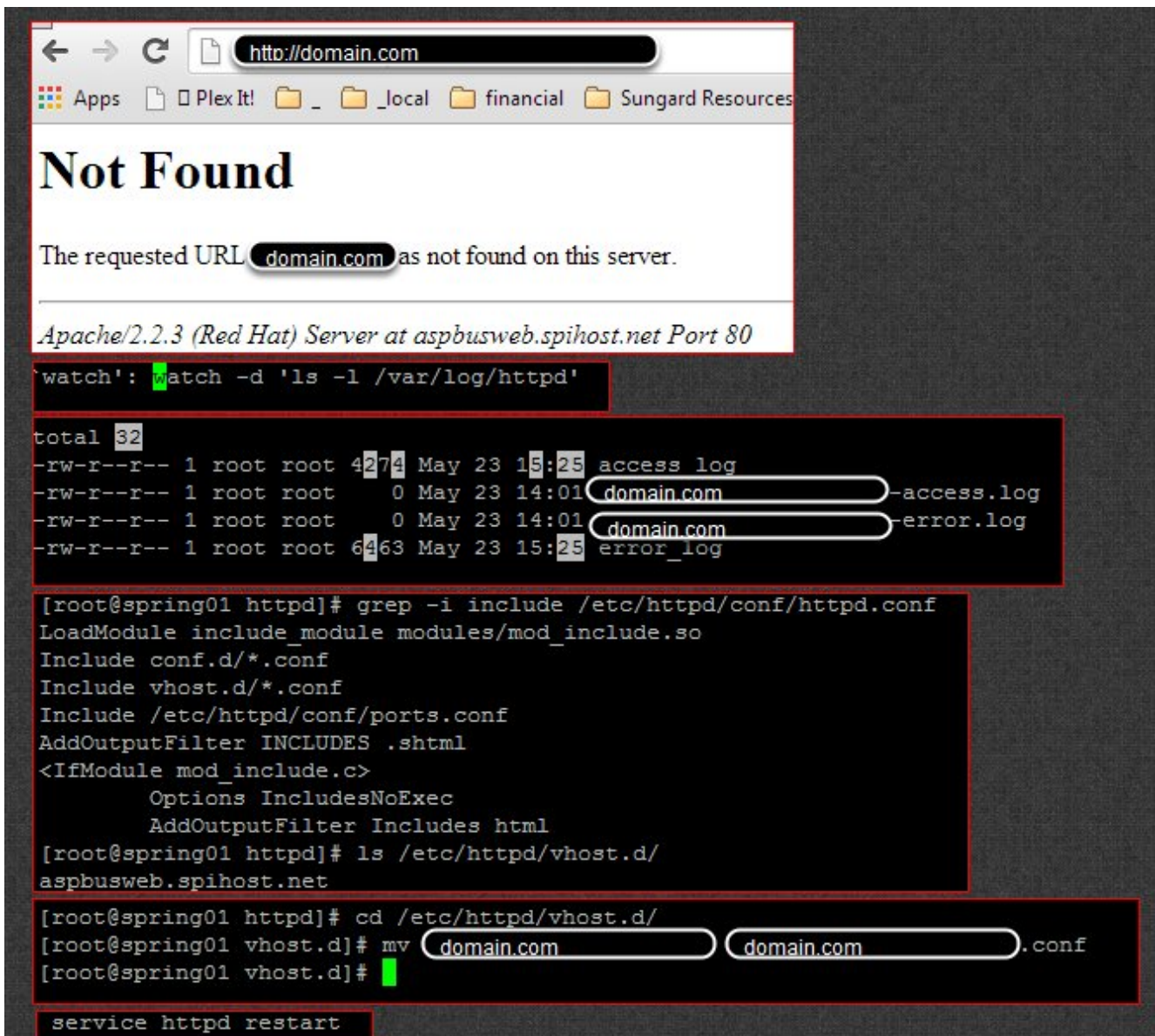
If you've made numerous changes, try restarting the Apache service again

```
service httpd restart
```

If all else fails, and if you have the option to do so, reboot the server

```
reboot
```

# Error – requested URL was not found on this server



The screenshot shows a web browser window with the address bar containing `http://domain.com`. The page displays a "Not Found" error message: "The requested URL `domain.com` as not found on this server." Below the error message, it identifies the server as "Apache/2.2.3 (Red Hat) Server at `aspbusweb.spihost.net` Port 80".

