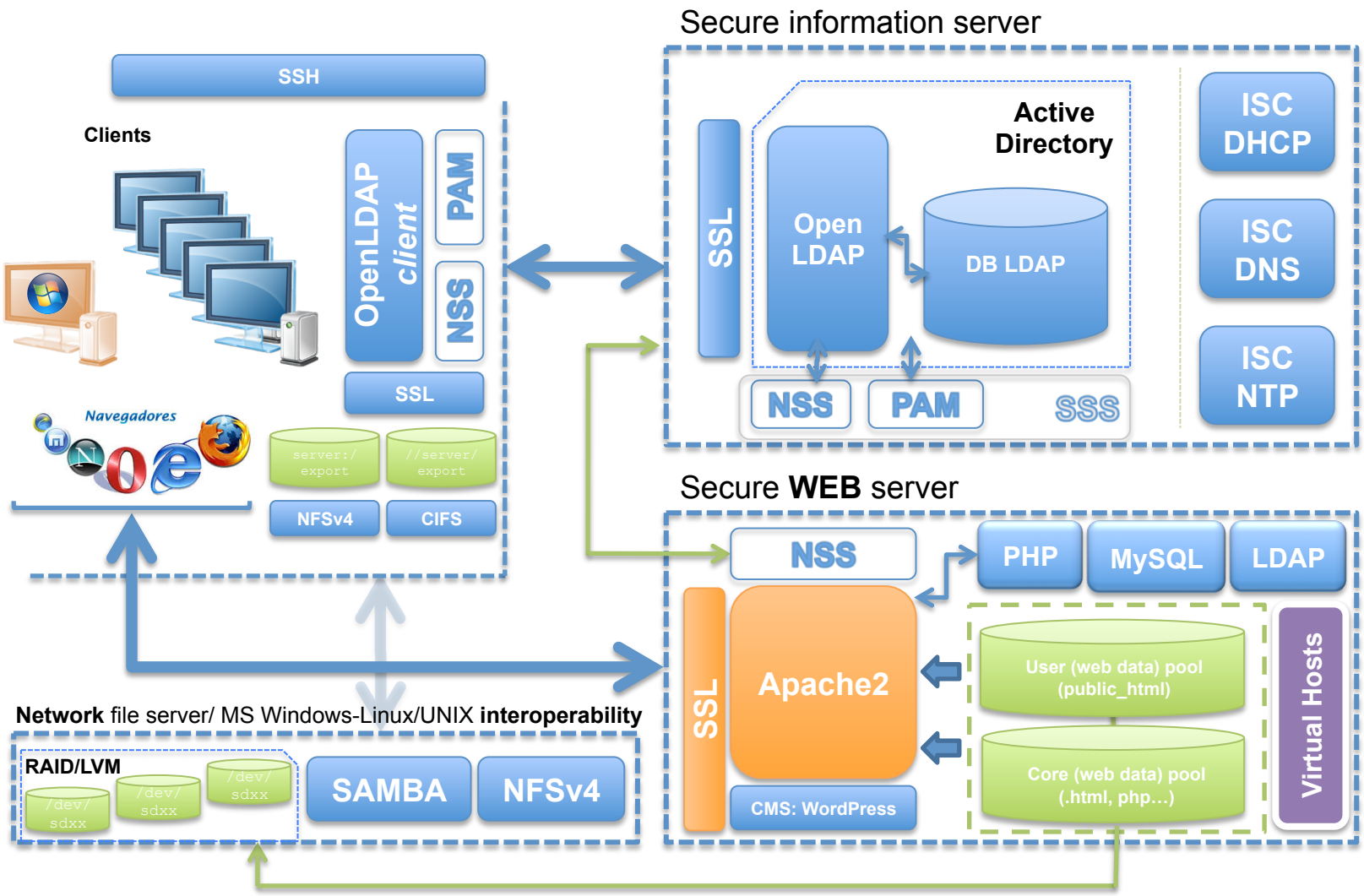
Computer System Design and Administration

Topic 9. Secure web service: HTTP Apache2 (over SSL)

José Ángel Herrero Velasco

Department of Computer and
Electrical Engineering

WEB service (HTTP): Puzzle

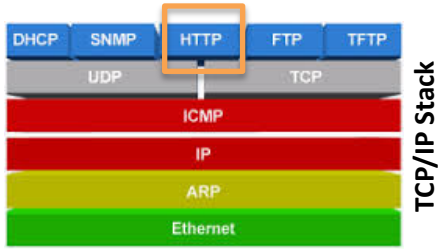


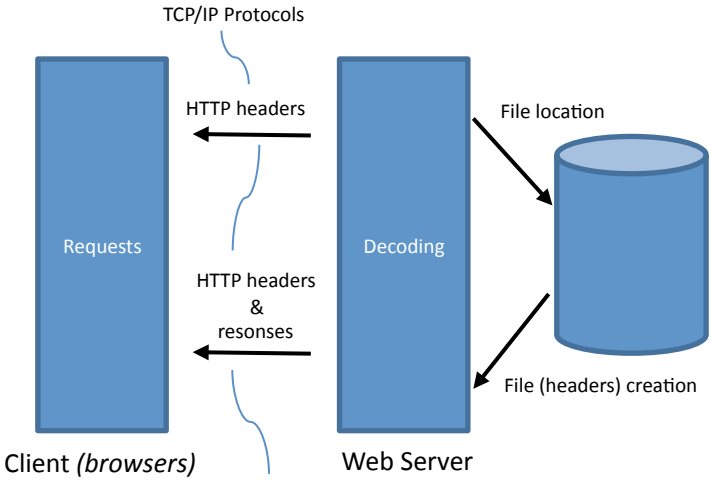
Target: WEB services

- Deployment and development of an INTERNET *secure* WEB **service** based on Apache server:
  - **Content management.**
  - **Access** control.
  - ***Virtualhost*** *concept* (management).
  - Secure communications: **TLS/SSL.**
- Installation, configuration and start-up of a Web Content Manager (**CMS**):
  - **Wordpress.**

