Instalación del servidor web Apache en una Raspberry PI 4

Instalación del servidor web Apache en una Raspberry PI 4

El servidor web **Apache** junto con **Nginx** copan los ⅔ de la implantación de servidores web en Internet. Vamos a ver como instalar el servidor Apache en un ordenador **Raspberry PI 4,** con soporte para conexiones seguras **https**. Vamos a utilizar un certificado gratuito desde **let's Encrypt,** un dominio gratuito desde **DuckDNS** y vamos a abrir los puertos necesarios en nuestro router para hacer accesible el servidor desde fuera de nuestra red de área local. Resumiendo, vamos a hacer el hosting de nuestro dominio en una máquina nuestra sin depender de un proveedor externo y por un mínimo precio: comprar una Raspberry PI 4, aunque el proceso será similar con cualquier otro ordenador que tengamos en casa sin más que instalar un sistema operativo basada en **Debian** como lo es Raspberry PI OS que es el que habrá que instalar en nuestra Raspberry PI 4.

La ventaja de Raspberry PI 4, aparte de su precio, es que el consumo de electricidad es mínimo, unos 4 w/hora frente a los 200-300 w/h , para arriba, de un ordenador tipo PC.¿ La desventaja?, que solo será eficaz con no muy altos requerimientos de tráfico, pero para una web personal de poco-medio tráfico será suficiente.

Necesitamos, por tanto, una Raspberry PI 4 con el sistema operativo Raspberry PI OS. En el artículo **Instalación de Raspberry PI OS en una Rasberry PI 4** en este mismo sitio web podemos ver el proceso.

Instalación del servidor web Apache

Nos conectamos mediante SSH a nuestra Raspberry PI 4 y abrimos una terminal de comandos. Cambiamos a modo superusuario:

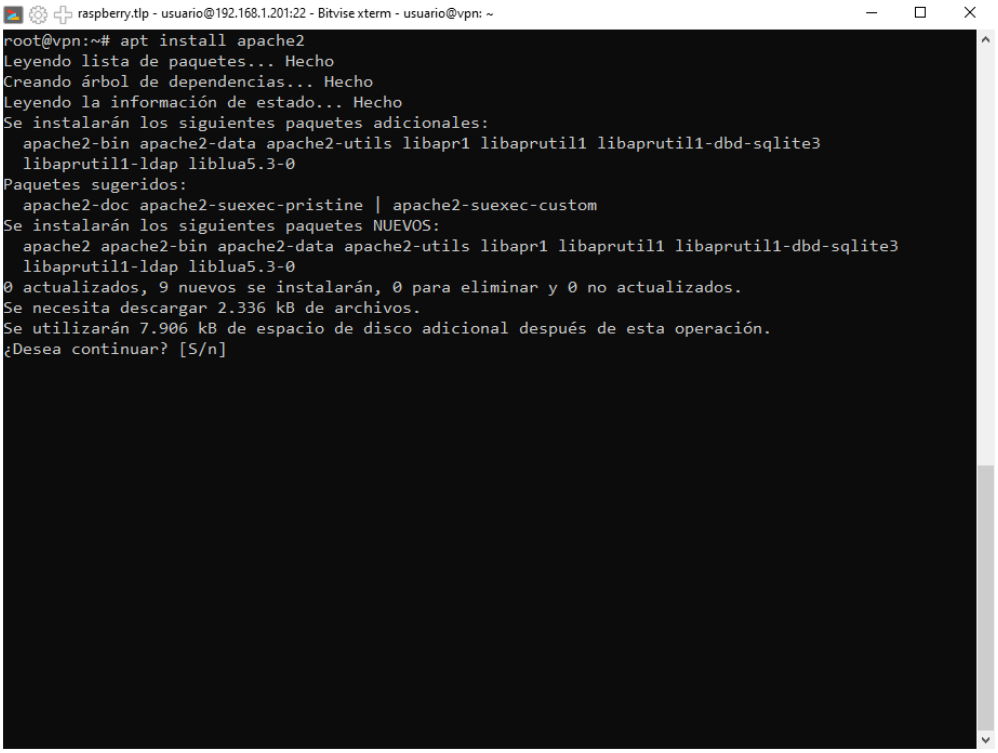
```
sudo su -
```

Actualizamos la lista de paquetes, y si hubiera alguna actualización pendiente actualizamos el sistema:

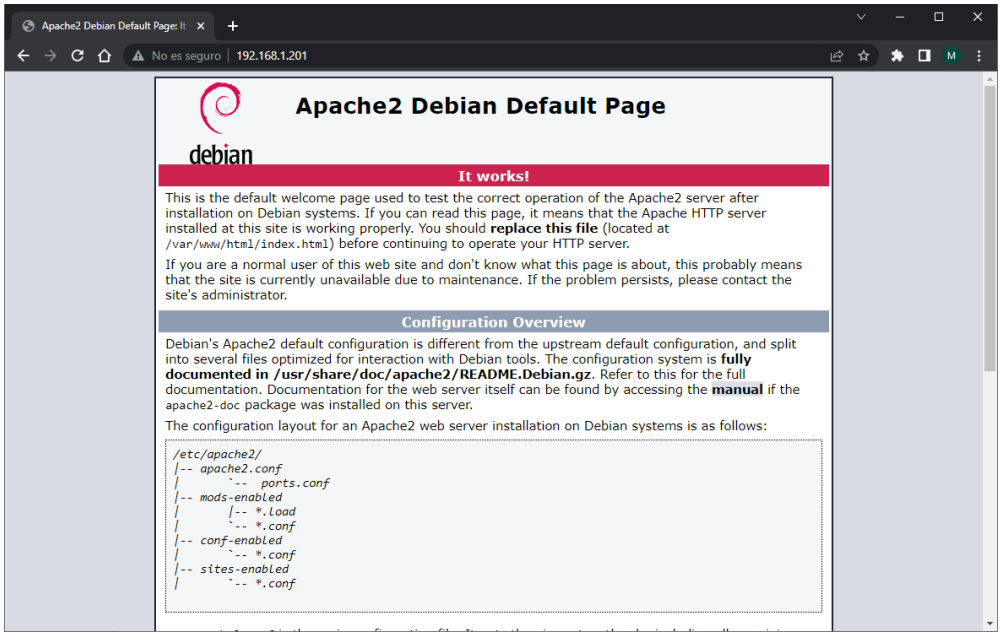
```
apt update
apt dist-upgrade
```

La instalación de Apache es muy sencilla:

```
apt install apache2
```



Si todo ha ido bien, tendremos un servidor web funcionando. Para comprobar no tenemos más acceder a la dirección IP de nuestra Raspberry PI 4 mediante un navegador:



Deberíamos ver la página por defecto del servidor. A partir de ahora todo lo que pongamos bajo **/var/www/html** podrá ser accedido a través de **http://*dirección_ip***. Si modificamos el archivo **index.html,** que ya se encuentra allí y es el responsable de la página por defecto de Apache:

```
<!DOCTYPE html">
<html lang="es">
  <head>
    <meta charset="UTF-8"/>
    <title>Web de Bond 008</title>
  </head>
  <body>
        <h1>Sitio Web Bond 008 en construcción</h1>
  </body>
</html>
```

Veríamos:

Para hacer accesible nuestro servidor desde el exterior necesitamos abrir en nuestro router el puerto **80** asociado al protocolo **http** por **TCP** y redirigir las peticiones a este puerto a nuestro servidor en la Raspberry PI 4. Este proceso dependerá de qué marca es nuestro router pero será muy parecido entre todos ellos. Lo primero es entrar a la administración del router. Para ello, desde un navegador, accedemos a la dirección de nuestro router que será habitualmente la misma dirección que la de nuestra puerta de enlace (si ejecutamos el comando **ipconfig** en Windows, o **ip route** en Linux podemos ver la dirección de nuestra puerta de enlace). Por ejemplo, en mi caso de router de Telefónica se vería:

Introduciendo la contraseña que aparece en la parte posterior del router, si no se ha cambiado ya, accederemos a la administración. Ahí, en **Menú → Puertos**

Pulsamos en **Añadir**. Se presupone que nuestra Raspberry PI 4 está en la dirección 192.168.1.201 y que nuestro servidor web está escuchando en el puerto 80. En el artículo sobre la instalación del sistema operativo se vio como poner una dirección fija.

Ya podremos acceder desde el exterior poniendo como url en un navegador la dirección ip pública externa de nuestro router. Para conocer cual es nuestra ip pública podemos acceder, por ejemplo, a la página https://miip.es que nos la dirán.

