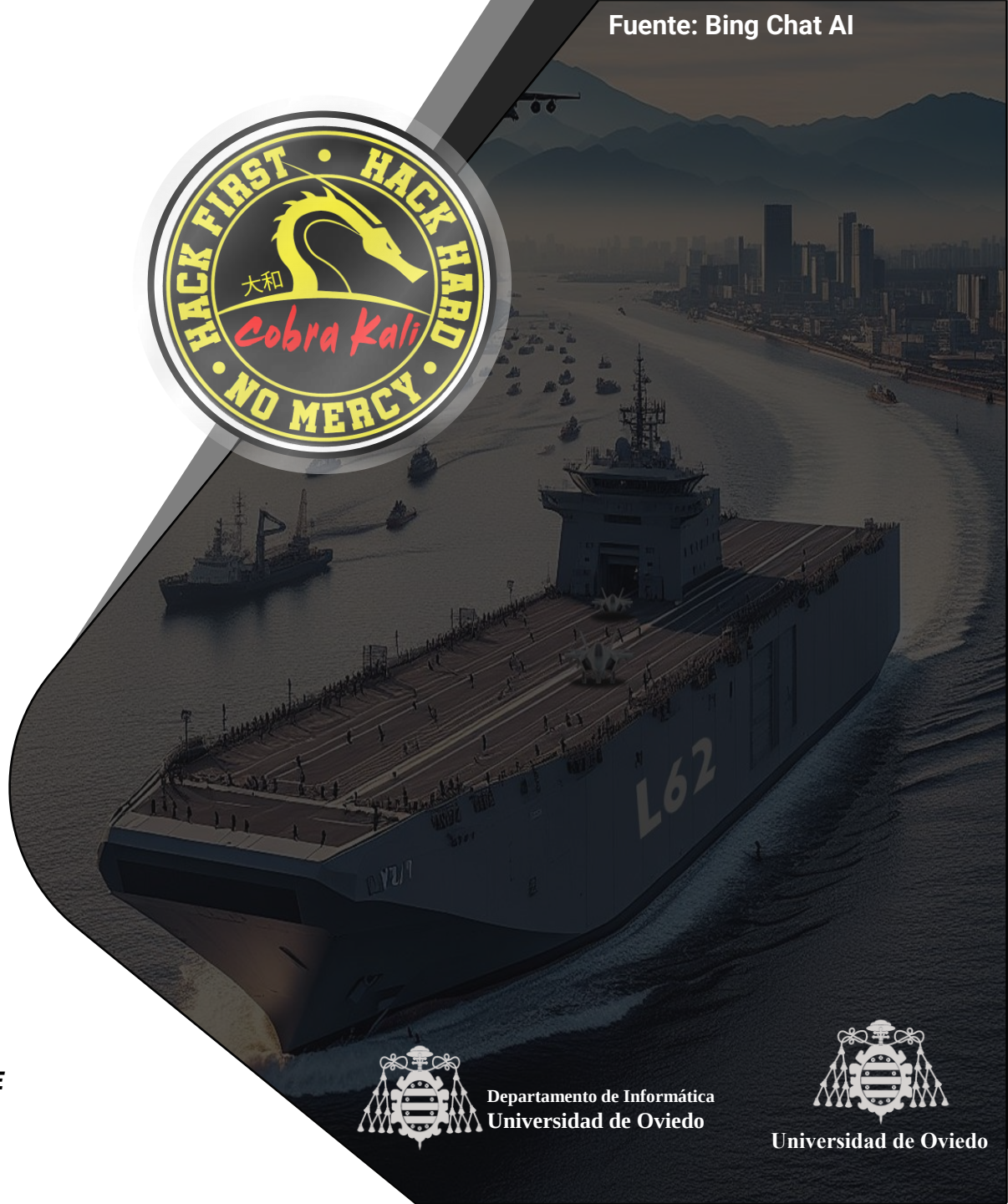
ASPECTOS GENERALES DE SERVIDORES WEB

Los servicios que “dan vida” a las webs que creamos

José Manuel Redondo López Proyecto “*L-62 Princesa de Asturias*” v1.2



Departamento de Informática
Universidad de Oviedo

Universidad de Oviedo

ÍNDICE

¿QUÉ VAS A APRENDER EN ESTE TEMA?

José Manuel
Redondo López

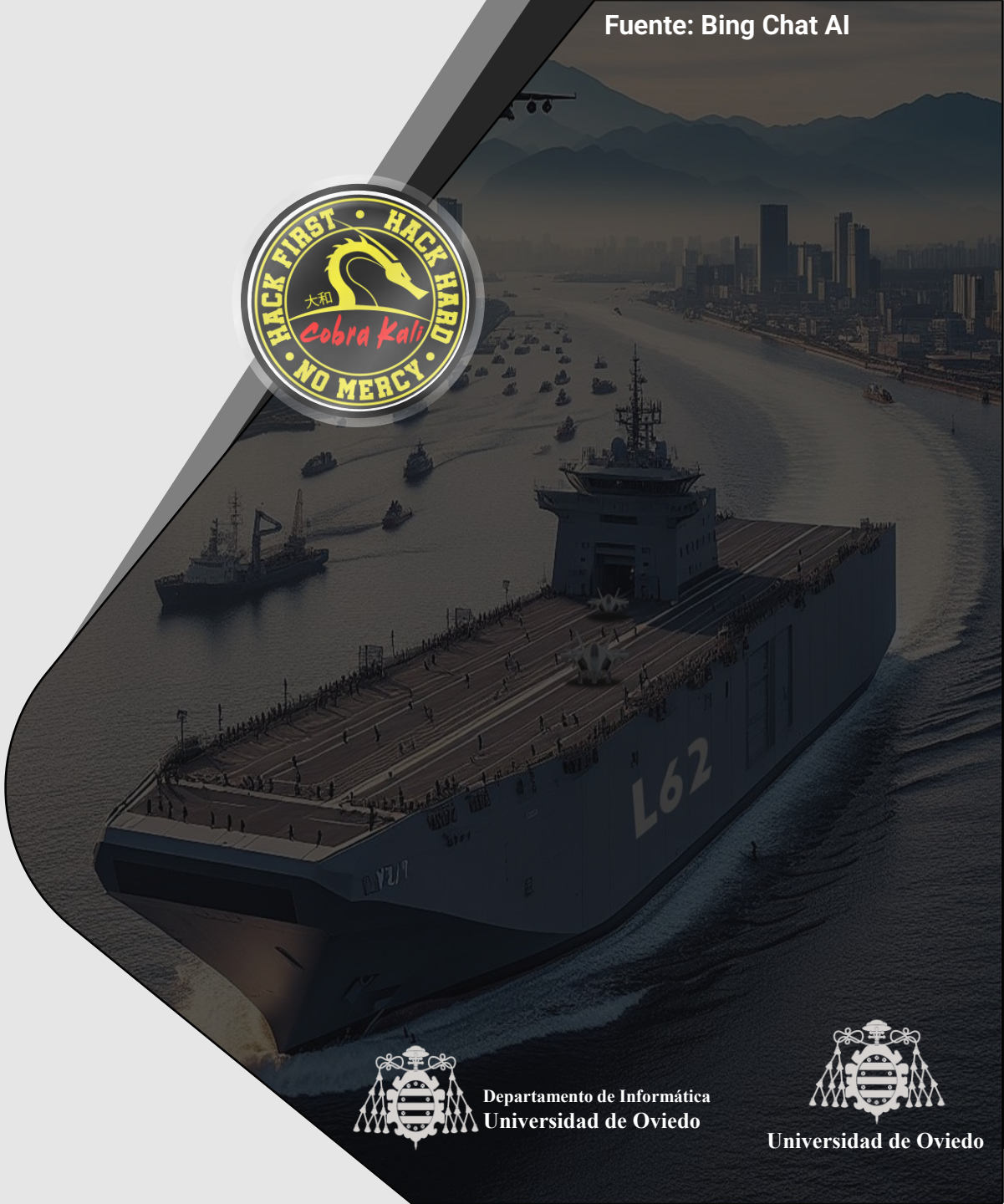
- **Este tema te va a dar las habilidades básicas para poner en marcha dos de los servidores web más usados en la actualidad: Apache 2 y Nginx**
  - Aprenderás que ambos tienen una serie de **elementos en común** que te facilita la "transición" entre ambos
  - Aprenderás qué son los **proxys**, los tipos que hay y para qué se usan
  - Podrás instalar, entender la estructura, configurar y **poner en marcha una web** y ampliar la funcionalidad, tanto del servidor Apache 2 como de Nginx
  - Podrás hacer que ambos servidores **sirvan varias webs a la vez**, y los criterios que usan para distinguirlas
  - Verás la forma de poder **restringir el acceso a una web** o a parte de la misma desde distintos clientes o a diferentes usuarios
- **Con todo ello, estarás capacitado para poner en marcha cualquier web que desarrolles con una configuración básica adecuada**

¿QUÉ TENDRÍAS QUE SABER YA PARA ENTENDERLO CORRECTAMENTE?

José Manuel
Redondo López

- **Este tema solo requiere que tengas un manejo a nivel usuario básico de Linux**
  - Concretamente operaciones básicas en consola (instalar software, navegar por carpetas...)
  - El material proporcionado para trabajar cuenta con una GUI también
  - También puede realizarse enteramente en Windows
- **Trabajaremos sobre Ubuntu (tipo Debian)**
  - Pero todas las operaciones se pueden reproducir en cualquier distribución tipo Red Hat
- **No se requiere ninguna otra habilidad concreta de administración**

< Ir al Índice



INTRODUCCIÓN

Qué son y para qué sirven los servidores web

¿Qué es un servidor web?

