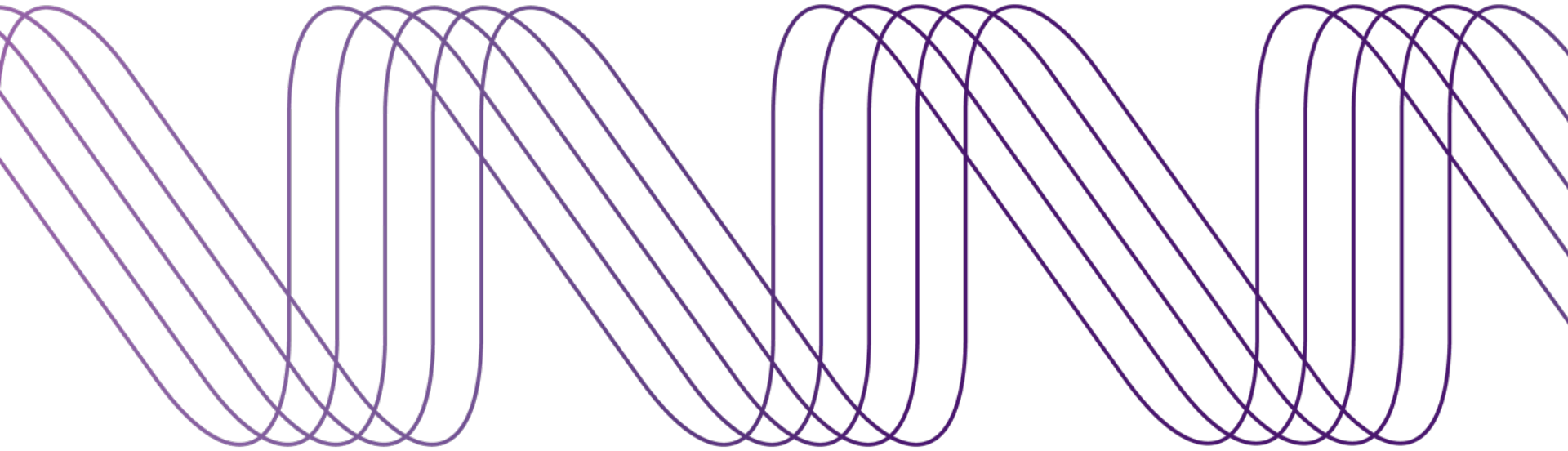
JISC netskills

# Coursework Review

# 1. Apache Quick Build

All about getting the first and most important component working...



# Objectives

- Access and download the latest version of the Apache source code Decompress and unpack the source code on your VM
- Configure the *makekefile* for Apache with specific additional options
- Use `make` to build and then install a working Apache 2 web server
- Make some common run-time configuration changes to Apache
- Re-compile Apache to add a new static module into the basic installation

# Assessed outcomes

- There should be a live, working webserver at:

  `http://vm-eliot-NNN.ncl.ac.uk`

- The server should be delivering the specified (and edited) index page from

  `/usr/local/www/index.html`

- The server-info module should be compiled into the web server and delivering its output at:

  `http://vm-eliot-NNN.ncl.ac.uk/my-server-information`

- The server configuration should be updated with the correct values for `ServerName` and `ServerAdmin`

# Key learning

- How to download, save and unpack files from the web using Linux command line tools (`lynx`, `gzip`, `tar`)
- How and when to run commands with the root user's permission set using `sudo`
- How to use a standard Linux source-code software installation technique basked on `make`
  - Run a configuration script (`configure`) to inspect the local system for the required dependencies and prepare a `makefile` to instruct the build process
  - Use `make` to read the `makefile` and construct the binaries and other files which make up a software application
  - Use `make install` to move the finished software into position
- How to use `config.nice` to make recompiling Apache easier

JISC netskills

# Key learning

- How to manually stop, start and restart an Apache web server
  `/usr/local/apache2/bin/apachectl start|stop|restart`
- The location of the core run-time configuration files for Apache
  `/usr/local/apache2/conf`
- The location of the core executable program files for Apache
  `/usr/local/apache2/bin`
- How to check for presence/absence of compiled (static) modules
  `/usr/local/apache2/bin/httpd -l`