La ip pública de nuestro router es dinámica, esto quiere decir que puede cambiar, por ejemplo, si lo reiniciamos. Si queremos que se acceda fácilmente a nuestra web, lo ideal es disponer de un nombre de dominio y que este apunte a la dirección pública de nuestro router, pero tenemos el problema de que esta dirección puede cambiar. Tenemos dos soluciones: contratar con nuestro proveedor de internet un ip fija con un sobrecoste, o utilizar alguno de los servicios de internet para DNS dinámicos, como por ejemplo **DuckDNS**. En esta misma web hay un artículo sobre como **Como obtener una URL o dominio propio, gratis y con dirección IP Dinámica DuckDNS**.

Tenemos ya un servidor web en nuestra Raspberry PI 4, con nuestro router con el puerto 80 abierto y redirigido a la Raspberry y un nombre de dominio, en el ejemplo, **bond008.duckdns.org**, que apunta a la dirección pública dinámica de nuestro router.

Para que el servidor Apache responda a peticiones sobre el dominio **bond008.duckdns.org** se debe crear un archivo de configuración del sitio, le llamaremos como el nombre del dominio y extensión **.conf** y lo colocaremos en la carpeta **etc/apache/sites-available** con contenido:

```
<VirtualHost *:80>
        ServerName bond008.duckdns.org
```

```
        ServerAdmin webmaster@localhost
        DocumentRoot /var/www/html
        ErrorLog ${APACHE_LOG_DIR}/error.log
        CustomLog ${APACHE_LOG_DIR}/access.log combined
</VirtualHost>
```

No es más que una copia del archivo **000-default.conf** que se encuentra en la misma carpeta y que es el archivo de configuración del sitio por defecto de Apache, al que se le ha añadido la línea

```
ServerName bond008.duckdns.org
```

Para activar el sitio ejecutamos

```
a2ensite bond008.duckdns.org.conf
```

Y reiniciamos el servicio

```
systemctl restart apache2
```

La tendencia actual en los navegadores es solo permitir la navegación web desde páginas seguras haciendo uso del protocolo **https** en lugar del inseguro **http**. Para hacer que nuestro servidor Apache atienda peticiones desde el exterior por el protocolo **https** se han de hacer varias cosas.

La primera es abrir el puerto **443 TCP** de nuestro router porque ese es el puerto por defecto asignado al protocolo https. La forma de hacerlo es similar a lo que hicimos para el puerto 80

Lo segundo viene de que, por defecto, la instalación de Apache no activa el soporte de comunicación SSL que es la que da soporte al protocolo https. La activación de funcionalidades en Apache se hace a través de la activación de módulos. Disponemos de los comandos **a2enmod, a2dismod** y **a2query** para activar, desactivar y listar los diferentes módulos. Así para activar el módulo **ssl** debemos hacer:

```
a2enmod ssl
```

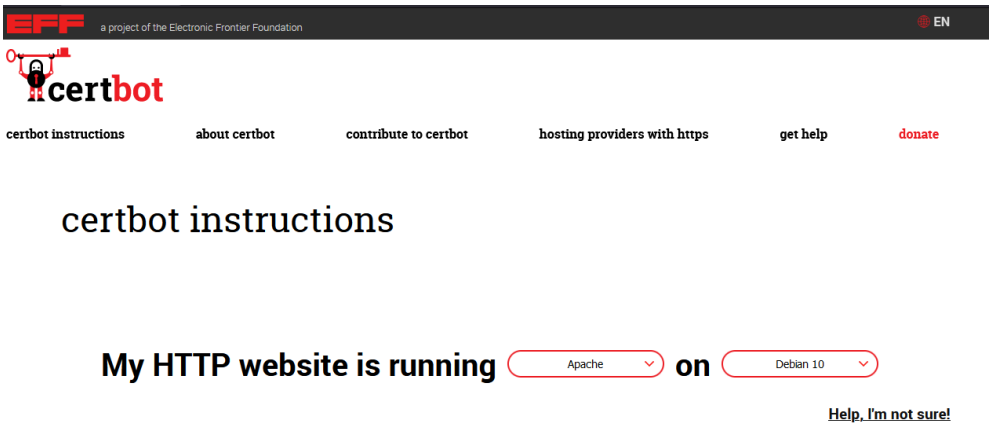
Y reiniciar el servicio para que los cambios se apliquen:

```
systemctl restart apache2
```

Lo tercero es contar con un certificado emitido por una entidad certificadora reconocida que valide los accesos mediante https a nuestro dominio registrado. Hay multitud de empresas que nos permiten obtener certificados legítimos casi siempre con un coste añadido, pero también las hay que los ofrecen de forma gratuita. La más extendida es **let's Encrypt** que nos permite con un proceso muy simple obtener de forma gratuita un certificado para uso con nuestro dominio. Los certificados let's Encrypt tienen una vida de seis meses por lo que habrá que renovarlos al termino de cada período. Por suerte todo la renovación se puede efectuar de forma automática con la ejecución de un simple comando. Vamos a ver como se obtiene el certificado.