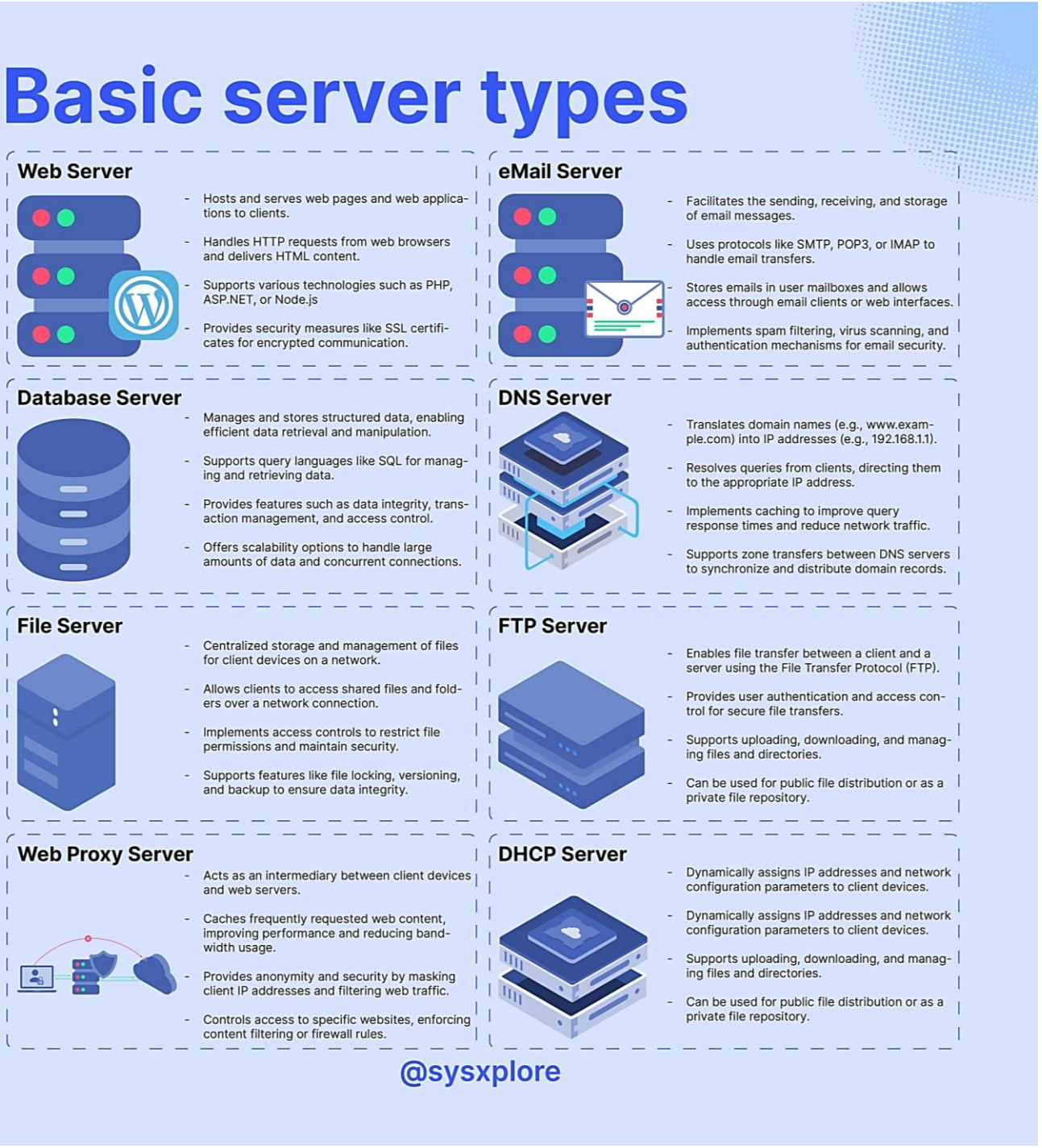
José Manuel
Redondo López

- **Son programas que muestran a sus clientes determinados recursos que se le solicitan del servidor en el que se instalan**
  - Una vez **los procesan y transforman** según su configuración, módulos instalados, etc.
  - Siempre que el cliente **tenga permiso para acceder** a ellos
- **Tienen una elevada complejidad y son extremadamente configurables**
  - Están pensados para ser útiles en un enorme conjunto de situaciones
  - Cuenta con **gran nº de opciones**
  - Están sometidos a un **gran nº de potenciales ataques y atacantes**
    - Un servidor web está expuesto a sus clientes
    - De ellos, muchos serán clientes estándar sin un fin malicioso...
    - ...pero otros sacarán partido de sus posibilidades para **hacer alguna clase de daño** al mismo o al contenido que sirve
- **Por ello, todo el software de la máquina debe estar todo lo actualizado posible**

¿SERVIDOR WEB?

José Manuel
Redondo López

- Un servidor web no es más que uno de los múltiples tipos de servidores que existen
- La mayor parte de servicios importante que usemos tienen un servidor específico asociado
- El principio siempre es el mismo: una máquina que hace cosas por nosotros



Seguridad de Servidores Web

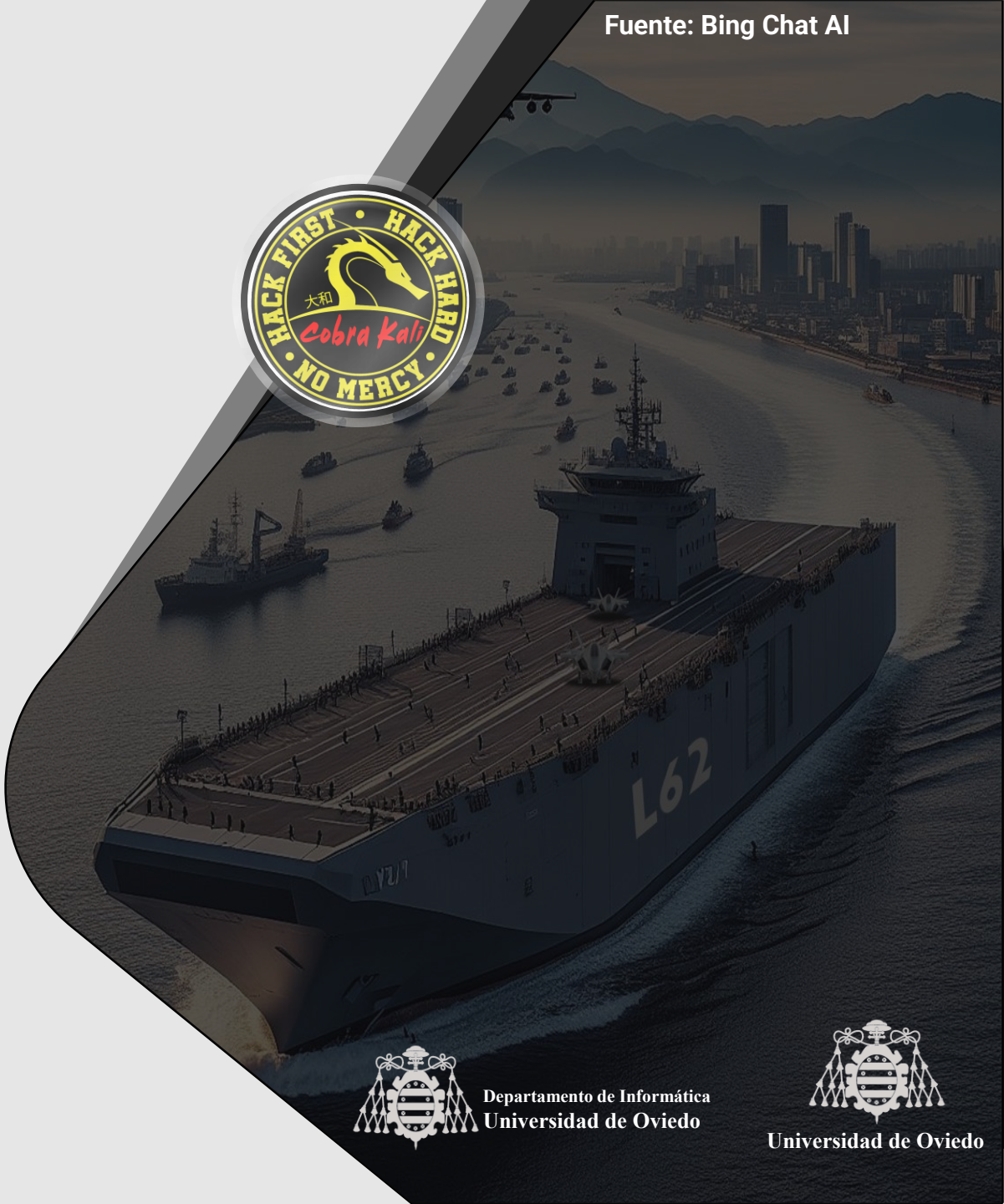
José Manuel
Redondo López

- **La seguridad de la web depende de**
  - **Cómo es la infraestructura** que los soporta (**esta asignatura**)
  - Cómo se ha mejorado la **seguridad de la infraestructura** que las sirve (“F-103 Blas de Lezo”)
  - Cómo está **configurado** el servidor web (“F-103 Blas de Lezo”)
  - Medidas de seguridad implantadas durante la construcción de las páginas web (“F-103 Blas de Lezo”, “F-113 Menéndez de Avilés”)
  - **Pruebas de seguridad** que se le hagan a la infraestructura y las aplicaciones web (“TK-210 Octubre Rojo”)
  - **Resistencia del resto de la infraestructura a intrusiones** cuando ya ha sido inicialmente comprometida, o post-explotación (“K-329 Belgorod”)
- **En esta introducción veremos una introducción que nos permitirá ponerlos en marcha sobre las infraestructuras que construyamos**

< Ir al Índice

Aspectos comunes a los servidores web más usados

Cosas que Apache 2 y Nginx tienen en común



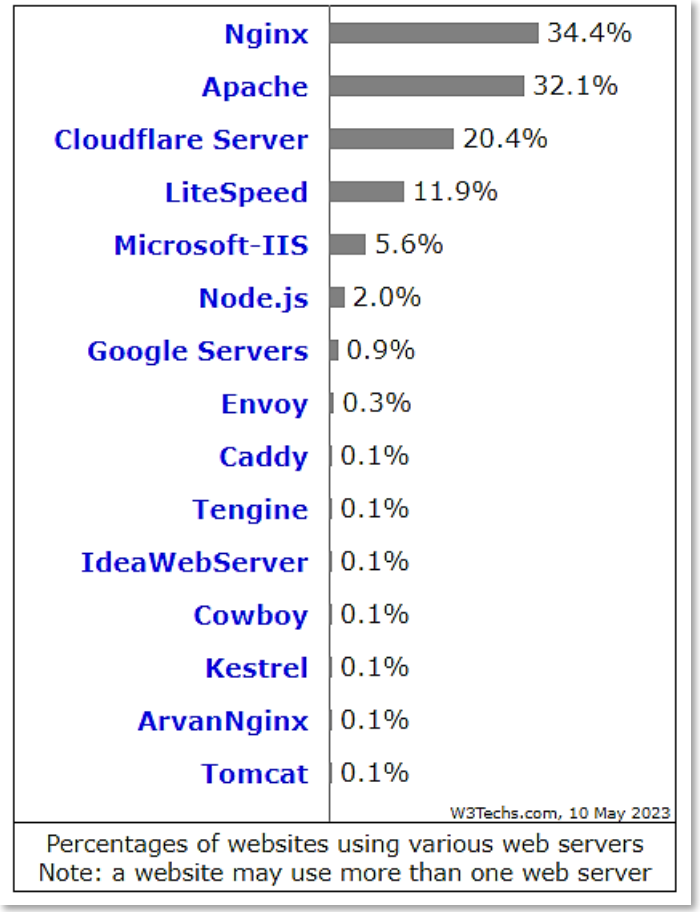
Departamento de Informática
Universidad de Oviedo

