3 - Algunas posibilidades

Vamos a ver en este capítulo algunas de las herramientas que considero pueden resultarnos de utilidad. El orden en que se muestran no se debe a nada por lo que si te decides a instalar alguna de estas herramientas puedes hacerlo con independencia de tener instaladas las que aparecen listadas con anterioridad. Lo que si "es requisito" para la gran mayoría de ellas es disponer de docker y docker-compose instalado si bien es cierto es que todas ellas pueden instalarse sin esta magnífica herramienta pero entonces perderemos la ventaja que nos aporta el trabajar con contenedores.
Espero que estas herramientas te resulten de interés y si conoces cualquier otra que crees que puede resultar interesante háznoslo saber para valorar la inclusión de la misma a estos materiales

- 3.16 Adguard. Navega por Internet sin anuncios y con seguridad
- 3.17 etc.

3.1 Portainer. Gestión de contenedores

portainer-logo.png

Imagen obtenida de https://www.portainer.io/

Esta herramienta sirve para...

Gestionar los distintos contenedores, imágenes, stacks,... que tengamos en nuestra Raspberry Pi a través de un entorno gráfico en lugar de hacerlo a través del terminal del sistema operativo. Cuenta con una versión BE (Business Edition) y otra CE (Community Edition), usaremos la 2ª.

Web de proyecto y otros enlaces de interés

Página web oficial: https://www.portainer.io/

Repositorio de la versión CE en github: https://github.com/portainer/portainer

Documentación del proyecto: https://docs.portainer.io/

Despliegue

> Si crees que instalar portainer del modo que a continuación se explica es complicado puedes instalarlo a través del método que explicamos en el capítulo 3.3 Linux Media Delivery System (LMDS). No te librará de utilizar la terminal pero quizás te resulte mas amigable.

En la propia documentación podemos encontrar como desplegar Portainer (https://docs.portainer.io/start/install-ce/server/docker/linux). Vamos a recopilar aquí qué hay que hacer y explicar los comandos

```
docker volume create portainer_data
docker run -d -p 8000:8000 -p 9443:9443 --name portainer --restart=always -v
/var/run/docker.sock:/var/run/docker.sock -v portainer_data:/data portainer/portainer-
ce:latest
```

En la primera línea creamos un volumen llamado portainer_data

En la segunda línea lanzamos, desplegamos un contenedor:

- -d (--detach): Ejecuta un contenedor en segundo plano.
- -p (--expose): Nos permite indicar qué puerto del contenedor se corresponde con qué puerto de la máquina anfitriona.
- --name: Nos permite establecer el nombre del contenedor.
- --restart: Nos permite establecer qué queremos que ocurra en caso de que el contenedor falle. En este caso establecemos que se reunicie siempre.
- -v (--volume): Nos permite mapear rutas del contenedor con rutas de la máquina anfitriona.
- El último parámetro que aparece en la ruta `portainer/portainer-ce:latest` es la imagen que se va a ejecutar.

> Visto y explicado cómo realizar la instalación según indica la documentación oficial, nosotros/as vamos a hacerlo de otro modo.

La forma que hemos visto con anterioridad funciona. Podéis usarla sin ningún problema. Ahora bien, dado que en este curso desconozco el nivel de partida de cada compañero/a que lo cursa voy a optar por utilizar un modo de despliegue semejante para cada servicio y, por ello, voy a hacer uso de docker-compose. Vamos allá:

Para ello accedemos al terminal y escribimos lo siguiente:

```
cd $HOME
mkdir portainer
cd portainer
nano docker-compose.yml
```

Dentro del fichero escribimos el siguiente contenido:

```
version: '2'
services:

  portainer:
    container_name: portainer
    image: portainer/portainer-ce
    restart: unless-stopped
    ports:
      - 9000:9000
    volumes:
      - /var/run/docker.sock:/var/run/docker.sock
      - ./volumes/data:/data
```

Para salir del fichero pulsaremos `control + x` y guardaremos los cambios. Posteriormente ponemos en marcha los contenedores con `docker compose up -d` Aparecerá en pantalla algo similar a

portainer-deploy.png

Elaboración propia

Si queremos comprobar que el contenedor está en marcha podemos ejecutar `docker ps --all` lo que nos mostrará todos los contenedor que hay en la máquina. Si queremos ver si, concretamente, está disponible el que acabamos de crear podemos ejecutar docker `ps --all | grep portainer`. Obteniendo unos resultados similares a los siguientes:

docker-ps-grep-portainer.png

Elaboración propia

También podemos tratar de acceder a la interface gráfica a través de un navegador web. Para ello accedemos a través del navegador la Raspberry Pi y al servicio Portainer del siguiente modo `http://<IP>:puerto` en mi caso tengo configurada la raspberry Pi con la IP `192.168.0.201` y portainer con el puerto `9000` por lo que escribo `http://192.168.0.201:9000` y así accedo a la interface web de portainer.

portainer-index.png

Elaboración propia

Funcionamiento

> En el resto del curso el despliegue de los distintos servicios lo haré siempre a través de comandos pero debes saber que con esta herramienta puedes hacer lo mismo en un entorno gráfico.

Como ya indicamos en la introducción esta herramienta es una interface web para docker lo cual facilitará enormemente la vida a aquellas personas que no estén acostumbradas a trabajar desde una interface de texto (terminal).

Nada mas acceder veremos una imagen como la que hemos visto un par de párrafos más arriba en ella podemos gestionar los usuarios, entornos, logs, parámetros de configuración... de este primer menú lateral no voy a comentaros nada. Si pinchamos en local veremos una imagen como la siguiente:

portainer-local.png

Elaboración propia

En la misma os he remarcado 4 zonas:

- zona 1 (verde): Tenemos acceso a los diferentes contenedores, imágenes, redes y volúmenes de nuestro entorno local.
- zona 2 (azul claro): Acceso a las mismas secciones que teníamos antes de entrar en el entorno local.
- zona 3 (rojo): Información del entorno.
- zona 4 (morado): Similar a la zona 1 con algo de información adicional.

Si accedemos a, por ejemplo, `Containers` tendremos un listado de todos los contenedores descargados:

portainer-containers.png

Elaboración propia

Dónde podemos ver si hay contenedores detenidos, fallando, sanos,... también podemos acceder al detalle de cualquiera de ellos y dentro del detalle detenerlo, reiniciarlo, pausarlo, borralo,... ver los logs, acceder al terminar del contenedor,... todo ello sin necesidad de conocer los comandos

docker que hay por detrás:

portainer-container-details.png

Elaboración propia

portainer-container-details-logs.png

Elaboración propia

Es bastante probable que nunca tengáis que recurrir a estas opciones pero si en alguna ocasión las necesitáis no tendréis que buscar que con `docker logs nombre_del_contenedor --follow` podéis hacer lo mismo que haciendo 4 clicks.

Acceder a las secciones de `Images` o `Volumes` nos puede resultar muy útil para de un vistazo ver, respectivamente, que imágenes o volúmenes no están en uso y así borrarlas para que dejen de ocupar espacio en nuestro disco duro.

portainer-image.png

Elaboración propia

Además, por si fuera poco, nos agrupa los contenedores por stacks de modo que nos facilita ver si los contenedores asociados a un determinado servicio (hay servicios que pueden requerir el funcionamiento de mas de 1 contenedor) están operativos o no.

portainer-stacks.png

Elaboración propia

portainer-stack-details.png

Elaboración propia

Y, ya para terminar pero no por ello menos importante, nos ofrece también la posibilidad de crear contenedores a partir de plantillas de aplicaciones preexistentes. Esta funcionalidad puede permitirnos desplegar servicios en segundos sin conocer ni un solo comando de docker (si bien no es lo deseable):

portainer-templates.png

Elaboración propia

Como nos ocurrirá en servicios posteriores, esta herramienta podría dar por si misma para un curso dedicado. Os presento aquellas cuestiones que me parecen mas relevantes a fin de que seáis capaces de continuar vosotros/as desde este punto de partida.

3.2 DIUN. Notificador de nuevas imágenes

diun-logo.png

Imagen obtenida de https://crazymax.dev/diun/

Esta herramienta sirve para...

enterarnos cuando una nueva imagen (para docker) está disponible. DIUN son las siglas de Docker Image Update Notifier.

