PARTE III
DE MAQUINA A SERVIDORA

Contenidos

9. Instalemos una servidora web en la Raspberry Pi

Existen diferentes programas para dar vida a una servidora web. Los más conocidos son Apache y nGinx. Pueden averiguar las características de cada uno y evaluar cuál les conviene. En este caso nosotras elegimos Apache y seguiremos los pasos para instalarlo. Básicamente consiste en ejecutar en la terminal de la Raspberry Pi (puede ser ya de manera remota desde nuestra computadora/ordenador o desde la propia Raspberry Pi) el comando para instalar paquetes en Raspberry Pi OS:

```
sudo apt install apache2
```

¡Sí!, así de fácil. Para comprobar que se ha instalado correctamente y que está activo podemos ir al navegador de nuestra computadora/ordenador y escribir la dirección IP de la Raspberry Pi (recuerden que todavía debemos estar conectadas a la misma red). Si no ha cambiado nada será la dirección IP que usaron para conectarse por SSH. Y si se la han olvidado recuerden ejecutar el comando **`hostname -I`** en la consola de la Raspberry Pi.

Debería aparecerles la página de Apache por defecto que indica que su servidor web está corriendo correctamente en la Raspberry Pi. Emocionante, ¿no? Esa página que nos muestra se llama **`index.html`** y está guardada en el directorio "*root* de Apache" **`/var/www/html/`** de nuestra Raspberry Pi. Si quieren cambiarla y hacer su propia página web de prueba pueden cambiarle el nombre a ese archivo para respaldarlo:

```
sudo mv /var/www/html/index.html  /var/www/html/index.html.old
```

Y luego crear un nuevo archivo **`index.html`** con lo que ustedes quieran:

```
sudo nano /var/www/html/index.html
```

Para ponerle contenido tienen que saber un poquito de la sintaxis de HTML. Pueden empezar con este ejemplo sencillo:

```
<html>
¡Hola amigas!
</html>
```

Guardamos, salimos y refrescamos la página en el navegador para ver si se reflejan los cambios. ¡Y voilá! Si no ven los cambios prueben volver a cargarla con **`Ctrl + F5`**. Si incluso así no se reflejan los cambios prueben abrir la página con una pestaña en modo incógnito o borrar la caché. En ocasiones se queda guardada en caché la versión antigua de la web.

De todos modos esta web sólo se puede ver desde las computadoras que están conectadas al *router* al que está conectado la servidora. Todavía no es accesible desde Internet.

En este punto pueden probar trastear/cacharrear/jugar/experimentar con HTML. Pueden poner, por ejemplo, una imagen: **`<img src="http://urldela.imagen.com/foto.jpg">`**. Hay muuuuuuuuucha documentación en Internet sobre HTML, plantillas, trucos, secretos, etc.

Más adelante veremos algunos archivos de configuración de Apache como **`/etc/apache2/conf-enabled/security.conf`**, **`/etc/apache2/apache2.conf`** o **`.htaccess`**. Recuerden que cada vez que modifiquemos cualquier parámetro en un archivo de configuración tenemos que reiniciar el servicio para que los cambios se hagan efectivos. En el caso de Apache debemos escribir:

```
sudo service apache2 restart
```

También se puede reiniciar haciendo:

```
sudo systemclt restart apache2
```

9.1 Cómo configurar dos o más webs en una servidora

Ahora, ¿se puede tener más de una web en una misma servidora? No sólo se puede sino que ¡no es muy complicado! Por cada web que quieran añadir, necesitarán:

- Un "*virtual host*" o "*vhost*". Esto es un archivo de este tipo **`/etc/apache2/sites-available/dominio.conf`** (más abajo tienen un ejemplo).
- Un "DocumentRoot" o ruta raíz donde poner los archivos que conformen cada web. Estas rutas suelen ser **`/var/www/html/dominio1.com/`**, **`/var/www/html/dominio2.com/`**, etc. Pueden crear esta carpeta con el comando: **`sudo mkdir /var/www/html/dominio1.com/`**