Universidad de Oviedo

ASPECTOS COMUNES A LOS SERVIDORES WEB MÁS USADOS

José Manuel
Redondo López

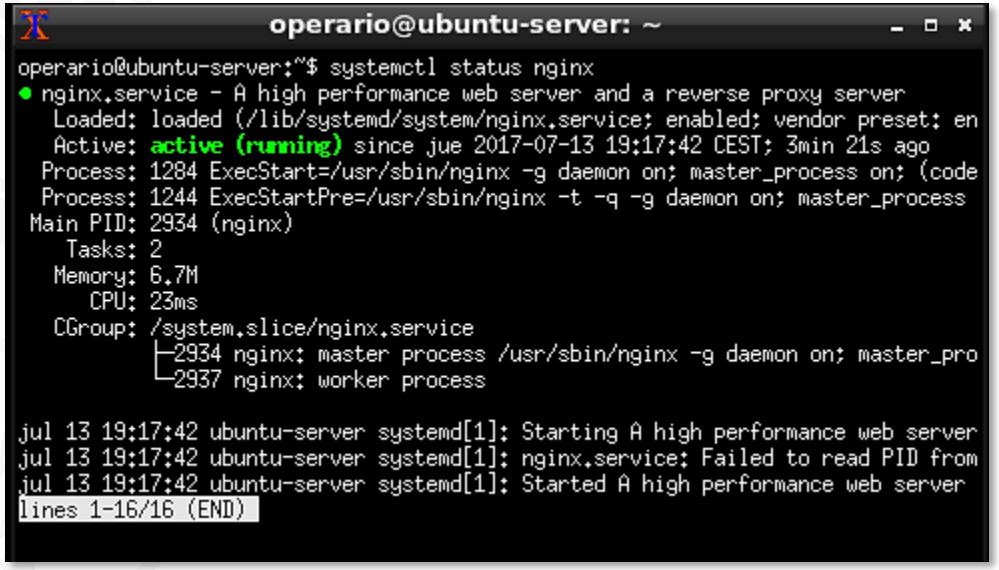
- Tanto **Apache** como **Nginx** (servidores web más usados) comparten aspectos en común
  - Salvo que lo cambiemos, **no pueden servir recursos fuera de la ruta** `/var/www/html`
    - **Ruta por defecto** de las páginas web
    - ¡Cuidado con lo que ponemos en esa ruta! Podríamos servir documentos privados por accidente **solo por copiarlos en ella**
  - A la hora de servir recursos, los accesos a los mismos se hacen **en nombre del usuario anónimo de Internet**
    - Es un **usuario y grupo del sistema** llamado normalmente `www-data`
    - Cuidado con darle demasiados permisos al mismo, o darle permisos de lectura sobre ficheros indebidos
  - Al instalarlos tienen una **funcionalidad básica**, ampliable mediante **módulos instalables**
    - Normalmente con el gestor de paquetes del SO (Ej.: `apt`)
    - Algunos vienen compilados con el ejecutable (**core**)



INSTALACIÓN Y COMPROBACIÓN DE FUNCIONAMIENTO

José Manuel
Redondo López

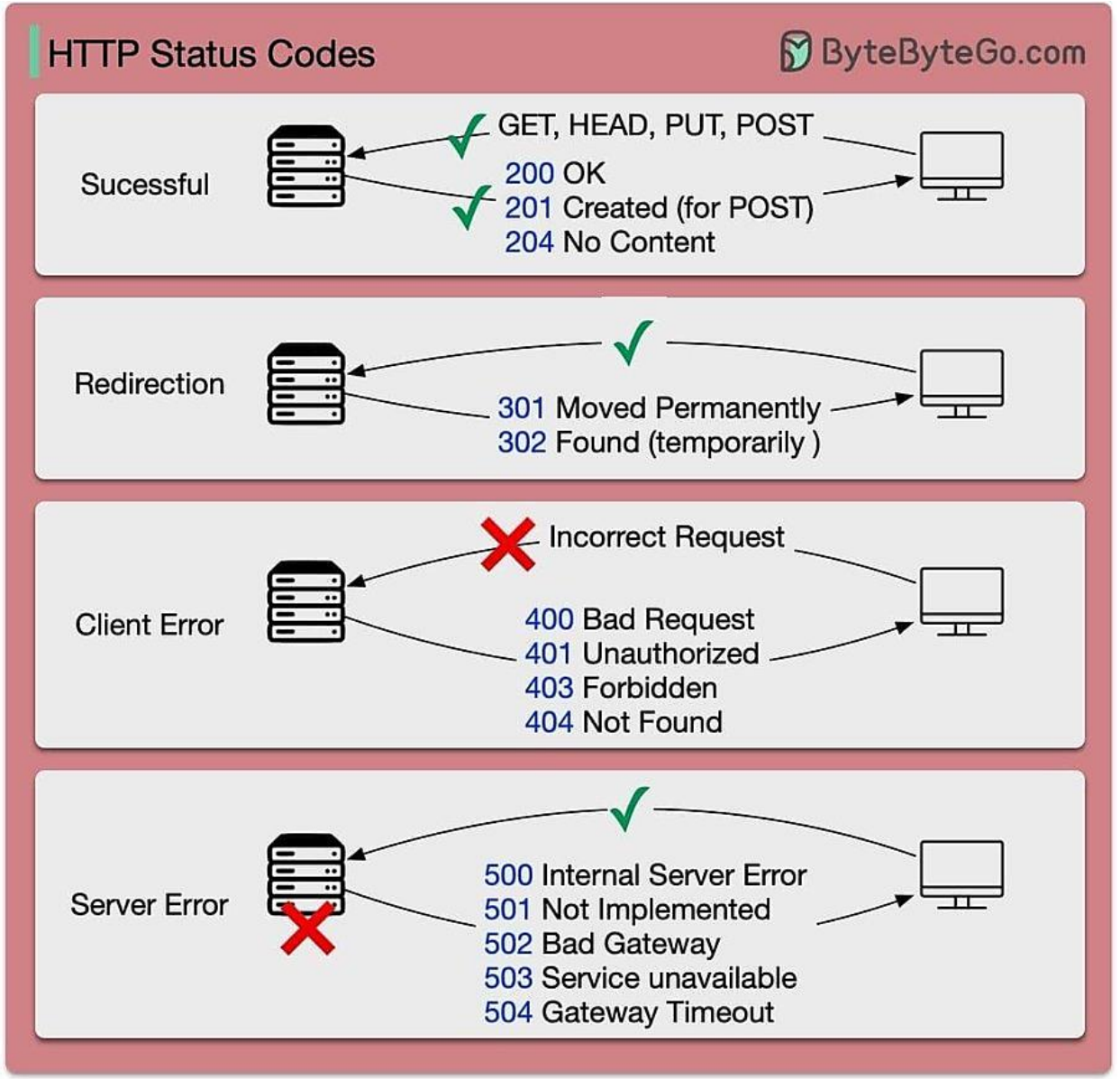
- Son multiplataforma (Windows, Linux...)
- Ambos se instalan como servicio en 2º plano
- Ambos permiten interactuar con ellos con las órdenes que controlan un servicio estándar
  - Ej.: En Ubuntu sería `sudo service (stop | start | restart | status) apache2 / ngnix`
- Al instalarse se auto arrancan
- **Cada vez que cambiemos su configuración hay que hacerles un `restart`**
  - **Ej.:** `sudo service apache2 restart`



RESPUESTAS DE UN SERVIDOR WEB

José Manuel
Redondo López

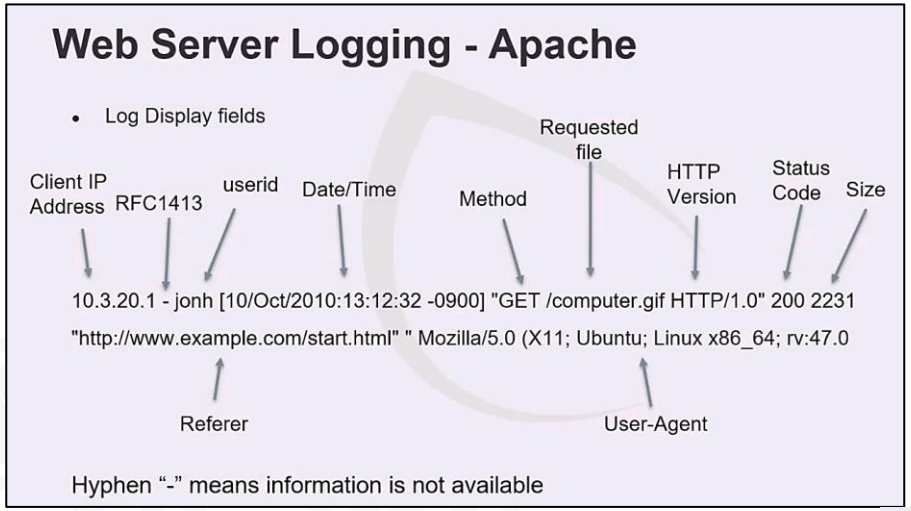
- Da igual el servidor web que uses, el protocolo HTTP asegura una respuesta idéntica a las peticiones
- Los servidores deberían reaccionar de una forma equivalente a las peticiones que se hagan a una misma web alojada
- Estas respuestas son los conocidos HTTP Status Codes

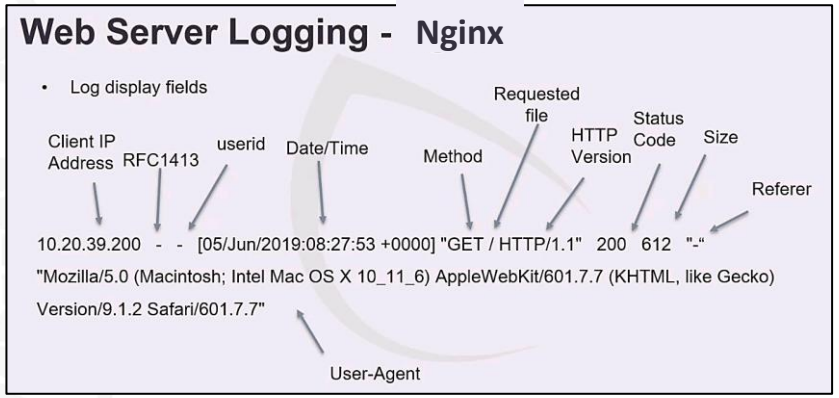


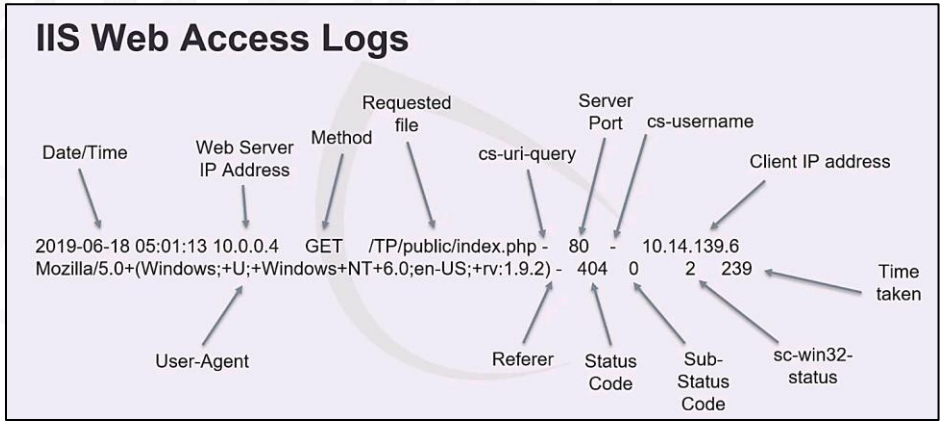
LOG DE UN SERVIDOR WEB

José Manuel
Redondo López

- TODOS los servidores web guardan log de sus actividades
- Cualquier petición (con éxito o no) deja huella en un fichero de log
- Todo lo que pasa al servidor está registrado
  - Se puede usar para depurar cuando una aplicación web no funciona
  - O también para ver si han sufrido un ataque
  - Esto habitualmente se analiza con herramientas automatizadas
    - Lo veremos en la "**F-103 Blas de Lezo**"