Web de proyecto y otros enlaces de interés

Web del proyecto: https://crazymax.dev/diun/

Repositorio de código: https://github.com/crazy-max/diun

Puesta en marcha

Si bien hay varios modos de desplegar el servicio DIUN nosotros, en este curso, vamos a optar por hacerlo a través de docker-compose pues creo es el modo mas sencillo en el que podemos hacer convivir varios servicios sin que unos acepten a otros. Para ello accedemos al terminal y escribimos lo siguiente:

```
cd $HOME
mkdir diun
cd diun
nano docker-compose.yml
```

Dentro del fichero escribimos el siguiente contenido

```
version: "3.5"

services:
  diun:
    image: crazymax/diun:latest
    container_name: diun
    command: serve
    volumes:
      - "./data:/data"
      - "./diun.yml:/diun.yml:ro"
      - "/var/run/docker.sock:/var/run/docker.sock"
    environment:
      - "TZ=Europe/Madrid"
      - "LOG_LEVEL=info"
      - "LOG_JSON=false"
    restart: always
```

Para salir del fichero pulsaremos `control + x` y guardaremos los cambios. Posteriormente ponemos en marcha los contenedores con `docker compose up -d` Aparecerá en pantalla algo similar a

diun-deploy.png

Elaboración propia

Y, si queremos, podemos ejecutar `docker ps | grep diun` para comprobar si entre todos los contenedores docker en ejecución hay alguno con el nombre diun. Veremos algo similar a

diun-grep.png

Elaboración propia

De acuerdo a la documentación sobre comandos que aparece en https://crazymax.dev/diun/usage/command-line/ podemos ejecutar cualquiera de los comandos que ahí aparecen ejecutando `docker exec diun comando` por ejemplo `docker exec diun diun image list` que nos mostrará algo similar a

diun-exec.png

Elaboración propia

De todos modos, **lo interesante de esta herramienta es que sea ella misma quién nos notifique cuando hay una nueva imagen sin necesidad de que nosotros/as ejecutemos nada**. Para ello hay que configurar las notificaciones de acuerdo a la documentación que aparece aquí https://crazymax.dev/diun/config/notif/ ¡Vamos allá! En la terminal escribiremos:

```
cd $HOME/diun
nano diun.yml
```

Dentro del fichero, que será en el cual establezcamos los métodos de notificación, escribimos el siguiente contenido:

```
watch:
  workers: 20
  schedule: "0 */6 * * *"
  firstCheckNotif: false

providers:
  docker:
    watchByDefault: true

notif:
  mail:
    host: localhost
    port: 25
    ssl: false
    insecureSkipVerify: false
    from: tu_email@tu_email.com
    to:
      - tu_email@tu_email.com
    templateTitle: "{{ .Entry.Image }} released"
    templateBody: |
      Docker tag {{ .Entry.Image }} which you subscribed to through {{ .Entry.Provider }}
provider has been released.
  telegram:
    token: tu_token_en_telegram
```

```
    chatIDs:
      - el_id_de_tu_chat
    templateBody: |
      Docker tag {{ .Entry.Image }} which you subscribed to through {{ .Entry.Provider }}
provider has been released.
```

A continuación reiniciamos el contenedor con los comandos

```
cd $HOME/diun
docker-compose down
docker-compose up -d
```

Para salir del fichero pulsaremos `control + x` , guardaremos los cambios y *¡et voilà!* ya están configuradas las notificaciones para Telegram y email. Deberás cambiar los valores a tus valores y establecer solo aquellos servicios a través de los que quieres que se te notifique.

Cómo actualizar la imágenes

En mi caso lo tengo configurado para que me notifique a través de un bot de Telegram por ello recibo notificaciones con este aspecto:

diun-notificacion.jpg

Elaboración propia

Si has instalado Portainer (lo hicimos en el capítulo anterior) es muy sencillo. Accedemos a través del navegador la Raspberry Pi y al servicio Portainer del siguiente modo `http://<IP>:puerto` en mi caso tengo configurada la raspberry Pi con la IP `192.168.0.201` y portainer con el puerto `9000` por lo que escribo `http://192.168.0.201:9000` y así accedo a la interface web de portainer.

portainer-index.png

Elaboración propia

Pincho en el entorno local y accedo a una pantalla como la siguiente:

portainer-dashboard.png

Elaboración propia

Selecciono imágenes, busco la que me interesa (en este ejemplo, la que corresponde a jackett) y pincho en ella. De modo que veré algo como

portainer-image-details.png

Elaboración propia

Y ahora elijo la opción que dice "Pull from regirtry" (la 2ª opción).

portainer-image-details-detail.png

Elaboración propia

Nos preguntará por el registro a usar pudiendo dejar la opción por defecto sin mayor problema y comenzará la descarga de la imagen. De este modo habremos descargado la última imagen disponible de, en este ejemplo, jackett.

Todo lo anterior podíamos haberlo hecho ejecutando desde el terminal `docker image pull linuxserver/jackett:latest`

> Con lo hecho hasta ahora habremos actualizado una determinada imagen PERO si algún contenedor está usando dicha imagen no pasará a utilizarla hasta que tiremos y levantemos el contenedor de nuevo

3.3 Linux Media Delivery System (LMDS). Centro de descargas

Esta herramienta sirve para...

disponer de un centro de descargas y centro de entretenimiento de diferente contenido multimedia.

Web de proyecto y otros enlaces de interés

- Web del proyecto: https://greenfrognest.com/
- Repositorio de código: https://github.com/GreenFrogSB/LMDS
- Configurar el servicio con un disco duro externo
  https://greenfrognest.com/LMDSUSBdrive.php#usbdrive
- Preguntas precuentes https://greenfrognest.com/faqlist.php

Instalación

En este caso no vamos a partir de un fichero docker-compose sino que vamos a clonar un respositorio de github y una vez clonado vamos a ejecutar un script que contiene. Dicho script se encargará de crear el fichero docker-compose.

```
cd $HOME
sudo apt-get update
sudo apt-get upgrade
sudo apt-get install git
git clone https://github.com/GreenFrogSB/LMDS.git ~/LMDS
```

```
cd ~/LMDS
./deploy.sh
```

En las líneas 2 y 3 actualizamos repositorios y el sistema. En la línea 4 instalamos git (si ya está instalado no lo instalará). En la línea 5 copiamos el repositorio de github dónde está el código. En la línea 6 accedemos a la carpeta LMDS. En la línea 7 ejecutamos el script de LMDS que nos irá preguntando qué queremos instalar a través de un menú interactivo como el que se ve a continuación:

LMDS_main_menu.png

La 1ª opción "Install Docker" nos permitirá instalar Docker y Docker-compose. Si no lo tenemos instalado deberemos seleccionarla. Si ya lo tenemos instalado podemos seleccionar directamente la 2ª opción "Build LMDS Stack" que nos mostrará una pantalla como la siguiente y que nos permitirá elegir qué servicios queremos instalar:

LMDS_stack.png

Algunos de los contenedores que aquí aparecen ya los hemos visto en este curso, es el caso de Portainer, y otros los veremos mas adelante, es el caso de Pi-Hole. Centrándonos en el caso que nos ocupa, la "construcción" de un centro de descargas mi recomendación es seleccionar:

- **Sonarr**: Si queremos rastrear series.
- **Radarr**: Si queremos rastrear películas.
- **Lidarr**: Si queremos rastrear música.
- **Bazarr**: Subtítulos.
- **Jackett**: Para que actúe de intermediario entre los programas antes indicados y el cliente de descargas.
- **Deluge** o **qBittorrent** o **Transmision**: Clientes de descargas (yo usaré Deluge en este curso)
- **Portainer**: No es necesario para hacer funcionar el centro de descargas pero recomendaría su instalación para facilitarnos la gestión del stack.

Cuando pulsemos ok el script se encargará de crear el fichero docker-compose. Ahora, como ya sabemos, ejecutaremos el comando `docker compose up -d` y todo se pondrá en marcha. Tras ejecutar el comando veremos algo similar s:

lmds-deploy.png

Elaboración propia

También podemos hacerlo desde Portainer si lo tenemos funcionando.

Si vemos el contenido del directorio veremos que aparece el fichero docker-compose-yml del cual podemos ver su contenido. En el mismo veremos que se han creado una serie de volúmenes. El contenido del directorio será semejante a lo que vemos en la siguiente imagen

LMDS_ls.png

> Se curioso/a y mira el contenido del fichero docker-compose.yml generado. Verlo es un buen modo de aprender.

En el subapartado Funcionamiento vamos a explicar como configurar el centro de descargas.

Os dejo además un vídeo de youtube donde, en inglés, nos indican lo mismo que os indico arriba

https://www.youtube.com/embed/oLxsSQIqOMw

> Estamos instalando todo este *stack* con LMDS por facilitar la tarea pero podemos instalar individualmente cada programa a través de 1 fichero docker-compose

Funcionamiento

> En mi caso tengo la raspberry conectada a la TV por cable HDMI por lo que únicamente debo seleccionar en la TV como entrada HDMI y ahí, con un ratón inalámbrico conectado a la raspberry, elegir el contenido a reproducir. Si no es vuestro caso deberéis recurrir a soluciones como Plex o Jellyfin, que el instalador también deja instalar.

En este caso creo que lo mas sencillo es recurrir a un vídeo dónde nos explican como usar en conjunto todas estas herramientas:

https://www.youtube.com/embed/mvufLzlOS4I?start=490

En el vídeo nos cuentan de un modo muy básico las diferentes posibilidades de cada herramienta pero si le dedicáis tiempo a ir mirando las diferentes configuraciones veréis que se tratan de programas muy potentes. Por ejemplo podemos seleccionar en qué idiomas queremos que busque el contenido o en qué calidad mínima estamos dispuestos a ver lo que descargue.