JISC netskills

# Key learning

- How to edit the core Apache configuration file `httpd.conf`
- The use of directives to control Apache behaviour

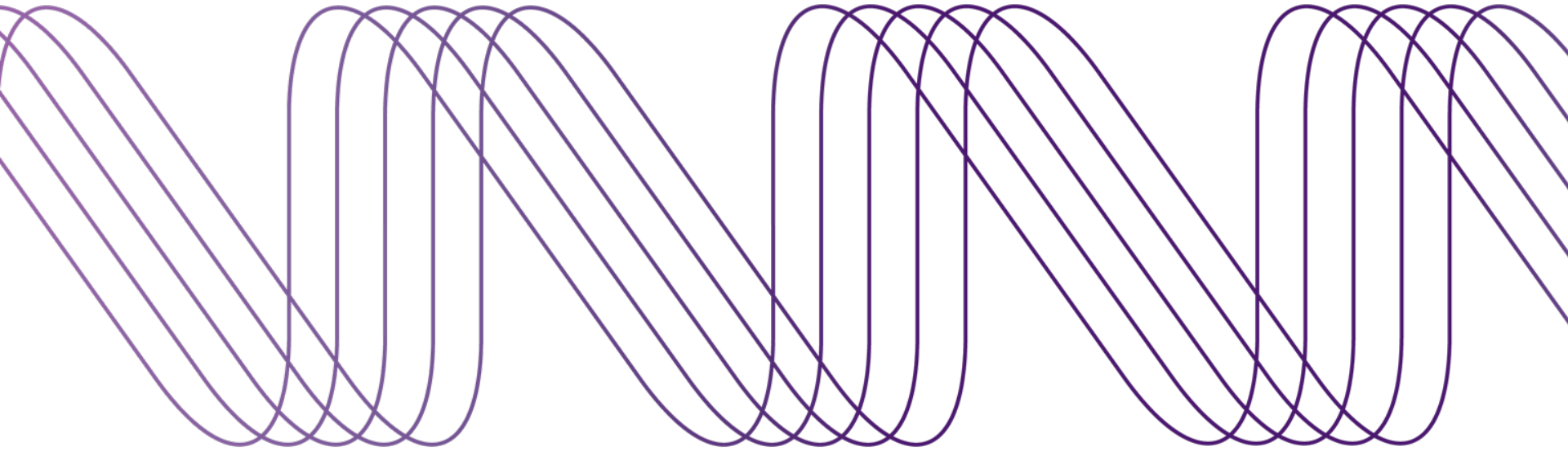
  `SomeDirective somevalue` e.g. `ServerName vm-eliot-001`
- How to use a simple configuration container to control the context (how, when and to what) Apache configuration is used

```
<Location /my-server-information>
  SetHandler server-info
</Location>
```

- The importance of restarting Apache to read updated config!

# 2. Configuring Apache (pts1-4)

How to use a few of the most commonly exploited features of Apache...Divided into 4 parts – each independent of the other



# Objectives (pt1)

- Use the tools supplied with Apache to create a username and password to restrict access to part of your site
- Create a new content area to be restricted by *password authentication*
- Add correct configuration to allow Apache to request a username/password combination and authenticate it before allowing access
- Successfully add a provided username/password combination to your password file

# Objectives (pt1)

- Create a second content area for your web site
- Use a *distributed configuration file* to prevent web clients from a specific IP address from accessing this content area
- Add correct configuration to allow Apache to read the distributed configuration file , process the directives it contains and (if necessary) allow them to override previously declared configuration

# Assessed outcomes (pt1)

- Clients should be presented with Apache's username and password prompt when attempting to access

  `http://vm-eliot-NNN.ncl.ac.uk/secure-area`

- Access to the content should be granted to the user **bob** using the password provided to you in hashed password file format