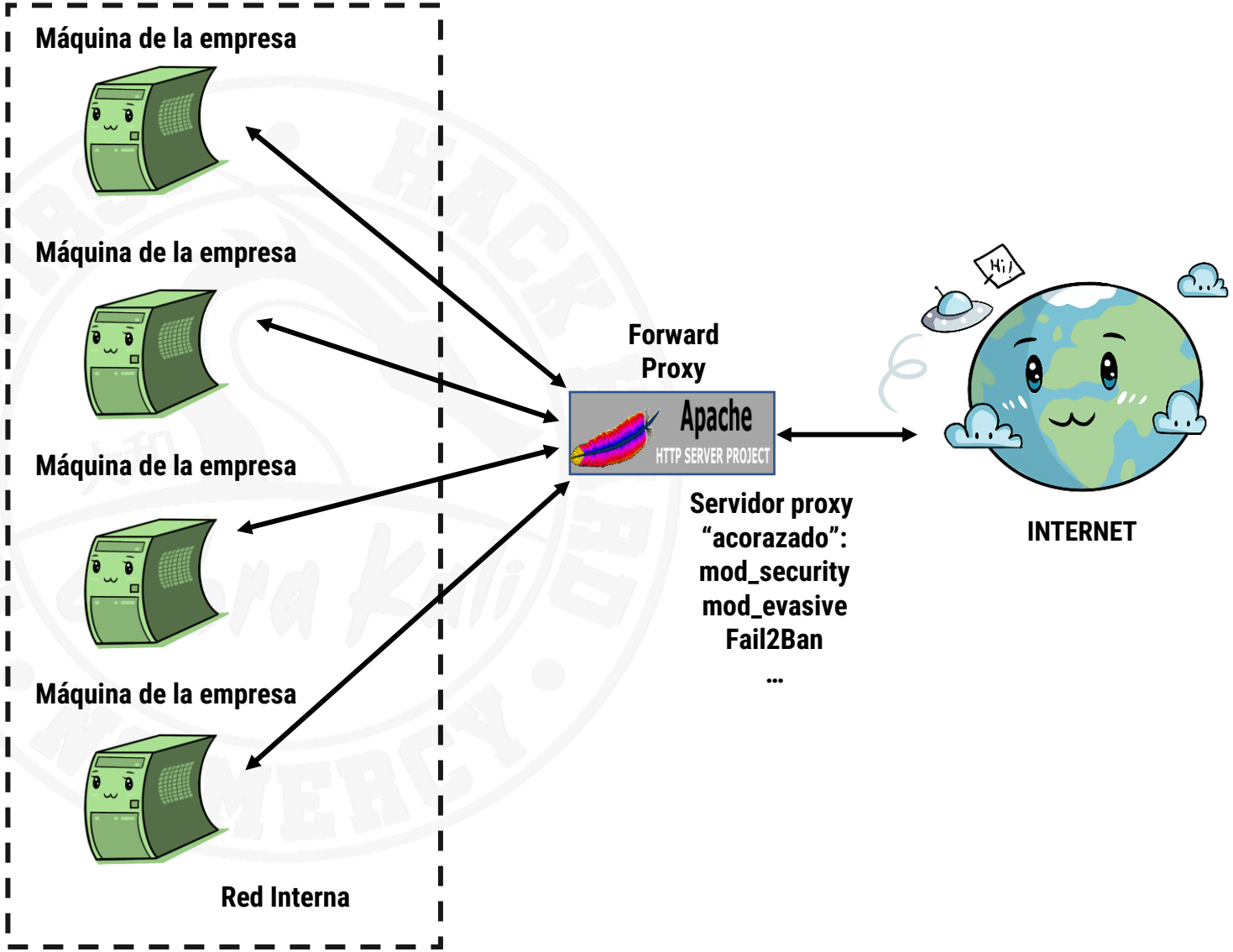


Fuentes: https://twitter.com/guillermoxd15

Concepto de Forward proxy

José Manuel
Redondo López

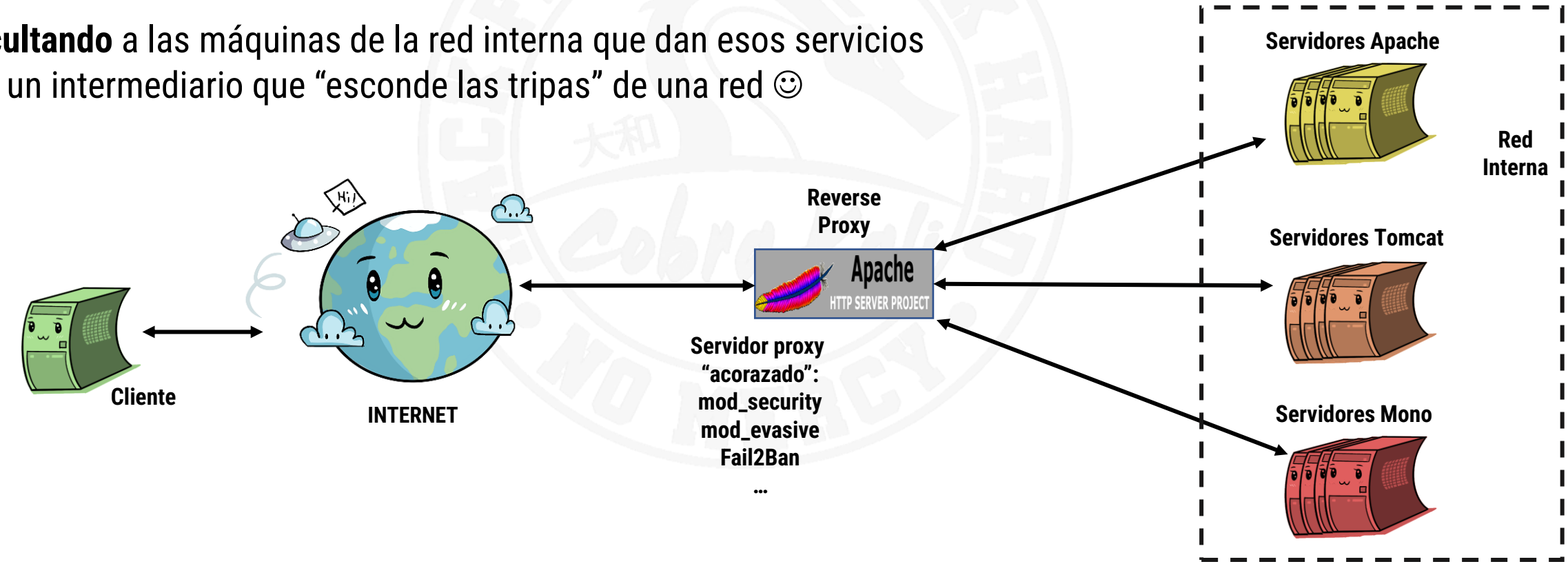
- Ambos servidores pueden funcionar como **forward proxy**
  - Tienen módulos para ello
- Son máquinas por la que pasa **todo el tráfico** de una determinada red
- Esto les permite
  - Filtrarlo
  - Controlarlo
  - Examinarlo



Reverse Proxy

José Manuel
Redondo López

- También ambos pueden funcionar como **reverse proxy** gracias a sus módulos
- Servidor para clientes de los servicios que hay "detrás" de él
  - **Ocultando** a las máquinas de la red interna que dan esos servicios
  - Es un intermediario que "esconde las tripas" de una red ☺



¿Crees que lo has entendido todo? ¡Autoevalúate!

José Manuel
Redondo López

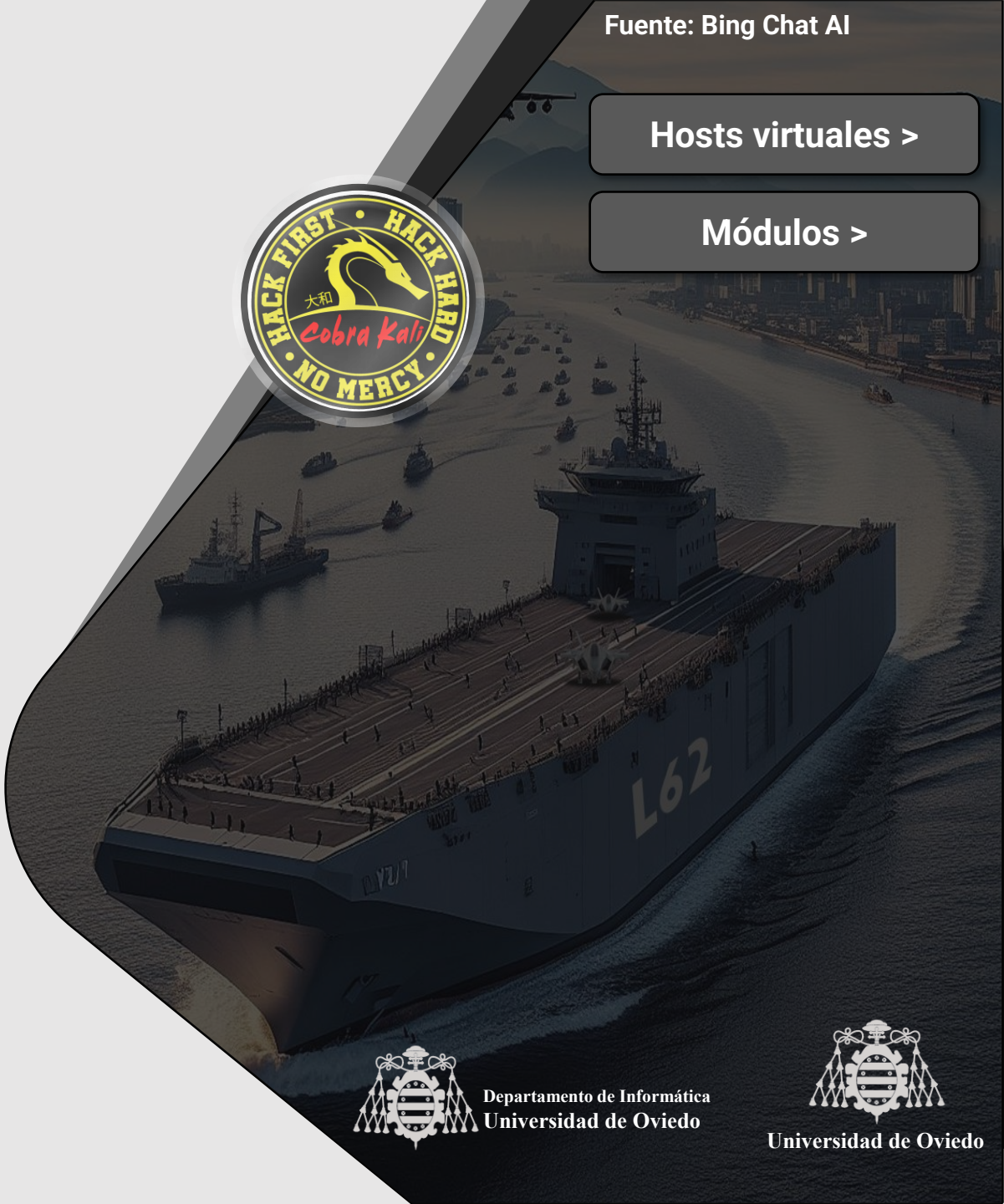
?