3.4 MultiBOB (Antes BOBcera). Convierte tu Raspberry Pi en una videoconsola retro

Esta herramienta sirve para...

pasárselo bien jugando a juegos de hace unas cuantas décadas. Para jugar es altamente recomendable adquirir un mando que funcione por USB y que preferiblemente sea inalámbrico. En mi caso adquirí este mando https://www.amazon.es/gp/product/B01KVC4K3O pero vale cualquiera que se pueda conectar por USB, WIFI o Bluetooth a la Raspberry Pi.

En el siguiente vídeo (de mas de 1h de duración) el creador del proyecto nos habla del mismo y nos muestra sus principales características:

https://www.youtube.com/embed/NO56iPNd8Tc

Web de proyecto y otros enlaces de interés

En este caso nuestro punto de partida es un canal de Telegram. El canal se llama "BOB - Mejores juegos viejunos para Raspberry pi0/1/2/3/4, Batocera,..." (podéis uniros al canal a través de este enlace https://t.me/BOB_retropie_windows_dudas) y en la web https://telegra.ph/Best-of-the-Best---Solo-los-mejores-juegos-09-25 tenemos toda la información para comenzar. En este caso no se trata de algo exclusivo para la Raspberry Pi sino que de modo muy sencillo también podéis usarlo en sistemas operativos Windows y dispositivos varios.

En el caso que nos ocupa, la Raspberry Pi, tenemos toda la información necesaria para ponerla en marcha en https://telegra.ph/BOBcera-juegos-BOB--Batocera-RaspberryPCWin-08-16 en ese caso no haremos uso de docker ni docker-compose sino que grabaremos la imagen en una tarjeta microSD, la introduciremos en la Raspberry Pi y ya estaremos listos/as para jugar.

> A la hora de descargar el fichero NO DESCARGUES TODO. Selecciona del fichero torrent únicamente aquello que se corresponda al dispositivo y/o sistema operativo que necesites.

El fichero .torrent, a fecha de escribir este texto son 800GB, por ello, a la hora de ponerlo a descargar selecciona que se descargue únicamente lo que necesites. Mita la siguiente imagen:

multibob-que-descargar.png

Imagen obtenida de https://telegra.ph/BOBcera-juegos-BOB--Batocera-RaspberryPCWin-08-16

3.5 Pi-hole. Privacidad en la navegación

pihole-logo.png

Imagen obtenida de https://github.com/pi-hole/docker-pi-hole/

Esta herramienta sirve para...

navegar por internet con menos publicidad y evitar muchos sistemas de rastreo. Encontrarás una solución similar en el capítulo 3.16 Adguard.

Web de proyecto y otros enlaces de interés

Web del proyecto: https://pi-hole.net/

Repositorio de código: https://github.com/pi-hole/docker-pi-hole/

Documentación instalación con docker-compose: https://github.com/pi-hole/docker-pi-hole/#quick-start

Despliegue

> Si crees que instalar portainer del modo que a continuación se explica es complicado puedes instalarlo a través del método que explicamos en el capítulo 3.3 Linux Media Delivery System (LMDS). No te librará de utilizar la terminal pero quizás te resulte mas amigable.

Si accedemos al repositorio de código encontraremos directamente el fichero docker-compose.yml que necesitamos en https://github.com/pi-hole/docker-pi-hole/#quick-start pero antes de crear el fichero haremos lo siguiente:

```
cd $HOME
mkdir pi-hole
cd pi-hole
nano docker-compose.yml
```

y dentro de este fichero copiaremos el contenido de la url anterior:

```
version: "3"

# More info at https://github.com/pi-hole/docker-pi-hole/ and https://docs.pi-hole.net/
services:
  pihole:
    container_name: pihole
    image: pihole/pihole:latest
    # For DHCP it is recommended to remove these ports and instead add: network_mode: "host"
    ports:
      - "53:53/tcp"
      - "53:53/udp"
      - "67:67/udp" # Only required if you are using Pi-hole as your DHCP server
      - "8088:80/tcp"
    environment:
      TZ: 'Europe/Madrid'
      WEBPASSWORD: 'VUESTRA-CLAVE'
    # Volumes store your data between container upgrades
    volumes:
      - './etc-pihole:/etc/pihole'
      - './etc-dnsmasq.d:/etc/dnsmasq.d'
    #   https://github.com/pi-hole/docker-pi-hole#note-on-capabilities
    cap_add:
      - NET_ADMIN # Required if you are using Pi-hole as your DHCP server, else not needed
    restart: unless-stopped
```

como en ocasiones anteriores, para guardar los cambios pulsaremos `control + x` y cuando nos pregunte aceptaremos. Una vez volvamos a estar en el terminal, escribiremos `docker compose up -d` para lanzar los servicios ubicados dentro del fichero docker-compose.yml. Veremos algo como:

pihole-deploy.png

> En el fichero docker-compose.yml puedes descomentar la línea WEBPASSWORD quitando el símbolo # inicial y, a continuación, establecer una contraseña de acceso que tu definas.

Funcionamiento

Lo más fácil y sencillo para hacer uso de esta herramienta es que en la configuración de la conexión de nuestros dipositivos (móviles, ordenadores, TVs,...) establezcamos como DNS primario la IP de nuestra raspberry. De este modo, cuando los dispositivos antes mencionados se conecten a internet nuestra Raspberry Pi a través de Pi-Hole actuará como servidor DNS y evitaremos una gran cantidad de publicidad y de rastreo en internet.

Si queremos ver las posibiliades de la herramienta accederemos a través del navegador la Raspberry Pi y al servicio Pi-hole del siguiente modo `http://<IP>:puerto` en mi caso tengo configurada la raspberry Pi con la IP `192.168.0.201` y Pi-hole con el puerto `8088` por lo que escribo `http://192.168.0.201:8088` y así accedo a la interface web.

pi-hole-web.png

Elaboración propia

A continuación accedemos al panel de administración pinchando en el enlace que aparece. Vuestra contraseña será la que hayáis establecido en el fichero docker-compose en el atributo WEBPASSWORD.

pi-hole-user-pass.png

Elaboración propia

Tras poner el usuario y contraseña accederéis al panel de control

pi-hole-dashboard.png

Elaboración propia

Una vez que estéis un rato navegando por internet en los dispositivos en los que habéis cambiado el DNS empezaréis a ver la gran cantidad de información que bloquea. También podréis ver cuándo y a qué sitios se conecta qué dispositivo.

pi-hole-dashboard-data.png

Elaboración propia

> Lo que os he contado con anterioridad es el uso más básico de este servicio. Os animo a leer en la documentación y en manuales sus diferentes posibilidades a fin de sacarle el máximo jugo posible.

3.6 ¿IP dinámica? No hay problema. DuckDNS

ducky_icon.png

Imagen obtenida de https://www.duckdns.org/

Esta herramienta sirve para...

facilitarnos el acceder a nuestra red de casa desde fuera de la misma.

Vamos a aclarar la afirmación anterior. Nuestro router (que conecta la red de nuestro domicilio con la red del exterior) tiene una IP en esa red exterior que va cambiando con el tiempo salvo que tengamos contratado con nuestro ISP el servicio de IP fija. Dado que aprenderse una IP es complicado y dado que ese dato cambia con el tiempo el servicio de duckdns es muy interesante pues nos permite "ponerle nombre" a nuestra IP (DNS) ya que es más sencillo aprenderse pabloruizsoria.duckdns.org que 148.3.110.57 que, como hemos dicho, cambia con el tiempo. Este servicio no solo va a "poner nombre" a nuestra IP sino que va a hacer que cuando nuestra IP cambie el nombre pase a apuntar al nuevo valor de la IP.

Estaríamos hablando de un servicio DDNS (DNS dinámico)

Web de proyecto y otros enlaces de interés

Web de duck DNS: https://www.duckdns.org/

Repositorio que usaremos para instalar este servicio: https://github.com/linuxserver/docker-duckdns

Despliegue

Primero deberemos acceder a la web https://www.duckdns.org/ aquí crearemos un subdominio con el nombre que nos interese. De esta web nos interesará tanto el nombre del subdominio como el campo token que aparece en pantalla.

duckdns-dashboard.png

Elaboración propia

Después, como con servicios anteriores, accedemos a la terminal y escribimos

```
cd $HOME
mkdir duckdns
cd duckdns
nano docker-compose.yml
```

Dentro del fichero escribimos el siguiente contenido

```
version: "2.1"
services:
  duckdns:
    image: lscr.io/linuxserver/duckdns:latest
    container_name: duckdns
    environment:
      - PUID=1000 #optional
      - PGID=1000 #optional
      - TZ=Etc/UTC #optional
      - SUBDOMAINS=subdomain1_que_hayas_configurado,subdomain2_que_hayas_configurado
      - TOKEN=token_que_aparece_en_la_web_duckdns_mira_mi_imagen
      - LOG_FILE=false #optional
    volumes:
      - /path/to/appdata/config:/config #optional
    restart: unless-stopped
```

Para salir del fichero pulsaremos `control + x` y guardaremos los cambios. Posteriormente ponemos en marcha los contenedores con `docker compose up -d` Aparecerá en pantalla algo similar a

duckdns-deploy.png

Elaboración propia

Funcionamiento

Si ahora accedéis a https://www.duckdns.org/ veréis que en el subdominio que acabáis de crear tiene un valor en current ip y que se ha actualizado hace poco.

