DAY 3!

Indexing, Authentication, Security and Performance

INDEXING

- Indexing is the process whereby Apache will dynamically produce a listing of the contents of a directory with `mod_autoindex`.
- To turn indexing on, the `Indexes` option must be present on the directory:
  - `Options +Indexes`

INDEXING

- There are two forms of indexing: basic and fancy.
- Basic is the default mode of indexing, creating a simple, alphabetized listing of the directory contents, with the names being links to the respective entries.
- Fancy indexing is enabled by adding the `FancyIndexing` option to the `IndexOptions` directive.
- With fancy indexing, the user can click column headers to re-sort the listing for any particular column.

ICONS

- When `mod_autoindex` creates the html for the directory listing, it will automatically use icons to help identify the types of various known files.

- This behavior is controlled and modified via the `AddIcon*` directives.

- Enabling, disabling and customizing indexing is a common task for the Apache administrator. The `mod_autoindex` documentation provides all the necessary information.

LAB

1. Create a few files and folders under the DocumentRoot for `wwwx`. Add some fake (empty) `.jpg, .bmp, .zip` and a `core` file.

2. Enable basic indexing for this directory (probably already enabled, but to practice, create a new Directory directive and explicitly set the Options)

3. Turn on Fancy Indexing. Figure out how to get the Index to always say “Apache ROCKS!” at the top of the listing.

AUTHENTICATION

- Gaining access to a website implementing access control generally involves two steps:
  - The authentication step, where Apache positively identifies the user as who they say they are.
  - The authorization step, where Apache decides if the identified user has sufficient rights to access the content being requested.

AUTHENTICATION

- As you can imagine, all of this is implemented via several modules.
- First, a discussion on Authentication types:
  - `mod_auth_basic`
  - `mod_auth_digest`

AUTH BASIC

- `mod_auth_basic` implements authentication using standard basic HTTP auth mechanisms.

- The HTTP protocol includes support for obtaining a username and password and returning it to the server in the request.

- This is what `mod_auth_basic` implements.

AUTH DIGEST

- `mod_auth_digest` implements a more secure form of authentication using an extension to HTTP, allowing authentication to occur without a password passing the network in plaintext.

- The net effect is the same - a correct username and password successfully identifies the user, allowing Apache to consider authorization.

AUTH PROVIDERS

- `mod_auth_basic` and `mod_auth_digest` implement the authentication type, allowing Apache to query and receive a username and password challenge response (fancy way of saying credentials). :)
- These modules do not check the username and password.
- That is handled via an authentication provider.
- Apache ships with 7 built-in auth providers, and more can be added as third party modules.

AUTH PROVIDERS

- **`mod_authn_alias`** Allows for creation of provider aliases
- **`mod_authn_anon`** Allows for anonymous authentication
- **`mod_authn_dbd`** Database authentication
- **`mod_authn_dbm`** DBM password file authentication
- **`mod_authn_default`** Always rejects (use as fallback)
- **`mod_authn_file`** Plain text password file
- **`mod_authnz_ldap`** LDAP authentication

AUTH PROVIDERS

- Now that a user has been *identified*, the authorization step can occur.

- In Apache, authorization is handled via one or more of the 6 included authorization modules, or a third party add-on.

AUTHORIZATION MODULES

- **`mod_authnz_ldap`** Queries LDAP Database
- **`mod_authz_dbm`** Queries DBM database
- **`mod_authz_default`** Always rejects (use as fallback)
- **`mod_authz_groupfile`** Uses plaintext group information
- **`mod_authz_owner`** Authorize by filesystem owners
- **`mod_authz_user`** Authorize simply by username

LAB

1. Using the documentation as a guide, set up basic HTTP authentication for `wwwX`. Use the file authentication provider and the user authorization module.

2. Clone the authorization information for wwwX into the configuration for vhostX. Switch the authentication module to digest instead of basic.

SECURITY

- There are many facets to web server security.
  - Operating system security
  - Attack vectors (server, cgi, module, etc)
  - Private content exposure
- Keeping Apache and Operating System patched, up to date, fire-walled and appropriately administered goes a long way towards security, and is all we have time to say in this class.