- *¿Eres capaz de enumerar qué cosas son comunes a Apache y Nginx? (ruta por defecto, usuario que sirva las webs...)*
- *¿Tienes claras las instrucciones de manejo básico del proceso que controla el servidor web?*
- *¿Entiendes los conceptos de forward proxy y reverse proxy y para qué pueden servir en una infraestructura?*

< Ir al Índice

Hosts virtuales >

Módulos >



Servidor Web Apache 2

Aspectos y conceptos generales

Departamento de Informática
Universidad de Oviedo

Universidad de Oviedo

Dónde y cómo se instala

José Manuel
Redondo López

- **Apache 2** puede instalarse fácilmente como un paquete del sistema de paquetes de cualquier distribución **Linux**
  - Ej. para **Debian**: `sudo apt apache2`
- Para **Windows**, puede compilarse uno mismo los binarios o instalar los binarios de algunos de los proveedores recomendados en su web oficial
  - https://httpd.apache.org/docs/2.4/es/platform/windows.html
  - No obstante, habitualmente para Windows suele usarse el servidor web de Microsoft, IIS

DÓNDE Y CÓMO SE INSTALA

José Manuel
Redondo López

- **Se instala por defecto en `/etc/apache2`, que dentro tiene**
  - `apache2.conf`: el archivo de configuración principal de Apache, con las **opciones globales** a todas las webs y que **incluye otros** `.conf` en el sistema de carpetas del servidor
    - Este será el **archivo de configuración principal** que usemos
  - Otros ficheros que normalmente **no necesitaremos tocar**
    - `envvars`: archivo en el que se establecen las **variables de entorno** de Apache
    - `magic`: instrucciones para determinar el tipo MIME en base a los primeros bytes de un archivo
    - `ports.conf`: almacena las **directivas** que determinan los puertos TCP en los que Apache escucha

```
/etc/apache2/
|-- apache2.conf
|       `--  ports.conf
|-- mods-enabled
|       |-- *.load
|       `-- *.conf
|-- conf-enabled
|       `-- *.conf
`-- sites-enabled
        `-- *.conf
```

```
drwxr-xr-x 8 root root  4096 Aug 31 16:21 .
drwxr-xr-x 1 root root  4096 Aug 31 16:21 ..
-rw-r--r-- 1 root root  7224 Jun 14 12:30 apache2.conf
drwxr-xr-x 2 root root  4096 Aug 31 16:21 conf-available
drwxr-xr-x 2 root root  4096 Aug 31 16:21 conf-enabled
-rw-r--r-- 1 root root  1782 Mar 23 02:00 envvars
-rw-r--r-- 1 root root 31063 Mar 23 02:00 magic
drwxr-xr-x 2 root root 12288 Aug 31 16:21 mods-available
drwxr-xr-x 2 root root  4096 Aug 31 16:21 mods-enabled
-rw-r--r-- 1 root root   320 Mar 23 02:00 ports.conf
drwxr-xr-x 2 root root  4096 Aug 31 16:21 sites-available
drwxr-xr-x 2 root root  4096 Aug 31 16:21 sites-enabled
root@802d3ac53895:/etc/apache2#
```

Estructura de directorios de Apache 2

José Manuel
Redondo López

- Directorios relacionados con la **configuración**
  - `conf-available`: contiene **todos los archivos de configuración** disponibles
  - `conf-enabled`: contiene **enlaces simbólicos** a ficheros de `/etc/apache2/conf-available`
    - Son configuraciones de módulos que **serán habilitados en el siguiente arranque** de Apache
    - También hay ficheros de configuración aparte, como `security.conf`
      - Contienen configuraciones de un aspecto concreto
      - "Descargan" al principal de contener opciones relativas a estos aspectos (**refactorización**)
- Directorios relacionados con los **módulos** que amplían la funcionalidad del servidor
  - `mods-available`: contiene los archivos de configuración de todos los módulos que **podrían cargarse**
  - `mods-enabled`: mantiene **enlaces simbólicos** a archivos de `/etc/apache2/mods-available`
    - Si se crea aquí un enlace simbólico a la configuración de un módulo, éste **se activará cuando se reinicie** Apache
- Directorios relacionados con los **sitios web** que sirve dicho servidor
  - `sites-available`: contiene archivos de configuración para **servidores virtuales** (Virtual Host)
    - Permiten que Apache pueda servir **múltiples sitios web** que tengan configuraciones separadas.
  - `sites-enabled`: contiene **enlaces simbólicos** al directorio `/etc/apache2/sites-available`
    - Si un archivo de configuración de `sites-available` tiene un enlace simbólico en este directorio, Apache permitirá el **acceso al sitio web configurado en él** la próxima vez que se reinicie

Glosario de términos

José Manuel
Redondo López

- **Módulo**: Extensión de la funcionalidad de **Apache**, que puede activarse o no
- **Host virtual**: Unidad de configuración de **Apache** a la que se asigna una web
  - Le permite alojar múltiples webs al mismo tiempo
  - Las webs se distinguen entre ellas por IP, puerto, nombre de dominio…
- **Contenedor de directivas**: Etiqueta que contiene directivas (opciones de configuración) de módulos y que determina a qué parte de los recursos se aplican
  - `<VirtualHost>`: Directivas para **un host virtual** (web) concreto
  - `<Directory>`: Directivas sobre los contenidos de **un directorio físico**
  - `<Location>`: Directivas sobre **una parte de una URI** de la petición
    - No tiene por qué corresponder a una carpeta física
  - `<Files>`: Directivas para **extensiones de archivos concretas**
  - Ficheros `.htaccess`: Directivas **solo para la carpeta** que los contiene
    - Sobrescribiendo a las directivas generales

```
<VirtualHost *:80>
        # The ServerName directive sets the request scheme, hostname and port th
at
        # the server uses to identify itself. This is used when creating
        # redirection URLs. In the context of virtual hosts, the ServerName
        # specifies what hostname must appear in the request's Host: header to
        # match this virtual host. For the default virtual host (this file) this
        # value is not decisive as it is used as a last resort host regardless.
        # However, you must set it for any further virtual host explicitly.
        #ServerName www.example.com

        ServerAdmin webmaster@localhost
        DocumentRoot /var/www/html

        # Available loglevels: trace8, ..., trace1, debug, info, notice, warn,
        # error, crit, alert, emerg.
        # It is also possible to configure the loglevel for particular
        # modules, e.g.
        #LogLevel info ssl:warn

        ErrorLog ${APACHE_LOG_DIR}/error.log
        CustomLog ${APACHE_LOG_DIR}/access.log combined

        # For most configuration files from conf-available/, which are
        # enabled or disabled at a global level, it is possible to
        # include a line for only one particular virtual host. For example the
        # following line enables the CGI configuration for this host only
        # after it has been globally disabled with "a2disconf".
        #Include conf-available/serve-cgi-bin.conf
</VirtualHost>
```

EJEMPLOS DE CONTENEDORES DE DIRECTIVAS

José Manuel
Redondo López

```
 GNU nano 2.9.3                                  vhost-macro.conf

<Macro VHost $domain>
<VirtualHost *:80>
        ServerName $domain
        ServerAlias www.$domain

        DocumentRoot "/www/$domain/www/public_html"

        <Directory "/www/$domain/www/public_html">
                Options -FollowSymLinks +MultiViews +Indexes
                AllowOverride all
                Require all granted
        </Directory>

        ErrorLog "/www/$domain/www/logs/error.log"
        CustomLog "/www/$domain/www/logs/access.log" combined
</VirtualHost>
</Macro>
```

```
 GNU nano 2.2.6          File: 000-default.conf                Modified

<VirtualHost *:80>
        # The ServerName directive sets the request scheme, hostname and port t$
        # the server uses to identify itself. This is used when creating
        # redirection URLs. In the context of virtual hosts, the ServerName
        # specifies what hostname must appear in the request's Host: header to
        # match this virtual host. For the default virtual host (this file) this
        # value is not decisive as it is used as a last resort host regardless.
        # However, you must set it for any further virtual host explicitly.
        #ServerName www.example.com

        ServerAdmin webmaster@localhost
        DocumentRoot /var/www/html

        <Location "/server-status">
        SetHandler server-status
        Require ip 192.168
        </Location>
```

```
NameVirtualHost *:80
NameVirtualHost *:443
<VirtualHost  *:80>
    ServerAdmin admin@ilovelinux.com
    ServerAlias www.ilovelinux.com ilovelinux.com
    DocumentRoot /var/www/html/ilovelinux.com/public_html
    ServerName ilovelinux.com
    ErrorLog logs/ilovelinux.com-error_log
    CustomLog logs/ilovelinux.com-access_log common
</VirtualHost>
<VirtualHost  *:443>
    ServerAdmin admin@ilovelinux.com
    ServerAlias www.ilovelinux.com ilovelinux.com
    DocumentRoot /var/www/html/ilovelinux.com/public_html
    ServerName ilovelinux.com
    ErrorLog /var/www/html/ilovelinux.com/error_log
    LogFormat "%v %l %u %t \"%r\" %>s %b" myvhost
    CustomLog /var/www/html/ilovelinux.com/access_log    myvhost
    SSLEngine on
    SSLCertificateFile /etc/httpd/ssl-certs/apache.crt
    SSLCertificateKeyFile /etc/httpd/ssl-certs/apache.key
</VirtualHost>
```

http://www.tecmint.com

- ¡Consulta el libro de Apache adjunto para más información sobre contenedores de directivas!

Hosts virtuales de Apache 2

Formas de hacer que un mismo servidor sirva varias webs distintas

Creación de host virtuales

José Manuel
Redondo López

- **Cada host virtual representa a una web distinta**
  - Permitiendo a un mismo servidor Apache **servir varias webs de forma transparente**
- **Módulos y host virtuales se manejan igual**
  - Cada uno **tiene su propio archivo de configuración**, con directivas que afectarán sólo a ese sitio web
  - Como vimos, estos archivos **deben estar en el directorio** `/etc/apache2/sites-available`
  - Es **aconsejable** que el nombre de los hosts virtuales sea **el del dominio** que representan
    - Y que sus archivos de configuración se llamen también así
    - ¡Eso facilitará el manejo más de lo que imaginas!
- **Tener varios hosts virtuales puede complicar la configuración, por lo que se recomiendan dos medidas**
  - Si hay configuraciones comunes a varios, se pueden **refactorizar a un fichero**
    - E integrarlas con `Include`
    - https://httpd.apache.org/docs/2.4/es/mod/core.html#include
  - Conviene hacer un **test de la configuración** antes de reiniciar el servidor con `apachectl configtest`
    - ¡Evitaremos quedarnos sin servidor temporalmente por un error en los ficheros!
    - https://httpd.apache.org/docs/2.4/programs/apachectl.html

Creación de host virtuales

José Manuel
Redondo López

- **Para activar y desactivar los hosts virtuales podemos usar los comandos `a2ensite` y `a2dissite`**
- **Los hosts virtuales activos en un momento dado pueden verse en `/etc/apache2/sites-enabled/`**
  - En este directorio debe existir **al menos un host virtual**,
  - El Default Virtual Host en el fichero `000-default`
- **Todos los hosts virtuales disponibles aparecen listados en el directorio `/etc/apache2/sites-available`**
  - **RECUERDA**: Estos son los que **se podrían activar** en un momento dado
  - ¡**NO** están activos salvo que también estén en `sites-enabled`!