Pueden tomar un ejemplo de *vhost* de **/etc/apache2/sites-available/000-default.conf**:

```
sudo cp /etc/apache2/sites-available/000-default.conf /etc/apache2/sites-available/dominio.conf
```

No olviden indicar la ruta en **`DocumentRoot`** y el dominio en **`ServerName`**:

```
<VirtualHost *:80>
    ServerAdmin webmaster@localhost
    ServerName domino.com
    DocumentRoot /var/www/html/dominio.com
    ErrorLog ${APACHE_LOG_DIR}/error.log
```

```
        CustomLog ${APACHE_LOG_DIR}/access.log
    </VirtualHost>
```

Una vez todo esté listo, los *vhosts* se activarán con el comando:

```
sudo a2ensite dominio.conf
```

Y tendremos que deshabilitar el *vhost* que Apache trae activado por defecto con **`sudo a2dissite 000-default.conf`**

Finalmente, para que la nueva la configuración tome efecto hay que reiniciar Apache:

```
sudo systemctl restart apache2
```

Aquí les dejamos un tutorial en español más detallado: https://www.digitalocean.com/community/tutorials/como-configurar-virtual-hosts-de-apache-en-ubuntu-16-04-es

> ***Nota 1:*** en la Parte V les hablamos sobre los permisos que tienen que tener los directorios *root* de Apache, pero se lo adelantamos aquí.
>
> El usuario **`pi`** debe ser parte del grupo **`www-data`**:
>
> ```
> sudo usermod -a -G www-data pi
> ```

El usuario **pi** y el grupo **www-data** tinen que ser los propietarios del directorio *root*:

```
sudo chown -R pi:www-data /var/www/html/dominio1.com
```

Y los permisos recomendados son **755** para directorios y **644** para archivos:

```
sudo find /var/www/html/dominio1.com -type d -exec chmod 755 {} \;
```

```
sudo find /var/www/html/dominio1.com -type f -exec chmod 644 {} \;
```

Nota 2: en el punto 11 explicaremos cómo configurar Let's Encrypt con **certbot**. Sigan los mismos pasos y el programa las guiará en la configuración de un certificado para cada domino. Al terminar verán que el *vhost* se ha editado por **certbot** y crea un nuevo archivo **/etc/apache2/sites-available/dominio-le-ssl.conf** indicando las nuevas configuraciones que necesita el certificado.

10. Hagamos la servidora accesible desde Internet

> **¡Atención!** Antes de seguir los pasos de este apartado es necesario securizar la servidora ya que en el momento en el que sea accesible desde Internet va a empezar a recibir intentos de conexión. No serán necesariamente personas que estén buscando meterse en nuestra servidora sino intentos automáticos de *bots* que quieren aprovecharse de infraestructura ajena. Les recomendamos saltar a la parte IV y securizar la servidora antes de continuar.

Una vez que tengan su pequeña servidora corriendo y securizada vamos a ver cómo lograr que cualquier persona en Internet pueda ver la página web alojada en nuestra servidora casera, "detrás" de un *router* casero. Es importante que repasen toda la parte IV antes de empezar con esta ya que cuando salgan a Internet estarán vulnerables a posibles ataques.

Lo que vamos a aprender ahora es cómo configurar el *router* para que las conexiones externas (que vienen desde Internet) se encaminen (se redirijan) a la servidora y nunca a otro dispositivo de la red. También veremos cómo asociar un dominio a la servidora.

Lo que viene ahora quizás les parezca más complejo con conceptos nuevos sobre redes y protocolos. Pónganse cómodas porque cuando funcione ¡será un subidón!