3.7 WireGuard. Servidor de VPN

wireguard-logo.png

Imagen obtenida de https://www.wireguard.com/

Esta herramienta sirve para...

crear una VPN de un modo extremadamente sencillo.

> Genial Pablo pero... ¿¡¿¡ para que quiero yo una VPN ?!?!

Hasta el momento hemos ido desplegando diferentes servicios a los que hemos asignado diferentes puertos y cuando nos hemos querido conectar a ellos hemos escrito la IP que tiene la Raspberry Pi **dentro de nuestra red** y el puerto que le hemos asignado en el fichero docker-compose. Ahora bien, es bastante probable que también queramos acceder a estos servicios desde **fuera de nuestra red**. Aquí básicamente se nos abren 2 posibilidades:

1. Acceder al router y "abrir" puertos.
2. Crear una VPN y conectarnos a ella.

Vamos a optar por la segunda opción por seguridad y comodidad. Al conectarnos a la VPN que creemos será como si estuviésemos conectados a la red de casa por lo que para conectarnos a nuestros servicios seguiremos utilizando la misma IP y puerto que en nuestro domicilio. Con ello conseguimos exponer menos puertos de nuestro router al exterior (**seguridad**) y no tener que configurar nada en el router ni aprender nada (**comodidad**).

Web de proyecto y otros enlaces de interés

Web del proyecto: https://www.wireguard.com/

Repositorio de código que podemos utilizar: https://github.com/linuxserver/docker-wireguard

Despliegue

Como en ocasiones anteriores vamos a hacer con docker-compose para ello accedemos al terminal y escribimos

```
cd $HOME
mkdir wireguard
cd wireguard
nano docker-compose.yml
```

y dentro del fichero escribiremos el siguiente contenido

```
version: "2.1"
services:
  wireguard:
    image: lscr.io/linuxserver/wireguard:latest
    container_name: wireguard
    cap_add:
      - NET_ADMIN
      - SYS_MODULE
    environment:
      - PUID=1000
      - PGID=1000
      - TZ=Europe/Madrid
      - SERVERURL=vuestrodominio.duckdns.org #optional
      - SERVERPORT=51820 #optional
      - PEERS=1 #optional. Numero de personas que se vayan a conectar a la VPN
      - PEERDNS=auto #optional
      - INTERNAL_SUBNET=10.13.13.0 #optional
      - ALLOWEDIPS=0.0.0.0/0 #optional
      - PERSISTENTKEEPALIVE_PEERS= #optional
      - LOG_CONFS=true #optional
    volumes:
```

```
    - ./config:/config
    - /lib/modules:/lib/modules #optional
  ports:
    - 51820:51820/udp
  sysctls:
    - net.ipv4.conf.all.src_valid_mark=1
  restart: unless-stopped
```

como en ocasiones anteriores, para guardar los cambios pulsaremos control + x y cuando nos pregunte aceptaremos. Una vez volvamos a estar en el terminal, escribiremos `docker compose up -d` para lanzar los servicios ubicados dentro del fichero docker-compose. El resultado será similar a:

wireguard-deploy.png

Elaboración propia

Funcionamiento

Si prestamos atención al fichero docker-compose veremos que, en el apartado `volumes`, hemos creado uno volumen llamado `config`. Si desde `$HOME/wireguard` listamos el contenido del directorio con `ls -l` veremos que hay un directorio llamado config. Si accedemos al mismo `cd config` y listamos el contendio veremos que se han creado tantas carpetas `peerX` como PEERS hayamos establecido en el fichero docker-compose. En mi caso tengo 3. Si accedemos a una de esas carpetas dentro hay 2 ficheros relevantes los ficheros peerx.conf y peerx.png. El 1º tiene la configuración del fichero para conectarnos a la VPN con esos datos y el 2º tiene una imagen con un código QR que, una vez escaneado, nos configura directamente la VPN.

wireguard-commands.png

Elaboración propia

Configuración desde el teléfono móvil

Desde nuestro teléfono Android accedemos a https://play.google.com/store/apps/details?id=com.wireguard.android e instalamos el cliente de VPN. Una vez instalada la APP pulsamos en el símbolo + y seleccionamos escanear desde código QR. Escaneamos el fichero png comentado en el párrafo anterior y ya está configurada la conexión. Ahora, cada vez que queramos conectarnos a nuestra VPN desde fuera de nuestra red activaremos la VPN y estaremos a todos los efectos conectados a nuestra red. Dejo una serie de capturas de

pantalla del proceso.

wireguard-add.jpg

Elaboración propia

wireguard-escanear-qr.jpg

Elaboración propia

wireguard-rename.jpg

Elaboración propia

wireguard-activado.jpg

Elaboración propia

Una vez hechos todos los pasos anteriores **y con la VPN activa** únicamente deberemos introducir en el navegador la IP que tiene nuestra Raspberry **en nuestra red local** y el puerto del servicio al que queramos acceder. De este modo nos estaremos conectando a este servicio desde fuera de nuestra red como si estuviéramos en ella.

3.8 Duplicati. Gestión de copias de seguridad

duplicati-fb-share-v1.png

Imagen obtenida de https://www.duplicati.com/

Esta herramienta sirve para...

Crear copias de seguridad

Web de proyecto y otros enlaces de interés

Web del proyecto: https://www.duplicati.com/

Repositorio de código: https://github.com/linuxserver/docker-duplicati

Despliegue

Como en ocasiones anteriores vamos a hacer con docker-compose para ello accedemos al terminal y escribimos

```
cd $HOME
mkdir duplicati
cd duplicati
nano docker-compose.yml
```

y dentro del fichero escribiremos el siguiente contenido

```
version: "2.1"
services:
  duplicati:
    image: lscr.io/linuxserver/duplicati:latest
    container_name: duplicati
    environment:
      - PUID=1000
      - PGID=1000
      - TZ=Europe/Madrid
      - CLI_ARGS= #optional
    volumes:
      - ./config:/config
      - ./backups:/backups
      - ./source:/source
    ports:
      - 8200:8200
    restart: unless-stopped
```

como en ocasiones anteriores, para guardar los cambios pulsaremos `control + x` y cuando nos pregunte aceptaremos. Una vez volvamos a estar en el terminal, escribiremos `docker compose up -d` para lanzar los servicios ubicados dentro del fichero docker-compose.yml. El resultado será similar al siguiente:

duplicati-deploy.png

Elaboración propia

Si accedéis en vuestro navegador a la IP de la raspberry y al puerto que hemos establecido (8200). En mi caso sería http://192.168.0.201:8200 deberíais ver algo como:

dupliciti-web.png

Elaboración propia

Funcionamiento

Vamos a crear una nueva copia de seguridad. Para ello, a modo de ejemplo, voy a crear una copia de seguridad de lo que tengo en un servidor web propio dónde alojo mi proyecto personal

trivinet.com (y que aprovecho la ocasión para recomendaros lo probéis con vuestro alumnado). De este servidor ya creo copias de seguridad y las saco a una Raspberry Pi diferente pero ahora voy a hacerlo de este modo. Vamos allá.

Seleccionaré la opción `Añadir copia de seguridad` y después `Configurar nueva copia de seguridad` para terminar pulsando `Siguiente`:

duplicati-1.png

Elaboración propia

En esta pantalla estableceré el `nombre` que quiero darle a la copia de seguridad y si quiero `cifrar` la misma. Terminaré pulsando `Siguiente`:

duplicati-2.png

Elaboración propia

En esta pantalla voy a indicarle el destino dónde quiero dejar la copia de seguridad. En este caso quiero conectarme a una máquina remota a través del protocolo SSH. Para ello debo darle la dirección de la máquina a la que voy a conectarme (IP y puerto), de igual modo debo indicar que ruta debe copiar y unas credenciales de acceso. Para asegurarme de que duplicati puede acceder pulsaré en conexión de prueba y una vez compruebe que todo es correcto seleccionaré siguiente:

duplicati-3.png

Elaboración propia

Ahora debo elegir qué quiero copiar de mi máquina. Tengo la posibilidad de filtrar y expluir aquellos ficheros que no me interese copiar. Una vez seleccionado pulsaremos siguiente.

Ahora nos encontramos en la pantalla en la cual estableceremos las fechas y horas en las que queremos que se lleve a cabo la copia de seguridad. Una vez establecido aquello que nos interese pulsaremos siguiente.

Ya nos encontramos en la última pantalla en la cual podemos establecer una serie de opciones generales y avanzadas. Tras marcar aquello que nos interese seleccionaremos Guardar.

Si todo ha sido satisfactorio veremos ahora en la pantalla de inicio que se ha creado una tarea con los datos que hemos ido marcado,

duplicati-4.png

Elaboración propia

Recuerda que debes comprobar periódicamente que las copias de seguridad se están realizando. Además debes probar, también periódicamente, que las mismas contienen los datos que quieres guardar y que son funcionales. No quieras descubrir en un momento de necesidad y aquellos que creías que estaba sucediendo correctamente estab ocurriendo o no.