Si vamos a https://letsencrypt.org/es/getting-started/

Nos dice que si tenemos acceso Shell a la máquina servidora, como es en nuestro caso, utilicemos un cliente ACME llamado **certbot** que nos permitirá automatizar la emisión, instalación y renovación del certificado. Pulsando sobre la palabra **certbot** nos remite a la página de la utilidad, en la que seleccionaremos en las listas desplegables que nuestro servidor es Apache y que nuestro sistema operativo es Debian 10 (Raspberry PI OS está basado en Debian)

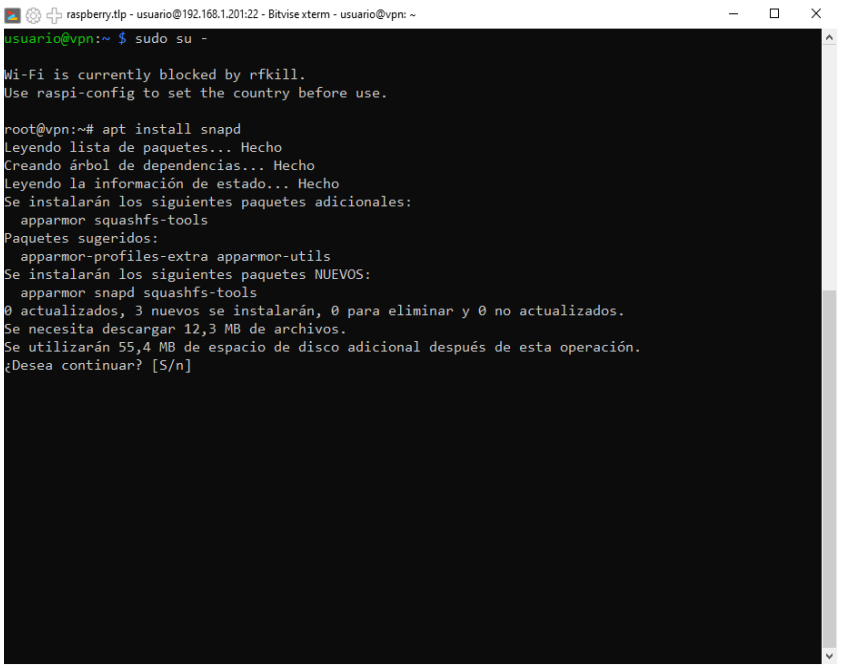


En la parte inferior de la página aparecerán las instrucciones para la instalación de la utilidad y su funcionamiento.

Vemos que se muestran los requerimientos para la instalación y que los cumplimos todos: tenemos un servidor http funcionando al cual hay acceso online, es decir, público desde internet a través de un nombre de dominio, que escucha en el puerto 80 y que tenemos acceso ssh a nuestro servidor.

Lo primero que se nos pide es la instalación del gestor de paquete **snapd**. Como cualquier instalación en los sistemas Debian lo hacemos con el comando **apt install.** Iniciamos una consola de comandos tras hacer login con SSH en el cliente Bitvise, entramos en modo superusuario e instalamos

```
sudo su -
apt install snapd
```



A continuación ejecutamos los siguientes comandos para asegurarnos que contamos con la última versión de **snapd**

```
snap install core
snap refresh core
```

A continuación instalamos **certbot** desde snap

```
snap install --classic certbot
```