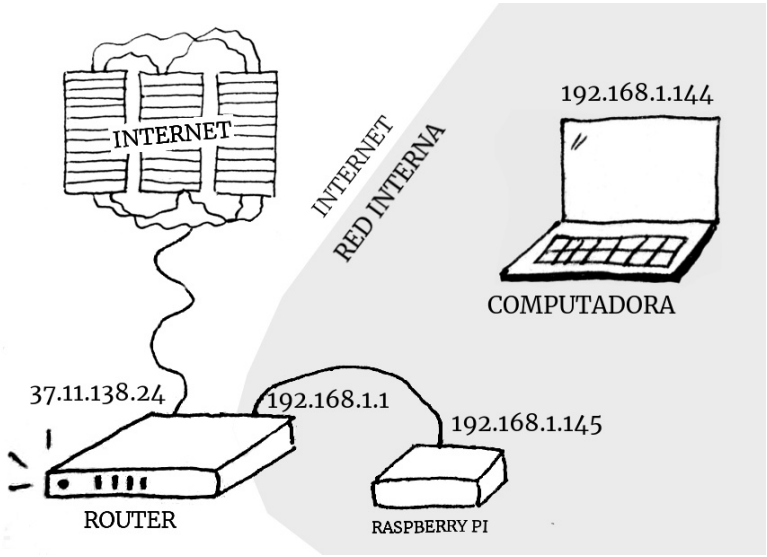
10.1 El router de casa o la comunidad

Comencemos accediendo al panel de control del *router*. ¿Se acuerdan cómo? Tienen que abrir un navegador y escribir en la barra de direcciones la dirección IP local del *router* (192.168.1.1 o 192.168.0.1). Entren con la usuaria y contraseña que le hayan configurado en la parte IV.

Una vez que ya estén dentro del panel de control de su *router* pueden echar un vistazo a los diferentes menús y opciones disponibles, ahora con más detalle. Como ya saben todos estos menús varían según el fabricante así que aquí no vamos a poder guiarlas paso a paso. Las animamos a guiarse por su intuición, un poco de inglés y algunos manuales en Internet. De todos modos los *routers* caseros son más o menos similares.

Algo importante de saber de los *routers* caseros es que -a diferencia de los móviles o las computadoras- en vez de tener una sola dirección IP con la que se los identifica en la red tienen dos direcciones IP: una privada o local y otra pública. Lo que hace el *router* es gestionar y encaminar el tráfico que hay entre la red local y la red de Internet. Por lo general, el *router* va a dejar salir todo el tráfico que los dispositivos de la red local estén enviando a Internet y deja entrar sólo el tráfico que éstos dispositivos hayan solicitado. Ordenadamente, redirige las peticiones que le llegan a cada quien que las haya pedido.

Una topología de red presente en una casa o comunidad pequeña suele ser así:



Intenten dibujar un pequeño esquema que refleje cómo es su red local con las direccione IP de cada dispositivo conectado. En la configuración del *router* suele haber un apartado donde lista las direcciones IP locales de los dispositivos conectados y la IP pública del *router*. La dirección IP pública del *router* también se puede consultar desde cualquier dispositivo conectado al *router*. Vayan al navegador y busquen "¿Cuál es mi dirección IP?", entre los resultados les saldrán un montón de webs que les dirán con qué IP pública están saliendo a Internet. Nosotras usamos https://wtfismyip.com/ o incluso el propio Duck Duck Go te la dice. Todos los dispositivos conectados a un mismo *router* salen con la misma dirección IP, es decir, con la IP pública del *router*.

Si la dirección IP pública que les está mostrando el navegador y la que ven en el *router* son diferentes puede ser que su proveedor de

Internet esté haciendo una cosa que se llama CGNAT para optimizar el número de direcciones IP que usa entre sus clientes. Esto nos pasó. Tuvimos que llamar a la compañía para que nos asignaran una dirección IP pública única (que no fija) y funcionó.

