# Overview of Lab

The Apache Lab will involve the installation and customization of setting up a web server. The Web server will be Apache 2.2. The lab will deliberately deviate from some of the “norms” or standard directories and naming conventions to show that changes from usual practice and defaults are easily done.

Overall the steps will be:

1. Install Apache and test default installation
2. Create the directories needed for the Web files
3. Customize Apache for displaying the Web files. (Allow directory structure to be shown)
   1. Install a simple set of supplied html files to be browsed
   2. Following a specific directory structure
4. Change the default for case of no file name given (Disallow root directory to be shown but allow a subdirectory structure to be shown)
5. Ensure the server can be browsed from another machine (e.g. the CentOS VM)

The whole lab will be done on one lab PC using 2 VMs: one Debian and one CentOS. Apache will be installed on Debian VM, the Web Server will be browsed from the CentOS VM.

Side benefits from this lab include:

* Learning how to install packages on a Debian System
* Re-enforce the use of the VM environment
* Re-enforce use of vi editor

# Part 1: New Linux Techniques

## Create and execute a script file

Note: the instructions say to use vi as your editor. Substitute your version of vi (vim or vim-tiny) as needed.

A script file is basically a small, low level program. The commands in the script file are executed by the shell (*bash* in this lab). Scripts are handy to automate tasks. Some of the things that can be done by a script:

* make decisions (using if statements)
* loop
* call other scripts to be executed
* read and write files
* execute system commands

Useful information:

* Parameters passed to a script are denoted by $1, $2, $3, … where $1 is the 1st parameter, $2 is the 2nd, etc.
  + The number of parameters is $#
* Conditionals are done by an if…elif…else…fi structure
  + The if and elif are followed by a command that evaluates to true or false
  + The elif and else are optional
    - elif can be repeated
    - Only one else may be used, if needed
  + The if is closed with the fi statement
  + To stop a script use exit n
    - n = 0 is a normal exit
    - n = 1 is an error exit
  + If a value needs to be checked the test command is used
    - Numbers use conditionals like –gt, -lt, and -eq (greater than, less than, equal)
      * E.g. test 1 –gt 2
    - Strings use = or != (equal and not equal)
  + There are other comparisons that can be done, check the internet
* Variables are case sensitive, and are UPPER case by convention for environment variables

### Script one

The first script will be a simple aid in changing your network configuration. It will stop the NIC, open the file interfaces to be edited, and then restart the NIC after you have saved the interfaces file.

Create a file named *netconfig.sh* using your version of vi. Note that .sh is a de facto extension used for script files. The file will reside in your home directory, but since it is working with network files an id with root authority will be need to execute it.

Change to your home directory:

cd ~

Start vi and enter the following text (note: the # lines are comments):

Save and exit vi. Make sure when you exit vi you have a file named netconfig.sh.

#!/bin/bash

NETCONFIGFILE=/etc/network/interfaces

RESOLVECONF=/etc/resolv.conf

ifdown $1

vi $NETCONFIGFILE

#vi $RESOLVECONF

ifup $1

echo 'nameserver 172.16.1.245' >> $RESOLVECONF

echo 'nameserver 172.16.1.219' >> $RESOLVECONF

* The first line tells the system which shell to run (bash in this case
* The next two line are variables that are used for the fully qualified name of the interfaces and resolv.conf files
  + To use a variable in a script you put a $ in front of the name
* The forth line stops the NIC card
  + $1 is the first (and only in this case) parameter passed to the script
* The fifth line starts the vi editor to edit the interfaces file
* The sixth line is commented out. It is to edit the resolv.conf file. To edit that file uncomment that line.
* The next line restarts the NIC card
* The last two lines add the IP address of two servers in the lab to the resolv.conf file

Use *chmod* to make the file *netconfig.sh* executable:

chmod 744 netcofig.sh

This will make the file readable, writeable and executable by your user id and only readable by those in the group and rest of the world. Note that although the file is now executable by your userid, it will fail the ifdown and ifup commands since only root can do those commands.

To run the script enter: ./netconfig.sh eth0

The script expects one parameter, the name of the NIC card, in this case *eth0*. Change *eth0* if it is different on your system. That beginning period and slash is important. The ./ is the "command" (it really means to run the program from the current directory) to run the following script program *netconfig.sh* in the PWD and that program is not in your PATH. Remember the script you created has a command to start vi. The script will pause at the vi command until you close vi. When vi opens make the desired changes to the interfaces file. When done save and close. The script will then continue by starting the NIC card, and then appending the nameserver information string to the end of the resolv.conf file.