Creación de host virtuales

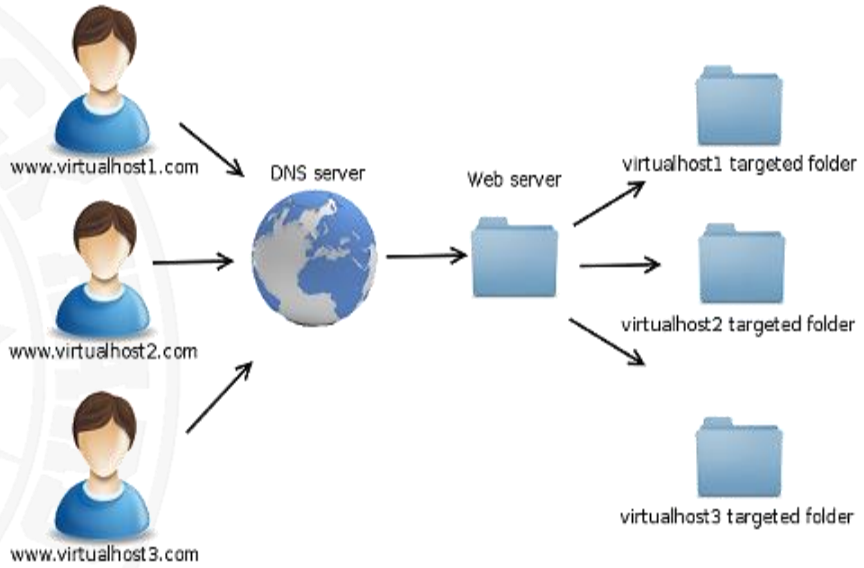
José Manuel
Redondo López

- **Hay dos tipos diferentes de host virtuales, basados en IP y basados en nombre de dominio**
- **Están basados en IP si un mismo servidor tiene múltiples IP asignadas**
  - Múltiples tarjetas de red, o una misma tarjeta con varias IP asignadas (ambas cosas son posibles)
  - Cada nombre de dominio que sirve se asocia a una de esas IP
  - Por tanto, **cada web tiene una IP diferente**, aunque están servidas por un mismo servidor físico
  - Esto es típico de **redes internas para pruebas**
    - Casos en los que usar la IP para acceder a una web no importe

Creación de host virtuales

José Manuel
Redondo López

- **Están basados en nombre de dominio cuando el servidor tiene una única dirección IP asociada**
  - Y **varios nombres de dominio que se asocian con esa IP**
  - Lo normal con **servidores expuestos a Internet**
- **Los diferentes sitios web se distinguen en este caso por cada nombre de dominio (DNS) asociado**
  - Requiere que los nombres de dominio usados tengan una **entrada válida en un servidor DNS** ajeno a Apache 2
    - Todos esos nombres dirigirán a la IP única del servidor
  - Apache usará cada nombre de dominio que le llegue a su IP para **escoger qué host virtual** atenderá la petición
  - También funcionan así los nombres en el fichero `hosts` del SO
    - Si no tenemos DNS, podemos usar este "truco" para pruebas



Host virtual basado en nombre de dominio. Fuente: https://blog.baehost.com/tutorial-como-configurar-virtual-host-en-un-cloud-server/

CREACIÓN DE HOST VIRTUALES

José Manuel
Redondo López

- **El siguiente ejemplo muestra dos nombres de dominio diferentes (`ejemplo.org` y `ejemplo.net`), que se alojarán en el mismo servidor**
  - Ambos están usando un `DocumentRoot` (directorio base donde se copian los contenidos de la web), logs de acceso y de error separados

```
#Virtual Host 1 – ejemplo.org
<VirtualHost*:80>
    ServerName www.ejemplo.org
    ServerAdmin admin@ejemplo.org
    DocumentRoot /var/www/html/www.example.org
    CustomLog /var/log/apache2/www.example.org-access_log
    ErrorLog/var/log/apache2/www.example.org-error_log
</VirtualHost>
#Virtual Host 2 – ejemplo.net
<VirtualHost *:80>
    ServerName www.ejemplo.net
    ServerAdmin admin@ejemplo.net
    DocumentRoot /var/www/html/www.example.net
    CustomLog /var/log/apache2/www.example.net-access_log
    ErrorLog/var/log/apache2/www.example.net-error_log
</VirtualHost>
```

CREACIÓN DE HOST VIRTUALES

José Manuel
Redondo López

- El proceso para crear un host virtual sería (ver imagen)
  - Ir a `/etc/apache2/sites-available`
  - **Crear un archivo** `site2.conf` con el contenido de la imagen
    - En este ejemplo el host virtual está asociado a la IP `192.168.56.3`
    - Usa sus propios ficheros de log de acceso y errores para no mezclar la información de otros sitios (**recomendado**)
  - **Crear el directorio** `/var/www/html/site2` para que haga de `DocumentRoot` del nuevo sitio web
  - **Mover las páginas web** a servir a ese directorio
  - **Ejecutar el comando** `a2ensite site2` para habilitar el host virtual
  - Reiniciar Apache
  - Ahora debería existir una entrada para `site2` en el directorio `sites-enabled` de Apache 2
- Más ejemplos: https://httpd.apache.org/docs/2.4/vhosts/examples.html

```
 GNU nano 2.2.6          Archivo: site2.conf

<VirtualHost 192.168.56.3>
        ServerName Site2
        DocumentRoot /var/www/html/site2
        ErrorLog /var/log/apache2/site2-error.log
        TransferLog /var/log/apache2/site2-access.log
</VirtualHost>
```

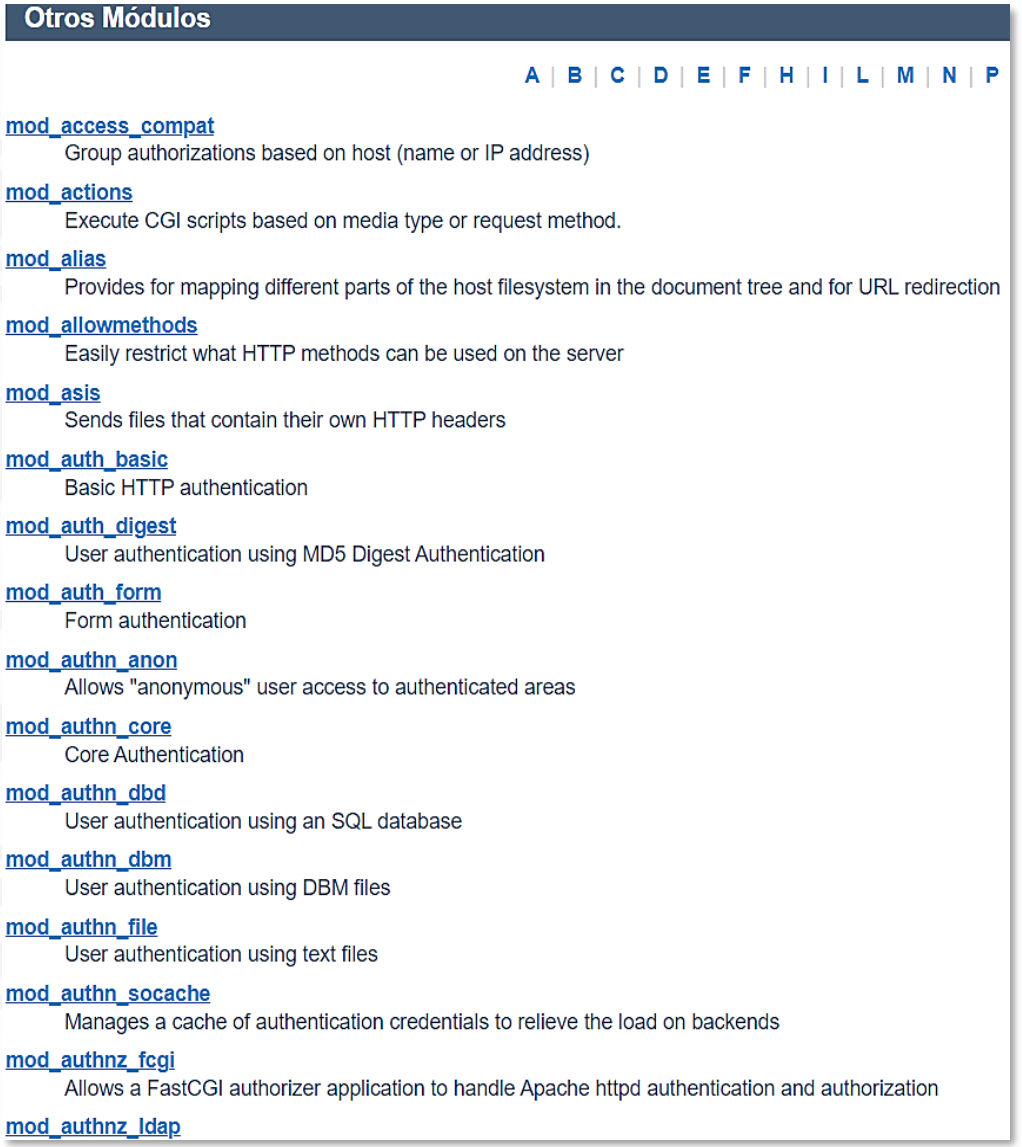
Módulos de Apache 2

Ampliando la funcionalidad del servidor

Procesamiento de peticiones de Apache

José Manuel
Redondo López

- **La funcionalidad de Apache está regida por los módulos que tenga activos**
  - Se pueden **ampliar sus funcionalidades** modificando su cadena de peticiones con los **módulos existentes en** https://httpd.apache.org/docs/2.4/es/mod/
    - Internacionalización independiente del lenguaje
    - Soporte de lenguajes de servidor adicionales
    - Activar filtros de entrada y salida personalizados que manipulen los contenidos como queramos
    - Compresión automática de contenidos
    - …
  - También se pueden **programar módulos nuevos**
    - Esto es un tema avanzado que no veremos
- **Solo tenemos que consultar el catálogo y elegir el que nos conviene**
  - Todos se suelen instalar de la misma forma



Otros Módulos

A | B | C | D | E | F | H | I | L | M | N | P

mod_access_compat
Group authorizations based on host (name or IP address)

mod_actions
Execute CGI scripts based on media type or request method.

mod_alias
Provides for mapping different parts of the host filesystem in the document tree and for URL redirection

mod_allowmethods
Easily restrict what HTTP methods can be used on the server

mod_asis
Sends files that contain their own HTTP headers

mod_auth_basic
Basic HTTP authentication

mod_auth_digest
User authentication using MD5 Digest Authentication

mod_auth_form
Form authentication

mod_authn_anon
Allows "anonymous" user access to authenticated areas

mod_authn_core
Core Authentication

mod_authn_dbd
User authentication using an SQL database

mod_authn_dbm
User authentication using DBM files

mod_authn_file
User authentication using text files

mod_authn_socache
Manages a cache of authentication credentials to relieve the load on backends

mod_authnz_fcgi
Allows a FastCGI authorizer application to handle Apache httpd authentication and authorization