Nota: Si tu compañía de Internet usa CGNAT puedes llamar y pedir que te asignen una IP pública única para tu *router* - esta IP no será fija pero sí única, normalmente cambia cada semana (las IP públicas fijas cuestan dinero). Si tu compañía te dice que no puede quitarte el CGNAT, puedes insistir e informarte de la situación del CGNAT en tu país. Una opción es llamar a otras compañías y preguntar si ofrecen un servicio sin CGNAT y en tal caso cambiar de compañía.

Alternativas al CGNAT - los "Servicios cebolla": otra opción que nos han planteado varias compañeras es configurar un "*Onion Service*" en la red Tor y así evitar el CGNAT. Pero entonces solo las personas conectadas a la red Tor podrán ver nuestra web y la url de la web será algo así: *ww6ybal4bd7szmgncyruucpgfkqahzdd.onion*. Con esto también nos evitamos la necesidad de un dominio y ganamos en privacidad. Aquí un tutorial de cómo configurar un "Servicio Cebolla": *https://community.tor-project.org/es/onion-services/setup/*.

Como dijimos antes, esta guía está pensada para montar una servidora web con una conexión casera. Puede ser la de la casa de alguna de ustedes, pero tengan en cuenta que esto conlleva riesgos: a través de la dirección IP se puede identificar nuestro proveedor de servicios de Internet (ISP), quien a su vez tiene los datos de la persona que contrató la conexión (nombre y apellidos, probablemente número de identificación, dirección postal, número de teléfono, etc.). Esos datos están protegidos en mayor o menor medida de acuerdo la compañía que hayamos contratado y al país donde residamos.

10.2 Asignar dirección IP local fija a la servidora

Antes de terminar con la configuración del *router* es importante asignar a nuestra servidora web una dirección IP local fija para que podamos "localizarla" siempre con la misma IP, ya que esta suele cambiar cada vez que conectamos nuestro dispositivo a la red local. Existen varias maneras fijar una dirección IP local a un dispositivo. Una sencilla es ir a algún apartado de la configuración del *router* y fijar una dirección IP según la dirección MAC del dispositivo. Tengan cuidado de no ponerle una dirección IP que ya esté usando otro dispositivo ya que se generaría un conflicto.

Hay *routers* caseros que lo tienen sencillo y permiten elegir directamente el nombre del dispositivo y asignarle una dirección IP. En otros tendrán que encontrar la dirección MAC de la tarjeta de red de la Raspberry Pi. Para hacerlo puede ejecutar el comando:

```
ip a
```

Y localizar un conjunto de 6 valores en hexadecimal del tipo: **`b8:27:eb:20:a0:9f`** (esto es sólo un ejemplo, cada dirección MAC

es única para cada placa de red). Tienen que mirar la sección de la interfaz cableada que se suele llamar **`eth0`**, ahí también podrán ver la dirección IP. No se confundan con la MAC de la interfaz inalámbrica que se suele llamar **`wlan0`**.

10.3 Configurar red DMZ

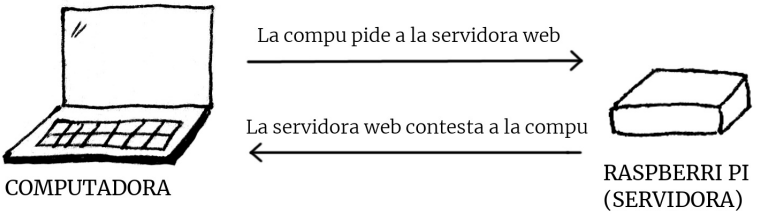
Ahora nuestro objetivo será configurar el *router* para que nuestra servidora sea accesible desde Internet y no sólo desde el salón de nuestra casa o comunidad. Para ello debemos abrir algunos puertos. Pero, ¿cómo?, ¿es esto seguro para los demás dispositivos que estén conectados a esa red? Vamos a ver cómo asegurarnos de no ponernos en riesgo.

