1. Servidor Apache – Host Virtuales.

El archivo de configuración de apache:

El servidor web Apache que nosotros hemos configurado tiene la misma dirección IP para todos nuestros proyectos y, por lo tanto, el mismo nombre DNS.

Sin embargo, en realidad un servidor web sirve las páginas para distintas empresas o instituciones, cada una de ellas con su propio nombre de dominio.

Si queremos tener distintas IP o distintos nombres en nuestro servidor Apache podemos hacerlo recurriendo a sitios web virtuales; así, tendremos en la misma máquina varios sitios funcionando gracias a un proceso que pasa desapercibido al usuario que visita nuestros sitios.

Los sitios Web virtuales pueden estar basados en IP, o basados en nombre.

- Servidores basados en IP → cada sitio Web tiene una dirección IP diferente. La IP que debemos poner en la definición de la directiva Virtualhost cambia, cada IP corresponde a una interfaz de red del servidor web.

  Este método no aporta ventajas sobre el anterior, es más, aún puede ser más difícil de mantener si las IP del servidor web se modifican con cierta frecuencia.

- Servidores basados en nombres → con una sola dirección IP están funcionando sitios Web con diferentes nombres (de dominio). El hecho de que estén funcionando en la misma máquina física pasa completamente desapercibido para el usuario que visita esos sitios Web. Es decir, asociamos a la IP del servidor los nombres de dominio que necesitamos.

Existe otro tipo de configuración:

- Basada en servidores principales → Método más complejo, solo tiene sentido cuando queramos tener varios archivos de configuración apache2.conf independientes donde cada uno organiza sus propios hosts virtuales.

1.1. El archivo de configuración.

Archivo httpd-vhosts.conf de Apache:

```
<VirtualHost IP_servidor:80>
        DocumentRoot /htdocs/empresa/
        ServerName www.empresa.com.
        ServerAlias empresa.com empresa.es www.empresa.es
</VirtualHost>
```

#IP_servidor:80 → es la IP de nuestro servidor con el puerto para http.

*#Cuando el servidor tiene interfaz única, y lo usamos solo para nuestras fases de ##desarrollo, podemos sustituir el valor por *:80, es decir:*

*#<VirtualHost *:80>*

#DocumentRoot es la ruta con la carpeta donde tenemos el proyecto.

#ServerName es el nombre de dominio escogido.

#ServerAlias, son todos los alias que aceptamos en el navegador para el #mismo dominio.

La directiva <VirtualHost *:80> nos indica que para nuestra IP, a través del puerto 80, somos capaces de atender o servir las páginas colocadas en el DocumentRoot, utilizando el nombre DNS especificado en ServerName.

En el caso de tener varias direcciones IP, entonces en la directiva VirtualHost, sustituiríamos el * por las IP para cada sitio.

La primera opción corresponde a una configuración basada en nombre y la segunda a una configuración basada en IP.

1.2. Directivas de configuración:

NameVirtualHost

Designa una dirección IP para una virtualización basada en nombre.

Por ejemplo: NameVirtualHost 44.25.2.1:80

<VirtualHost>

```
<VirtualHost addr[:port] [addr[:port]] ...> ... </VirtualHost>
```

Contiene la directiva que se aplica al nombre o la IP indicada.

Addr puede ser:

- La IP del servidor virtual.
- El nombre DNS para la IP del servidor virtual (no recomendado).
- El carácter * para todas las IP.

Ejemplo:

```
<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>
```

ServerName

`ServerName [scheme://]fully-qualified-domain-name[:port]`

El nombre del host y el Puerto que el servidor usa para identificarse. Responden al esquema de nombre cualificado de dominio.

Por ejemplo: ServerName www.ejemplo.es

ServerAlias

`ServerAlias hostname [hostname] ...`

Nombres alternativos para referirnos al ServerName.

Por ejemplo:

ServerName servidor.dominio.com

ServerAlias servidor servidor1 .dominio. com servidor1

ServerPath

En el caso de usar Virtual Host por nombre, será la ruta URL para el host.

1.3. Servidores virtuales basados en nombre.

Es la forma más simple de configurar Hosts Virtuales. Solo necesitamos configurar nuestro servidor DNS para asignar la IP al nombre y configurar Apache para que reconozca los diferentes nombres.

Directivas que se usan:
`DocumentRoot`

`NameVirtualHost`

`ServerAlias`

`ServerName`

`ServerPath`

`<VirtualHost>`

Debemos tener claro que tendremos designada una sola IP que nos servirá para atender todas las peticiones de los distintos sitios con nombres diferentes. Por eso usaremos el valor * para NameVirtualHost y para <VirtualHost>.

Ejemplo:

```
NameVirtualHost *:80

<VirtualHost *:80>
ServerName www.domain.tld
```

```
ServerAlias domain.tld *.domain.tld
DocumentRoot /www/domain
</VirtualHost>

<VirtualHost *:80>
ServerName www.otherdomain.tld
DocumentRoot /www/otherdomain
</VirtualHost>
```

2. Servidores Web Seguros.

2.1. Protocolo HTTPS

HTTP (Hypertext Transfer Protocol = Protocolo de Transferencia de Hipertexto). No es un protocolo seguro:

- Intercambio de información en texto plano (sniffing).
- No se garantiza que los equipos involucrados en la transferencia sean seguros.
- Robo y falsificación de cookies y/o parámetros (robo de identidad, suplantación de web).
- Vulnerable entre clientes y servidores.
- Vulnerable en las aplicaciones.