mod_authnz_ldap

INSTALAR, ACTIVAR Y DESACTIVAR MÓDULOS

José Manuel
Redondo López

- Para poder usar un módulo **hay que instalarlo**
  - Seguramente con **el gestor de paquetes del SO (Ej.:** `apt`**)**
  - Esto lo colocará al menos en el directorio `mods-available`
- Para activar un módulo en `mods-available` pero aún no en `mods-enabled`
  - `sudo a2enmod <nombre del fichero .load del módulo>`
- Para desactivarlo
  - `sudo a2dismod <nombre del fichero .load del módulo>`
  - Hay que intentar **minimizar el nº de módulos**
- En ambos casos hay que **reiniciar Apache** para que el cambio tenga efecto
  - `sudo service apache2 restart`
- Cada módulo activo añade **nuevas directivas** que podremos usar en los ficheros de configuración

```
actions.conf            dir.load                proxy_express.load
actions.load            dump_io.load            proxy_fcgi.load
alias.conf              echo.load               proxy_fdpass.load
alias.load              env.load                proxy_ftp.conf
allowmethods.load       expires.load            proxy_ftp.load
asis.load               ext_filter.load         proxy_hcheck.load
auth_basic.load         file_cache.load         proxy_html.conf
auth_digest.load        filter.load             proxy_html.load
auth_form.load          headers.load            proxy_http.load
authn_anon.load         heartbeat.load          proxy_http2.load
authn_core.load         heartmonitor.load       proxy_scgi.load
authn_dbd.load          http2.conf              proxy_uwsgi.load
authn_dbm.load          http2.load              proxy_wstunnel.load
authn_file.load         ident.load              ratelimit.load
authn_socache.load      imagemap.load           reflector.load
authnz_fcgi.load        include.load            remoteip.load
authnz_ldap.load        info.conf               reqtimeout.conf
authz_core.load         info.load               reqtimeout.load
authz_dbd.load          lbmethod_bybusyness.load request.load
authz_dbm.load          lbmethod_byrequests.load rewrite.load
authz_groupfile.load    lbmethod_bytraffic.load sed.load
authz_host.load         lbmethod_heartbeat.load session.load
authz_owner.load        ldap.conf               session_cookie.load
authz_user.load         ldap.load               session_crypto.load
autoindex.conf          log_debug.load          session_dbd.load
autoindex.load          log_forensic.load       setenvif.conf
brotli.load             lua.load                setenvif.load
buffer.load             macro.load              slotmem_plain.load
cache.load              md.load                 slotmem_shm.load
cache_disk.conf         mime.conf               socache_dbm.load
cache_disk.load         mime.load               socache_memcache.load
cache_socache.load      mime_magic.conf         socache_redis.load
cern_meta.load          mime_magic.load         socache_shmcb.load
cgi.load                mpm_event.conf          speling.load
cgid.conf               mpm_event.load          ssl.conf
cgid.load               mpm_prefork.conf        ssl.load
charset_lite.load       mpm_prefork.load        status.conf
data.load               mpm_worker.conf         status.load
dav.load                mpm_worker.load         substitute.load
dav_fs.conf             negotiation.conf        suexec.load
dav_fs.load             negotiation.load        unique_id.load
dav_lock.load           proxy.conf              userdir.conf
dbd.load                proxy.load              userdir.load
deflate.conf            proxy_ajp.load          usertrack.load
deflate.load            proxy_balancer.conf     vhost_alias.load
dialup.load             proxy_balancer.load     xml2enc.load
```

Ejemplo de módulo: Restricción de acceso

https://httpd.apache.org/docs/2.4/es/mod/mod_authz_host.html

José Manuel
Redondo López

- `mod_authz_host` **permite restringir/permitir accesos a rutas o ficheros del servidor por**
  - IP del cliente (incluyendo subredes)
  - Listas de IPs
  - Nombres de dominio
  - IPV6
  - Presencia o no de variables de entorno
  - Sólo máquina local
  - Método HTTP
  - Expresiones evaluadas con variables de entorno (gran flexibilidad)
- **Los problemas con el servidor o con alguno de sus módulos se ven en sus logs**
  - `/var/log/apache2/access.log`: Log de cada acceso al servidor
  - `/var/log/apache2/error.log`: Log de cada error que ocurra en el servidor
  - ... (ficheros de log particulares que puedan crear los módulos)

Ejemplo de módulo: Módulos de autenticación

José Manuel
Redondo López

- Consiste en requerir una identidad de usuario vía HTTP para ver ciertos contenidos o partes de una web
- Se puede conseguir con medios locales o externos
  - **Locales**: Ficheros de diferente tipo existentes en el disco duro de la máquina con usuarios y claves
    - Basic (https://httpd.apache.org/docs/2.4/es/mod/mod_auth_basic.html)
    - Digest (https://httpd.apache.org/docs/2.4/es/mod/mod_auth_digest.html)
    - ...
  - **Externos**: Almacenes de usuarios y claves a los que podamos acceder desde el servidor
    - Datos de usuarios en LDAPs (https://httpd.apache.org/docs/2.4/es/mod/mod_authnz_ldap.html)
    - Usando BBDD externas (MySQL…) (https://httpd.apache.org/docs/2.4/es/mod/mod_authz_dbd.html)
    - ...
- En general debemos ir a la lista de módulos, buscar por la funcionalidad que queramos y leernos bien la documentación e instrucciones de puesta en marcha
  - ¡Esto es así **para cualquier cosa** que queramos hacer con el servidor!

¿Crees que lo has entendido todo? ¡Autoevalúate!

José Manuel
Redondo López

?

Conceptos de Apache 2

- *¿Eres capaz de recordar los principales ficheros y directorios que contienen la configuración, módulos y sitios web del servidor web Apache 2 y para qué se usan?*
- *¿Sabes lo que son y lo que permiten los módulos de Apache 2?*
- *¿Sabes lo que son y lo que permiten los hosts virtuales de Apache 2?*
- *¿Entiendes lo que son los contenedores de directivas de la configuración de Apache 2 y los distintos ámbitos de aplicación que tienen los mismos?*

Hosts virtuales

- *¿Entiendes lo que es un host virtual y cómo se crean?*
- *¿Recuerdas qué dos tipos de host virtuales existen y en qué escenarios es más típico usarlos?*
- *¿Sabes cómo definir un host virtual adecuadamente, de forma que sus contenidos estén separados de los de otros hosts virtuales?*

Módulos

- *¿Entiendes que Apache 2 tiene una librería de módulos muy grande y cómo activar cualquiera de ellos?*
- *¿Entiendes que existe un módulo para restringir el acceso por un gran nº de criterios?*
- *¿Entiendes que existen muchos módulos para autenticar al usuario contra distintos almacenes de nombres de usuario / password?*

< Ir al Índice

Fuente: Bing Chat AI

Contextos >

Server blocks >

Módulos >

Servidor Web Nginx

Aspectos y conceptos generales

Departamento de Informática
Universidad de Oviedo

Universidad de Oviedo

Instalación de Nginx

José Manuel
Redondo López

- **Hay dos versiones de Nginx**
  - La versión open source (que veremos en esta presentación)
  - Una de pago llamada Nginx Plus: https://www.nginx.com/products/nginx/
- **Está disponible en los repositorios de software oficiales de cualquier Linux**
  - Su instalación en formato paquete es trivial
  - Ej.: en Debian/Ubuntu `sudo apt-get install nginx`
- **Para Windows: https://nginx.org/en/docs/windows.html**
  - Aunque vuelvo a insistir que en Windows es mejor usar el IIS de Microsoft
- **Al igual que Apache 2 ...**
  - La funcionalidad del servidor se puede ampliar con **módulos**
  - Se pueden alojar múltiples webs en un mismo servidor
    - En Nginx se llaman server blocks en lugar de hosts virtuales, pero el concepto es el mismo

Ficheros importantes (Debian)

José Manuel
Redondo López

- **Al igual que con Apache, es importante tener claras una serie de localizaciones**
  - `/var/www/html`: Directorio donde están por defecto las webs que sirve, se puede cambiar
  - **Distintas partes de la configuración del servidor**:
    - `/etc/nginx`: Carpeta donde se guarda la configuración de Nginx
    - `/etc/nginx/nginx.conf`: Fichero principal de configuración global (como `apache2.conf`)
    - `/etc/nginx/sites-available`: Contiene las configuraciones de las webs que se pueden servir
      - Los "server blocks" que mencionamos antes
    - `/etc/nginx/sites-enabled`: Ficheros de configuración de **server blocks** en `sites-available` enlazados, representando las configuraciones de todas las webs que el servidor sirve realmente
    - `/etc/nginx/snippets`: Contiene ficheros con **fragmentos de configuraciones** que pueden ser usados en otras configuraciones, refactorizándolas
  - **Logs del servidor**
    - `/var/log/nginx/access.log`: Logs de cada acceso al servidor
    - `/var/log/nginx/error.log`: Logs de cada error que ocurra en el servidor
    - ... (otros ficheros de log creados por los módulos de Nginx)

Contextos de configuración de Nginx

Organizando la configuración del servidor

Contextos de configuración de Nginx

José Manuel
Redondo López

- Para trabajar con **Nginx** es muy importante entender **cómo se estructuran** sus ficheros de configuración
  - Tienen **forma de árbol**, formado por bloques entre { } llamados **contextos**
  - Cada contexto representa un **área** en el que la serie de configuraciones que contiene **se aplica**
  - Los contextos **pueden anidarse** unos dentro de otros, formando la estructura de la configuración
    - Si un contexto está dentro de otro, **hereda todas las directivas de su contenedor**
    - Además, un contexto "hijo" puede **sobrescribir el valor** de cualquier directiva heredada de sus padres
- Muchas directivas están pensadas para ser usadas solo en determinados tipos de contextos
  - Ngnix **dará un error de configuración** si tratamos de usarlas en otros
  - La **documentación** de cada directiva contiene esta información

Contextos "Core"

José Manuel
Redondo López

- **Hay unos contextos principales ("Core") en la configuración de Nginx**
  - **El principal**: El único contexto que **no está contenido en otro**
    - Cualquier directiva que exista **fuera de un bloque** de un contexto pertenece al contexto principal
    - Contiene configuraciones que afectan **a toda una aplicación**
    - Si se importa un fichero de configuración desde otro, las directivas del main context del fichero importado **pasan a formar parte** del contexto que hace la importación del fichero
      - No se añaden al main context del fichero que lo importa
    - Muchas de las directivas que van dentro del mismo **no pueden ser heredadas**
      - Los contextos hijos no pueden redefinirlas
    - **Contiene configuraciones como**: el usuario y grupo que usan los procesos worker, el nº de procesos worker, afinidad por determinadas CPUs, el fichero de log de errores, …
    - No lo modificaremos en este curso

```
# Todo lo definido aquí
# fuera de otro contexto
# pertenece al principal
. . .
http {
    . . .
}
```

CONTEXTO "HTTP"

José Manuel
Redondo López

- **El contexto "`events`"**: Solo puede haber uno en toda la configuración de Nginx
  - Con él se configura cómo gestiona las conexiones el servidor a nivel general
  - Tampoco vamos a usarlo en este curso
- **El contexto "`http`"**: Si Nginx se usa como servidor web o proxy, **la mayoría de la configuración se incluirá en este contexto**
  - Define cómo gestiona el servidor **cualquier conexión tipo HTTP o HTTPs**
  - Se debe definir como parte del main context, no anidado en otro
  - Se usa para establecer **cómo se comportan por defecto** todos los server blocks que definamos en el servidor
    - Cada server block puede dar un valor concreto a muchas de las directivas que definamos aquí si la configuración por defecto no le sirve
  - **Contiene directivas como**: localización por defecto de **logs de peticiones** y error, páginas de error, compresión de peticiones, configurar el parámetro Keep Alive, reglas de optimización…
  - **Más información** http://nginx.org/en/docs/http/ngx_http_core_module.html#http

```
# Contexto principal
…
events {
    . . .
}
http {
    . . .
}
```

Contexto "Server"

José Manuel
Redondo López

- Se declaran dentro del `http`, **uno por cada server block**
  - **Cada petición solo puede ser atendida por un solo contexto** `server`
- Se usa información de la petición para seleccionar el adecuado, según una de estas directivas
  - `listen`: La dirección **IP / puerto** que el contexto `server` atiende
  - `server_name`: Sólo si hay varios bloques `server` asociados a la misma IP y puerto, con **nombres de dominio** distintos
    - Nginx usa la cabecera HTTP Host Header para seleccionar el que encaje con el valor de esta directiva
- Las directivas de `server` pueden redefinir muchas de las opciones definidas en `http`
  - Además de configurar ficheros para **intentar responder a ciertas peticiones** (`try_files`), hacer **redirecciones** (`return`), **reescrituras** (`rewrite`) y **modificar valores** (`set`)
  - **Más información**
    - http://nginx.org/en/docs/http/ngx_http_core_module.html#server

```
# main context
http {
    # http context
    server {
        # Primer contexto server
    }
    server {
        # Segundo contexto server
    }
}
```

Estos contextos son la forma de conseguir que un mismo Nginx sirva varias webs distintas (lo veremos en la siguiente sección)