Para "abrir esas puertas" en el router hay varias opciones. Una opción que nos recomendaron es configurar una red DMZ (las siglas significan "zona desmilitarizada" en inglés) en el router. Lo que hace esta red es que cualquier petición externa que reciba el *router* y que considere "desconocida" la dirija a la dirección IP local que le asignemos. En este caso será la de nuestra servidora que ahora tiene dirección IP local fija.

Con "Petición externa desconocida" nos referimos a una conexión que venga de Internet y que ningún dispositivo de la red local ha pedido. La situación es la opuesta a cuando estamos consultando un sitio web desde nuestra computadora. En ese caso recibimos la conexión que hemos pedido y por lo tanto para el *router* esta petición es "conocida".

Una servidora web sí tiene que poder aceptar peticiones desconocidas, que vengan desde "afuera", para poder ofrecer los contenidos web alojados en ella. Este es el sentido "pasivo" de "servir" una web. La servidora estará a la espera de que lleguen esas peticiones desconocidas para satisfacerlas.

La red DMZ aísla al resto de dispositivos de la red interna de conexiones desconocidas y redirige el tráfico que queremos a la servidora. Quizás les ayuda este diagrama sobre peticiones web:



En la primera conexión la servidora recibe una conexión "desconocida", algo que ella no ha pedido. En la segunda conexión la computadora recibe una conexión conocida que es la respuesta a una petición que ella ha hecho.

En el *router* que nosotras usamos la configuración de la red DMZ está en "Seguridad > Security" o puede estar también en "Configuración avanzada". Ahí se activa y se pone el "LAN host" (el dipositivo de la red local) al que se van a redirigir las conexiones externas

desconocidas. Ahí tendremos que poner la dirección IP de la servidora. ¿Se acuerdan cómo averiguarla? Tenemos que escribir en la consola: **`hostname -I`**.

Una alternativa a configurar una red DMZ en el *router* es configurar el "*Port forwarding*" para que la servidora sea accesible desde Internet y dirija las conexiones no solicitadas a una dirección IP específica. Tendrán que configurarlo para los puertos 80, 443 y para el que hayan elegido para las conexiones SSH (en nuestro caso el 2251, por ejemplo), por lo menos. Esto nos ha dado varios dolores de cabeza así que te recomendamos la red DMZ.

Para comprobar que esta configuración funcionó pueden poner en su navegador la dirección IP pública asignada al *router* y les debería mostrar su página web. ¡Espectacular!

10.4 Gestionar DNS dinámicos

Como nuestra red es casera nuestro proveedor de servicios de Internet no nos asigna una dirección IP pública fija para el *router*. Es decir, un día nos conectamos con una, a la semana con otra, etc. Es una cuestión de que no hay suficientes direcciones IP públicas para todos los *router* que hay en el mundo. Si quieres una IP pública fija tienes que pagar un extra al proveedor de servicios de Internet. Los servidores grandes que pertenecen a instituciones o empresas suelen pagar por este servicio.

En este caso quienes quisieran visitar nuestra web tendrían que enterarse de cuál es nuestra dirección IP de la semana. Para solventar

esto necesitamos un servicio de DNS dinámico que nos permita vincular un dominio con una dirección IP dinámica pública como la que tenemos en una conexión casera. Lo que hace este servicio es estar atento a los cambios de la dirección IP pública de nuestro router y actualizarla en la vinculación con nuestro dominio.

El servicio que vincula direcciones IP públicas con nombres de dominio se llama DNS (*Domain Name System*). Existen servidores DNS que proveen este servicio. Pero primero necesitamos un dominio.

