LAB 22

WEB SERVER ADMINISTRATION

What You Will Learn

- Where to configure some common services for webservers
- Some practical hands on walkthroughs with domain registration
- How to host multiple domains on 1 IP address
- How to set up an open source log statistics package.

Approximate Time

The exercises in this lab should take approximately 90 minutes to complete.

Fundamentals of Web Development, 2nd Ed

Randy Connolly and Ricardo Hoar

Textbook by Pearson
http://www.funwebdev.com

Date Last Revised: May 20, 2017

PREPARING DIRECTORIES

1 This Lab, unlike most of the previous ones, cannot easily be contained within a folder. At some point, you will go beyond the development of a site and actually launch a website with a domain.

2 This lab will walk you through some projects and may assume at times you have access to a host and a domain name you want to register. Later, in Lab 21, for example we continue to build on the assumption that you have a live domain you can work with.

GETTING STARTED IN THE REAL WORLD WEB

Exercise 22.1 — REGISTER A DOMAIN

1 Choose a domain name that reflects the website or business you want to launch. Some considerations are short length (hence funwebdev.com and not fundamentalsofwebdevelopment.com).

2 Once you have a domain name in mind, you must select a registrar to work with. This is a very competitive industry with literally dozens of registrars to choose from. While most all registrars offer the same services, not all do. The authors suggest researching and discussing which registrar to use with your classmates.

3 Now visit the registrar and make an account. Be aware that your mailing address may be used in the WHOIS registration of your domains.

4 Type the domain you want and see if it's available. Sometimes the domain is not available and the registrar will suggest similar domains. Other times, the registrar will select you purchase additional domains with different TLD. You can register just the one for now.

5 Choose whether to have a secret registration (private WHOIS info) or public. You will pay more for private registration.

6 Finally you will have to say no to hosting, email, and other "special offers" unless you really want to get hosting right away.

With the domain registered you have to consider where to host the website. Again, this is best left as a research question, where you compare popular hosts to decide which ones offer the features you want at the price you want.

Once you make the choice find out what their name servers are called.

Finally change you name server records with the registrar to point to eh DNS servers of your host. Although most registrars offer an easy to use online form to change name servers, some require you to phone in or send a special request.

Exercise 22.2 — Find Out Who Owns a Domain

1 Sometimes determining the ownership of a domain is as easy as doing a WHOIS on the domain name. This is so popular that there are even websites to do WHOIS queries on your behalf.

2 Choose a domain whose ownership you are interested in and run a whois command on the domain.

```
whois funwebdev.com
```

tells us it's a domain registered to one of the authors.

```
Domain Name: FUNWEBDEV.COM
Registry Domain ID: 1741031189_DOMAIN_COM-VRSN
Registrar WHOIS Server: whois.bluehost.com
Registrar URL: http://www.bluehost.com/
Updated Date: 2014-01-24T17:14:59Z
Creation Date: 2012-08-27T22:33:49Z
Registrar Registration Expiration Date: 2014-08-27T22:33:49Z
Registrar: FastDomain Inc.
Registrar IANA ID: 1154
Registrar Abuse Contact Email: support@bluehost.com
Registrar Abuse Contact Phone:
Reseller: BlueHost.Com
Domain Status: clientTransferProhibited
Registry Registrant ID:
Registrant Name: RICARDO HOAR
```

Exercise 22.3 -- Checking Name Servers

1 This exercise shown you how to test and see how a particular name server is responding to DNS requests.

2 To begin, determine the name servers for a test domain. In our case to determine the name servers for funwebdev.com type at the command line:

```
dig NS funwebdev.com
```

The output will tell us what our local DNS resolver (192.168.1.1 in my case) has determined is the nameserver for the domain:

```
; <<>> DiG 9.8.5-P1 <<>> NS funwebdev.com
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 31842
;; flags: qr rd ra; QUERY: 1, ANSWER: 5, AUTHORITY: 0, ADDITIONAL: 0

;; QUESTION SECTION:
;funwebdev.com.          IN  NS

;; ANSWER SECTION:
funwebdev.com.    59599  IN  NS     ns1.linode.com.
funwebdev.com.    59599  IN  NS     ns5.linode.com.
funwebdev.com.    59599  IN  NS     ns2.linode.com.
funwebdev.com.    59599  IN  NS     ns4.linode.com.
funwebdev.com.    59599  IN  NS     ns3.linode.com.

;; Query time: 50 msec
;; SERVER: 192.168.1.1#53(192.168.1.1)
;; WHEN: Sat Apr 05 11:28:49 MDT 2014
;; MSG SIZE  rcvd: 128
```

As you can see our local DNS resolver identifies the name servers from linode, which is where we are hosted at the time of writing.