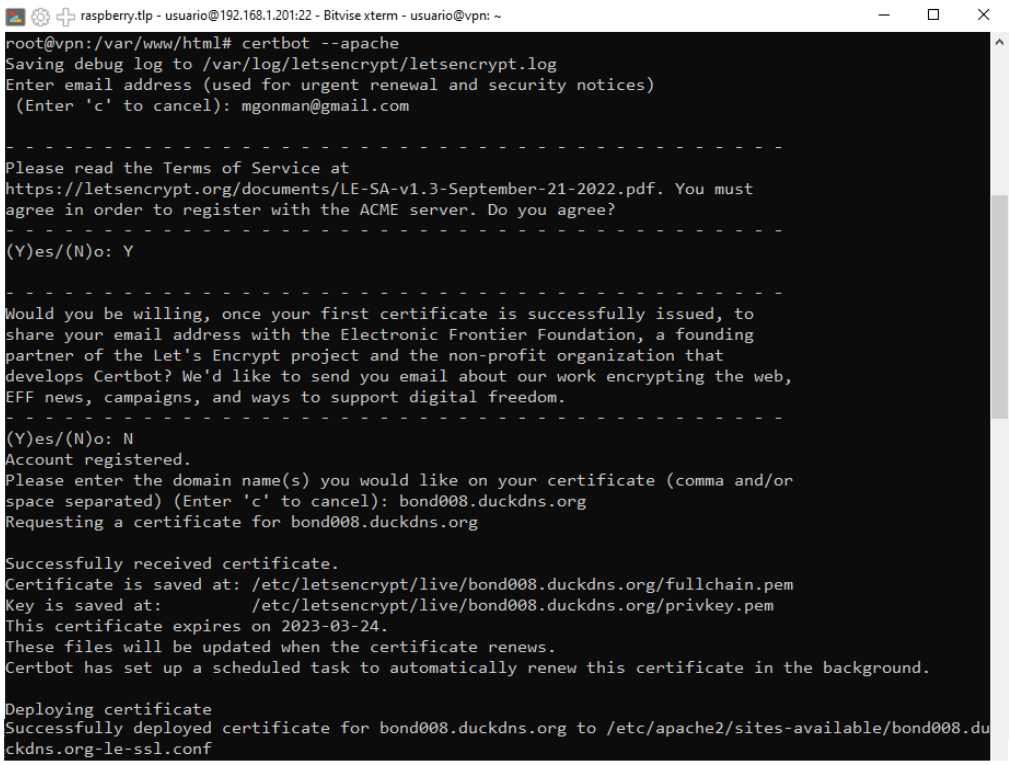
Ejecutamos el siguiente comando para asegurarnos que certbot puede ser ejecutado

```
ln -s /snap/bin/certbot /usr/bin/certbot
```

No nos queda más que ejecutar:

```
certbot --apache
```

Se lanzara un proceso en el que nos solicitará un correo electrónico para comunicación con let's encrypt, se nos solicitará aceptar los términos del servicio, nos preguntarán que si queremos recibir correos con noticias sobre el proyecto y por último y lo más importante junto con el correo primero el nombre del dominio que queremos registrar para el certificado. Se pueden incluir varios dominios enumerándolos separados por coma.



Al terminar el proceso, si todo ha ido bien, ya tendremos instalado el certificado y configurado Apache para servir mediante https. Certbot habrá creado en **sites-available** un nuevo archivo **bond008.duckdns.org-le-ssl.conf** con la configuración del sitio con https, habrá modificado **bond008.duckdns.org.conf** para redirigir las peticiones **http** a **https** y habrá configurado el planificador de tareas para que se renueve de forma periódica el certificado. No tendremos que hacer ya nada más. El contenido de los archivos será:

bond008.duckdns.org.conf

```
<VirtualHost *:80>
        ServerName bond008.duckdns.org
        ServerAdmin webmaster@localhost
        DocumentRoot /var/www/html
        ErrorLog ${APACHE_LOG_DIR}/error.log
        CustomLog ${APACHE_LOG_DIR}/access.log combined
RewriteEngine on
RewriteCond %{SERVER_NAME} =bond008.duckdns.org
RewriteRule ^ https://%{SERVER_NAME}%{REQUEST_URI} [END,NE,R=permanent]
</VirtualHost>
```

bond008.duckdns.org-le-ssl.conf

```
<IfModule mod_ssl.c>
<VirtualHost *:443>
        ServerName bond008.duckdns.org
        ServerAdmin webmaster@localhost
        DocumentRoot /var/www/html
        ErrorLog ${APACHE_LOG_DIR}/error.log
        CustomLog ${APACHE_LOG_DIR}/access.log combined
SSLCertificateFile
/etc/letsencrypt/live/bond008.duckdns.org/fullchain.pem
SSLCertificateKeyFile
/etc/letsencrypt/live/bond008.duckdns.org/privkey.pem
Include /etc/letsencrypt/options-ssl-apache.conf
</VirtualHost>
</IfModule>
```