HTTP: The protocol

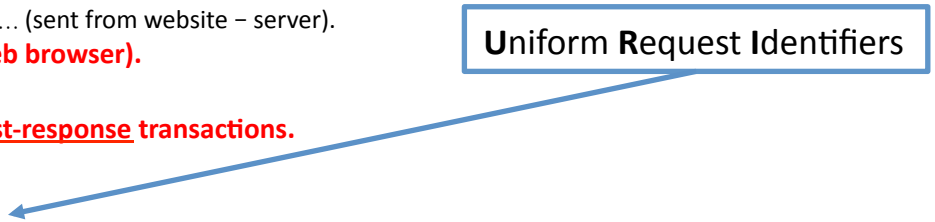
- **The HTTP** (**H**ypertext **T**ransfer **P**rotocol) **protocol:**
  - **Definition:**
    - The *Hypertext* Transfer Protocol (HTTP) is an *application-level* protocol for **distributed**, **collaborative**, **hypermedia** information systems:
      - **Enables exchanging information (text, multimedia...).**
      - **Most *common method* of information exchange on WWW - Since 1990.**
      - **Syntax and semantics.**
  - The information exchanged:
    - The ***HyperText Markup Language (HTML):***
      - **Hypertext is structured text that uses logical links between nodes with contents (plain text):**
        - » Message exchange methods:
          - `HEAD`, `GET`, `POST`…
      - **Used by software elements on the WEB architecture:**
        - » Clients (browsers), servers and proxies.
      - **HTTP is the protocol to exchange or hypertext transfer.**
    - Other content → **Multimedia.**
  - Operation protocol:
    - **Request-response** protocol:
      - **It is a client-server model.**
    - **Transaction oriented:**
      - **HTTP transaction: header + content.**
  - Stateless protocol:
    - → HTTP server does not retain information about client sessions (status).
    - Some web APPs implement states:
      - ***Cookies* (sessions).**
  - Run over **TCP:**
    - Port **80** (default).





HTTP: The protocol

- Items and **main features:**
  - **The user agent (UA):**
    - **Web browsers, crawlers, mobile apps...**
  - ***Cookies:***
    - **Small piece of data about HTTP sessions (client – server):**
      - » State, activity, authentication… (sent from website – server).
    - **Stored on clients (by the user's web browser).**
  - **HTTP sessions:**
    - **Sequence of HTTP network request-response transactions.**
    - **Manage a web resource request.**
  - **HTTP resources:**
    - **Depict a web content → URL (URI) and RDF (abstract syntax).**
  - **Request methods:**
    - **Indicate the desired action to be performed.**
    - **GET, POST, HEAD, PUT, DELETE, CONNECT...**
  - **HTTP authentication:**
    - **Even thought the default way of using the WEB is in *anonymous mode.***
    - **Headers: WWW-Authentication & Authorization / Basis and Digest (HTTP 1.1).**
  - **Safe connections:**
    - **HTTP supports SSL:**
      - » *Encrypted* connections (**https**).
    - **TCP port 443.**
  - **Persistent connections:**
    - **Enable transferring several "request/responses" using only one TCP session.**
    - **Reduce request latency perceptibly.**
  - **HTTP Cache:**
    - **Mechanism for web documents caching.**
    - **The HTTP server stores copies of documents passing through it:**
      - » It reduces the wait time and the network load.
  - **HTTP messages:**
    - **Plain-text (ASCII) messages → Header + Body.**

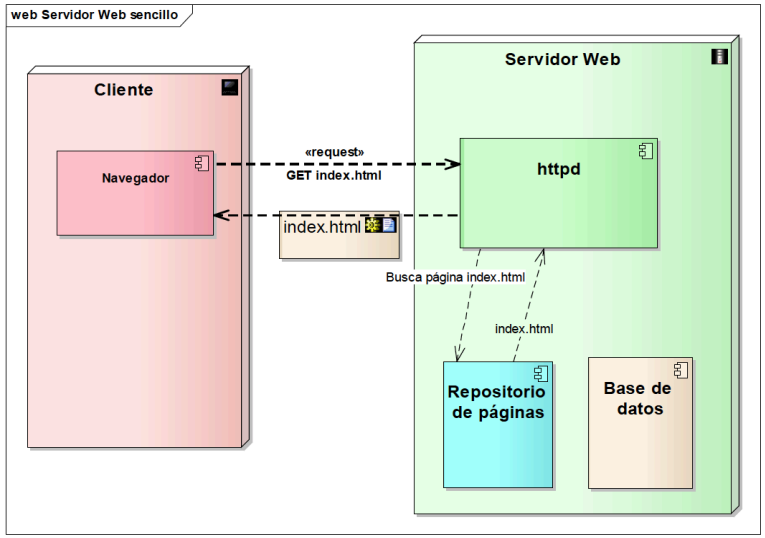


HTTP: The protocol

- Developed *initially* by:
  - Tim Berners-Lee at CERN.
- Evolved and coordinated by:
  - World Wide Web Consortium (**W3C**).
  - Internet Engineering Task Force (**IETF**).
- Main standards:
  - **HTTP/1.1** (RFC 2068)…
  - URI (RFC 3986).
  - **HTML** 2.0 (RFC 1866)…
  - XML…
  - Java & Javascripts (applets).
- Protocol versions:
  - HTTP/1.0 (**RFC 7540**):
    - **May 1996.**
  - HTTP/1.1:
    - **June 1999.**
  - HTTP/2 (**RFC 7540**):
    - **May 2015.**