3.9 File Browser. Explorador de ficheros en remoto

filebrowser-logo.png

Imagen obtenida de https://github.com/filebrowser/filebrowser

Esta herramienta sirve para...

gestionar carpetas y directorios a partir de una interface gráfica.

Web de proyecto y otros enlaces de interés

Página web del proyecto: https://filebrowser.org/

Repositorio de código: https://github.com/filebrowser/filebrowser

Puesta en marcha

Como en ocasiones anteriores vamos a hacer con docker-compose para ello accedemos al terminal y escribimos

```
cd $HOME
mkdir file-browser
cd file-browser
nano docker-compose.yml
```

y dentro del fichero escribiremos el siguiente contenido

```
version: "3"

services:
  filebrowser:
    image: hurlenko/filebrowser
    user: "1000:1000"
    ports:
      - 8080:8080
    volumes:
      - /:/data
      - ./config:/config
    environment:
      - FB_BASEURL=/filebrowser
    restart: always
```

como en ocasiones anteriores, para guardar los cambios pulsaremos `control + x` y cuando nos pregunte aceptaremos. Una vez volvamos a estar en el terminal, escribiremos `docker compose up -d` para lanzar los servicios ubicados dentro del fichero docker-compose. El resultado será similar al siguiente:

file-browser-deploy.png.png

Elaboración propia

Si ahora accedemos al servicio como venimos haciendo veremos algo similar a:

file-browser-admin.png

Elaboración propia

El usuario y contraseña por defecto es `admin`.

Funcionamiento

Una vez que hemos accedido con el usuario y contraseña por defecto accederemos a un entorno similar al que nos presenta Google Drive y a través del cuál podremos ver el contenido de nuestros directorios, crear ficheros, gestionar usuarios y sus permisos,... también podemos cambiarlo a

castellano si nos es necesario.

filebrowser-home.png

Elaboración propia

Con esta solución podemos resolver los espacios de almacenamiento gratuito que a día de hoy presentan soluciones como google drive y sus 15 GB gratuitos. En mi caso, conectado a la Raspberry Pi, tengo conectado un disco duro de 4TB en al cual almaceno una copia de seguridad de mis fotos, películas, series,...

> En caso de que tengas problemas de almacenamiento en tu cuenta personal y gratuita de Google puedes hacer una copia de seguridad de tus datos con Google Takeout y una vez que tienes esos datos en tu poder puedes borrarlos de Google volviendo a conseguir espacio. Con soluciones como la que hemos visto en este apartado y en el siguiente podrás acceder a tu contenido multimedia sin limitaciones de espacio y sin pagar suscripciones.

3.10 PhotoPrism. Alternativa a las nubes de fotos

photoprism-logo.png

Imagen obtenida de https://www.photoprism.app/

Esta herramienta sirve para...

gestionar tu contenido multimedia de un modo avanzado. Haciendo uso de la demo https://demo.photoprism.app/library/browse podrás hacerte una idea de sus posibilidades.

Web de proyecto y otros enlaces de interés

Web: https://www.photoprism.app/

Repositorio: https://github.com/photoprism/photoprism

Puesta en marcha

Como en ocasiones anteriores vamos a hacer con docker-compose para ello accedemos al terminal y escribimos

```
cd $HOME
mkdir photoprism
cd photoprism
nano docker-compose.yml
```

y dentro del fichero copiaremos el siguiente contenido (adaptado del fichero visto en https://dl.photoprism.app/docker/docker-compose.yml):

```
version: '3.5'

services:
  photoprism:
    image: photoprism/photoprism:latest
    depends_on:
      - mariadb
    ## Don't enable automatic restarts until PhotoPrism has been properly configured and
tested!
    ## If the service gets stuck in a restart loop, this points to a memory, filesystem,
network, or database issue:
    ## https://docs.photoprism.app/getting-started/troubleshooting/#fatal-server-errors
    # restart: unless-stopped
    security_opt:
      - seccomp:unconfined
      - apparmor:unconfined
    ports:
      - "2342:2342" # HTTP port (host:container)
    environment:
      PHOTOPRISM_ADMIN_USER: "admin"                 # superadmin username
      PHOTOPRISM_ADMIN_PASSWORD: "insecure"          # initial superadmin password (minimum 8
characters)
      PHOTOPRISM_AUTH_MODE: "password"               # authentication mode (public, password)
      PHOTOPRISM_SITE_URL: "http://photoprism.me:2342/"  # server URL in the format
"http(s)://domain.name(:port)/(path)"
      PHOTOPRISM_ORIGINALS_LIMIT: 5000               # file size limit for originals in MB
(increase for high-res video)
      PHOTOPRISM_HTTP_COMPRESSION: "gzip"            # improves transfer speed and bandwidth
utilization (none or gzip)
      PHOTOPRISM_LOG_LEVEL: "info"                   # log level: trace, debug, info, warning,
error, fatal, or panic
      PHOTOPRISM_READONLY: "false"                   # do not modify originals directory
(reduced functionality)
```

```
      PHOTOPRISM_EXPERIMENTAL: "false"              # enables experimental features
      PHOTOPRISM_DISABLE_CHOWN: "false"             # disables updating storage permissions
via chmod and chown on startup
      PHOTOPRISM_DISABLE_WEBDAV: "false"            # disables built-in WebDAV server
      PHOTOPRISM_DISABLE_SETTINGS: "false"          # disables settings UI and API
      PHOTOPRISM_DISABLE_TENSORFLOW: "false"        # disables all features depending on
TensorFlow
      PHOTOPRISM_DISABLE_FACES: "false"             # disables face detection and recognition
(requires TensorFlow)
      PHOTOPRISM_DISABLE_CLASSIFICATION: "false"    # disables image classification (requires
TensorFlow)
      PHOTOPRISM_DISABLE_RAW: "false"               # disables indexing and conversion of RAW
files
      PHOTOPRISM_RAW_PRESETS: "false"               # enables applying user presets when
converting RAW files (reduces performance)
      PHOTOPRISM_JPEG_QUALITY: 85                   # a higher value increases the quality
and file size of JPEG images and thumbnails (25-100)
      PHOTOPRISM_DETECT_NSFW: "false"               # automatically flags photos as private
that MAY be offensive (requires TensorFlow)
      PHOTOPRISM_UPLOAD_NSFW: "true"                # allows uploads that MAY be offensive
(no effect without TensorFlow)
      # PHOTOPRISM_DATABASE_DRIVER: "sqlite"        # SQLite is an embedded database that
doesn't require a server
      PHOTOPRISM_DATABASE_DRIVER: "mysql"           # use MariaDB 10.5+ or MySQL 8+ instead
of SQLite for improved performance
      PHOTOPRISM_DATABASE_SERVER: "mariadb:3306"    # MariaDB or MySQL database server
(hostname:port)
      PHOTOPRISM_DATABASE_NAME: "photoprism"        # MariaDB or MySQL database schema name
      PHOTOPRISM_DATABASE_USER: "photoprism"        # MariaDB or MySQL database user name
      PHOTOPRISM_DATABASE_PASSWORD: "insecure"      # MariaDB or MySQL database user password
      PHOTOPRISM_SITE_CAPTION: "AI-Powered Photos App"
      PHOTOPRISM_SITE_DESCRIPTION: ""               # meta site description
      PHOTOPRISM_SITE_AUTHOR: ""                    # meta site author
    working_dir: "/photoprism" # do not change or remove
    ## Storage Folders: "~" is a shortcut for your home directory, "." for the current
directory
```

```
    volumes:
      # "/host/folder:/photoprism/folder"                # Example
      - "~/Pictures:/photoprism/originals"               # Original media files (DO NOT
REMOVE)
      # - "/example/family:/photoprism/originals/family" # *Additional* media folders can be
mounted like this
      # - "~/Import:/photoprism/import"                  # *Optional* base folder from which
files can be imported to originals
      - "./storage:/photoprism/storage"                  # *Writable* storage folder for
cache, database, and sidecar files (DO NOT REMOVE)

  ## Database Server (recommended)
  ## see https://docs.photoprism.app/getting-started/faq/#should-i-use-sqlite-mariadb-or-mysql
  mariadb:
    ## If MariaDB gets stuck in a restart loop, this points to a memory or filesystem issue:
    ## https://docs.photoprism.app/getting-started/troubleshooting/#fatal-server-errors
    restart: unless-stopped
    image: mariadb:10.10
    security_opt: # see https://github.com/MariaDB/mariadb-docker/issues/434#issuecomment-
1136151239
      - seccomp:unconfined
      - apparmor:unconfined
    command: mysqld --innodb-buffer-pool-size=512M --transaction-isolation=READ-COMMITTED --
character-set-server=utf8mb4 --collation-server=utf8mb4_unicode_ci --max-connections=512 --
innodb-rollback-on-timeout=OFF --innodb-lock-wait-timeout=120
    ## Never store database files on an unreliable device such as a USB flash drive, an SD
card, or a shared network folder:
    volumes:
      - "./database:/var/lib/mysql" # DO NOT REMOVE
    environment:
      MARIADB_AUTO_UPGRADE: "1"
      MARIADB_INITDB_SKIP_TZINFO: "1"
      MARIADB_DATABASE: "photoprism"
      MARIADB_USER: "photoprism"
      MARIADB_PASSWORD: "insecure"
      MARIADB_ROOT_PASSWORD: "insecure"
```

como en ocasiones anteriores, para guardar los cambios pulsaremos `control + x` y cuando nos pregunte aceptaremos. Una vez volvamos a estar en el terminal, escribiremos `docker compose up -d` para lanzar los servicios ubicados dentro del fichero docker-compose. Le va a costar un buen rato extraer las imágenes y empezar el despliegue, paciencia. El resultado será similar al siguiente:

photoprism-deploy.png.png

Elaboración propia

Se paciente, le cuesta un par de minutos arrancar. Si tras esa breve pausa accedemos al servicio como venimos haciendo, en este caso en el puerto 2342, veremos algo similar a:

photoprism-web.png

Elaboración propia

El usuario y contraseña por defecto son `admin` y `insecure`. Fíjate que vienen establecidos en el fichero docker-compose.