HTTPS (Hypertext Transfer Protocol Secure= Protocolo de Transferencia de Hipertexto Seguro.

HTTPS es la versión segura de HTTP, tiene el objetivo de que si se intercepta la información, no pueda verse el contenido.

3. SSL y Apache.

SSL son las siglas en inglés de Secure Socket Layer (en español capa de conexión segura).

Este protocolo ha sido sucedido por TLS, que son las siglas en inglés de Transport Layer Security (en español seguridad de la capa de transporte). Versiones de TLS tienen un equivalente en SSL, por ejemplo, TLS 1.2 corresponde a SSL 3.3; de ahí que aún sea común que se refiera a este protocolo como SSL.

SSL/TLS utiliza el método de cifrado de clave pública (cifrado asimétrico) para la autenticación del servidor.

Características SSL/TSL:

- Se ejecutan en una capa entre los protocolos de aplicación (http, smtp o ftp) y el protocolo de transporte TCP.
- HTTPS, FTPS, SMTPS, POPS, IMAPS, etc., se basan en SSL/TSL.
- También es posible implementarlo sobre UDP.
- Configuración habitual:

  - El servidor de la comunicación tiene que estar autenticado.
  - El servidor emite el Certificado digital.

Ejemplos de aplicaciones SSL/TLS:

- Comercio electrónico en Internet -> HTTPS
- Correo electrónico seguro -> SMTPS, IMAPS, POPS.
- Autenticación y cifrado en tráfico de voz (VoIp).

En Apache, el módulo SSL no viene activado, así que si queremos utilizarlo tendremos que activarlo.

4. Cifrado de clave pública o asimétrico:

Trabaja con el par: cave pública (**kpub**) y clave privada (**kpriv**).

- La kpub suele publicarse para que sea conocida por cualquiera.
- La kpriv no interesa que sea conocida, tan solo es para el propietario de esta.

Tanto kpub como kpriv son necesarias para que la comunicación sea posible. Así:

- La información cifrada mediante kpub solo puede ser descifrada mediante kpriv.
- La información cifrada mediante kpriv solo puede ser descifrada mediante kpub.

Apache puede configurarse como AC (Autoridad de Certificación), pero los navegadores pueden desconfiar de los certificados creados.

5. HTTPS:

- http://www.educantabria.es
- https://yedra.educantabria.es

Para asegurar la información, el protocolo HTTPS utiliza certificados, que cuando se validan transfieren la información cifrada.

A la hora de cifrar la información, debemos tener en cuenta que es un proceso que consume tiempo de computación, así que debemos decidir:

- Qué parte de la información transferida entre cliente y servidor tiene que ir cifrada y cual no. Por ejemplo, solamente es necesario que sea cifrada la autenticación a dicha información.
- Si tenemos que configurar el servidor para que la información esté cifrada en todo el dominio o solo en el intento de acceso a la misma.

Apache Web Server puede emitir certificados, pero en algún navegador pueden ser interpretados como amenazas o peligros.

¿Por qué un Certificado puede interpretarse como una amenaza?

¿Confiarías en un DNI que no estuviese certificado por una entidad de confianza como el Ministerio de Interior?

Lo mismo sucede con los navegadores, solo confían en los certificados emitidos por su lista de **Entidades Certificadoras**, donde verifican, autentican y dan validez a los certificados.

Por eso, aunque podamos crear nuestros certificados en un servidor web, si salimos a Internet pueden ser interpretados como peligrosos por el navegador, que alertará con un aviso de problema de seguridad.

¿Cómo funciona https?

- HTTPS utiliza el puerto 443, en vez del puerto 80 (http).
- HTTPS cifra la información mediante SSL (Secure Socket Layers) y así crear un canal de transferencia de cifrado.

SSL es un protocolo que proporciona privacidad e integridad entre dos aplicaciones utilizando http.

¿Cómo se transfiere información con http y con https?

1. Con http → Escribo la URL http://www.educantabria.es

- Se traduce el dominio DNS a IP.
- Se busca en la IP obtenida la página solicitada por el puerto 80 (puerto TCP asignado por defecto al protocolo HTTP).
- Si el servidor tiene la página, la ofrece al navegador.

2. Con https → Escribo la URL https://yedra.educantabria.es

- Se traduce el dominio DNS a IP.
- Se busca en la IP obtenida la página solicitada por el puerto 443 (puerto TCP asignado por defecto al protocolo HTTPS).
- Se inicia negociación SSL, entre otras cosas, el servidor envía su certificado.
- Si el certificado es firmado por una Entidad Certificadora de confianza, acepta el certificado y cifra la comunicación.
- Se transfiere la página web de forma cifrada.

6. Protocolos y puertos:

Normalmente el servidor web espera el protocolo HTTP a través del puerto 80 y HTTPS a través del puerto 443.

Los puertos se pueden cambiar y configurar en el servidor web, pero cualquiera que quiera acceder a esas páginas debe saber el puerto TCP de la solicitud. Por ejemplo:

- http://www.miweb.local:8080 → El servidor web espera a miweb.local en el puerto 8080.
- https://www.miweb.local:4333 → El servidor web espera a miweb.local en el puerto 4333.