

3.2 DIUN. Notificador de nuevas imágenes

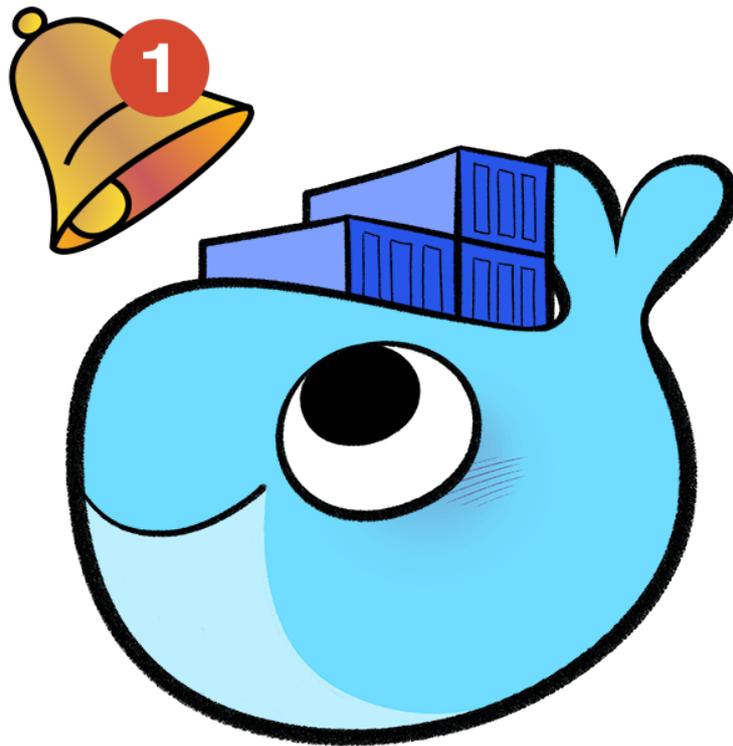


Imagen obtenida de <https://crazymax.dev/diun/>

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

```
pablo@raspberrypicatedu:~/portainer $ cd $HOME
mkdir diun
cd diun
nano docker-compose.yml
pablo@raspberrypicatedu:~/diun $ docker compose up -d
[+] Running 4/4
 ✓ diun 3 layers [██████] 0B/0B Pulled
 ✓ c41833b44d91 Pull complete
 ✓ e6d170451ea7 Pull complete
 ✓ dd0eb591c47f Pull complete
[+] Running 2/2
 ✓ Network diun_default Created 0.1s
 ✓ Container diun Started 4.8s
pablo@raspberrypicatedu:~/diun $
```

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

```
pi@mediacenter:~/diun $ docker ps | grep diun
d8813643241c crazymax/diun:latest "diun serve" 2 minutes ago Up About a minute
diun
```

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

```
pi@mediacenter:~$ docker exec diun diun image list
+-----+-----+-----+-----+-----+
| NAME | MANIFESTS COUNT | LATEST TAG | LATEST CREATED | LATEST DIGEST |
+-----+-----+-----+-----+-----+
| docker.io/crazymax/diun | 1 | latest | 2022-12-29T11:25:29Z | sha256:fa80af32a7c61128ffda667344547805b3c5e7721ecbbafd70e35bb7bb7c989f |
+-----+-----+-----+-----+-----+
| TOTAL | 1 | | | |
+-----+-----+-----+-----+-----+
```