> Este servicio está al limite en cuanto a la capacidad de la Raspberry Pi 4 modelo B de 4 GB. Valora si la solución que hemos visto en el capítulo anterior es suficiente para ti.

3.11 PaperMerge. Gestión documental

papermerge-logo.jpg

Imagen obtenida de https://twitter.com/papermerge

Esta herramienta sirve para...

realizar la gestión documental a través de una interface web sencilla de utilizar. Cuenta con OCR por lo que podremos buscar textos dentro de estos documentos.

Web de proyecto y otros enlaces de interés

Página web: https://papermerge.com/

Repositorio de los proyectos que componen esta herramienta https://github.com/papermerge

Puesta en marcha

> La documentación del proyecto https://docs.papermerge.io/Installation/docker-compose.html recomienda no utilizar docker-compose para un sistema en producción.

Como en ocasiones anteriores vamos a hacer con docker-compose para ello accedemos al terminal y escribimos

```
cd $HOME
mkdir papermarge
cd papermarge
nano .env
```

y dentro del fichero copiaremos el siguiente contenido:

```
APP_IMAGE=papermerge/papermerge
APP_TAG=latest
PAPERMERGE_JS_IMAGE=papermerge/papermerge.js
PAPERMERGE_JS_TAG=latest

TIMEZONE=Europe/Madrid

DB_USER=postgres
DB_NAME=postgres
DB_PASSWORD=postgres
DB_HOST=db
DB_PORT=5432

USE_HOSTNAME=papermerge.local

REDIS_HOST=redis
REDIS_PORT=6379

SECRET_KEY=12345abcdxyz

SUPERUSER_USERNAME=admin
SUPERUSER_EMAIL=admin@example.com
SUPERUSER_PASSWORD=admin
```

como en ocasiones anteriores, para guardar los cambios pulsaremos `control + x` y cuando nos pregunte aceptaremos.

Hasta ahora nunca habíamos utilizado ningún fichero `.env` pero lo correcto es establecer determinadas configuraciones en los mismos. Los ficheros `.env` son ficheros que no se comparten en el respositorio de código mientras que los ficheros `docker-compose.yml` si. Lo que suele hacerse

es compartirse algún fichero como `env-sample` o de nombre similar de modo que cualquier persona pueda ver ahí los parámetros que debe configurar pero sin ver tus valores reales y le basta con renombrarlo a `.env`.

Ahora si. Vamos a crear el fichero `docker-compose.yml` para ello escribimos en el terminal

```
nano docker-compose.yml
```

y dentro del fichero copiamos el siguiente contenido:

```
version: '3.7'
# Any top-level key starting with x- in a Docker Compose file will be
# ignored
x-backend: &backend  # yaml anchor definition
  image: ${APP_IMAGE}:${APP_TAG}
  volumes:
    - media_root:/app/media
    - xapian_index:/app/xapian_index
  environment:
    # PAPERMERGE__<section>__<variable>
    - PAPERMERGE__MAIN__SECRET_KEY=${SECRET_KEY}
    - PAPERMERGE__DATABASE__TYPE=postgres
    - PAPERMERGE__DATABASE__USER=${DB_USER}
    - PAPERMERGE__DATABASE__NAME=${DB_NAME}
    - PAPERMERGE__DATABASE__PASSWORD=${DB_PASSWORD}
    - PAPERMERGE__DATABASE__HOST=${DB_HOST}
    - PAPERMERGE__REDIS__HOST=${REDIS_HOST}
    - PAPERMERGE__REDIS__PORT=${REDIS_PORT}
    - PAPERMERGE__MAIN__TIMEZONE=${TIMEZONE}
    # path where xapian index data is stored
    - PAPERMERGE__SEARCH__PATH=/app/xapian_index
    - DJANGO_SUPERUSER_USERNAME=${SUPERUSER_USERNAME}
    - DJANGO_SUPERUSER_EMAIL=${SUPERUSER_EMAIL}
    - DJANGO_SUPERUSER_PASSWORD=${SUPERUSER_PASSWORD}
    - DJANGO_SETTINGS_MODULE=config.settings
services:
  worker: # celery worker
```

```
    <<: *backend
    command: worker
  ws_server:  # websockets server / daphne
    <<: *backend
    command: ws_server
    labels:
      - "traefik.enable=true"
      - "traefik.http.routers.ws_server.rule=Host(`${USE_HOSTNAME}`) && PathPrefix(`/ws/`)"
  backend:  # rest api backend / uwsgi
    <<: *backend
    labels:
      - "traefik.enable=true"
      - "traefik.http.routers.backend.rule=Host(`${USE_HOSTNAME}`) && PathPrefix(`/api/`)"
  db:
    image: postgres:14.4
    volumes:
      - postgres_data:/var/lib/postgresql/data/
    environment:
      - POSTGRES_USER=${DB_USER}
      - POSTGRES_DB=${DB_NAME}
      - POSTGRES_PASSWORD=${DB_PASSWORD}
  redis:
    image: 'redis:6'
    ports:
      - '6379:6379'
    volumes:
      - redis_data:/data
  traefik:
    image: "traefik:v2.6"
    command:
      #- "--log.level=DEBUG"
      - "--api.insecure=true"
      - "--providers.docker=true"
      - "--providers.docker.exposedbydefault=false"
      - "--entrypoints.web.address=:80"
    ports:
```

```
      - "80:80"
      - "8080:8080"
    volumes:
      - "/var/run/docker.sock:/var/run/docker.sock:ro"
  frontend: # emberjs
    image: ${PAPERMERGE_JS_IMAGE}:${PAPERMERGE_JS_TAG}
    labels:
      - "traefik.enable=true"
      - "traefik.http.routers.traefik.rule=Host(`${USE_HOSTNAME}`) && PathPrefix(`/`)"
volumes:
  postgres_data:
  media_root:
  xapian_index:
  redis_data:
```

como en ocasiones anteriores, para guardar los cambios pulsaremos `control + x` y cuando nos pregunte aceptaremos.

Ahora en el equipo desde el que vayamos a acceder al servicio en el terminal escribiremos:

```
sudo nano /etc/hosts
```

Y en dicho fichero añade al final del mismo el texto

```
192.168.0.201       papermerge.local
# En lugar de 192.168.0.201 vosotros/a pondréis la IP de vuestra Raspberry Pi
```

como en ocasiones anteriores, para guardar los cambios pulsaremos `control + x` y cuando nos pregunte aceptaremos.

Una vez volvamos a estar en el terminal de la Raspberry Pi, escribiremos `docker compose -f docker-compose.yml --env-file .env up -d` para lanzar los servicios ubicados dentro del fichero docker-compose. Le va a costar un buen rato extraer las imágenes y empezar el despliegue, paciencia. El resultado será similar al siguiente:

papermerge-deploy.png

Elaboración propia

En esta ocasión, aprovechando que hemos modificado el fichero /etc/hosts vamos a acceder a este servicio a través de la dirección http://papermerge.local y veremos algo como:

papermerge-web.png

Elaboración propia

El usuario y contraseña por defecto son `admin` y `admin`. Fíjate que vienen establecidos en el fichero .env en los valores SUPERUSER_USERNAME y SUPERUSER_PASSWORD.

> Este servicio está al limite en cuanto a la capacidad de la Raspberry Pi 4 modelo B de 4 GB.

3.12 Pi alert. ¿Intrusos en tu red?

pialert-logo.jpg

Imagen obtenida de https://devpost.com/software/pialert

> Este proyecto lleva mas de 2 años sin ser actualizado

Esta herramienta sirve para...

detectar nuevos equipos conectados en nuestra red de modo que si aparece algún equipo nuevo del cuál no estamos al tanto detectaremos una conexión no permitida.