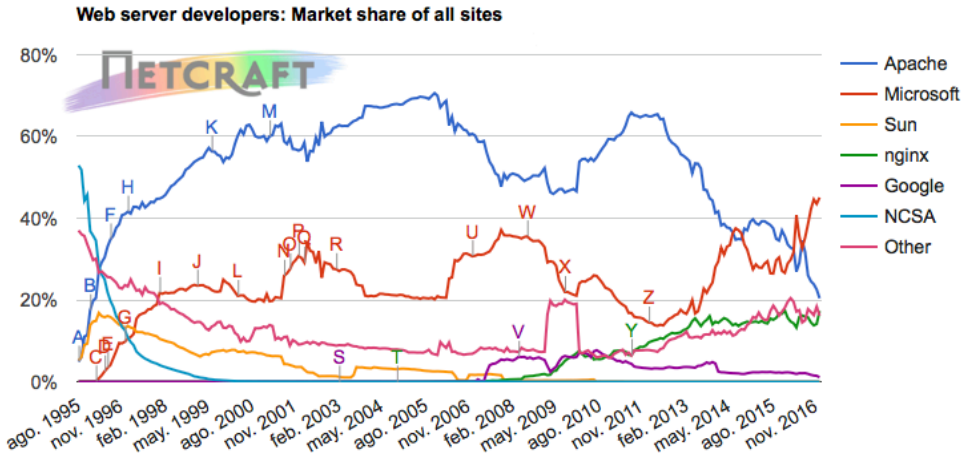
HTTP: The protocol

- HTTP **operation:**
  - *Client–Server* computing model (***for example***)
    - **client → server:**
      - Browser submits a HTTP *request* message:
        - URL (URI) → http://www.elpais.es (default port 80).
      - *Previously:* the URL must be translated from FQDN to IP address (**DNS**).
    - **server → client:**
      - Server returns a *response message* to the client:
        - It contains full status information about the request.
        - It is possible that server requests **user validation (credentials)** to the client.
    - **client → server:**
      - Browser submits a request in a new message (Header): **GET method** (for example).
    - **server:**
      - Server *maps* the URL to a file or app under the document directory.
      - Creates a new *response message:*
        - Header + Request resource (file, app...).
    - **server → client:**
      - Server sends the *response message.*
    - **client:**
      - Browser receives the *response* from server.
      - Browser **formats** the response information and **displays** it.
    - **server:**
      - Server closes the opened TCP connection used.



HTTP: Protocol implementations

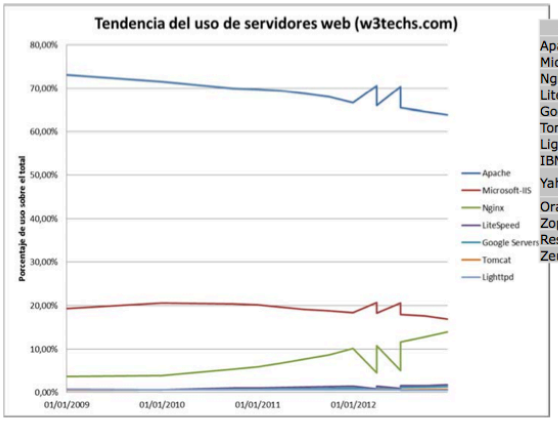
- Some of the most relevant **implementations** of HTTP (server side):
  - **Apache:**
    - **Tomcat (javaserver & java servlet).**
  - *Nginx.*
  - Cherokee.
  - **Internet information Services – IIS** (Microsoft).
  - **WebSphere** Application Server (IBM).



Source: http://www.netcraf.com

Apache: Web service features

- **The Apache HTTP server:**
  - **Definition**: *Multi-platform* **HTTP** server distributed under *open source* license:
    - **HTTP protocol implementation → httpd.**
    - UNIX (Linux), MS Windows, Solaris, Novel Netware, OS X…
  - Originally based on the **NCSA HTTPd** server (Rob McCool):
    - Since 1995.
    - One of the vital keys in the growth of the **WWW.**
    - Currently (since 1999), it is a project of **Apache Software Foundation.**
  - Main features:
    - Open source:
      - **Distribution license: Apache:**
        - » No GPL.
        - » https://www.apache.org/licenses/LICENSE-2.0.
    - **Highly configurable and scalable.**
    - **Extensible functionality:**
      - **Modular architecture.**
      - **Loadable Dynamic Modules.**
    - **Very widely used *(popular):***
      - **Most popular version distributed: 1.3:**
        - » current: 2.4.
    - **Proxy server** capabilities.
    - HTTP/2.
    - IPv6.
    - TLS/SSL...



	12/2012
Apache	63,90%
Microsoft-IIS	16,90%
Nginx	13,90%
LiteSpeed	1,80%
Google Servers	1,30%
Tomcat	0,70%
Lighttpd	0,50%
IBM Servers	0,30%
Yahoo Traffic Server	0,20%
Oracle Servers	0,20%
Zope	0,10%
Resin	0,10%
Zeus	0,10%

W3Techs
Web Technology Surveys
http://w3techs.com

Apache: Apache 2 improvements…

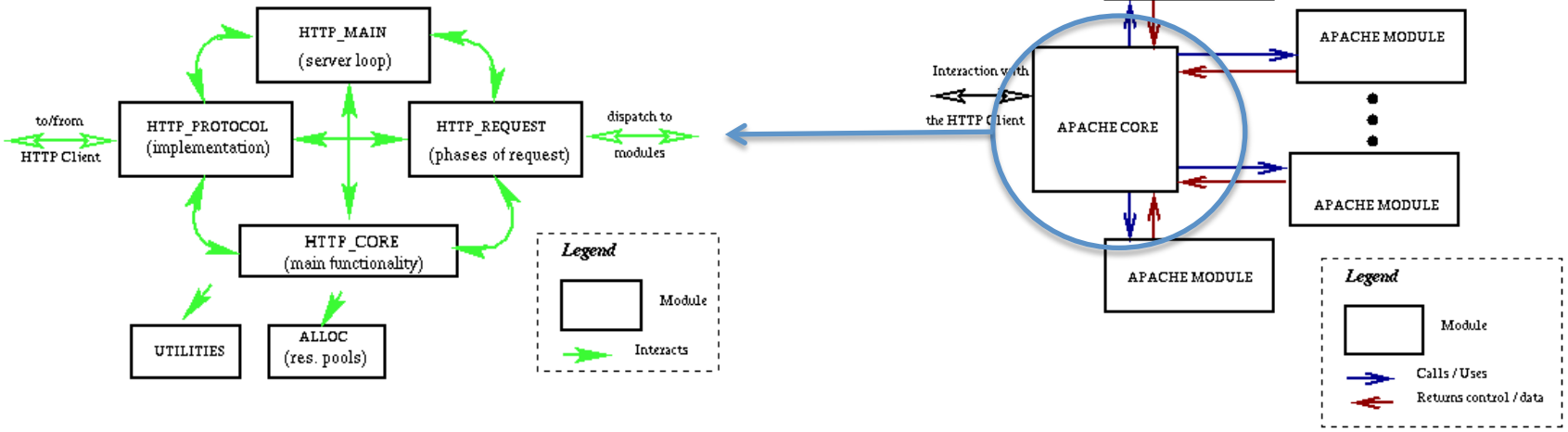
- **Apache 2.x:**
  - Provides a number of **improvements** over Apache 1.3:
    - New **Build System:**
      - **`autoconf` and `libtool` → Similar to other packages.**
      - **Moves the main part of request processing into *Multi-Processing Modules* (MPMs).**
    - Simplified configuration:
      - **Many confusing directives have been simplified.**
    - ***Virtualhosts*** definition (**Name-** and **IP address-based** virtual servers):
      - **One server (IP address) ←→ multiple domain names (URLs – websites).**
    - **Filtering:**
      - **Which acts on the stream of content as it is delivered to or from the server.**
    - New Apache **API:**
      - **Useful for new modules (functionality) creation.**
    - UNIX *multithreading* (**Pthreads**):
      - **On Unix systems with POSIX threads support, Apache now can be run in a hybrid multiprocess.**
    - Better support for **non-Unix** platforms:
      - **BeOS, OS/2 & Windows, MPMs… Native Windows NT Unicode Support.**
    - **Load balancing.**
    - **It supports:**
      - **IPv6 protocol.**
      - **TLS/SSL security.**
      - **XML, CGI, GZIP, WebDAV.**
      - ***Scripting* languages support*:***
        - » **PHP, perl, Phyton, Tcl.**
      - **Java: J2EE.**
      - **Tomcat.**
    - And more...:
      - **Multilanguage Error Responses, Regular Expression Library Updated...**