Conseguir un dominio

Para proceder a vincular un dominio con la dirección IP dinámica debemos comprar un dominio. En caso de no tener dinero para comprar un dominio propio podemos optar por servicios gratuitos como https://www.noip.com, que ofrecen DNS dinámico y prestan subdominios del tipo http://hola-amiga.noip.com o http://hola.ddns.net. En la propia página de NoIp hay mucha docu- mentación sobre cómo utilizar el servicio. La única desventaja de la versión gratuita es que hay que renovarla cada mes (¡pst!, pero nos avisan para no olvidarnos). Sin embargo, creemos que puede dar problemas a la hora de configurar los certificados SSL/TLS usando Let's Encrypt.

Nosotras compramos un dominio de nivel superior .red. Originalmente había unos pocos: .com, .edu, .gov, .intl, .org, .net y .mil, además de los dominios geofráficos de nivel superior: .ar para Argentina, .br para Brasil, etc. Pero ahora hay muchísimos. Puedes ver una lista completa en la página de Wikipedia de *List of Internet top level*

domains. Existen muchos lugares para comprar dominios: varían en precio, condiciones, prestaciones, etc. Si queremos comprar un dominio de país (.gt, .es, .py, etc.) debemos acudir a las agencias nacionales. Para dominios genéricos podemos buscar cualquier otro servicio. Algunos proyectos activistas/éticos que ofrecen dominios son: 1984hosting.com, greenhost.net o gandi.net.

Debemos tener en cuenta que cuando compremos cualquier dominio nos pedirán datos personales de contacto: nombre, dirección postal, correo electrónico, etc. Esos datos (llamados informalmente 'WHOIS') serán públicos salvo que pidamos que no los muestren. Algunas compañías cobran por este servicio y otras no. Es importante hacer una evaluación de riesgos para saber qué datos dar y cómo protegerlos. También será necesario pensar cómo hacer el pago. Algunos proyectos como https://1984hosting.com o https://njal.la permiten hacer la compra con criptomonedas en vez de tener que usar una tarjeta de crédito. Incluso https://qwass.com/ acepta Faircoins, una criptomoneda ética. Esto nos puede ayudar a proteger nuestra privacidad.

Para comprar el dominio tendremos que crearnos un perfil para acceder a un panel de control que nos permitirá gestionar una serie de parámetros que veremos más adelante. ¡Usen una buena contraseña y a guardarla en su gestor de contraseñas!

> Recomendamos el uso de **gestores de contraseñas** como KeepasXC: las ayudará a generar contraseñas fuertes, tenerlas ordenadas y guardarlas cifradas.

DNS dinámicos

Después de rebuscar servicios de DNS dinámicos nos recomendaron el que provee Hurricane Electric (https://he.net). Para poder usar sus servicios de DNS hay que hacerse una cuenta en https://dns.he.net/ donde podremos configurar los registros DNS. Una vez que nos hayamos dado de alta vamos al panel de administración de nuestro dominio en la web donde lo compramos. Busquen donde se puedan configurar los servidores de DNS. Allí tendrán que configurar como servicio de DNS a los DNS de he.net:

ns1.he.net

ns2.he.net

ns3.he.net

ns4.he.net

ns5.he.net

Quizás haya que esperar unos minutos hasta que la configuración se actualice. Luego volvemos al panel de control de dns.he.net y añadimos un nuevo registro de tipo A con los siguientes datos:

name: nuestrodominio.net

IPv4 Address: 1.1.1.1

TTL: actualización del registro en segundos (puedes poner 300 - 5 minutos)

Check el cuadradito "Enable entry for dynamic DNS"

Una vez que añadimos el registro debemos pinchar/hacer clic en el símbolo de las flechitas en círculo (como de actualizar). Se nos abrirá una ventana de dialogo pidiendo asignar una contraseña al registro. Se la ponemos, confirmamos y aceptamos. ¡A guardar esa contraseña en el Keepass!

> **¡Ojo!** Ponemos la dirección IP 1.1.1.1 que es una dirección cualquiera para poder comprobar luego de manera más sencilla que se ha cambiado a la dirección correcta ¡sigan leyendo!