Web de proyecto y otros enlaces de interés

Repositorio de código original: https://github.com/pucherot/Pi.Alert

Imagen que utilizaremos: https://registry.hub.docker.com/r/jokobsk/pi.alert

Despliegue

Como en ocasiones anteriores vamos a hacer con docker-compose para ello accedemos al terminal y escribimos

```
cd $HOME
mkdir pialert
cd pialert
```

```
nano docker-compose.yml
```

y dentro del fichero escribiremos el siguiente contenido

```
version: "3"
services:
  pialert:
    image: jokobsk/pi.alert
    ports:
      - "20211:20211/tcp"
    environment:
      - TZ=Europe/Madrid
    restart: unless-stopped
    volumes:
      - ./pialert_db:/home/pi/pialert/db
      - ./config/pialert.conf:/home/pi/pialert/config/pialert.conf
```

como en ocasiones anteriores, para guardar los cambios pulsaremos control + x y cuando nos pregunte aceptaremos. Una vez volvamos a estar en el terminal, escribiremos `docker compose up -d` para lanzar los servicios ubicados dentro del fichero docker-compose. Veremos algo similar a:

pialert-deploy.png

Elaboración propia

Si accedéis en vuestro navegador a la IP de la raspberry y al puerto que hemos establecido (20211). En mi caso sería http://192.168.0.201:20211 deberíais ver algo como:

pialert-web.png

Elaboración propia

Funcionamiento

Nada mas acceder accederás a una pantalla como la anterior. En la mismas verás 4 partes claramente delimitadas:

- un menú lateral a la izquierda
- una parte central dividida en 3 franjas horizontales:

- botonera de dispositivos
- gráfica con las horas y dispositivos encontrados (conectados y no conectados)
- listado de elementos encontrados (nombre, tipo, IP, MAC, estado,...)

Del menú lateral nos interesará acceder a la sección de Configuración (Settings).

pialert-settings.png

Elaboración propia

En este menú debes prestar especial atención a la opción que dice `Subnets to scan`, es decir, subredes a escanear. En mi caso, la red de mi casa es la 192.168.0.0/24 ¡ojo, lo habitual suele ser 192.168.**1**.0/24! y la interface de red que usa mi raspberry es eth0 de ahí mi configuración. En otras opciones que aparecen mas abajo podéis cambiar el idioma o configurar que os lleguen alertas de intrusión por email, MQTT (lo veremos en el apartado 4 de domótica). Webhooks u otras opciones.

Si pincháis en el nombre de un dispositivo aparecerá una pantalla como la siguiente

pialert-device.png

Elaboración propia

En la cual podréis establecer algunas características del dispositivo como su nombre. Esto os ayudará en el futuro, en las estadísticas, a identificar el dispositivo.

3.13 Change detection. Monitoriza cambios en una web

changedetection-logo.png

Imagen obtenida de https://changedetection.io/

Esta herramienta sirve para...

que se nos notifique cuando una determinada web (o parte de una web) cambie. Es especialmente útil cuando queremos, por ejemplo, enterarnos de una nueva noticia que aparezca en la web del colegio de nuestros hijos/as o cuando estamos esperando una calificación de un proceso selectivo y no queremos estar accediendo continuamente a la página web a comprobarlo. Este servicio se encarga de monitorizar por nosotros/as la web que le indiquemos y avisarnos si hay cambios.

Web de proyecto y otros enlaces de interés

Web del proyecto: https://changedetection.io/

Repositorio del proyecto: https://github.com/dgtlmoon/changedetection.io

Despliegue

Como en ocasiones anteriores vamos a hacer con docker-compose para ello accedemos al terminal y escribimos

```
cd $HOME
mkdir change-detection
cd change-detection
```

```
nano docker-compose.yml
```

y dentro del fichero escribiremos el siguiente contenido

```
version: '3.2'
services:
    changedetection:
      image: ghcr.io/dgtlmoon/changedetection.io
      container_name: changedetection
      hostname: changedetection
      volumes:
        - changedetection-data:/datastore
      ports:
        - 5000:5000
      restart: unless-stopped
volumes:
  changedetection-data:
```

como en ocasiones anteriores, para guardar los cambios pulsaremos control + x y cuando nos pregunte aceptaremos. Una vez volvamos a estar en el terminal, escribiremos `docker compose up -d` para lanzar los servicios ubicados dentro del fichero docker-compose. Veremos algo similar a:

changedetection-deploy.png

Elaboración propia

Si accedéis en vuestro navegador a la IP de la raspberry y al puerto que hemos establecido (5000). En mi caso sería http://192.168.0.201:5000 deberíais ver algo como:

changedetection-dashboard.png

Elaboración propia

Funcionamiento

El funcionamiento es tremendamente sencillo. Por ejemplo, para borrar las 2 páginas webs que vienen monitorizadas por defecto únicamente deberemos marcar los checkboz que aparecen a su izquierda y marcar `delete`.

Si queremos añadir una nueva página a monitorizar es suficiente con introducir una url en el recuadro superior y pulsar `Watch`. Si introducimos una url y pulsamos en `Edit > Watch` entonces tendremos acceso a una pantalla como la siguiente

changedetection-create.png

Elaboración propia

en la cual podremos establecer la frecuencia de comprobaciones o las notificaciones a recibir de esta web en concreto.

Si lo preferimos también podemos acceder a la configuración global en el menú superior en la opción de Settings que nos dará acceso a una configuración general de nuestro servicio como vemos a continuación

changedetection-settings.png

Elaboración propia

> Si estableces un chequeo demasiado constante de una página web puede que la web en cuestión te bloquee el acceso a la misma al considerar que estás atacando la web. Sufrirías un *baneo*.

3.14 Paperless-ngx. Gestión documental

paperless-logo.png

Imagen obtenida de https://docs.paperless-ngx.com/

Esta herramienta sirve para...

la gestión documental. Al igual que vimos con PaperMerge, cuenta con OCR.

Web de proyecto y otros enlaces de interés

Web del proyecto: https://docs.paperless-ngx.com/

Repositorio en github: https://github.com/paperless-ngx/paperless-ngx

Despliegue

En esta ocasión vamos a variar ligeramente la forma de realizar el despliegue. Dentro del respositorio, la empresa desarrolladora ha creado un script que automatiza la instalación vía docker-compose y, dado que queremos simplicidad, haremos uso del mismo

> Cuidado con ejecutar cualquier script que encontréis en internet pues podría ser malicioso.

Vamos allá, accedemos al terminal y escribimos:

```
cd $HOME
mkdir paperless
cd paperless
```

y ahora si, lanzamos el script del siguiente modo

```
bash -c "$(curl -L https://raw.githubusercontent.com/paperless-ngx/paperless-ngx/main/install-paperless-ngx.sh)"
```

aparecerá un asistente que nos irá realizando una serie de preguntas para configurar el servicio y que tendrá un aspecto similar a este (remarco en un cuadrado rojo lo que he ido respondiendo):

paperless-deploy1.png

paperless-deploy2.png

paperless-deploy3.png

paperless-deploy4.png

Elaboración propia

Una vez terminemos (le costará un buen rato), como en ocasiones anteriores, accederemos al servicio a través del navegador escribiendo http://nuestra_ip_local:puerto En mi caso es http://192.168.0.201:8000

paperless-login.png

Elaboración propia

Si tienes problemas para acceder vuelve al terminal y escribe `docker-compose run --rm webserver createsuperuser` te pedirá que introduzcas un nuevo usuario y contraseña. A continuación tira el servicio con `docker-compose down` y cuando termine vuelve a levantarlo con `docker-compose up -d` . Con esto deberías poder acceder sin problema.

Funcionamiento

La primera vez que accedas verás un aspecto similar al siguiente:

paperless-dashboard.png

Elaboración propia

La herramienta tiene muchas posibilidades pero quizás lo mas interesante sea acceder a la sección documentos y allí arrastrar aquellos documentos que nos interese conservar (nóminas, facturas de la luz/gas/teléfono,...) Muchos de estos documentos están accesibles mientras somos clientes de una compañía pero dejan de estar a nuestro alcance cuando dejamos de ser clientes y, en ocasiones, nos pueden resultar de utilidad. En la imagen posterior he arrastrado una nómina

paperless-documents.png

Elaboración propia

La potencia de la herramienta reside en que gracias a OCR y el sistema de etiquetas que incluye podemos ir almacenando documentos de modo que en el futuro resulte muy fácil encontrar la documentación que nos interese. Así, si en el ejemplo anterior, pulsamos en el lápiz que aparece justo del documento subido a modo de ejemplo podremos configurar una serie de parámetros:

paperless-detail.png

Elaboración propia

Una vez clasificados los documentos podemos utilizar el buscador superior o el ubicado en documentos par localizar aquella información que nos interese.

paperless-search.png

Elaboración propia

> Este servicio es también bastante exigente para el hardware que incorpora una Raspberry Pi modelo 4. De todos modos, recuerda que lo que estamos haciendo con docker y docker-compose sobre la Raspberry Pi puedes replicarlo en cualquier otro sistema operativo que permita su ejecución y con un hardware mas potente.

3.15 Jellyfin. Gestión de medios

jellyfin-logo.png

Imagen obtenida de https://jellyfin.org/

Esta herramienta sirve para...

gestionar nuestros ficheros de audio, vídeo, imagen, ebooks,... de un modo muy sencillo a través de una interface web.