For more *details,* see…:
http://httpd.apache.org/docs/2.0/upgrading.html.

Apache: Architecture

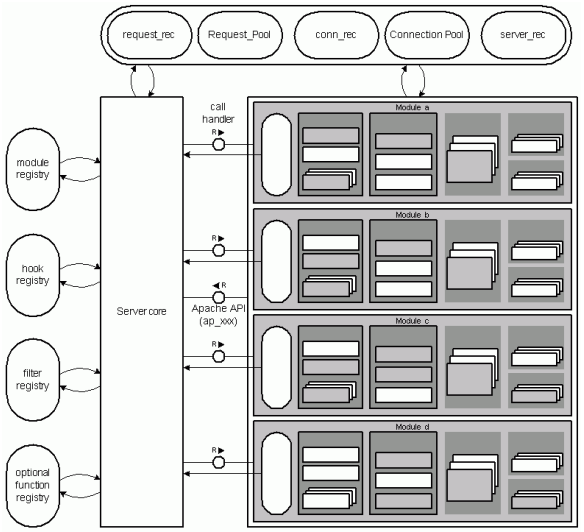
- **Apache 2:**
  - **Modular (design) architecture:**
    - Apache *httpd* has always accommodated a wide variety of environments through its **modular design.**
    - *Core* + many **modules:**
      - **Most of the functionality falls over modules *(pieces)*.**
      - **For example:**
        - » **SSL**, WebDAV, **PHP**, Python, mysql...

Apache *"core"* components



Apache: Dynamic load modules

- **Extend** Apache functionality:
  - Allows for Apache to perform **additional functions.**
  - By selecting which modules to load either at **compile-time** or at **run-time.**
- **Installation and load:**
  - Modules are installed as **independent software pieces.**
  - **Static & Dynamic** load:
    - *"On line"* connect and disconnect.
    - Service has *not to be* restarted (but *reload).*
    - Independent of each other.
  - **MPM** → Multi-processing modules:
    - Better **performance.**
- For each module:
  - It contains:
    - A **module-info** (handlers information):
      - **Essential for module registration by the core.**
    - ***"Handlers":***
      - **A set of actions needed to resolve a `HTTP_REQUEST`.**
      - **Handlers for hooks, for dealing with configuration directives, filters...**
  - Some of them can use their own *configuration file:*
    - `/etc/apache2/conf.d` → `/etc/apache2/mod-available`.



Source: http://www.fmc-modeling.org.

Apache 2: Service installation

- **Installation:**
  - Linux Debian:
    - **Core** & *"default"* modules:
      `$ apt-get install apache2`
    - …Additional modules:
      `$ apt-get install libapache2-mod-xxx`

- **Server *(daemons)* configuration** (main flags):
  `$ vi /etc/default/apache2`
    - This file defines the options about `htcacheclean` service:
      - **It is a service included in Apache which is used to manage the cache functionality.**
    - **`HTCACHECLEAN_RUN`:**
      - **Exec mode:**
        - `htcacheclean: auto`
    - **`HTCACHECLEAN_MODE`:**
      - **Exec mode → *Standalone* daemon or cron jobs.**
    - **`HTCACHECLEAN_SIZE`:**
      - **Cache size: 300M.**
    - **`HTCACHECLEAN_OPTIONS`:**
      - **Additional options.**