3 This technique is far more interesting when you have just updated your name servers somewhere, and you local DNS resolver is not yet updated. Imagine for a moment that we had just updated our A (host) records, but our DNS resolver still had the old IP address showing. In that case we could check what the new name servers (69.93.127.10) would reply with by specifying them in the dig command using the @ syntax as follows:

```
dig A funwebdev.com @ns1.linode.com
```

Which would tell us:

```
;; ANSWER SECTION:
funwebdev.com.    86400  IN  A      198.58.103.61
;; Query time: 89 msec
;; SERVER: 69.93.127.10#53(69.93.127.10)
;; WHEN: Sat Apr 05 11:35:47 MDT 2014
;; MSG SIZE  rcvd: 364
```

4 Next time you update a DNS record, use this exercise to confirm that your change is correct before the propagation of the change updates your local DNS resolver.

SYSTEM ADMINISTRATION

Exercise 22.4 — CONTROL APACHE

1 Starting and stopping apache in production can not require you to click on buttons since most production servers are headless (no monitor or GUI system). This exercise walks you through starting and stopping apache from the command line.

2 If you are running on a Red Hat/CentOS webserver then the commands to start and stop apache are

```
/etc/init.d/httpd start
/etc/init.d/httpd stop
```

On A Mac computer the commands are

```
apachectrl start
apachectrl stop
```

And on other Linux installations might be similar but with a different path. Try restarting apache and see that the server is restarted correctly (that is you can surf to the home page url and get a response).

3 Now make a small edit to your apache configuration file httpd.conf (often located in /etc/httpd/). Search for the line that defines which index file to serve (Search for DirectoryIndex) and change it so that wacky.html is the file served (it will normally be index.html index.php by default)

```
DirectoryIndex wacky.html
```

Restart apache and navigate to a folder URL. You should get an error (since no index file named wacky.html is found).

Create the file wacky.html with some small markup to let you know if you've reached to right file. Refresh the page (on the URL with the folder only) and you should now see the contents of wacky.html.

Exercise 22.5 – Set Up Secure HTTPS

1 This exercise requires that you have completed the generation of a self signed certificate in Lab 16 (Exercise 16.2). In that exercise you generated a X.507 certificate (server.crt) and a private key (server.key).

There may also be come challenges, since every default installation comes with different modules installed, confuguration files and other details. You will need to make use of the official Apache documentation if you get stuck:

http://httpd.apache.org/docs/2.2/mod/mod_ssl.html

2 This next steps require that you modify the apache configuration files (as root). You can either edit the httpd.conf file, or use one of the child configuration files referenced from that file.

In most cases a folder contains files with options for various extra modules like httpd-ssl.conf for the secure server.

3 Find the current listen Directive (should be "isten 80). Add (or uncomment) a second line to make the server listen on the secure port (443)

```
Listen 443
```

Turn on the SSL engine:

```
SSLEngine on
```

And make sure the mod_ssl module is enabled.

```
LoadModule ssl_module libexec/apache2/mod_ssl.so
```

Now restart apache to make the changes take effect. Your old URLs should work, but not yet the secure URLS (https).

4 This next step has you point apache to use the certificate and key you generated. Make sure you are referencing the file you created.

```
SSLCertificateFile "/private/etc/apache2/server.crt"
SSLCertificateKeyFile "/private/etc/apache2/server.key"
```