Configurar ddclient

Ahora volvamos a nuestra Raspberry Pi para configurar el DNS dinámico. Lo haremos con un programita llamado **`ddclient`** que se comunicará con el servicio de DNS dinamico de he.net y le mandará nuestra nueva dirección IP cada vez que cambie. Para instalar el servicio ddclient ejecutemos:

```
sudo apt install ddclient
```

Para congifurarlo abrimos:

```
sudo nano /etc/ddclient.conf
```

Y escribamos la siguiente configuración estándar:

```
pid=/var/run/ddclient.pid
protocol=dyndns2
```

```
ssl=yes
use=web
server=dyn.dns.he.net
login=nuestrodominio.net
password='contraseña'
nuestrodominio.net
```

En el campo *login* tienen que poner el dominio que hayan comprado y en el campo *password* la contraseña que hayan asignado al registro A. Es importante ponerla entre comillas simples. En la ultima línea tienen que poner de nuevo el dominio.

Además hay que configurar que el ddclient funcione como *daemon*, es decir, como servicio que corra todo el tiempo. Para eso hay que editar:

```
sudo nano /etc/default/ddclient
```

Y cambiar las opciones para que quede así:

```
run_dhclient="false"
run_ipup="false"`
run_daemon="true"`
daemon_interval="300"
```

Para que se hagan efectivos los cambios debemos reiniciar el servicio:

```
sudo service ddclient restart
```

Para ver si la actualización de la dirección se hizo o qué errores pudieron darse pueden ejecutar:

```
sudo ddclient -daemon=0 -debug -verbose -noquiet
```

Y también se pueden revisar los *logs*:

```
cat /var/log/syslog* | grep ddclient
```

Si configuraron todo correctamente deberían ver los cambios en la página de dns.he.net. Si actualizamos la web, en el registro A (*A Record*) donde antes decía 1.1.1.1 ahora debería aparecer la dirección IP pública de la conexión casera donde tenemos conectada nuestra servidora. Para comprobar que su dirección IP está asociada a su dominio pueden hacer desde cualquier terminal:

```
ping nuestrodominio.net
```

Tengan en cuenta que quizás tarde un ratito en actualizarse la IP ya que tiene un TTL de 300 segundos (5 minutos). Así que esperen ese tiempo al menos antes de entrar en pánico.

Si después de un tiempo creen que no se está actualizando correctamente la IP pueden borrar la caché del servicio:

```
sudo mv /var/cache/ddclient/ddclient.cache /var/
cache/ddclient/ddclient.cache.old
```

Y volver a reiniciar el servicio y ver los *logs*. ¡Ojo!, no se pongan a reiniciar mil veces porque el servicio de DNS les puede "banear" por mandarle demasiada información ;).

¿Qué qué pasa si quieren tener varias webs en la misma máquina? Pues necesitan agregar en la configuración de ddclient los datos del

segundo dominio. Es decir, el archivo de configuración **`/etc/ddclient.conf`** quedaría así:

```
pid=/var/run/ddclient.pid
protocol=dyndns2
ssl=yes
use=web
server=dyn.dns.he.net

# Primer dominio
login=labekka.red
password= xxxxxxxxxxx
labekka.red

# Segundo dominio
login=midominio.red
password=xxxxxxxxxxxx
midominio.red
```

A nosotras nos sirvió consultar esta documentación:

- https://www.bidon.ca/en/random/2011-06-16-using-dynamic-dns-feature-dnshenet
- https://joatbloginterim.wordpress.com/2013/06/18/setting-up-ddclient-on-the-raspberry-pi/

Recuerden que a partir de ahora para conectarse a la servidora por SSH pueden ejecutar el siguiente comando desde cualquier computadora conectada a Internet:

```
sudo ssh -p 2251 pi@nuestrodominio.net
```

11. Configuremos Let's Encrypt

Para que el tráfico de nuestra web vaya cifrado a través del protocolo HTTPS necesitamos configurar un certificado SSL/TLS. Estos certificados pueden ser de pago y algo complejos de configurar en Apache. Sin embargo, con la herramienta Certbot (https://certbot.eff.org/), desarrollada por la organización EFF, podemos configurar un certificado de Let's Encrypt (https://letsencrypt.org/) fácilmente y de forma gratuita.