The terminal window shows the following commands and output:

```
watch: watch -d 'ls -l /var/log/httpd'
```

```
total 32
-rw-r--r-- 1 root root 4274 May 23 15:25 access_log
-rw-r--r-- 1 root root 0 May 23 14:01 domain.com-access.log
-rw-r--r-- 1 root root 0 May 23 14:01 domain.com-error.log
-rw-r--r-- 1 root root 6463 May 23 15:25 error_log
```

```
[root@spring01 httpd]# grep -i include /etc/httpd/conf/httpd.conf
LoadModule include_module modules/mod_include.so
Include conf.d/*.conf
Include vhost.d/*.conf
Include /etc/httpd/conf/ports.conf
AddOutputFilter INCLUDES .shtml
<IfModule mod_include.c>
    Options IncludesNoExec
    AddOutputFilter Includes html
[root@spring01 httpd]# ls /etc/httpd/vhost.d/
aspbusweb.spihost.net
```

```
[root@spring01 httpd]# cd /etc/httpd/vhost.d/
[root@spring01 vhost.d]# mv domain.com domain.com.conf
[root@spring01 vhost.d]#
```

```
service httpd restart
```

In this case, I created the config file for the domain under `vhosts.d`, but had forgotten to give it a `.conf` file extension. doh!

Note how I used the `watch` command to 'watch' for changes to log files under `/var/log/httpd`.

This functions much like `inotifywait` for troubleshooting using log files.



---

# Protect Apache Webfolder From Unauthorized Access

In this article, I cover how you can easily implement a secure, web-accessible file depot using Linux, Apache, PHP, and an LDAP Authentication Backend (in this case, Microsoft Active Directory).

## Overview

The configuration in question employs a simple flat text file named `.htaccess` to force authentication for a particular web path.

This will be accomplished through a dedicated user object that serves to *glue (or bind)* the authentication session to the LDAP instance and a security group that further restricts access by membership.

The end result:

The only user accounts allowed access to the url in question are those adhering to these constraints:

- Only user objects contained under the defined LDAP path
- Only those user objects that are members of the MySpecialGroup security group

This is the environment from which I accomplished this:

Web server details:

OS: CentOS 6.x

Apache Version: 2.2.15

PHP Version: 5.3.3

LDAP:

Active Directory on Windows Server 2008 R2

In this article, we are assuming the following:

- The URL in question is *http://filedopot.contoso.com*
- The physical path to the web folder is */var/www/vhosts/filedopot.contoso.com*
- The Active Directory Domain is *contoso.com*
- The IP Address for the Domain Controller is 192.168.1.1
- The LDAP binding user account is named *MyServiceAccount*
- The Organizational Unit containing the binding account is located under *contoso.com*
- Service Accounts:

*ou=service accounts,dc=contoso,dc=com*

\* With the full path to the binding account user object being:

*cn=myserviceaccount,ou=service accounts,dc=contoso,dc=com*

- The Organizational Unit containing the user objects is located under *contoso.com*
- MyOU
- Users:

*ou=users,ou=myou,dc=contoso,dc=com*

- The LDAP security group is named *myspecialgroup*
- The Organizational Unit containing the security group objects is located under *contoso.com*
- Groups:

*ou=groups,dc=contoso,dc=com*

\* With the full path to the security group object being:

*cn=myspecialgroup,ou=groups,dc=contoso,dc=com*

See: [Appendix](#) for more information on Apache .htaccess files.

Let's proceed with the general workflow, shall we?