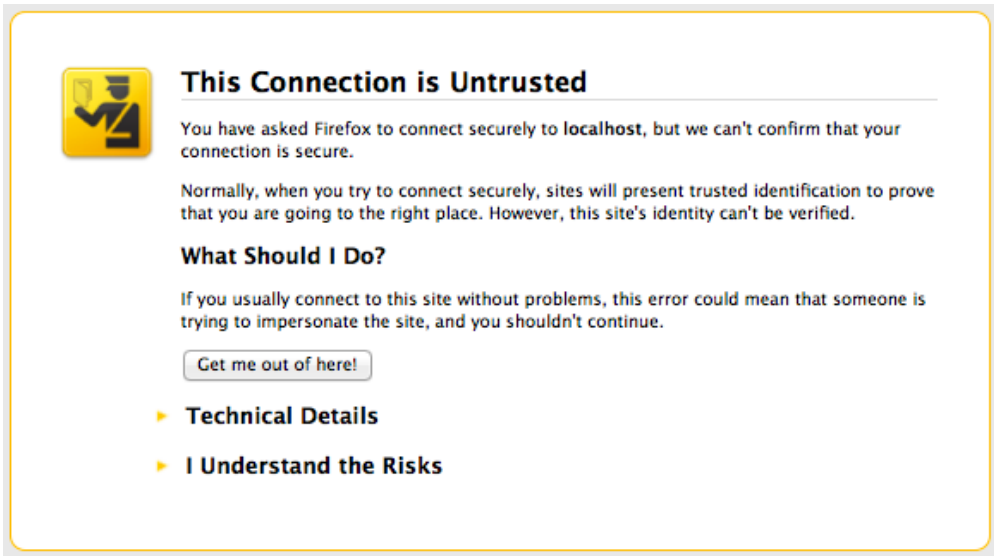
Be extra careful that your private key is not publically accessible! Again restart apache

Depending on the details of your configuration, you should be able to access a secure version of your site.

Unfortunately, many things could complicate your installation including:

- Does the certificate domain name match the domain you are using (localhost)?
- Is the mod_ssl module enabled? (It may or may not be installed)
- Are you using Virtual Hosts (see next exercise)

After overcoming any particular issues related to your setup, you should be able to load the page, and get the warning (due to a self signed certificate) as shown in Figure 22.1. Accept the exception and you can now access the secure version of your site!



This Connection is Untrusted

You have asked Firefox to connect securely to **localhost**, but we can't confirm that your connection is secure.

Normally, when you try to connect securely, sites will present trusted identification to prove that you are going to the right place. However, this site's identity can't be verified.

What Should I Do?

If you usually connect to this site without problems, this error could mean that someone is trying to impersonate the site, and you shouldn't continue.

Figure 22.1 The warning screen that shown the apache server is working!

Exercise 22.6 – Hosting 2 Domains on One IP address

1 One of the great abilities of apache is to easily manage multiple domains on the same machine, even using the same IP address.

2 Open your apache configuration file (httpd.conf) using as root.

Navigate to the bottom of the file. This is where Virtual Hosts are normally defined (some installations define virtual hosts in a separate file, Determine if this is the case and decide where you want to add your VirtualHost directives.

3 If you use name based Virtual Hosting then you can share 1 IP address between many domains. *Note: You cannot easily share 1 IP address between multiple SSl domains.*

Add a directive to tell apache you will be using Name Based Virtual Hosts. The *:80 means to listen on any IP address on port 80.

```
#
# Use name-based virtual hosting.
#
NameVirtualHost *:80
```

4 Now below define a Virtual Host using the `<VirtualHost>` container and directives inside that container.

```
#
# VirtualHost example:
# Almost any Apache directive may go into a VirtualHost container.
# The first VirtualHost section is used for all requests that do not
# match a ServerName or ServerAlias in any <VirtualHost> block.
#
<VirtualHost *:80>
    ServerAdmin webmaster@example.com
    DocumentRoot "/var/www/html/funwebdev.com/"
    ServerName funwebdev.com
    ServerAlias www.funwebdev.com
</VirtualHost>
```

Since our Virtual Host will answer requests for funwebdev.com and www.funwebdev.com the directives

```
    ServerName funwebdev.com
    ServerAlias www.funwebdev.com
```

Are used inside of the VirtualHost container. When a request arrives the desired host is looked at to dtermine which folder to serve from.

5 Now add a second virtual host, to prove to yourself you can indeed host two domains off one server. This second example will use a subdomain, although it can be a completely different domain. Place another VirtualHost container under the first one, this time with the new server name.

```
# Virtual Host for your second domain on this machine.
<VirtualHost *:80>
    ServerAdmin webmaster@example.com
    DocumentRoot "/var/www/html/examples/"
    ServerName examples.funwebdev.com
</VirtualHost>
```

The last step is to restart apache and see if you were successful. Consider if your DNS records are not updated using a /etc/hosts entry to facilitate testing.

Now if you surf to the first domain you will see files served from the folder you specified, and if you try domain two you should see files from the second folder! Note: *you can now add more domains as required, and even add VirtualHost containers for secure sites. Note: these Listen on port 443 not 80.*

Exercise 22.7 – Simple Folder Protection

1 Create a folder named topsecret off the root of your server and inside place a file named index.html that simply contains the text "you have accesses the secret file".