  `bob:umDolxdRNr4aY`

- The configuration for this behaviour should be in a `<Location>` block in `httpd.conf`

```
<Location /secure-area>
  AuthType             basic
  AuthName             "Login to access VIP area"
  AuthBasicProvider    file
  AuthUserFile         /usr/local/apache2/conf/users
  Require              valid-user
</Location>
```

JISC netskills

# Assessed outcomes (pt1)

- A web client connecting from `128.240.148.134` should be denied access to (and receive a `401` response from)

  `http://vm-eliot-NNN.ncl.ac.uk/restricted-area`

- The IP restriction should be in a working `.htaccess` file at

  `/usr/local/www/restricted-area/.htaccess`

- The config to allow `.htaccess` files should be in a `<Directory>` block in `httpd.conf` and be as minimal as possible

```
Deny from 128.240.148.134
```

```
<Directory /usr/local/www/restricted-area>
  AllowOverride Limit
</Directory>
```

# Key learning (pt1)

- How to create and add extra content areas to a web site
- How Apache configuration containers are used to apply local context to one or more directives
- The difference between `<Location>` and `<Directory>` containers
- How to create and use simple password files to store user information
- How to manually add users to an Apache password file
- How to configure Apache to check a password file for authenticated user information before continuing to process am incoming request

# Key learning (pt1)

- How to configure Apache to allow directives from distributed configuration files (`.htaccess`) to be applied
- That distributed configuration is only read *on-access*, not at server start-up (and that errors in a `.htaccess` file cause an HTTP 500!)
- How the `AllowOverride` directive can be used to restrict the directive types allowed in a `.htaccess` file

# Objectives (pt2)

- To add support for SSL over HTTP (HTTPS) to allow encryption of requests and responses to the web server
- To use OpenSSL to create a private key and self-signed server certificate
- To configure Apache to use a second VirtualHost to listen for and respond to encrypted communication on port 443

JISC netskills

# Assessed outcomes (pt2)

- A working server delivering the *same* content at both

  `http://vm-eliot-NNN.ncl.ac.uk` and

  `https://vm-eliot-NNN.ncl.ac.uk`
- Clients accessing over HTTPS should be presented with a self-signed server certificate containing the specified information (location, validity etc.)

JISC netskills

# Key learning (pt2)

- How to use OpenSSL to create a private key and self-signed certificate
- How to include core configuration at server-start-up from files linked to `httpd.conf`
- Content being delivered over HTTPS is handled by a second *virtual host* which could be configured differently to the main server if required
- The use of the `Listen` directive to specify which network ports Apache should respond to
- How web clients react when presented with potentially unsafe server certificates

# Objectives (pt3)

- Configure and use Analog to display information from an Apache server log
- Deliver the processed log file at a specific URL

# Assessed outcomes (pt3)

- There should be an Analog output page at

  `http://vm-eliot-NNN.ncl.ac.uk/my-server-log`
- The report be configured as specified and should contain:
  - DNS resolved host names for clients accessing your web site
  - No hits in the host report for the marking server
- The images used to create the bar charts etc. should be present and working

# Key learning (pt3)

- How to use a command line tool to analyse a log file
- The type of information Apache stores in a "combined" format log file
- The effect of deleting/renaming a log file and restarting the server

# Objectives (pt4)

- Create three customized error pages
- Update the core web server configuration to use the customized error pages in place of the built-in server defaults for 401, 403 and 404 HTTP status responses
- Ensure that the correct pages are being delivered as a result of the correct response.
- Create a favicon to be used by web clients to identify the tabs, windows and bookmarks for your site