Apache 2: Service configuration

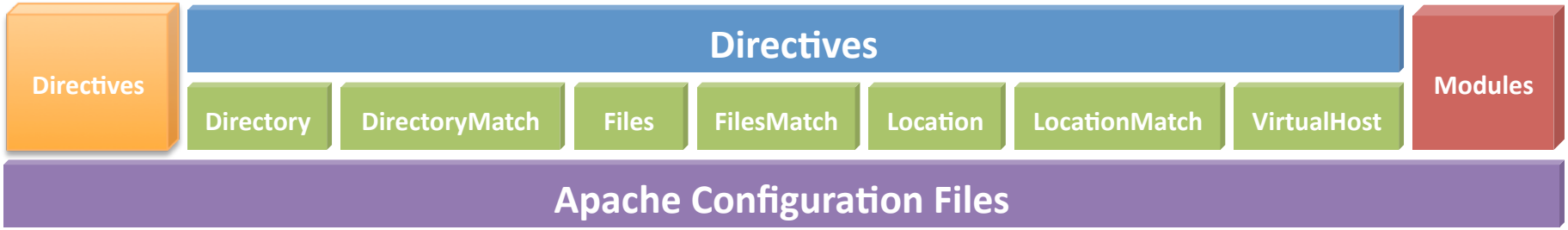
- **Service configuration:**
  - **Directory tree:**
    - **`/etc/apache2`:**
      - **"Root" directory.**
    - **`/etc/apache2/conf.d`:**
      - **Configuration files for *specific features* (functions) of apache:**
        - » Charset, PHP, security...
    - **`/etc/apache2/mods-available/{*.load,*.conf}`:**
      - **Modules files and their configuration files.**
    - **`/etc/apache2/mods-enables/`→`{*.load}`:**
      - **Only contents *symbolic links* to module files:**
        - » If exists, then the **module** is enabled.
    - **`/etc/apache2/sites-available`:**
      - **Configuration files for Apache2 "virtualhost", defaults included.**
      - **And other website definitions.**
    - **`/etc/apache2/sites-enable/`→**
      - **Only contents *symbolic links* to website files:**
        - » If exists, then the **"virtualhost"** or **website** is enabled.

Apache 2: Service configuration

- **Service configuration:**
  - **Files:**
    - `/etc/apache2/apache2.conf`:
      - **Main configuration file (root).**
      - **Overall features and parameters for the httpd service.**
    - `/etc/apache2/http.conf` **(deprecated):**
      - **Its content is distributed among the different configuration files.**
      - **It is usually empty.**
    - `/etc/apache2/sites-available/*`:
      - **80 % of the Apache2 configurations:**
        - » `Apache2 directories, virtualhost…`
    - `/etc/apache2/mod-available/*.conf`.
      - **15 % of the Apache2 configurations:**
        - » Configuration for apache2 **modules.**
    - `/etc/apache2/envvars`:
      - **Environment variables for httpd service.**
    - `/etc/apache2/magic`:
      - `mod_mime_magic` **configuration. It defines the document types (MINE) to be transferred.**
    - `/etc/apache2/ports.conf`:
      - **Directives to define the TCP ports and IP address for httpd service.**
    - **Every Apache configuration file shows the same syntax:**
      - **ASCII files.**
      - **Sections.**
      - **Directives.**
      - **Each file has:**
        - » 2 configuration **blocks:**
          - *Global* parameters:
            - **Directives.**
          - *Section* parameters:
            - **Directives.**

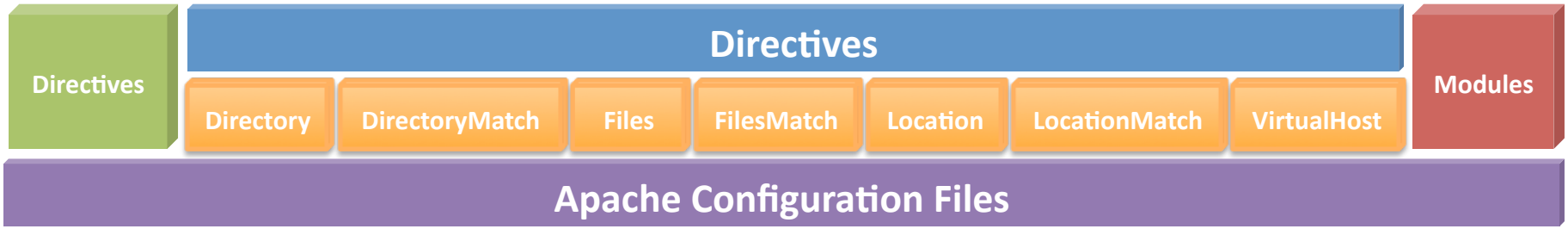
The **directives scope** can be:
- **Global:**
  - All web service.
- **By section:**
  - Subset of **directories…**
  - **Virtualhost.**

Apache 2: Service configuration



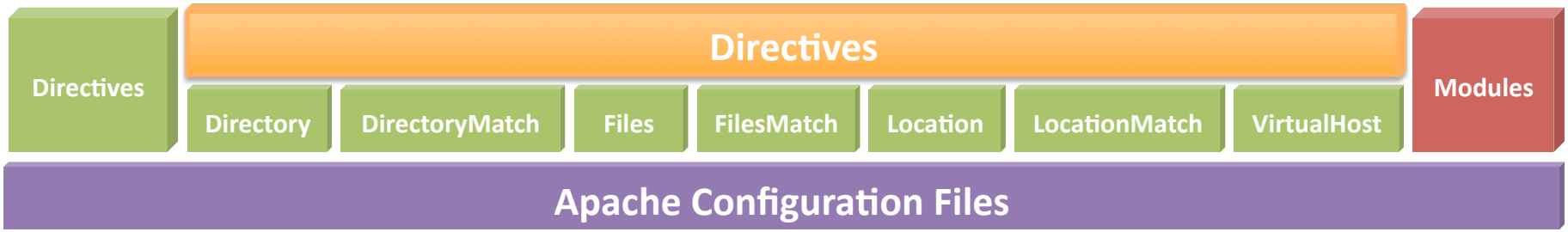
- Main GLOBAL **directives:**
  - `ServerName:`
    - *Hostname* (and port) that the server uses to identify itself.
  - `DocumentRoot:`
    - Directory that forms the main document tree (**web contents**) visible from the Web.
  - `Listen:`
    - IP addresses and ports that the server listens to.
  - `ServerRoot:`
    - Base directory for the server installation.
  - `TimeOut:`
    - Amount of time the server will wait for certain events before failing a request.
  - `User:`
    - The userID with which the server will answer requests.
  - `LogFormat:`
    - Describes a format for using in a log file.
  - `DirectoryIndex:`
    - List of resources to look for when the client requests a directory.