Server blocks de Nginx

Formas de hacer que un mismo servidor sirva varias webs distintas

Contexto "Server": Manejando server blocks

José Manuel
Redondo López

- Si te das cuenta, los **server block** hacen lo mismo que los host virtuales en **Apache**
- Crear varias webs para que **Nginx** las sirva (varios **server blocks**) es muy sencillo
  - Primero **creamos los directorios** que van a servir su contenido
    - `sudo mkdir -p /var/www/html/web1.com/html`
    - `sudo mkdir -p /var/www/html/web2.com/html`
  - **Cambiamos el propietario** de estos directorios por nuestro usuario
    - Para no tener que hacer `sudo` en cada operación
    - Usamos la variable de entorno **`$USER`**, que contiene nuestro usuario actual
      - `sudo chown -R $USER:$USER /var/www/html/web1.com/html`
      - `sudo chown -R $USER:$USER /var/www/html/web2.com/html`
  - Hecho esto, debemos **copiar el contenido a servir** por cada web en cada uno de los directorios creados
    - Al igual que en Apache, **por defecto se sirven los documentos** `index.html` presentes en una localización a la que se accede vía web

Contexto "Server": Manejando server blocks

José Manuel
Redondo López

- Ahora ya podemos configurar los **server blocks** de cada web
  - Para ello **copiamos la configuración del server block por defecto** como punto de partida
    - `sudo cp /etc/nginx/sites-available/default /etc/nginx/sites-available/web1.com`
  - Y modificamos lo necesario para nuestro primer server block
- La imagen es un ejemplo de cómo es este fichero del **server block** por defecto
  - Que iremos modificando paso a paso para adaptarlo a nuestro nuevo server block

```
server {
        listen 80 default_server;
        listen [::]:80 default_server;

        root /var/www/html;
        index index.html index.htm index.nginx-debian.html;

        server_name _;

        location / {
                try_files $uri $uri/ =404;
        }
}
```

Contexto "Server": Creación de un server block

José Manuel
Redondo López

- **Sólo uno de los server blocks puede tener la opción `default_server` activa**
  - El `default_server` es el server block que se cargará al intentar servir una petición que no encaje con ninguno de los server blocks registrados en el servidor
- **Podemos dejar el server block por defecto para estas funciones**
  - O bien eliminarlo y activar como `default_server` uno de los que vamos a crear
- **Es importante que no haya más de un server block que lleve `default_server`**

```
server {
        listen 80;
        listen [::]:80;

        . . .
}
```

Contexto "Server": Creación de un server block

José Manuel
Redondo López

- La siguiente modificación consiste en cambiar el parámetro `root` para que encaje con el directorio que hemos creado antes para ese server block
  - Ahora el servidor ya sabrá **de donde leer** las páginas correspondientes a este server block

```
server {
        listen 80;
        listen [::]:80;

        root /var/www/html/web1.com/html;

}
```

Contexto "Server": Creación de un server block

José Manuel
Redondo López

- Luego, tenemos que cambiar el parámetro `server_name`
  - Para que sea el nombre del dominio asignado a nuestro primer server block
  - También podemos añadirle alias a dicho nombre (nombres adicionales)
- El resto de configuración podemos dejarla como está

```
server {
        listen 80;
        listen [::]:80;

        root /var/www/example.com/html;
        index index.html index.htm index.nginx-debian.html;

        server_name web1.com www.web1.com;
        location / {
                try_files $uri $uri/ =404;
        }
}
```

Contexto "Server": Creación de un server block

José Manuel
Redondo López

- Ahora hacemos lo mismo para nuestro segundo server block
  - `sudo cp /etc/nginx/sites-available/web1.com /etc/nginx/sites-available/web2.com`
- Siguiendo los pasos vistos, deberíamos tener un fichero como este

```
server {
        listen 80;
        listen [::]:80;

        root /var/www/web2.com/html;
        index index.html index.htm index.nginx-debian.html;

        server_name web2.com www.web2.com;

        location / {
                try_files $uri $uri/ =404;
        }
}
```

Contexto "Server": Creación de un server block

José Manuel
Redondo López

- Ahora ya podemos activar ambos server blocks
  - Mediante la creación de un **enlace simbólico** a los ficheros de configuración en el directorio que hemos mencionado
  - `sudo ln -s /etc/nginx/sites-available/web1.com /etc/nginx/sites-enabled/`
  - `sudo ln -s /etc/nginx/sites-available/web2.com /etc/nginx/sites-enabled/`
- Como en Apache, la configuración se puede modularizar en distintos ficheros con `include`
  - https://www.nginx.com/resources/wiki/start/topics/examples/full/
- Con esto tendremos tres server blocks operativos:
  - `web1.com`: Para peticiones a `web1.com` y `www.web1.com`
  - `web2.com`: Para peticiones a `web2.com` y `www.web2.com`
  - `default`: Para peticiones HTTP (puerto 80) **que no encajen** con las dos anteriores

Contexto "Server": Creación de un server block

José Manuel
Redondo López

- **Ahora ya podemos reiniciar Nginx**
  - Pero antes conviene **comprobar que el fichero de configuración no tiene ningún error** con: `sudo nginx -t`
  - Si no hay errores, podemos reiniciar el servidor con: `sudo service nginx restart`
- **Y ahora nuestro servidor podrá atender estos nombres de dominio**
  - En nuestras pruebas no tenemos servidor DNS
  - Habría que cambiar el fichero `/etc/hosts` (en Linux) o equivalente en Windows para que reconociera ese nombre de dominio

Contexto "Location"

José Manuel
Redondo López

- **Se pueden definir varios contextos `location` dentro de uno `server`**
  - Para sitios que estén en la **parte de la URL** que contienen
  - La localización que sirve la petición se seleccionará **a través de un algoritmo**
- **Deben estar dentro de contextos `server` y pueden anidarse**
  - Creando conjuntos de reglas especializados según se llega a localizaciones más concretas
- **Más información**
  - http://nginx.org/en/docs/http/ngx_http_core_module.html#location

```
# main context
server {
    # server context
    location /match/criteria {
        # Primer contexto location
    }
    location /other/criteria {
        # Segundo contexto location
        location nested_match {
            # Primer contexto location anidado
        }
        location other_nested {
            # Segundo contexto location anidado
        }
    }
}
```

Contexto "Location"

José Manuel
Redondo López

- **El algoritmo usado para elegir una `location` se explica aquí:**
  - https://www.digitalocean.com/community/tutorials/understanding-nginx-server-and-location-block-selection-algorithms
- **Ejemplos**
  - Esta localización encaja con peticiones a `/clientes`, `/clientes/martinez/index.html` o `/clientes/index.html`
    - `location /clientes {. . .}`
  - Esta localización encaja con `/clientes`, pero no con `/clientes/martinez/index.html` o `/clientes/index.html`
    - `location = /clientes {. . .}`
  - Esta localización encaja con peticiones a imágenes como `magneto.jpg`, pero no con `ciclope.GIF`
    - `location ~ \.(jpe?g|png|gif|ico)$ {. . .}`
  - Esta sin embargo encaja con ambas imágenes
    - `location ~* \.(jpe?g|png|gif|ico)$ {. . .}`
  - Esta localización encaja con `/coches/honda_civic.html`
    - Y no se hará ningún procesamiento de expresiones regulares en otros bloques `location` para determinar si hay otro encaja con la URI
    - `location ^~ /coches {. . .}`

Módulos de Nginx

Ampliando la funcionalidad del servidor

Módulos Nginx

José Manuel
Redondo López

- **Al igual que Apache, Nginx tiene una gran cantidad de módulos instalables**
  - https://www.nginx.com/resources/wiki/modules/
  - https://www.nginx.com/products/nginx/modules/
- **Para instalarlos**
  - Si se instala uno de un desarrollador certificado, se deben seguir sus instrucciones de instalación
  - Si es un módulo de terceros, basta con instalarlo vía gestor de paquetes del SO (Ej.: `apt`)
    - Usando su `module-name`, que podemos encontrar en su documentación
  - Tras instalarlo, en el contexto principal de `/etc/nginx/nginx.conf`, **añadimos una directiva** `load_module` por cada módulo a añadir
    - Los módulos suelen instalarse en `/etc/nginx/modules`
    - Ej.: `load_module modules/module-name.so;`
- **Finalmente, comprobamos la configuración y recargamos Nginx**
  - `nginx -t && nginx -s reload`
- **Al igual que en Apache 2, siempre instalar LOS MÍNIMOS MÓDULOS POSIBLES**

Restricción de acceso

José Manuel
Redondo López

- **Se puede restringir el acceso a partes de un servidor con el módulo `ngx_http_access_module`**
  - http://nginx.org/en/docs/http/ngx_http_access_module.html
- **Permite restringir/permitir por IP del cliente (incluyendo subredes), listas de IPs, nombres de dominio, IPV6, ...**
  - Las reglas se comprueban en orden hasta que encaja la 1ª
  - En el ejemplo se permite el acceso a las redes IPV4 `20.2.1.0/16` y `192.168.1.0/24` excepto a la dirección `192.168.1.1`
  - También se permite acceder a la red IPv6 `2001:0db8::/32`
- **La sintaxis es**
  - `allow/deny dirección IP | CIDR | unix: | all;`
  - `unix` representa a los sockets Unix (conexiones entre aplicaciones en el propio servidor)

```
 location / {
    deny  192.168.1.1;
    allow 192.168.1.0/24;
    allow 20.2.1.0/16;
    allow 2001:0db8::/32;
    deny all;
}
```

AUTENTICACIÓN

José Manuel
Redondo López

- **Al igual que con Apache 2, se puede conseguir con medios locales o externos**
  - **Locales**: Ficheros de diferente tipo existentes en el disco duro de la máquina con pares usuario/clave
    - Basic, Digest, sobre `/etc/shadow`…
  - **Externos**: Datos de usuarios en LDAPs, Active Directory, BBDD MySQL,…
- **Nginx soporta los mismos modos de autenticación que Apache con módulos**
  - Basic: https://www.nginx.com/resources/admin-guide/restricting-access-auth-basic/
  - Contra usuarios del SO y contra LDAP (a través de PAM, Pluggable Authentication Modules):
    - http://www.doublecloud.org/2014/01/nginx-with-pam-authentication/
    - https://unix.stackexchange.com/questions/165520/authenticate-users-via-ldap-using-nginx
  - …

¿Crees que lo has entendido todo? ¡Autoevalúate!

José Manuel
Redondo López

?

Conceptos de Nginx

- *¿Eres capaz de recordar los principales ficheros y directorios que contienen la configuración, módulos y sitios web del servidor web Nginx y para qué se usan?*
- *¿Entiendes la equivalencia de módulos y server blocks de Nginx con entidades de Apache 2?*
- *¿Entiendes lo que son los contextos de configuración de Nginx y los distintos ámbitos de aplicación que tienen los mismos?*

Server blocks

- *¿Entiendes lo que es un server block y cómo se crean?*
- *¿Entiendes las equivalencias entre un server block y un host virtual de Apache 2?*
- *¿Sabes cómo definir un server block adecuadamente, de forma que sus contenidos estén separados de los de otros hosts virtuales?*

Módulos

- *¿Entiendes que Nginx tiene una librería de módulos muy grande y cómo activar cualquiera de ellos?*
- *¿Entiendes que existe un módulo para restringir el acceso por un gran nº de criterios?*
- *¿Entiendes que existen muchos módulos para autenticar al usuario contra distintos almacenes de nombres de usuario / password?*

ASPECTOS GENERALES DE SERVIDORES WEB