# Assessed outcomes (pt4)

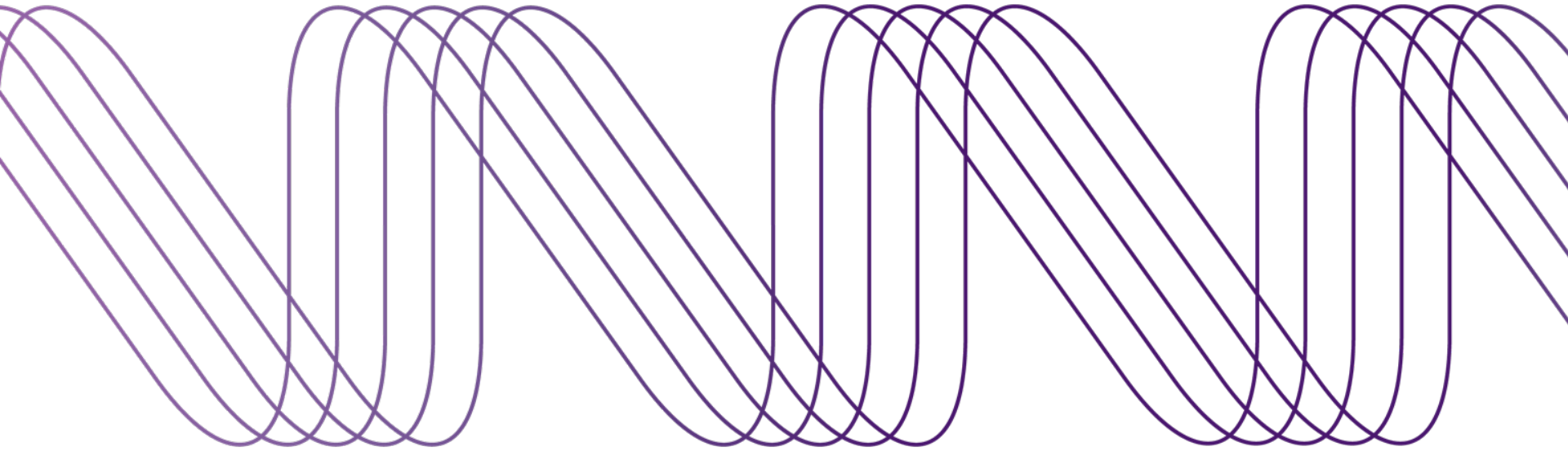
- The server core configuration should be updated to use the custom error pages
- Client requests resulting in 401, 403 and 404 HTTP server responses should return the custom error and not:
  - The server default page
  - An additional error code (e.g. a "404 for a 404")
- The error pages should be directly accessible from

  `http://vm-eliot-NNN.ncl.ac.uk/401.html` etc.
- A working favicon image should be seen in tabbed windows containing pages from your web site. It should be directly accessible from

  `http://vm-eliot-NNN.ncl.ac.uk/favicon.ico`

# 3. Server Side Includes

A bit of common Apache functionality
Be clear about the context(s) in which configuration is permitted and created



# Objectives

- Use `.htaccess` to enable SSI functionality by file extension in a specific part of your web site
- Use conditional statements in SSI (xSSI) to deliver additional information to clients requesting from specific IP-based locations
- Enable SSI by file permission in a site-wide context
- Use an "include within and include"
- Allow Apache to deliver different files as the "index" page for a directory

# Assessed outcomes

- Any client should receive a working SSI-driven page containing the current date, in the specified format, when accessing:

  `http://vm-eliot-NNN.ncl.ac.uk/includes`

- A client accessing from a `128.240.*` IP address should receive specific additional content in particular:
  - The file system path to the current page
  - The User Agent (browser) information provided by the client
- Any client should receive a working SSI-driven page containing the current date/time and last modified date/time from