ACCESS CONTROLS

- Apache provides several mechanisms to control access to content:
  - Remote host
  - Environment variables
  - mod_rewrite.. bum bum BUMMMMMM!

REMOTE HOST ACCESS CONTROLS

- There are three directives supporting remote host access control:
  - `Order by`
  - `Allow from`
  - `Deny from`

REMOTE HOST ACCESS CONTROLS

- If `Allow` and `Deny` directives are mixed, then a keen understanding of the `Order by` directive is required to understand Apache's behavior.

- Or, things can be simplified by deciding on a default deny or default allow policy, and then only using the appropriate `Allow/Deny` directive to create the exceptions.

- First, syntax...

HOST ACL SYNTAX

- The allow and deny directives take the form:
  - `allow from <host|network|ALL>`
  - `deny from <host|network|ALL>`
- The order directive controls the access control behavior:
  - `order allow,deny` — Allows explicitly allowed clients and **denies everyone else**. Anyone matching both the allow and deny are denied.
  - `order deny,allow` — Denies explicitly denied clients and **allows everyone else**. Anyone matching both the allow and deny are allowed.

BEST PRACTICE

- As you can see, if you go with:

  `Order allow,deny`

  This will create a default deny, and exceptions can be listed in `Allow From` directives. Adding `Deny From` directives only muddies the water.

- The same can be said in reverse for `Order deny,allow`

WHERE CAN THE TAGS BE USED?

- These access control directive are applied through a per-Directory or per-File basis.

- The `allow`, `deny` and `order` directives are placed inside one of the following tags:

    - `<Directory>`

    - `<File>`

ENVIRONMENT ACLS

- Access control can be handled via environment variables as well, though it's not as common:

```
SetEnvIf User-Agent BadBot GoAway=1

Order deny,allow

Deny from env=GoAway
```

MOD_REWRITE ACLS

- Access control can even be provided by `mod_rewrite`!
- Using the `F` flag on a rule, which generates a 403 forbidden response when the rule matches, arbitrary access controls can be created.
- See the documentation for details

LAB

1. Using Apache access controls, configure you virtual hosts such that one of your neighbors can get to `wwwX`, and another can't.

2. Also using Apache access controls, configure `vhostX` to only be accessible by your machine and one neighbor.

3. Test all possible scenarios.

PERFORMANCE TUNING

- Performance tuning an Apache server involves benchmarking the existing setup, adjusting and tuning some configuration values, and benchmarking again to determine improvements.

- Apache includes an excellent benchmarking tool known as ‘ab’ - Apache Benchmark.

BENCHMARKS

- A benchmark is a specific measure of performance, taken in a repeatable fashion such that outside influences are minimized and operational characteristics of the machine and operating system are matched for every measurement.
  - In other words, every time a benchmark is taken, it's taken in the same manner and under the same conditions
  - This allows for meaningful comparison of benchmarks

BENCHMARK BEST PRACTICES

- If possible, shut down all services that won't be needed
- Run the benchmark at least 5 times in a row, and average the results. For better accuracy, run the test 10-20 times in a row and throw out the top and bottom 10% metrics. Then average the resultant set.
- Document everything! Conditions, commands, sequences, timing, every individual result and how the final benchmark was calculated.

TUNING

- As far as tuning is concerned, that rabbit hole goes on nearly forever.. Consider:
  - Hardware, operating system, Apache version, loaded modules, configuration
- Further, consider:
  - Network capabilities, dynamic vs static content, database latencies, network congestion
- Tuning involves everything listed above, and more!

TUNING

- As far as Apache configuration tuning is concerned, there are nearly limitless options. To name a few:
  - Consider removing unused modules
  - Simplify/remove mod_rewrite use
  - Tune MPM parameters to match hardware/os capabilities
  - Implement dynamic content with modules instead of plain CGI

LAB

1. Use ab to benchmark wwwX. Try various concurrent and total request counts, and observe the results.

2. Use ab and the same concurrent/total parameters to test your cgi script, myscript. Notice a drop in performance? Is it significant? Theorize as to reasons for the observed behavior.

3. Bonus: Adjust MPM parameters and observe ab output changes. Try a minimized configuration, say with only 10 servers running max, or a 0 spare server configuration. Then try a much larger configuration - is there an upper limit on performance improvements?

```
slideshow.end();
```