Confirm and document\*\* that the script executed properly by checking the contents of the interfaces file and the resolv.conf file. You can make any changes you wish when editing the interfaces file. Hint: you might use this script when installing Apache (the next Step in this lab). Note that when the script is executed the first time the ifdown command may generate an error if the network has not been started yet.

You will probably want to keep this script (with minor modifications) to run in later labs when you need to edit your interfaces file.

### Script two:

This script will copy one of two preconfigured interfaces templates to interfaces. The two templates are interfaces.static and interfaces.dhcp. The script will be called go.sh and will have one parameter which will be either "static" or "dhcp". When the script is run it will stop the NIC, copy the appropriate file to interfaces, then restart the NIC.

First make two copies of interfaces:

* Change to the networks file: cd /etc/network
* Create the templates:
  + cp interfaces interfaces.static
  + cp interfaces interfaces.dhcp
* Edit both templates with the values you wish to have in them

Then create the following script named *go*:

#!/bin/bash

IFILE=/etc/network/interfaces

ifdown eth0

if test $1 = "static" ; then

cp $IFILE.static $IFILE

echo "Static interfaces loaded!"

elif test $1 = "dhcp" ; then

cp $IFILE.dhcp $IFILE

echo "DHCP interfaces loaded!"

else

echo "Parameter must be static or dhcp"

fi

ifup eth0

To run the script:

* Make the script executable:
  + chmod 755 go
* Run the script:
  + ./go static
    - Will load the static
  + ./go dhcp
    - Will load the dhcp
  + ./go dog
    - Will print the error message

\*\*Document the script working and not working (bad parameter)

Here is an alternate version (*go2*) that does not use an if clause and has a second parm for the NIC. This version checks for the correct number of parameters, but does not check their value. It assumes they have been typed in correctly. Create the script, make sure it is executable, and then test it. Try it with no parameters, with the correct parameters, then with statics spelled wrong, and then finally with the wrong NIC interface, e.g. eth9.

#!/bin/bash

IFILE=/etc/network/interfaces

if test $# -ne 2 ; then

echo "Two parameters must be passed, the state and the NIC id"

exit 1

fi

ifdown $2

cp $IFILE.$1 $IFILE

ifup $2

\*\*Document the working and one of the tests with a bad parameter.

**Summary:**

Scripts can make repetitive tasks easier. Scripts by convention end in .sh, but that is for human use only. The permissions for the script must be eXecutable for the owner, group or world to run them. The scripts go and go2 have no extension and they work perfectly well.

# Part 2: Apache

## Step 1: Install Apache

The latest version of Apache is version 2.2. This package is available as an option from your previous install of the Debian OS. You will use the CLI to install.

**If needed, change the VM network configuration to use DHCP, then restart the VM (or just the NIC)! You could use the script file you created above to do this.**

## Install

Do a CLI based install of Apache: apt-get install apach**e2**

### Test to see if install was successful

Perform a quick check to see if the base Apache installation worked.

Open up the *Ice Weasel* browser:

***Applications 🡪 Internet 🡪 Iceweasel Web Browser***

Browse “yourself”: in the URL entry box enter: **localhost**, and hit enter. That request should be directed to the default installation for Apache and display a web page that says: **It Works**!

\*\*Be sure to capture the successful web page message for your lab report.

### Take a quick tour of the Web Server

Look around in the Web Server installation to see what directories and files were installed:

* Default configuration files are located at **/etc/apache2.**
* Default document root for apache2 isat **/var/www.**

\*\* Document what you found (text)

## Step2: Create directories and copy files

In this procedure you will begin to set up and customize the environment for *your* Web Server. This will consist of creating your directory structure and copying pre-existing html files into them. Later the Apache server will be customized to display these files.

The home directory for the server will be ***apachelab*** with one subdirectory ***ITIS2110***.  *Apachelab* will have two html files: *home.html* and *wf1.html*, and one image file: *constitution.jpg.* Directory *ITIS2110* will have two files: *test1.html* and *test2.html*, and two images: *Sunset.jpg* and *Winter.jpg*.

* **Open a terminal with root authority:**

***Applications -> Accessories -> Root Terminal***

* **Create the directories at the root level (/)**

**mkdir /apachelab**

**mkdir /apachelab/ITIS2110**

* **Copy the files to these directories**

**The files are available in multiple places on the *hades*.lab network. It is your responsibility to get the files in the correct directories. The Hades server may or may not be directly accessible from the Debian VM, depending on what IP address your VM is using. You might be able to use the Iceweasel browser on the Debian VM to display the files (browse them) and then use the browser to copy them to the temporary directory. The “gottcha” on this step is the target directory is not accessible (wrong permissions) by the standard user id and the browser most likely will only run from a standard user id. Don’t forget to also copy the picture files.**

**Locations:**

* **lab302-web.hades.lab (172.16.1.250)**
  + **/apachelab**

**\*\* Show your directory structure and content**

## ****Step3: Configure Apache****

Now it is time to reconfigure the default configuration of Apache to serve your files. The configuration file is called *default* and is in the */etc/apache2/sites-available* directory.