Web de proyecto y otros enlaces de interés

Web oficial https://jellyfin.org/

Imagen del proyecto en docker hub: https://hub.docker.com/r/jellyfin/jellyfin/

Despliegue

Como en ocasiones anteriores vamos a hacer con docker-compose para ello accedemos al terminal y escribimos

```
cd $HOME
mkdir jellyfin
cd jellyfin
nano docker-compose.yml
```

y dentro del fichero escribiremos el siguiente contenido

```
version: '3.3'
services:
    jellyfin:
        volumes:
            - './config:/config'
            - './cache:/cache'
            - '/direccion/donde/tengais/vuestros/media:/media'
        network_mode: host
        image: 'jellyfin/jellyfin:latest'
        ports:
          - "8089:80"
```

como en ocasiones anteriores, para guardar los cambios pulsaremos control + x y cuando nos pregunte aceptaremos. Una vez volvamos a estar en el terminal, escribiremos `docker compose up -d` para lanzar los servicios ubicados dentro del fichero docker-compose. Veremos algo similar a:

jellyfin-deploy.png

Elaboración propia

Si accedéis en vuestro navegador a la IP de la raspberry y al puerto que hemos establecido (8089). En mi caso sería http://192.168.0.201:8096 deberíais ver algo como:

jellyfin-home.png

Elaboración propia

Ahora es cuestión de ir siguiendo las instrucciones para crear un nuevo usuario y añadir bibliotecas de medios (carpetas dónde están los documentos de tipo media, es decir, vídeos, audios e imágenes). Una vez terminemos aparecerá una pantalla de acceso para introducir un usuario y contraseña.

jellyfin-login.png

Elaboración propia

Funcionamiento

Si durante la instalación hemos establecido alguna biblioteca de medios al acceder nos aparecerá en la pantalla principal (le costará un rato que aparezca todo el contenido):

jellyfin-fome.png

Elaboración propia

De la pantalla anterior quizás lo mas relevante es la posibilidad de enviar contenido a otros dispositivos como un chromecast, fire tV o algunos modelos de televisores o pantallas táctiles al igual que podemos hacer desde las apps de Netflix, Disney+, HBO,... El botón en cuestión es

jellyfin-cast.png

Elaboración propia

Desde ajustes > Panel de control > Bibliotecas puedes añadir nuevas bibliotecas de medios. Voy a enseñaros lo que podría ser la configuración de una de ellas:

jellyfin-add-media.png

jellyfin-add-media2.png

jellyfin-add-media3.png

Elaboración propia

Tras establecer la nueva biblioteca de medios y regresar al panel de control veremos que ahí empieza a aparecer el contenido relacionado con esta bilioteca de medios:

jellyfin-dashboard.png

Elaboración propia

> En mi caso no utilizo un sistema como este pues tengo la raspberry pi conectada a la TV a través de HDMI. También tengo un teclado y un ratón inalámbrico conectados. Cuando quiero ver algo utilizo la aplicación kodi en lugar de este solución.

Y, cuando estoy fuera de casa, conecto la VPN (wireguard) y con kodi desde mi teléfono móvil visualizo el contenido a través de kodi pero esta es también una buena solución.

3.16 Adguard. Navega por Internet sin anuncios y con seguridad

adguard-logo.png

Imagen obtenida de https://adguard.com/es/welcome.html

Esta herramienta sirve para...

como dicen en su web:

> “
> Bloquea todo tipo de anuncios
> Elimina elementos web molestos
> Ahorra tráfico y acelera carga de páginas
> Funciona para navegadores y aplicaciones
> Mantiene la funcionalidad y la apariencia de sitios

Web de proyecto y otros enlaces de interés

Web del proyecto https://adguard.com/es/welcome.html

Repositorio de código: https://github.com/AdguardTeam/AdGuardHome

Imagen docker oficial: https://hub.docker.com/r/adguard/adguardhome

Despliegue

En esta ocasión he tenido muchas dudas sobre como recomendaros el despliegue del servicio pues es posible la instalación de un modo muy sencillo a través de cualquiera de estas 3 opciones

- vía **script**: simplemente ejecutando: `curl -s -S -L https://raw.githubusercontent.com/AdguardTeam/AdGuardHome/master/scripts/install.sh | sh -s -- -v`
- vía **snap** (sistemas linux): en la snap store desde este enlace https://snapcraft.io/adguard-home
- vía **docker**

Por mantener cierta coherencia con lo trabajado en el curso voy a optar por seguir trabajando con docker y docker-compose pero lo cierto es que fuera del curso había sido mi 3ª elección debido a la simplicidad de los otros 2 sistemas. Vamos allá:

Accedemos a la terminal y escribimos

```
cd $HOME
mkdir adguard
cd adguard
nano docker-compose.yml
```

Dentro del fichero escribimos el siguiente contenido

```
version: '3.3'
services:
    adguardhome:
        image: adguard/adguardhome
        container_name: adguardhome
        restart: unless-stopped
        volumes:
            - './workdir:/opt/adguardhome/work'
            - './confdir:/opt/adguardhome/conf'
        ports:
            - '53:53/tcp'
            - '53:53/udp'
            - '67:67/udp'
```

```
      - '80:80/tcp'
      - '443:443/tcp'
      - '443:443/udp'
      - '3000:3000/tcp'
      - '853:853/tcp'
      - '784:784/udp'
      - '853:853/udp'
      - '8853:8853/udp'
      - '5443:5443/tcp'
      - '5443:5443/udp'
```

Para salir del fichero pulsaremos `control + x` y guardaremos los cambios.

> Si algún otro servicio está utilizando los puertos que en este servicio vamos a utilizar se generará un conflicto y puede que ninguno de los servicios funcione o, mas probable, el último que pongamos en marcha.

> Para usar un puerto diferente puedes cambiar el valor que aparece ANTES de los : por un valor que no esté en uso. También puedes acceder al directorio dónde se encuentra el otro servicio y ejecutar `docker-compose down`.

Posteriormente ponemos en marcha los contenedores con `docker compose up -d`. Si accedemos, como en ocasiones anteriores, a http://IP:PUERTO siendo en mi caso http://192.168.0.201:3000 veremos algo como:

adguard-deployed.png

Elaboración propia

Funcionamiento

Una vez hemos accedido a la pantalla anterior es momento de configurar el servicio a través de la interface gráfica. A través de 5 pasos estableceremos los valores de configuración. No es necesario cambiar nada de la configuración. Si que deberás establecer un usuario y contraseña de acceso. En el paso 4 te dirá como debes configurar tus dispotivios para que adguard pueda hacer su tarea (si ya los tenías configurados con pi-hole no deberás hacer nada adicional). Una vez configurado la interface gráfica te pedirá el usuario y contraseña que has establecido y accederás a un panel de

control como el siguiente:

adguard-login.png

Elaboración propia

3.17 etc.

Como hemos visto con anterioridad, nuestra Raspberry Pi es un ordenador a todos los efectos y en el mismo tenemos un sistema operativo completo con lo que podemos llevar a cabo cualquier tarea que pueda automatizarse.

Por ejemplo, en mi caso, tengo una raspberry pi 2 model B con raspbian que cada noche a una determinada hora se conecta al servidor web dónde alojo el recurso didáctico trivinet.com y copia a la SD de esta raspberry Pi la última copia de seguridad que he hecho en ese servidor web. De este modo tengo la copia de seguridad en 2 ubicaciones distintas por si ocurriese algún desastre.

¿cómo programar tareas para que se ejecuten en fechas/concretas?

En sistemas operativos linux tenemos cron mientras que en sistemas operativos windows existe una herramienta llamada tareas programadas. En este curso nos centraremos en cómo hacerlo en Rasberry Pi os (anteriormente llamado Rasbian).

Para ver el contenido del fichero cron ejecutaremos el comando `crontab -l` mientras que para editar su contenido ejecutaremos `crontab -e` Pongo a continuación una captura de pantalla del resultado de ejecutar el primer comando visto en la máquina antes mencionada

crontab_l.png

En la imagen anterior podemos ver varias líneas que comienzan por # estas líneas son comentarios, es decir, cron las ignora. Si nos fijamos en la única línea que tenemos que no comienza por # tenemos lo siguiente

```
0 5 * * * /home/pi/scripts/copiaSeguridad.sh
```

¿y esto qué significa Pablo? pues que siempre que la fecha y hora sea:

- 0 minutos
- 5 horas
- de cualquier día (numérico)
- de cualquier mes
- de cualquier día de la semana (lunes a domingo)

se ejecutará el comando `/home/pi/scripts/copiaSeguridad.sh` Es decir, yo ahí estoy automatizando el que este comando se ejecute siempre a las 5 de la mañana.

En el siguiente enlace https://geekflare.com/es/crontab-linux-with-real-time-examples-and-tools/ podemos ver mas información sobre el uso de cron

Una vez que sabemos como programar tareas "únicamente" quedaría la creación del script cuaestión esta que requeriría de un curso por si solo.

> Te animo a que si tienes un script que crees que puede resultar útil a cualquier compañero/a nos lo facilites a fin de recopilarlo en esta sección. Si por el contrario tienes la necesidad de un script para una tarea concreta háznoslo saber y trataremos de darle solución y publicarlo aquí