  `http://vm-eliot-NNN.ncl.ac.uk/index.html`

- The core server configuration has been appropriately updated

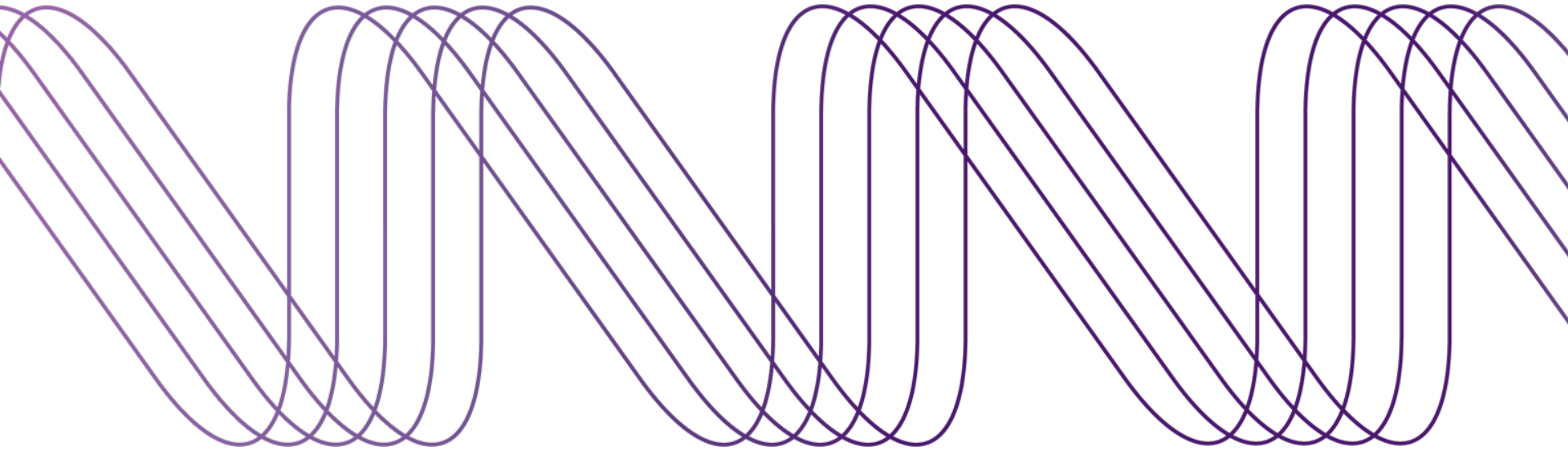
JISC netskills

# Key learning

- How SSI can be used to perform simple site-automation tasks
- The "environment variables" Apache uses and the type of information they hold
- The relationship between Apache configuration and the context in which it should be applied
- How to set configuration directives to apply across the whole site.... and how the `AllowOverride` should be used sparingly, in context and only permit the functions required
  - i.e. not `AllowOverride All` across the whole web site!
- How to use `DirectoryIndex` to control which documents the server will return as the "index" page for a directory

# 4. Installing PHP

Add dynamic processing capability using external scripts/programs and provide the hooks for MySQL



# Objectives

- Re-compile Apache to add support for Dynamic Shared Objects (DSO) modules
- Access and download the latest version of the PHP source code Decompress and unpack the source code on your VM
- Configure the *makekefile* for PHPwith specific additional options
- Use `make` to build and then install a working instance of PHP and the DSO Apache module `mod_php`
- Make some run-time configuration changes to Apache to allow specific files to be processed by PHP
- Build PHP with support for connecting to and querying MySQL database servers

# Assessed outcomes

- The standard output from `phpinfo()` should be delivered at:

  `http://vm-eliot-NNN.ncl.ac.uk/php/`

- A suitable PHP configuration file (`php.ini`) should be loaded at run time and must specify:
  - Display of run-time errors
  - Use of short opening tags in PHP scripts
  - The correct time zone for the server
- A PHP script should run and access the MySQL server on `vm-eliot-000` and return a page as specified, containing data pulled dynamically from a database, when a client accesses:

  `http://vm-eliot-NNN.ncl.ac.uk/php/dbtest/`

# Key learning

- Many Linux programs share the same source compilation and installation routine
- Apache needs to be compiled with support for DSO modules in order to use them
- DSO modules built using the tools installed as part of Apache (`apxs`) and are loaded dynamically at server start-up (and what happens if they are missing!)
- It is PHP *not* Apache that needs MySQL support in a basic LAMP stack
- The MySQL support in PHP enables it to act as a client to connect to MySQL servers

www.netskills.ac.uk

JISC netskills

JISC Netskills is a JISC Advance service hosted at Newcastle University