When you surf to domain/topsecret/

You should see the "you have accesses the secret file" message. This exercise will secure that folder using apache's ,htaccess mechanism so that you need a password to see the file.

2 On your server generate a file that wil contain a username and password for use with this secured folder. You must use the htpasswd command on the command line to generate the file. To generate a file names passwords.txt and a user named ricardo I type:

```
Htpasswd –c passwords.txt ricardo
```

You will be prompted to enter a password twice (I chose password). Looking at the file you will see that the username is plaintext but the password has been hashed as follows:

```
ricardo:$apr1$5ltx503N$28kiJOihbUTevof/WkvBM.
```

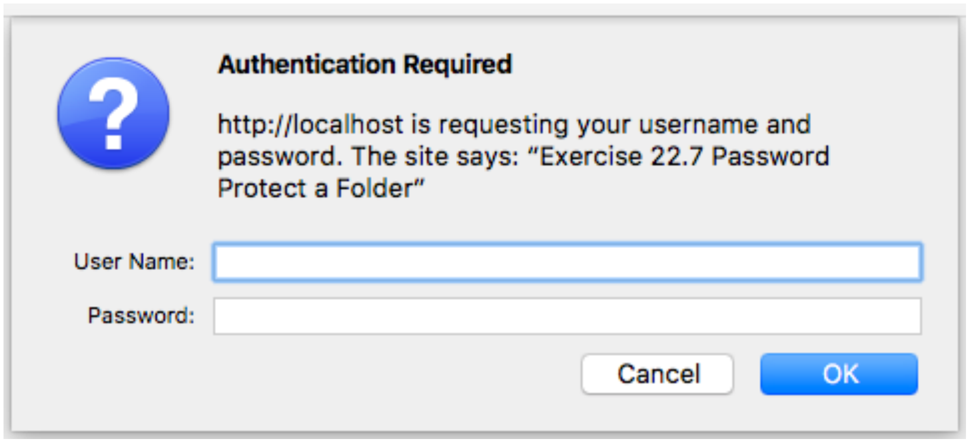
3 Now move that file to a secure place that is not accessible through the webserver. I moved mine to /var/passwords.txt

4 In the folder you want to protect create a file named .htaccess (make sure there's a dot in front of the name.)

Inside that file paste the following directives:

```
AuthUserFile /var/passwords.txt
AuthName "Exercise 22.7 Password Protect a Folder"
AuthType Basic
require valid-user
```

5 Now when you try to access the folder you should be prompted for your credentials as shown in Figure 22.2. *Note that you did not need to restart the server (one of the advantages of .htaccess files)*


Authentication Required

http://localhost is requesting your username and password. The site says: "Exercise 22.7 Password Protect a Folder"

User Name:

Password:

Figure 22.2 Screenshot of the password prompt associated with the .htaccess folder protection

Note: Many hosting platforms such as cPanel and others allow you to set these same folder levelprotection through a web interface.

Exercise 22.8 – Define Unique Logs

1 This exercise requires that you have configured VirtualHosts containers from Exercise 22.6, You will now define unique logs for each domain so that the logs are not all combined together.

2 As root, open the httpd.conf configuration file where you defined your Virtual Hosts.

Add a single **CustomLog** directive to each container defining the location for the logs as follows (changes in red).

```
#
# VirtualHost example:
# Almost any Apache directive may go into a VirtualHost container.
# The first VirtualHost section is used for all requests that do not
# match a ServerName or ServerAlias in any <VirtualHost> block.
#
<VirtualHost *:80>
    ServerAdmin webmaster@example.com
    DocumentRoot "/var/www/html/funwebdev.com/"
    ServerName funwebdev.com
    ServerAlias www.funwebdev.com
    CustomLog /var/www/logs/funwebdev_log combined
</VirtualHost>

# Virtual Host for your second domain on this machine.
<VirtualHost *:80>
    ServerAdmin webmaster@example.com
    DocumentRoot "/var/www/html/examples/"
    ServerName examples.funwebdev.com
    CustomLog /var/www/logs/examples_log combined
</VirtualHost>
```

3 Now go the location specified on the command line and touch the log files as follows:

```
touch funwebdev_log
touch examples_log
```

4 Now restart apache, and visit both domains once.

Look at the two distinct log files and confirm that they both have entries that match the pages you surfed to.