# Determine LDAP Path to User Objects

The image shows a screenshot of the Active Directory Users and Computers console. The left pane shows the hierarchy: Active Directory Users and Computers > contoso.com > MyOU > Users. The right pane shows the properties of a user object in the 'MyOU Properties' window. The 'distinguishedName' attribute is highlighted, and its value is shown in the 'String Attribute Editor' window as 'OU=Users,OU=MyOU,DC=Contoso,DC=com'. Below the screenshot is a diagram illustrating the LDAP path as nested rectangles representing organizational units. The outermost rectangle is labeled 'Domain Component = com'. Inside it is a rectangle labeled 'Domain Component = contoso'. Inside that is a rectangle labeled 'Organizational Unit = MyOU'. Inside that is the innermost rectangle labeled 'Organizational Unit = Users'. Red arrows point from the labels 'dc = com', 'dc = contoso', 'ou = MyOU', and 'ou = Users' to their respective levels in the hierarchy.

Apache needs to know what bucket holds the user objects that will be allowed to authenticate.

As illustrated in this example, the user objects are contained in the **Users** Organizational Unit which is a child of the **MyOU** Organizational Unit within the **contoso.com** domain.

In like fashion, we can determine the paths to the binding user object and the security group.

## Create The .htaccess File

With the LDAP information defined, we can now build a .htaccess file for our given web folder.

Login to the machine in question • navigate to your web root •

create the .htaccess file

```
cd /var/www/vhosts/iledopot.contoso.com
vi ./htaccess
```

According to our LDAP settings, the contents of this file should be:

```
AuthType Basic
AuthName "Network Credentials Required"
AuthBasicProvider ldap
AuthLDAPURL
"ldap://192.168.1.1:389/ou=users,ou=myou,dc=contoso,dc=com?sAM
AccountName?sub?(objectClass=*)"
AuthLDAPBindDN "cn=myserviceaccount,ou=service
accounts,dc=contoso,dc=com"
AuthLDAPBindPassword "somepassword"
Require ldap-group
cn=myspecialgroup,ou=groups,dc=contoso,dc=com
```

## Test Access & Troubleshoot

1. Attempt navigation to the url in question:  
*http://iledopot.contoso.com*
2. Verify that you are prompted for credentials.
3. Try entering in a valid username and password combination.
4. If Problems, you can troubleshoot access by producing a live view of the site's apache error log, e.g.:

```
tail -f /var/log/httpd/contoso.com-error.log
```

[divider]

## Appendix

[divider]

# Sources

URL	Description	Apache .htaccess files
	<a href="http://httpd.apache.org/docs/2.2/howto/htaccess.html">http://httpd.apache.org/docs/2.2/howto/htaccess.html</a>	

## Automating IIS

This Article presents some examples of automating IIS using PowerShell and the appcmd.exe tool

### References I used

Title	URL IIS AppCmd Quick Reference	PowerShell Snap-In	PowerShell Snap-In
	<a href="http://blogs.web.com/robtech/archive/2012/04/23/iis-appcmd-quick-reference.aspx">http://blogs.web.com/robtech/archive/2012/04/23/iis-appcmd-quick-reference.aspx</a>	<a href="https://github.com/jehoroki/iiswebadminpowershell/blob/master/PowerShellSnap-In.ps1">https://github.com/jehoroki/iiswebadminpowershell/blob/master/PowerShellSnap-In.ps1</a>	<a href="http://www.iis.net/learn/manage/powershell/powershell-snap-in-creating-web-sites-web-applications-virtual-directories-and-application-pools">http://www.iis.net/learn/manage/powershell/powershell-snap-in-creating-web-sites-web-applications-virtual-directories-and-application-pools</a>

## PowerShell Commands

This applies to my environment: Before you can utilize PowerShell commands for IIS automation, you'll need to ensure two things:

- PowerShell Script Execution is allowed for your session

```
[code language="powershell"]Set-Executionpolicy Bypass -Scope Process[/code]
```

- IIS Web Administration PowerShell module is properly loaded

```
[code language="powershell"]Import-Module WebAdministration[/code]
```

In the following end-to-end scenario we will execute the following steps:

