Technology Transfer Partners

APACHE HTTP SERVER PROJECT

An Introduction to Apache Web Server

Joe Doupnik
jrd@netlab1.net
jdoupnik@microfocus.com
Mindworksuk and Micro Focus
Prof (ret.) Univ of Oxford

UNIVERSITY OF OXFORD

MICRO FOCUS

An overview to start things

Apache web server configuration files for SUSE Linux are in several spots:

/etc/sysconfig/apache2	file for core configuration, also used by YaST menu
/etc/apache2/*	the many main Apache config files
/usr/share/apache2/*	docs about Apache
/var/log/apache2/*	log files
/usr/lib64/apache2*	module binaries (many, for different MPMs)

MF OES adds its supplemental files located in /etc/opt/novell/httpd/*

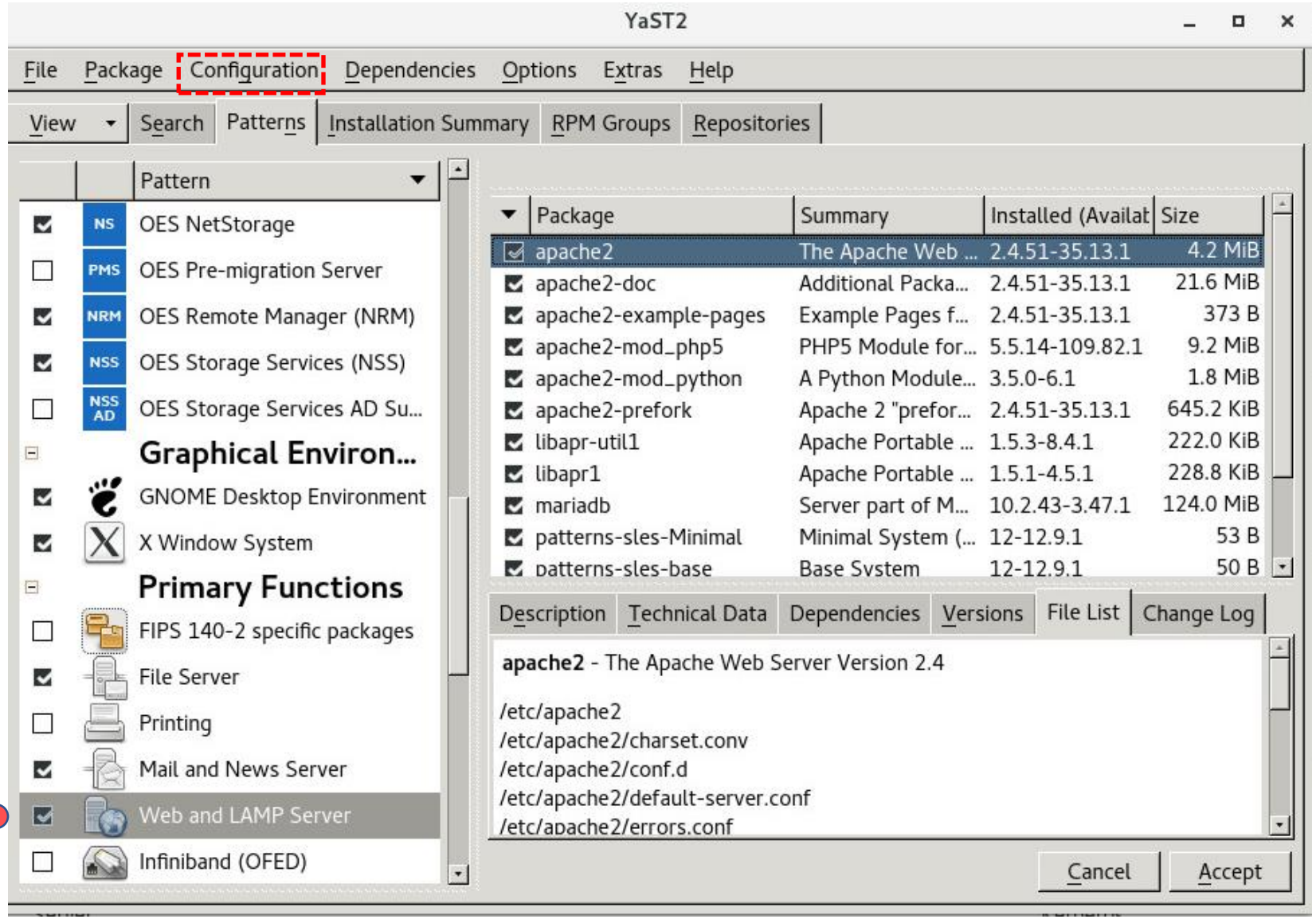
The manual is on httpd.apache.org, and optionally as local URL /manual.

Most of our changes are in /etc/apache2 subdirectories, particularly in *conf.d* where most application web config files reside. The web server is a core plus many many modules which do the interesting work.

Apache runs as user **wwwrun** in group **www**, as defined in file */etc/apache2/uid.conf*.

Providing access to the file system is a dual level process (Apache itself and the file system), as we discuss shortly.

YaST, Software Management to select Apache

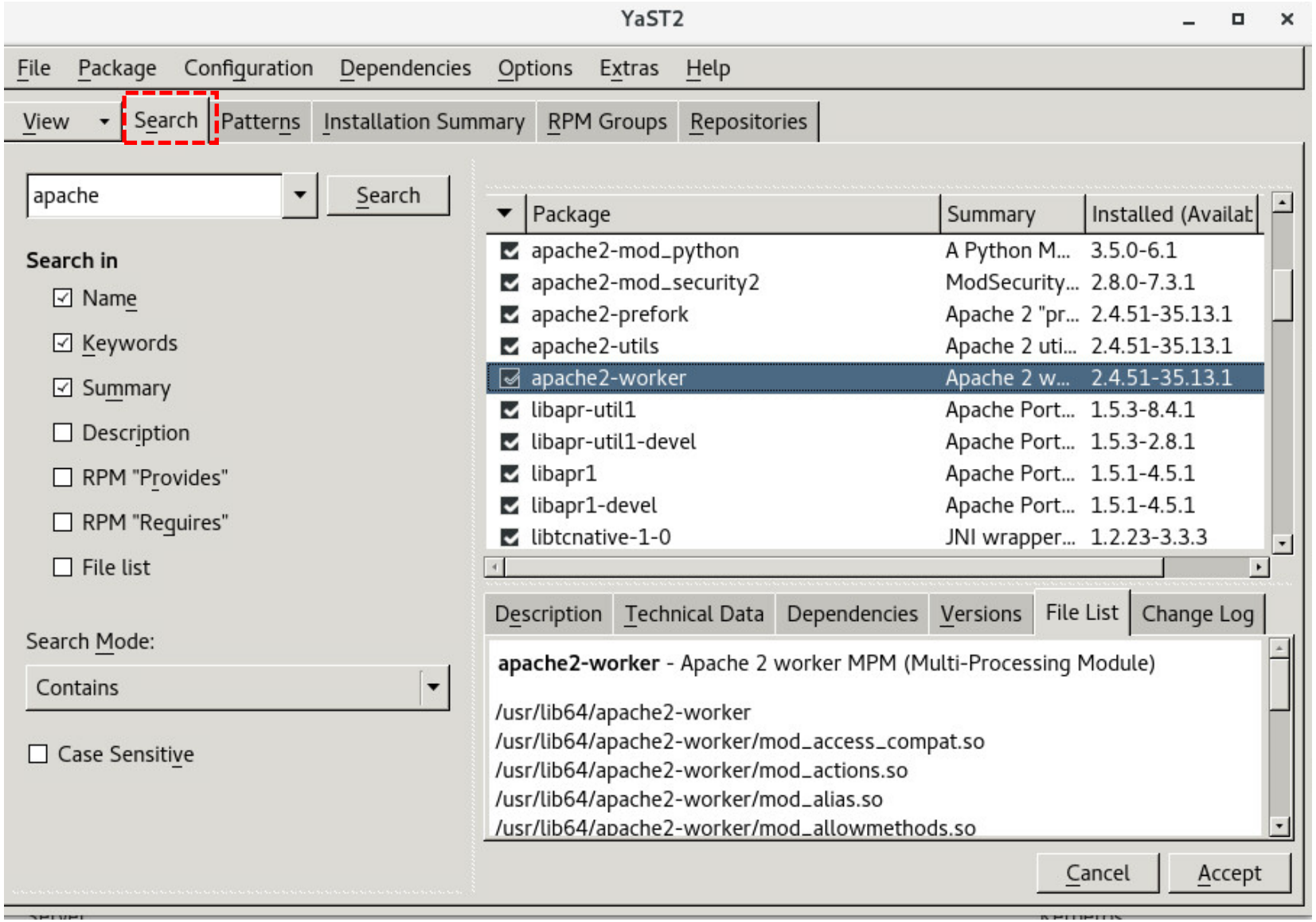


The first step is use Patterns for Apache core to install some parts.

Further steps are needed to activate and configure the result.

This example is with MF OES2018 atop SLES 12.

YaST, Apache module selection

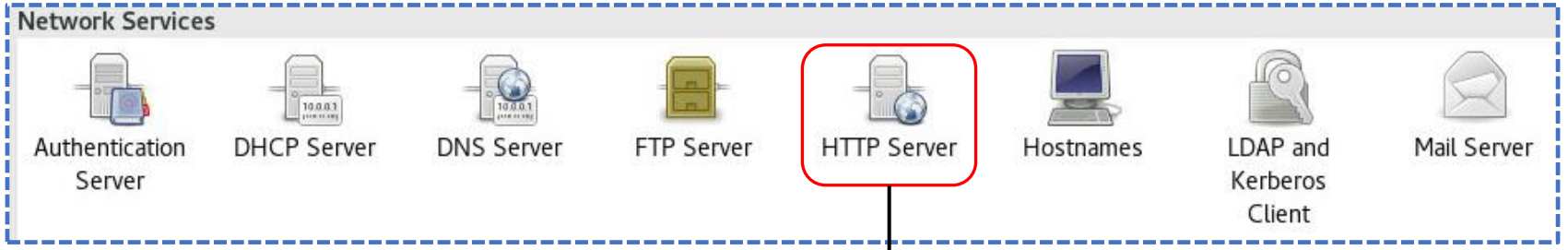


Over time add modules and features from the distribution sources.

Modules complied manually (from the net) will not appear here unless made as an RPM. However, their run code may be stored with regular modules.

Some Apache configuration scenic locations

YaST:



File /etc/sysconfig/apache2, some files in /etc/apache2/

Apache main configuration files live in /etc/apache2:

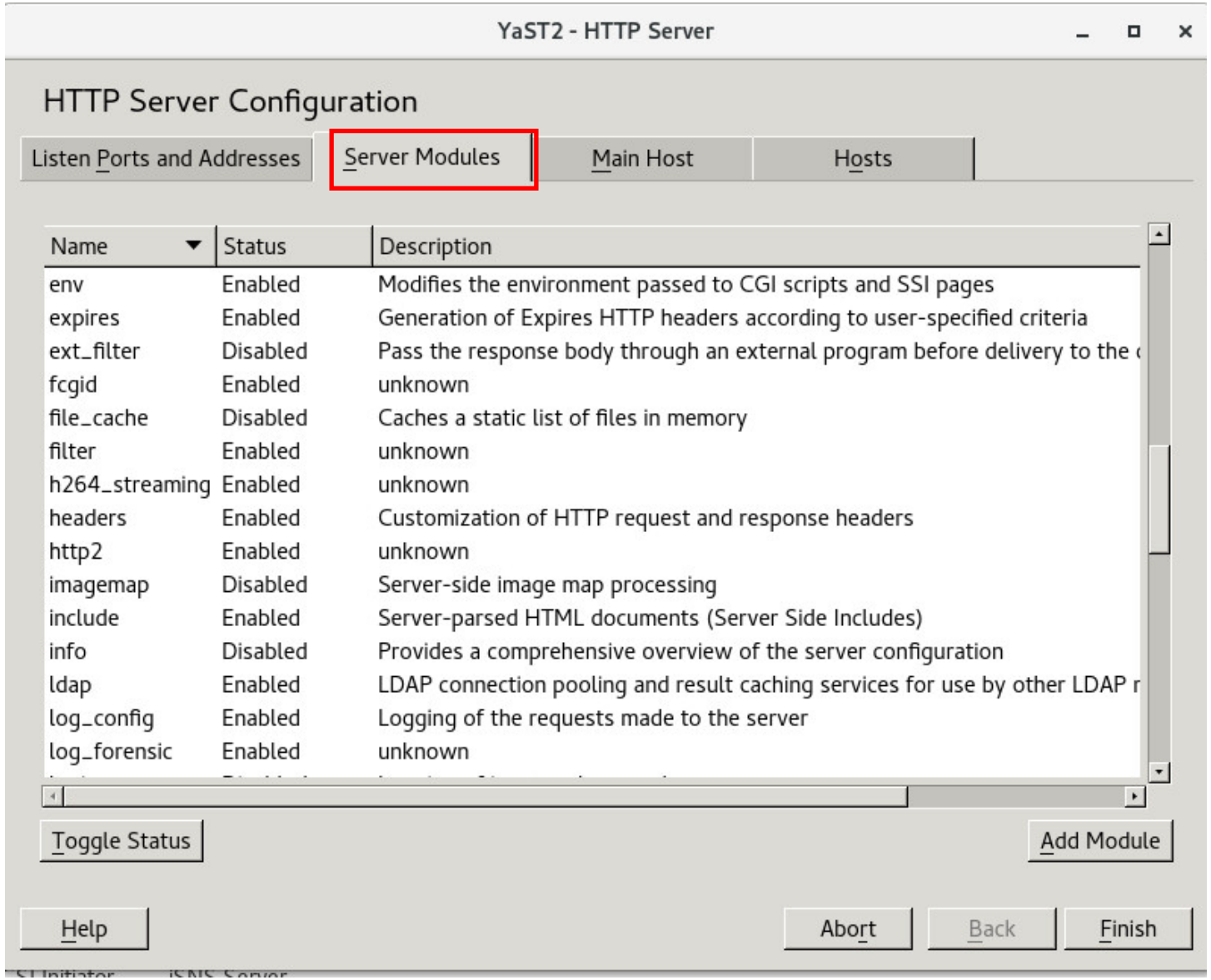
```
# ls /etc/apache2
charset.conv            mime.types                    mod_status.conf          ssl.csr
conf.d                  mod_autoindex-defaults.conf   mod_userdir.conf         ssl.key
default-server.conf     mod_cgid-timeout.conf         mod_usertrack.conf       ssl.prm
errors.conf             mod_info.conf                 server-tuning.conf       sysconfig.d
global.conf             mod_log_config.conf           smt-mod_perl-startup.pl  uid.conf
httpd.conf              mod_mime-defaults.conf        ssl-global.conf          vhosts.d
listen.conf             mod_perl-startup.pl           ssl-global.conf.rpmnew
loadmodule.conf         mod_reqtimeout.conf           ssl.crl
magic                   mod_security2.d               ssl.crt
```

Lots here → conf.d, default-server.conf, sysconfig.d, vhosts.d

Module binaries reside in /usr/lib64/apache2*

Apache startup script *start_apache2* is in /usr/sbin

YaST HTTP Server, enabling modules, sort of



This menu modifies file */etc/apache2/sysconfig.d/load_modules.conf*. I find this YaST HTTP GUI to be insufficient.

I activate a module by manually inserting its short name in list APACHE_MODULES= in text file */etc/sysconfig/apache* and restart Apache. Configs will do the rest.

Wise managers double check these files after changes. See also utility *apachectl –h* for checks.

Commentary in file /etc/sysconfig/apache2

```
#
# [It might look silly to not simply edit httpd.conf for the LoadModule statements.
# However, since the LoadModule statements might need an absolute path to the modules,
# switching between MPMs can be quite a hassle. It's easier to just give the names here.]
#
# * list of all modules shipped with the base distribution:
#
#    actions alias asis auth_basic auth_digest authn_alias authn_anon
#    authn_dbd authn_dbm authn_default authn_file authnz_ldap authz_dbm
#    authz_default authz_groupfile authz_host authz_owner authz_user
#    autoindex bucketeer cache case_filter case_filter_in cern_meta cgi
#    charset_lite dav dav_fs dav_lock dbd deflate dir disk_cache dumpio
#    echo env expires ext_filter file_cache filter headers ident imagemap
#    include info ldap log_config log_forensic logio mem_cache mime mime_magic
#    negotiation optional_fn_export optional_fn_import optional_hook_export
#    optional_hook_import proxy proxy_ajp proxy_balancer proxy_connect
#    proxy_ftp proxy_http proxy_wstunnel reqtimeout rewrite setenvif speling
#    ssl status substitute suexec unique_id userdir usertrack version
#    vhost_alias
#
#   see http://httpd.apache.org/docs-2.2/mod/ !
#
# * It pays to use IfDefine statements... like
#    <IfModule mod_xyz.c>
#        ....
#    </IfModule>
#
# * In the APACHE_MODULES variable, you can use mod_xyz or just xyz syntax.
#   You may also name an absolute path if you like.
```

Cover words for not creating a straight forward solution in the first place.

We should edit */etc/sysconfig/apache2*

Hand editing can do a much better job than the present YaST. Manually edit file *apache2* to define the list and order of Apache modules to load, and tend other basic admin details.

An example entry in text file /etc/sysconfig/apache2/:

```
# apache's default installation
# APACHE_MODULES="authz_host actions alias asis auth autoindex cgi dir imap include log_config
mime negotiation setenvif status userdir"

# your settings
APACHE_MODULES="actions alias ssl auth_basic authn_core authn_dbm authn_file authnz_ldap
authz_core authz_groupfile authz_host authz_user autoindex cgi dav dav_fs dav_lock deflate  dir
env expires fcgid filter h264_streaming headers http2 include ldap log_config log_forensic mime
negotiation perl proxy proxy_ajp proxy_http reqtimeout rewrite setenvif socache_shmcb speling
status substitute suexec unique_id userdir"
```

Yes, that's a lot of "stuff", needed to handle my many situations. We add items as needs arise.

Partial decoding of how modules are enabled

- SUSE's preferred agent is use YaST, item HTTP Server. It has menus which know only simple patterns which can/do conflict with our manual settings. We have a problem.
- Thus I prefer manual editing of configuration files.
- A puzzling part is selecting Apache modules for loading. YaST | HTTP finds module binaries in */usr/lib64/apache2* and in /usr/lib64/*apache2-*.* YaST offers a menu of modules to enable. Those which are enabled are listed by short name in item APACHE_MODULES= in text file */etc/sysconfig/apache2, a*nd by full name in file */etc/apache2/sysconfig.d/loadmodule.conf*. YaST screens may not show all modules.
- What works for me is manually placing a module's short name into primary list *APACHE_MODULES=* in file ***/etc/sysconfig/apache2***. Order is important there. Then restart Apache to have config menus executed. Lastly review Apache startup list */etc/apache2/sysconfig.d/loadmodule.conf* to verify the module is listed and in your stated order. Don't edit this *loadmodule.conf* file; edit */etc/sysconfig/apache2*.
- Check via command apachectl -t -D DUMP_MODULES and review logs for problems.

Comments on YaST's /etc/sysconfig/apache2

- In addition to routine admin items requiring attention in this file is the choice of **Apache MPM** (multiple processor module).
- The default MPM is "**prefork**", meaning each new incoming connection is handled by creating a new/fresh image of the whole Apache application. That is resource expensive, but it is easy to use in the beginning.
- Better is MPM "**worker**" which uses one Apache image in *threaded mode* to handle many connections at once by one process. MPM **event** has less support. See */etc/apache2/server-tuning.c*onf for MPM configuration details.
- However, language interpreters (PHP, Perl, Python etc) are **not thread-safe**, thus with MPM **worker** we add helper module **mod_fcgid** to support PHP and omit listing module "php" in *Apache2-modules=.* For Perl programs include its FCGI module and see mod-fcgid docs. For Python consider using Apache **mod_wsgi**.
- Mod_fcgid has its configuration file within */etc/apache2/conf.d*. It is shown next.
- Module **ssl** needs to appear early in the list *Apache-modules=* (as item #3 or so) to satisfy loading dependencies of other modules.

File /etc/apache2/conf.d/mod_fcgid.conf

Bottom of the file where I put my (JRD) configuration choices

```
<FilesMatch "\.php$">                                  PHP programs (.php) are detected by this line.
    AddHandler fcgid-script  .php                      handle .php files using fcgid-script
    FCGIWrapper /srv/www/cgi-bin/php  .php             PHP fast-cgi helper for .php files
    Options +ExecCGI                                   permit execution of these cgi scripts
</FilesMatch>
##JRD addition, do not buffer output
    OutputBufferSize 0                                 don’t buffer output data
    MaxRequestLen   40000000                           large file transfer limit, make it be large
    MaxRequestsPerProcess -1                           long live fcgid
    BusyTimeout 86400
    IdleScanInterval 3
```

This module creates forked helpers (language etc) for the threaded Apache core

See the Apache httpd manual for mod_fcgi and its configuration information

Alias map a URL path to a file system location

Alias Directive

Description:	Maps URLs to filesystem locations
Syntax:	`Alias [URL-path] file-path|directory-path`
Context:	server config, virtual host, directory
Status:	Base
Module:	mod_alias

The `Alias` directive allows documents to be stored in the local filesystem other than under the `DocumentRoot`. URLs with a (%-decoded) path beginning with *URL-path* will be mapped to local files beginning with *directory-path*. The *URL-path* is case-sensitive, even on case-insensitive file systems.

```
Alias "/image" "/ftp/pub/image"
```

This and similar figures are from the Apache manual

URLs are for people, easy to remember.
File system locations are internal details.
Directive *Alias* relates the two views.
URL said “/foo”, Alias can report it as file “/etc/bar” to the rest of the script.

Directive *DocumentRoot* specifies a file system root location for unaliased URLs.
Alias converts to actual, not relative, file system locations.

Default *DocumentRoot* is /srv/www/htdocs, but we usually redefine it to be elsewhere.

<Directory> Grant Apache access to the file system

<Directory> Directive

Description:	Enclose a group of directives that apply only to the named file-system directory, sub-directories, and their contents.
Syntax:	`<Directory directory-path> ... </Directory>`
Context:	server config, virtual host
Status:	Core
Module:	core

`<Directory>` and `</Directory>` are used to enclose a group of directives that will apply only to the named directory, sub-directories of that directory, and the files within the respective directories. Any directive that is allowed in a directory context may be used. *Directory-path* is either the full path to a directory, or a wild-card string using Unix shell-style matching. In a wild-card string, `?` matches any single character, and `*` matches any sequences of characters. You may also use `[]` character ranges. None of the wildcards match a `/` character, so `<Directory "/*/public_html">` will not match `/home/user/public_html`, but `<Directory "/home/*/public_html">` will match. Example:

```
<Directory "/usr/local/httpd/htdocs">
  Options Indexes FollowSymLinks
</Directory>
```

Alias /posh “/usr/local/fancy/place”
<Directory “/usr/local/fancy”>
  blah blah must dress appropriately here etc
</Directory>

Grants Apache access to a particular file system area and creates a container for directives using that area. The file system must also permit Apache user *wwwrun:www* to access the area.

Directive *Alias* translates a URL token into a file system location and thus normally directly preceeds a <Directory> clause.

Example, restricted access to area and program

```
Alias /config  /home/search/config.php        config is URL/progam for admin use only
<Directory /home/search>                      Restrict area access by IP and username
    <requireall>                              Must satisfy all “require” conditions
    require ip 11.22.33.100 10.0.0.1/24 127.0.0.1      admin moves about
    AuthType Basic
    AuthBasicProvider ldap
    AuthName "search"
    AuthLDAPUrl  ldap://example.com/o=oucs?uid?one?(objectClass=user)
    require ldap-user admin jrd                The second “require” condition
    </requireall>                         (There is also  <requireany> and <requirenone>)
</Directory>
```

<Location> Rules for URLs, no file system grants

<Location> Directive

Description:	Applies the enclosed directives only to matching URLs
Syntax:	`<Location URL-path|URL> ... </Location>`
Context:	server config, virtual host
Status:	Core
Module:	core

The `<Location>` directive limits the scope of the enclosed directives by URL. It is similar to the `<Directory>` directive, and starts a subsection which is terminated with a `</Location>` directive. `<Location>` sections are processed in the order they appear in the configuration file, after the `<Directory>` sections and `.htaccess` files are read, and after the `<Files>` sections.

`<Location>` sections operate completely outside the filesystem. This has several consequences. Most importantly, `<Location>` directives should not be used to control access to filesystem locations. Since several different URLs may map to the same filesystem location, such access controls may by circumvented.

Clause to contain rules based on a URL, but it does not grant access to the file system

<Location dinner>
  blah blah how to behave at meal time
</Location>

Has conditions (directives) to be applied if the URL contains a particular string.

<LocationMatch> allows a regular expression to specify URL strings.

See later slide about syntax nuances.

Shielding Solr via a web proxy server

Let admin's IP and localhost based query agents be proxied to search engine Solr. Solr is running in a Java VM on localhost port 8983.

https://example.com/solr invokes the controlling clause below and relays (proxies) traffic to and from the independent Solr process.

```
<Location /solr>                          Relay/proxy data to external web program solr
    require ip 11.22.33.100 127.0.0.1     Only these IP's are permitted
    proxypass "http://localhost:8983/solr" keepalive=on
    proxypassreverse "http://localhost:8983/solr"
</Location>
```

Note: *proxypass* and *proxypassreverse* should use the same URL field

A minor industry has arisen to embellish Apache proxying. See the Apache manual.

Masquerading (proxy to/from local workers)

- The proxy modules of Apache enable smart relaying by Apache of web traffic between the outside world and back end worker machines, while pretending that Apache itself does the real work. Apache also acts as the terminator of outside world SSL/TLS connections. It hides outsourcing.
- Proxying provides a convenient user-friendly web experience and hides the back room worker details. It also shields the back end from many SSL/TLS subversion attempts and similar bad guy things. Apache as a web firewall.
- The pair of Apache directives are **proxypass** and **proxypassreverse**. They should be used as a matched pair. <Proxy foo> can be a proxy-specific container for directives. There are many Proxy related directives.
- This subject is discussed in presentation Hiding Tomcat behind Apache on netlab1.net which has Vibe and Filr examples.

<Location> nuances (its fine print)

In the example below, where no trailing slash is used, requests to /private1, /private1/ and /private1/file.txt will have the enclosed directives applied, but /private1other would not.

```
<Location "/private1">
    #  ...
</Location>
```

In the example below, where a trailing slash is used, requests to /private2/ and /private2/file.txt will have the enclosed directives applied, but /private2 and /private2other would not.

```
<Location "/private2/">
    # ...
</Location>
```

Review this item about handling trailing slashes

When to use <Location>

Use <Location> to apply directives to content that lives outside the filesystem. For content that lives in the filesystem, use <Directory> and <Files>. An exception is <Location "/">, which is an easy way to apply a configuration to the entire server.

For all origin (non-proxy) requests, the URL to be matched is a URL-path of the form /path/. *No scheme, hostname, port, or query string may be included.* For proxy requests, the URL to be matched is of the form scheme://servername/path, and you must include the prefix.

The URL may use wildcards. In a wild-card string, ? matches any single character, and * matches any sequences of characters. Neither wildcard character matches a / in the URL-path.

Regular expressions can also be used, with the addition of the ~ character. For example: ⇦

```
<Location ~ "/(extra|special)/data">
    #...
</Location>
```

Discussion about using trailing shashes, and using regular expressions to detect key words

Another example, <Directory> + <Location>

```
ScriptAlias  /mailman /usr/local/mailman/cgi-bin/    This spot has executable cgi scripts
<Directory /usr/local/mailman/cgi-bin>               Grant permission to enter this spot
    Options +Indexes +FollowSymLinks +ExecCGI        Local facilities permitted here
    SetHandler cgi-script                            Run scripts via helper cgi-script
    require ip 11.22.33.44/28 10.0.0.1/24 127.0.0.1  Client IPs which can use this area
</Directory>
<Location  "/mailman/listinfo/test">                 Oh, and while we are here…
    require ip 11.22.33.100 10.0.0.1/24 127.0.0.1    Allow this from only admin's IPs
</Location>
```

Users say https://example.com/mailman and Apache converts this into a file system path to a cgi-bin worker. How thoughtful of it.

Quotes are actually needed only if a string contains spaces and the like, but we often use them anyway.

<IfModule> module-specific rule sets

<IfModule> Directive

Description:	Encloses directives that are processed conditional on the presence or absence of a specific module
Syntax:	`<IfModule [!]module-file|module-identifier> ... </IfModule>`
Context:	server config, virtual host, directory, .htaccess
Override:	All
Status:	Core
Module:	core
Compatibility:	Module identifiers are available in version 2.1 and later.

The `<IfModule test>...</IfModule>` section is used to mark directives that are conditional on the presence of a specific module. The directives within an `<IfModule>` section are only processed if the *test* is true. If *test* is false, everything between the start and end markers is ignored.

The *test* in the `<IfModule>` section directive can be one of two forms:

- *module*
- !*module*

In the former case, the directives between the start and end markers are only processed if the module named *module* is included in Apache httpd -- either compiled in or dynamically loaded using `LoadModule`. The second format reverses the test, and only processes the directives if *module* is **not** included.

The *module* argument can be either the module identifier or the file name of the module, at the time it was compiled. For example, `rewrite_module` is the identifier and `mod_rewrite.c` is the file name. If a module consists of several source files, use the name of the file containing the string `STANDARD20_MODULE_STUFF`.

Rules which apply only if the named module is enabled within Apache.

Example of an <IfModule > specific container

```
# cat /etc/apache2/conf.d/mp4.conf
<IfModule mod_h264_streaming.c>
    AddHandler h264-streaming.extensions .mp4 .webm .ogv
    AddHandler smooth-streaming.extensions .ism
    AddType video/webm .webm
    AddType video/mp4 .mp4
    AddType video/x-m4v .m4v
    AddType video/ogg .ogv
    AddType video/quicktime .mov
</IfModule>
```

If this module is enabled

Lots of module-specific items

<Files blah> and **<FilesMatch blah>** perform similar containment based on the filename in the URL. Example usage:

```
<Files ~ "\.(cgi|shtml|phtml|php3?)$">
    SSLOptions +StdEnvVars
</Files>
```

Also see **mod_deflate** for on the wire gzip/gunzip of selected files.

Helping users, mod_speling

Add this clause into /etc/apache2/default-server.conf and/or vhosts to correct simple typos. It requires *speling* be in list APACHE_MODULES=

```
<IfModule mod_speling.c>
## CheckSpelling alone does case insensitivity and one-character typo corrections
    CheckSpelling on
    CheckCaseOnly on
</IfModule>
```

Mod_speling manual snippet

Apache Module mod_speling

Available Languages: en | fr | ja | ko

Description:	Attempts to correct mistaken URLs by ignoring capitalization, or attempting to correct various minor misspellings.
Status:	Extension
Module Identifier:	speling_module
Source File:	mod_speling.c

Summary

Requests to documents sometimes cannot be served by the core apache server because the request was misspelled or miscapitalized. This module addresses this problem by trying to find a matching document, even after all other modules gave up. It does its work by comparing each document name in the requested directory against the requested document name **without regard to case**, and allowing **up to one misspelling** (character insertion / omission / transposition or wrong character). A list is built with all document names which were matched using this strategy. **Erroneous extension** can also be fixed by this module.

If, after scanning the directory,

- no matching document was found, Apache will proceed as usual and return an error (`404 - document not found`).
- only one document is found that "almost" matches the request, then it is returned in the form of a redirection response (`301 - Moved Permanently`).
- more than one document with a close match was found, then the list of the matches is returned to the client, and the client can select the correct candidate (`300 - Multiple Choices`).

Several directives are associated with this module. Their default on/off settings:

Checkbasenamematch	on
CheckCaseOnly	off
CheckSpelling	off

See the manual for details.

Another mixed ingredients example

```
# cat /etc/apache2/conf.d/phpLdapPasswd.conf
Alias  /cpw  /home/user/phpLdapPasswd
<Directory /home/user/phpLdapPasswd>
        Options FollowSymLinks
        DirectoryIndex index.php
        require ip 11.22.33.44/28 127.0.0.1
        RewriteEngine On                    Rewrite triad redirects the caller to try again with https, not  http
        RewriteCond %{HTTPS} off            Report True (do the rule below) if https was not used
        RewriteRule (.*) https://%{HTTP_HOST}%{REQUEST_URI} [QSA,R]     query string append, redirect
</Directory>
<Location /cpw/hidden>
        require all denied                  Don’t go here, I warn you. Apache rebuffs everyone at that door.
</Location>
```

```
<?php
// Include the configuration and functions, and display an error message
// if unable to do so.
if ((!@include './hidden/config.php') || (!@include './hidden/functions.php')) {
?>
<html>
<head>
<title>Fatal Error</title>
</head>
<body>
  … on and on
```

Ordinary file *index.html* is replaced here by running a PHP program.
Directive *DirectoryIndex* does the replacement.

Summary thus far

- Clause <Directory> picks out URL details and relates them to file access places. <Directory>, <Location>, <Proxy> and similar <…> clauses can assign area behaviour conditions.
- Within these clauses can be a variety of appropriate directives, for which we must read the fine print in documentation and run verification experiments. Apache has many many directives.

- Directives stated outside of clauses can apply globally, thus be careful.
- Documentation indicates whether such directives must be within a clause (context:) and if so in which kind. Experiment and check log files.

Default host and virtual hosts (ghosts?)

- File */etc/apache2/default-server.conf* defines the default port 80 host behaviour. It includes defining *DocumentRoot* to relate URL filenames to where in the file system those names begin.
- There can be additional "apparent" web servers, named vhosts, which respond on different ports and perhaps different IP and DNS names.

- One standard vhost is */etc/apache2/vhosts.d/vhost-ssl.conf* which normally services SSL/TLS connections on port 443. Each such vhost can use its own DocumentRoot, certificates and configuration details.

- All are brought into play by script */etc/apache2/httpd.conf*. Other details are in adjacent scripts, such as *listen.conf* for IP:port combos.
- Notice that "*Include filename*" statements are often placed near the end of .conf files.
- Reviewing each file is highly recommended.

<VirtualHost> documentation snippet

<VirtualHost> Directive

Description:	Contains directives that apply only to a specific hostname or IP address
Syntax:	`<VirtualHost addr[:port] [addr[:port]] ...> ... </VirtualHost>`
Context:	server config
Status:	Core
Module:	core

`<VirtualHost>` and `</VirtualHost>` are used to enclose a group of directives that will apply only to a particular virtual host. Any directive that is allowed in a virtual host context may be used. When the server receives a request for a document on a particular virtual host, it uses the configuration directives enclosed in the `<VirtualHost>` section. *Addr* can be any of the following, optionally followed by a colon and a port number (or *):

- The IP address of the virtual host;
- A fully qualified domain name for the IP address of the virtual host (not recommended);
- The character `*`, which acts as a wildcard and matches any IP address.
- The string `_default_`, which is an alias for `*`

```
<VirtualHost 10.1.2.3:80>
  ServerAdmin webmaster@host.example.com
  DocumentRoot "/www/docs/host.example.com"
  ServerName host.example.com
  ErrorLog "logs/host.example.com-error_log"
  TransferLog "logs/host.example.com-access_log"
</VirtualHost>
```

Vhost-ssl.conf summary

- File */etc/apache2/vhosts.d/vhost-ssl.conf* is for the standard SSL/TLS capable web server which listens on port 443 for https:// connections.
- Related files live within a <VirtualHost name> container to separate them from the default and other virtual hosts. Home rule in action.
- File */etc/apache2/listen.conf* defines how Apache listens to the network, including what virtual hosts require for comms.
- File */etc/apache2/ssl-global.conf* is of use if there are many vhosts using the same certificate. It is included by each vhost's .conf file.
- Handling of certificates and SSL/TLS is full of detail and I suggest reviewing presentation SSL/TLS for system admins v2.1 which is available on netlab1.net and has those details.