* **Open the configuration file for editing.   
  Note: it will be easiest to use vi from a root terminal since you will be changing files that your user id will not have permission to change.   
  From a root terminal:  
  *vi /etc/apache2/sites-available/default***
* Change the default document root  
  Look for the line “DocumentRoot /var/www/”. Change the directory name to the new server root base directory */apachelab/*. The line should now read: “DocumentRoot /apachelab/”.
* Allow root indexes (show directory structure):  
  Find the first <Directory… tag. Change the line following from:   
  Options FollowSymLinks  
  to:  
  Options Indexes FollowSymLinks  
  **Note:** there may be slight variations on the Options line, just ensure the Indexes word is added if need be.
* Edit the second <Directory… tag to refer to the ITIS2110 directory and remove the Redirection  
  - change the /var/www/ to /apachelab/ITIS2110   
  - If there is RedirectMatch directive put a comment mark (#) in front of it

Restart the Apache server:

There are two methods:  
***service apache2 restart***

-or-

***/etc/init.d/apache2 restart***

You will use Iceweasel again to browse the **localhost**. You should now be able to browse the new web pages. Ensure all the links work. Document your effort. Try browsing **localhost** with no file name. You should see the directory structure on the returned document. Try it also with **localhost/ITIS2110** as the URL. You should see another directory structure for that directory.

\*\*Again, document your effort.

## Step4: Change access to forbidden for the root directory

In Step 4 you will make browsing the directory structure for the root “forbidden”.

* **Forbidden directory browser for** document root **“/”**

**Look for this line “Options Indexes FollowSymLinks” and delete “Indexes”**

* **Allow directory browsing for sub directory “/apachelab/ITIS2110”**

**The previous command forbids directory listings for root and all it’s subdirectories. Now we want to allow listings for the subdirectory ITIS2110 and its subdirectories. Double-check that the Directory tag for the ITIS2110 has the Options Indexes somewhere in it. E.g.**

**<Directory /apachelab/ITIS2110>**

**Options Indexes FollowSymLinks Multiviews**

**</Directory>**

There may be other values with the Options.

* **Restart the apache server**

***/etc/init.d/apache2 restart***

* Test the configuration

Applications -> Internet -> Iceweasel Web Browser

* Use the URL **localhost** again to browse the Web server root directory. You should see a **Forbidden 403** message pop up.
* Try again with the URL ***localhost/ITIS2110***, you should see the index of the directory

\*\* Document your results for both the blocked and unblocked browses

## Step 5: Ensure the Web Server is accessible by other clients

In this step you will ensure that the Web Browser works from another environment (e.g. workstation). To do this, the Debian VM will be used as the Web Server and another VM, e.g. CentOS, will be used as the client to do the browsing.

* On Debian, use ifconfig or similar tools to find the IP address of the servers. Assume it is 192.168.1.123 in this case.
* Start another VM with CentOS as the OS.
  + Start a browser in the CentOS VM (Ice Weasel or such).
  + Use the browser to open the Debian Web page
  + Enter the Debian IP address in the CentOS URL box
    - **192.168.1.123** will be used as the example in this case
  + You should get the Forbidden message.
  + Change the URL to **192.168.1.123/home.html**
  + You should see the home web page.
    - Use the links on the home page to ensure all four web pages can be navigated.

\*\*Document your results.

# Deliverables:

A well written lab report. The lab report should have an overview and a summary as well as documenting the work done and results (note the elements marked with a \*\*), including the minimal following details:

## Part 1

Documentation of \*\* items

## Part 2

**Listing of the Apache “*default*” file after install**

Screen shot of the browser after visiting “***localhost”*** (default install data)

### Step 2

Listing of the affected directories and contents

### Step3

**Listing of the Apache “*default*” file after Step 3 changes**

**Screen shots of the web browser**

* **After visiting “*localhost*”**
* After visiting “***localhost/itis2110”***

### ****Step 4****

**Listing of the Apache “*default*” file after Step 4 changes**

**Screen shot of the web browser**

* **After visiting “*localhost***” after forbidden directory browser for document root (shows Forbidden message)
* After visiting “***localhost/itis2110”*** (shows no forbidden message, just the directory)

### Step 5

**Screen shot for web browser from another VM (CentOS)**

* **Visiting “*debian\_ vm\_ ip*”**
* Visiting “***debian\_ vm\_ ip /itis2110”***

**Reminder:** documenting text data with screenshots will result in point deductions