Antes de proceder necesitan haber realizado el paso anterior: conseguir un dominio y tener los DNS dinámicos bien configurados. En https://certbot.eff.org/lets-encrypt/debianstretch-apache hay una guía sobre cómo configurarlo (aunque está un poco desactua- lizada, la verdad). De hecho, dice que hace falta configurar los *backports* para poder descargar Certbot pero hemos comprobado que no es necesario. Así que no lo hagan.

Para instalar Certbot hay que ejecutar el comando:

```
sudo apt install certbot python-certbot-apache
```

Una vez que tengamos instalado Certbot sólo debemos seguir los pasos de la configuración que nos solicita. Tendrán que indicar el do-

minio de su servidora y un correo electrónico. Comencemos la instalación corriendo el asistente:

```
sudo certbot --apache
```

El asistente de configuración también nos preguntará si queremos redirigir las peticiones **`http`** a **`https`**. En nuestro caso diremos que sí tecleando el número 2 y apretando *enter*. Si el dominio no está bien asociado a una dirección IP el asistente no terminará de generar los certificados.

Los certificados generados se pueden ver en **`/etc/letsencrypt/live/dominio/`**. Sólo duran 3 meses pero se renuevan de manera automática. Si quieren comprobar que las renovaciones funcionan pueden ejecutar:

```
sudo certbot renew --dry-run
```

`--dry-run` es un modificador muy útil para probar un comando: lo ejecuta pero sin ejecutarlo realmente y nos devuelve el resultado de lo que ocurriría si lo ejecutáramos. Chilero, ¿no? Ojalá hubiera algo así para la vida humana.

Ahora pueden ir al navegador, teclear su dominio y comprobar que tienen su certificado configurado. Verán un candadito verde en la barra de navegación y que la conexión se hace por el protocolo HTTPS. También pueden comprobar que la redirección de HTTP a HTTPS funciona borrando la s y corroborando que se añade automáticamente.

Si tienen una Raspberry Pi viejita quizás tengan algunos problemas. A nosotras nos pasó con una Raspberry Pi 1 Model B+ porque el

Certbot que está en los repositorios de Debian Stretch (la versión de Debian 9 sobre la cuál está "hecha" Raspbian Stretch) no es compatible con su arquitectura. Acá encontramos algunas orientaciones para estos casos: https://community.letsencrypt.org/t/certbot-on-raspbian-illegal-instruction/15813. Pero no logramos configurarlo. Tuvimos que usar una Raspberry Pi más nueva.

Otra opción también es usar el **`certbot-auto`** que se propone para cualquier sistema operativo cuando no hay desarrollado uno específico: https://certbot.eff.org/lets-encrypt/pip-apache. Con este último paso ya tenemos configurados los certificados SSL/TLS.

¡Felicitaciones! Esta parte de la guía ha tenido muchos pasos pero de extrema relevancia para poder habitar Internet sin exponernos. Y aunque ahora estamos más protegidas no tenemos que confiarnos. Esta es una tarea cotidiana de revisión, actualización y testeo permanente. Al fin y al cabo, aunque sea una máquina, a la servidorcita también hay que cuidarla. Les recordamos la importancia de documentar su proceso: anotar qué hicieron, cómo, con qué dificultades se encontraron y cómo las resolvieron. Una suerte de diario que registre todas las decisiones que hemos tomado a lo largo del proceso. Así podremos reconstruirlo, multiplicarlo y volver a él cada vez que necesitemos.

¿Les parece si desarrollamos nuestra página web?

¡Vamos a la parte V! <3