What Apache does when viewing a directory

- If an incoming URL specifies a filename then it is sought in the URL provided path/filename, and that path is relative to the *DocumentRoot* file system location.
- If no filename is stated, only a path (relative to *DocumentRoot*), then Apache looks in that directory for a default index file, typically named *index.html*. Directive *DirectoryIndex* defines such default filenames.
- If a DirectoryIndex file is not present then a directory listing is normally performed. "*Options Indexes*" and module mod_autoindex are required to show a listing.
- Ah ha! We can show directory listings (if we wish).
- We must be careful to not carelessly expose directories or specific files. Protection involves using Apache directives to forbid kinds of access, often conditional on the caller's identity. <Directory> and <Location> clauses are standard filtration places.

An example directory listing, my improvements

Index of /novttp/files/AsiaPac2019/presentations/Fri

Name	Last modified	Size
http3.pdf	2019-11-26 15:58	2.0M
http3.pptx	2019-11-26 15:58	3.8M
Risk_Service_TTP.pdf	2019-12-16 14:30	1.2M
Risk_Service_TTP.pptx	2019-12-10 06:03	1.7M
Sharepoint_Integration_6Dec.pdf	2019-12-16 14:32	209K
Sharepoint_Integration_6Dec.pptx	2019-12-06 09:07	250K
TTP - SCC Workflows (2).pdf	2019-12-16 14:01	1.5M
TTP - SCC Workflows (2).pptx	2019-12-06 08:23	2.0M
UpgradingIDM_LessonsLearned.pdf	2019-12-11 08:29	1.3M

<Directory foo>
Options Indexes FollowSymLinks MultiViews IncludesNoExec
IndexStyleSheet “/index.css”

State **Indexes** for a listing
My embellishments

Default appearance has been improved by my file *index.*css
which is enabled by directive IndexStyleSheet “/index.css” .

Blocking some bad guy penetration methods

These items can be placed in <Directory> clauses or placed outside of them and thus apply globally.

```
<IfModule mod_rewrite.c>
# Fail OPTIONS, CONNECT etc and HTTP/1.0 requests, and require https://
# ^ is "starts with", $ is "ends with"
    RewriteEngine On
    RewriteCond %{REQUEST_METHOD}  ^(CONNECT|DELETE|HEAD|OPTIONS|PUT|TRACE|TRACK)
    RewriteRule ".*"  "-"  [F,L,END]                    F is report failure, L is last, END is end this rule set
      RewriteEngine On                                    ".*"  "-" is replace incoming command text with -
      RewriteCond %{THE_REQUEST} HTTP/1\.0$             HTTP/1.0 is old and vulnerable to abuse
      RewriteRule ".*"  "-"  [F,L,END]
    RewriteEngine On                                     Please repeat the request using https, not http
    RewriteCond %{HTTPS} off
    RewriteRule (.*)  https://%{HTTP_HOST}/%{REQUEST_URL}  [QSA,L,R=301]      R is redirect, args are repeated
</IfModule>
```

Also see presentations IPtables & blocking the bad guys and SSL/TLS for system admins v2.1

Further bad guy defense

Normally applied globally, but can be within a <Directory> clause.
Redirect pests to their local host. We are being “kind” rather than nasty.
Redirect permanently if this-URL-phrase to this-new-request-line

```
ReDirectPermanent /admin https://127.0.0.1/
ReDirectPermanent /wp-login.php https://127.0.0.1/
ReDirectPermanent /pub/wp-login.php https://127.0.0.1/
ReDirectPermanent /cgi-bin/ViewLog.asp https://127.0.0.1/
ReDirectPermanent /4VG* https://127.0.0.1/
ReDirectPermanent /ViewLog.asp https://127.0.0.1
ReDirectPermanent /.env https://127.0.0.1
ReDirectPermanent /HNAPI https://127.0.0.1
ReDirectPermanent /boaform https://127.0.0.1
```

Making progress with this matter

- Apache web server is sophisticated, and rather complicated.
- We can approach such matters by reviewing an initial installation and then make small incremental changes of interest. Research on the Internet is often needed to find hints and examples. Best to also skim the documentation, but don't let anyone know you have done that.
- After some experience we master configuration details for particular use situations. That leaves other features for if/when a need arises.
- Thus look at the initial installation, tinker with it, and progress in small steps. A test setup is often safer than changing production gear.

- Help about writing web pages: w3schools and Mozilla.
- Discussion about using Certificates and cryptography: SSL/TLS for system admins
- A web based HTML validator: w3c.org. Browsers can have HTML validator plugins.

MindWorks Inc. Ltd
210 Burnley Road
Weir
Bacup
OL13 8QE UK

Telephone: +44 (0) 170 687 1900
Fax: +44 (0) 170 687 8203
Web: www.mindworksuk.com
Email: training@mindworksuk.com