Apache 2: Service configuration



- Main **SECTION directives:**
  - **<Directory>:**
    - Enclose a group of directives that *apply only* to the named file-system **directory, sub-directories**, and their **contents**.
  - **<DirectoryMatch>:**
    - Enclose directives that apply to the contents of file-system directories matching a *regular expression*
  - **<Files>:**
    - Contains directives that apply to matched filenames
  - **<FilesMatch>:**
    - Contains directives that apply to regular-expression matched filenames
  - **<Location>:**
    - Applies the enclosed directives only to matching `URLs`
  - **<LocationMatch>:**
    - Applies the enclosed directives only to regular-expression matching `URLs`
  - **<VirtualHost>:**
    - Contains directives that apply only to a **specific hostname or IP address**
  - **<IfDefine>:**
    - Encloses directives that will be processed only if a test is true at startup (`-D` option)
  - **<IfModule>:**
    - Encloses directives that are processed conditional on the *presence* or *absence* of a **specific module.**

Apache 2: Service configuration



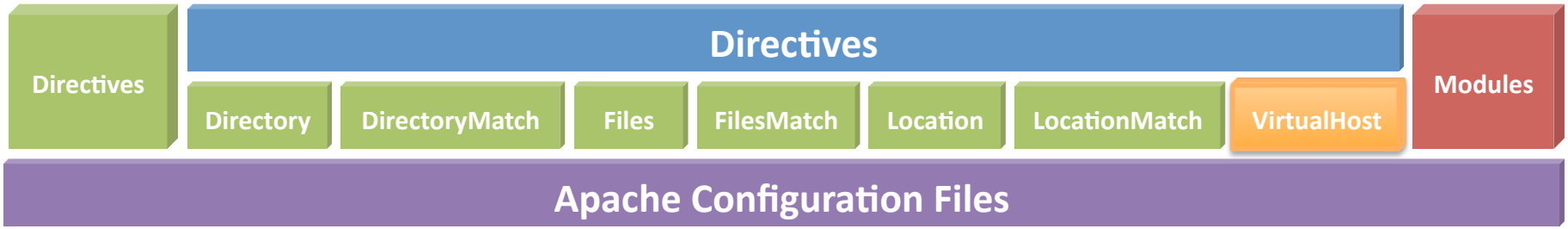
- Specific directives for **control of access to section (content directory):**
  - These directives can be defined within the `<directory>` directive or within a specific file:
    - `.httaccess` → ***"distributed configuration files".***
    - **Located in the directory (file system) that I want to protect.**
  - **Access Restrictions:**
    - **When a user tries to access, browser asks for user credentials.**
    - ***username:password.***
  - **Directives:**
    - **`AllowOverride`:**
      - » Types of directives that are allowed in `.htaccess` files (All, None, *Authconfig…).*
    - **`Allow/Deny`:**
      - » Controls **which hosts (IPs)** can access an area of the server:
        - `Allow from All.`
        - `Allow from 156.35.171.`
        - `Allow from 156.35.171.1/255.255.0.0.`
        - `Deny from 192.168.1.104 192.168.1.205.`
    - **`Order`:**
      - » Controls the default access state and the order in which `Allow` and `Deny` are evaluated :
        - `Order Allow, Deny.`
        - `Order Deny, Allow.`

Apache 2: Service configuration

- **AuthType:**
  - Type of user authentication (**Basic** or **Digest**).
- **AuthName:**
  - Authorization **realm** for use in HTTP authentication.
- **AuthUserFile:**
  - Sets the name of a *text file* containing the list of **users** and **passwords** for authentication.
  - It may be created by apache2 using the command:
    - **$ htpasswd.**
- **Required:**
  - Establishes what specific user(s)/group(s) can access the content (`user ..., group ..., valid-user`).

- **More directives:**
  - **Options:**
    - Configures what features are available in a particular directory:
      - **None/All FollowSymLinks:**
        - » Types Allow (or not) to follow symbolic links in this directory.
      - **+/-Indexes:**
        - » If a URL that maps to a directory is requested, and there is no `DirectoryIndex` (E.g.: *index.html)* in that directory, then `mod_autoindex` will return a **formatted listing** of the directory.

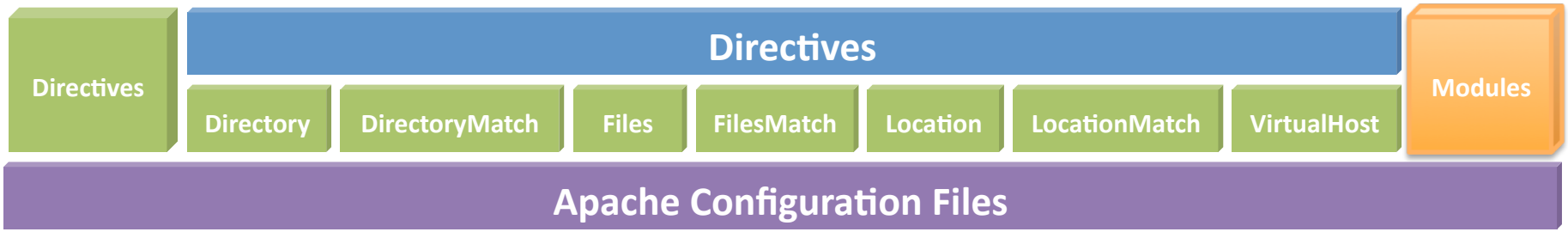
Apache 2: Service configuration



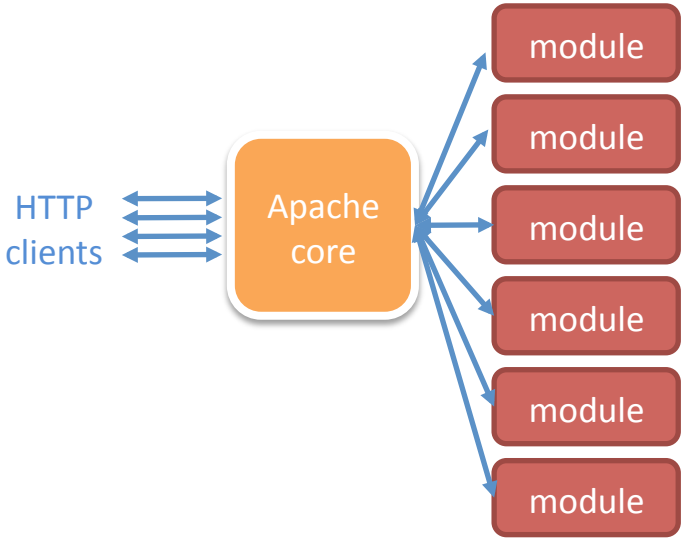
- **Virtualhost** directive:
  - Enables running more than one web site (such as *company1.example.com* and *company2.example.com)* on a **single machine:**
    - **"IP-based" → Meaning that you have a different IP address for every web site.**
    - **"Name-based" → Meaning that you have multiple names running on each IP address.**
  - There is no magic:
    - **You *must* have the names in your local DNS *(authoritative),* resolving your IP address, or nobody else will be able to see your web site.**
  - Configuration:
    - **Usually, Virtualhosts configuration is located on particular file:**
      - » `/etc/apache2/sites-<enable|available>/`
  - For each "virtualhost", define one section:

```
<VirtualHost [IP address:Port]>
        ServerAdmin
        DocumentRoot → Directory that forms the main document tree visible from the web.
        ServerName → Virtualhost name (FQDN).
        ServerAlias
        ErrorLog → PATH to the log (error) file for this section (virtual host).
        CustomLog → PATH to the log (special) file for this section (virtual host).
                <Directory "/www/vh1">
                        […]
                </Directory>
</VirtualHost>
```

Apache 2: Service configuration



- Directives for loading and operation of **modules:**
  - Apache HTTP Server: **core +** modules.
  - Perform **additional functions.**
  - **Loading:**
    - **Run-time.**
    - **Compile-time.**
  - **Directives:**
    - `LoadModule:`
      - » Load a particular module.
      - » *Compile-time.*
    - `<IfModule modulo>:`
      - » Encloses directives that are processed conditional on the *presence* or *absence* of a **specific module.**
    - `</IfModule>.`



Apache 2: Service configuration

- **Modules:**
  - Main "core" modules:
    - **`core:`**
      - **Basic functionality.**
    - **`worker:`**
      - Multi-Processing Module (**MPM**) implementing a hybrid multi-threaded multi-process web server.
  - MPMs (Multi-Processing Modules):
    - **`mpm_common, perchild, prefork, worker:`**
      - They are responsible for **binding** to *network ports* on the machine, accepting requests, and dispatching *children* to handle the requests.
  - Others:
    - **`mod_access:`**
      - Accesss control.
    - **`mod_auth_ldap:`**
      - LDAP user authentication.
    - **`mod_perl:`**
      - Perl dynamic pages.
    - **`mod_php:`**
      - PHP dynamic pages.
    - **`mod_python:`**
      - Python dynamic pages.
    - **`mod_ssl:`**
      - SSL/TLS secure communications.
    - **`mod_security:`**
      - App level filtering (Security).

Apache 2: Service configuration

- **Directory** of **WEB contents** (html, php…):
  - **`/var/www`** (default):
    - In every *Virtualhost,* it must be established.
    - Default:
      - **`/etc/apache2/sites-available/default.`**
    - Through **`DocumentRoot`** directive.
  - Fully configurable:
    - More details: http://httpd.apache.org/docs/current/.
- **Directory** of **user** WEB content:
  - **`$HOME/public_html:`**
    - Be created *manually.*
    - Have "proper" UNIX permissions (**access/property rights**).
  - Enabled by loading its corresponding module:
    - Module: **`mod_userdir.`**
    - File configuration: **`userdir.conf.`**
- **Logs** files:
  - **`/var/log/apache2/access.log:`**

```
156.35.14.7 - - [11/Jun/2007:20:44:55 +0200] "GET /icons/blank.gif HTTP/1.0" 200 148 "http://
156.35.171.157/""Mozilla/4.0 (compatible; MSIE 7.0; Windows NT 5.1; InfoPath.1; .NET CLR
2.0.50727; .NET CLR 3.0.04506.30)"
```

  - **`/var/log/apache2/error.log:`**

```
[Mon Jun 11 20:42:48 2007] [notice] Apache/2.0.55 (Ubuntu) configured -- resuming normal operations
[Mon Jun 11 20:44:55 2007] [error] [client 156.35.14.7] File does not exist: /var/www/favicon.ico
```

Apache 2: Service operation

- **Start & Stop** of Web service (Apache2):
  - **`$ service apache2 start.`**
  - **`$ service apache2 stop.`**
  - **`$ service apache2 reload.`**
  - Apache utility:
    - `$ apache2ctl start.`
    - `$ apache2ctl stop.`
    - `$ apache2ctl status.`

Does not stop service

- **Server configuration check:**
  - Shows information about server configuration:
    - **`$ apache2ctl -t`:**
      - **Syntax checking.**
    - **`$ apache2ctl –M`:**
      - **Loaded modules list.**
    - **`$ apache2ctl –S`:**
      - **Virtualhost list.**
    - **`$ apache2ctl -V`:**
      - **Compile options.**

Apache 2: Service operation

- Load & unload of **“virtualhosts”** or **“web sites”**:
  - **`$ a2ensite <vh_name>.`**
  - **`$ a2dissite <vh_name>.`**
  - Service (apache) restart is not necessary:
    - `$ service apache2 reload.`

- Load & unload of dynamic **“modules”**:
  - **`$ a2enmod <mod_name>.`**
  - **`$ a2dismod <mod_name>.`**
  - It is necessary:
    - To install previously the module packages from repositories (or source):
      - **`$ apt-get install libapache2-mod-xxxx.`**
    - To restart the apache service:
      - **`$ service apache2 restart.`**

Apache 2: Security features

- **Basics (previously):**
  - Disable modules *(features)* loaded by default which **are not** necessary:
    - It can avoid possible *security holes.*
    - How to:
      - **(1) List the modules loaded:**
        - » `$ apache2ctl -M`
      - **(2) Disable them:**
        - » `$ a2dismod <mod_name>`
  - Ensure that the **public contents** are the ones they should be:
    - Sections:
      - `<Directory>`
  - Disable **"browsing"** by content directories:
    - Directive:
      - `Options -Indexes`
  - Disable the execution of `.cgi` files:
    - Directive:
      - `Option -ExecCGI`
  - Do not allow apache to follow **symbolic links:**
    - Directive:
      - `Options -FollowSymLinks`
  - Minimize **waiting times** to the maximum:
    - Directive:
      - `Timeout <seconds>`
  - Limit **http requests** to the maximum:
    - Directives:
      - `LimitRequestBody, LimitRequestFields, LimitRequestFieldSize and LimitRequestLine`

Apache 2: Security features

- **Access control:**
  - **IP level:**
    - Which clients may or may not access the Web service according to their **IP address.**
    - Module:
      - `mod_authz_host`
    - Directives:
      - **`Allow from / Deny from / Order`** **(`<Directory>` Section):**
        - For example:

```
Order deny,allow
Deny from all
Allow from unican.es
```

  - **User/group level:**

htaccess mechanism

- Which user(s)/group(s) may or may not access the Web contents according to their **credentials:**
  - ***"username" & "password".***
  - **The management of user credentials can be done by apache itself.**
- Modules:
  - `mod_auth_basic, mod_authn_file, mod_authz_groupfile, mod_authz_user`
- How to:
  - **(1) Determine the content to be protected, through the directive `AllowOverride`:**

```
<Directory "/www/htdocs">
AllowOverride AuthConfig
</Directory>
```

  - **(2) Create a `.htaccess` file in that directory (content) that you want to protect:**

```
AuthType Basic
AuthName "Web Site: Login with user id"
AuthUserFile "/www/passwords/password.file"
AuthGroupFile "/www/passwords/group.file"
Require valid-user
```

  - **(3) Create the credentials file: (`/www/passwords/<password/group>.file`):**

```
$ htpasswd -c /www/passwords/password.file jherrero
```

- More details:
  - http://httpd.apache.org/docs/current/howto/htaccess.html.

Apache 2: Security features

- *Centralized* **user authentication management:**
  - Control of user access can be delegated to systems such as **LDAP** or Kerberos.
  - **LDAP:**
    - Modules: `mod_ldap, mod_authnz_ldap`
    - Directives:

```
» LDAPVerifyServerCert off
» LDAPTrustedMode SSL
» LDAPTrustedGlobalCert CA_BASE64 <Full PATH to CA.cert>
» AuthBasicProvider ldap
» AuthzLDAPAuthoritative off
» AuthLDAPUrl "ldaps://<LDAP hostname (FQDN):636/ou=people,dc=localdomain?uid"
» AuthLDAPBindDN "cn=admin,dc=localdomain"
» AuthLDAPBindPassword "<password>"
» AuthLDAPGroupAttribute memberUid
» AuthName "[LDAP] Restricted Access"
» require valid-user
» require user <username>
» […]
```

Beware of the UNIX **permissions** of this file...

    - All of these must be defined in the configuration part of the **content:**

```
» <Directory "/www/htdocs">
» AuthType Basic
» AuthName "Web Site: Login with LDAP user id"
» […]
» </Directory>
```

    - More details:
      - http://httpd.apache.org/docs/2.4/mod/mod_authnz_ldap.html.

Apache 2: Security features

- **Kerberos:**
  - Modules: `mod_auth_kerb`
  - Directives: `AuthName "[Kerberos] Restricted Access”`:
    - **AuthType kerberos**
    - **KrbAuthRealms <REALM>**
    - **KrbMethodNegotiate On**
    - **KrbMethodK5Passwd On**
    - **Krb5Keytab <Full PATH to the credentials file .keytab>**
    - **require valid-user**
    - **[…]**
  - All of these must be defined in the configuration part of the **content:**
    - **<Directory "/www/htdocs">**
    - **AuthType Kerberos**
    - **AuthName "Web Site: Login with KRB5 user id”**
    - **[…]**
    - **</Directory>**

Apache 2: Security features

- **Encrypted communication:**
  - Apache server supports encrypted communications based on **TLS/SSL:**
    - "plain" → port 80.
    - "crypt" → port **443.**
  - To be taken into consideration:
    - SSL does not support (design failure) multiple *viltualhost.*
    - Only one **secure** (virtual) **server** for each IP address.
  - Procedure:
    - Previously, you must create the **TLS/SSL certificate/key** for the http service:
      → Self-signed certificates can be used…
      - Certificate: `web_server-04.localdomain.cert`
      - Key: `web_server-04.localdomain.key`
    - Re-configure the default secure (TLS/SSL) *virtualhost:*

```
» <VirtualHost a.b.c.d:443>
»         ServerAdmin sistemas@localdomain
»         <Directory "/var/www/localdomain/
»                   DocumentRoot /var/www/localdomain/html
»                   ServerName localdomain
»                   ScriptAlias /cgi-bin/ /var/www/localdomain/cgi-bin">
»     SSLOptions +StdEnvVars
»         </Directory>
»         SSLEngine on
»         SSLProtocol all -SSLv2
»         SSLCipherSuite ALL:!ADH:!EXPORT:!SSLv2:RC4+RSA:+HIGH:+MEDIUM:+LOW
»         SSLCertificateFile /etc/ssl/tls/certs/web_server-04.localdomain.cert
»         SSLCertificateKeyFile /etc/ssl/private/web_server-04.localdomain.key
»         CustomLog logs/localdomain-ssl_request_log \
»          "%t %h %{SSL_PROTOCOL}x %{SSL_CIPHER}x \"%r\" %b »
»         Errorlog logs/localdomain-ssl_error_log
»         TransferLog logs/localdomain-ssl_access_log
» LogLevel warn
» </VirtualHost>
```

  - More details:
    - http://httpd.apache.org/docs/2.4/ssl/.

Apache 2: Security features

- **Running Apache in a *chroot* jail:**
  - Creates a secure *"enclosure"* (jail):
    - Apache *daemon* and threads will be run in a **"*closed*" runtime environment:**
      - **In a different root file system.**
    - There is no need for *setuid-root* programs:
      - **Which can be used to gain root access and break out of jail.**
  - `chroot`:
    - Enables running apps in isolation, both on **process** level and **file system** level.
    - Limits the damage that a regular user (or hacker) can cause using a local shell.
  - Apache server supports an internal mechanism *to run itself* in a *chroot* jail:
    - Modules:
      - `mod_security, mod_chroot`
    - Directives:
      - `chrootDir /var/www`