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

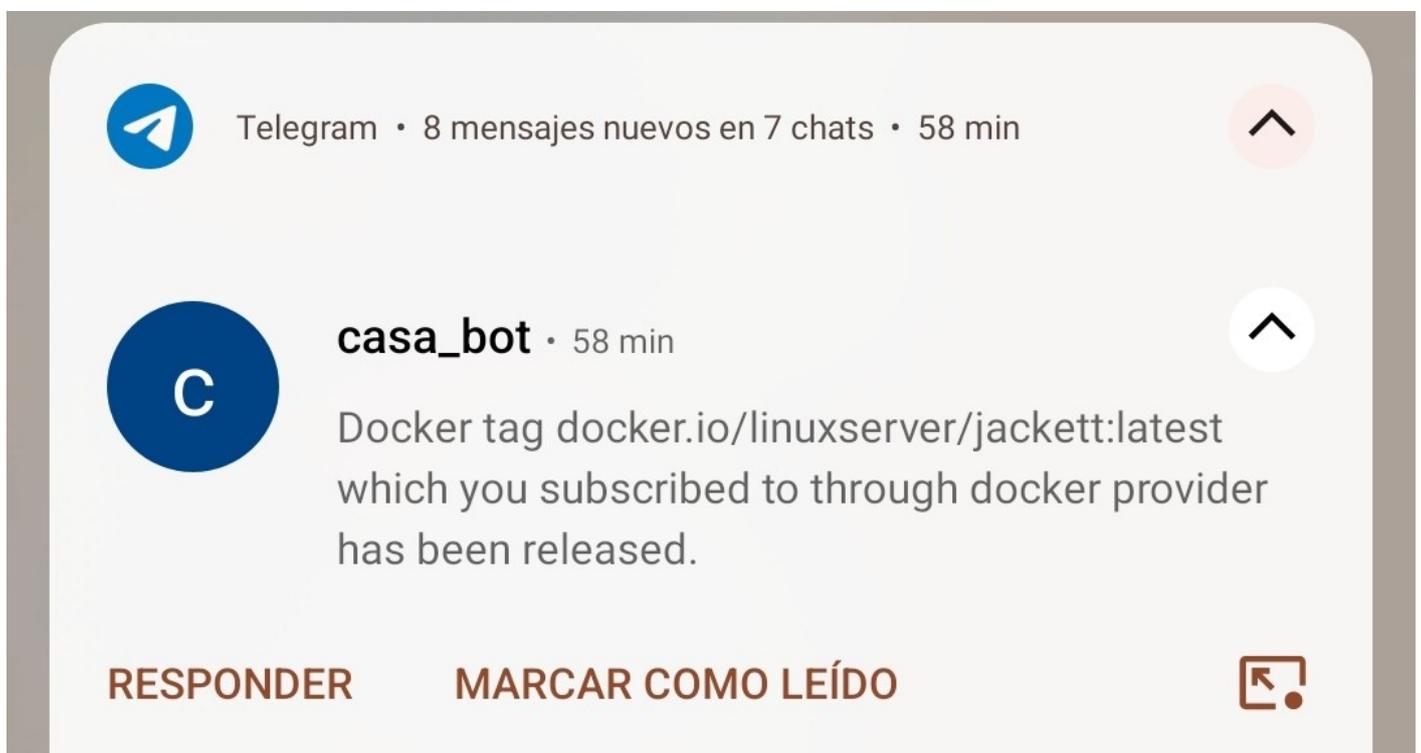
```
cd $HOME/diun
docker-compose down
```

```
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.

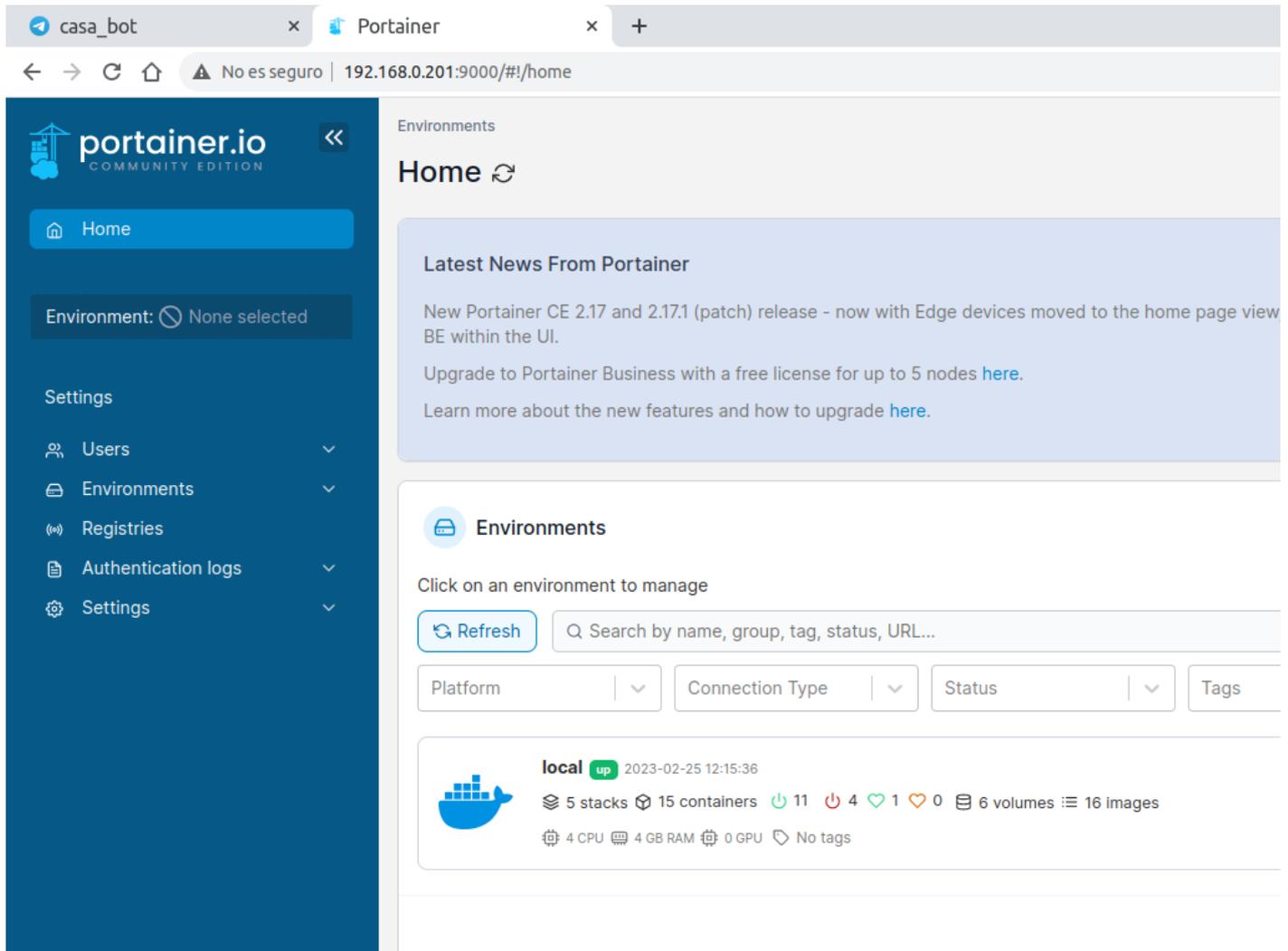
Cómo actualizar la imágenes

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



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.



Elaboración propia

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

The screenshot shows the Portainer.io Dashboard for a local environment. The left sidebar contains navigation options like Home, local, Dashboard, App Templates, Stacks, Containers, Images, Networks, Volumes, Events, Host, Settings, Users, Environments, Registries, Authentication logs, and Settings. The main area displays 'Environment summary' with a 'Dashboard' header. Below this, there's an 'Environment info' section with details: Environment (local, 4 GB, Standalone 20.10.5+dfsg1), URL (/var/run/docker.sock), GPU (none), and Tags (-). A summary grid shows: 5 Stacks, 15 Containers (10 running, 0 stopped, 1 healthy, 0 unhealthy), 16 Images (4.9 GB), 6 Volumes, 9 Networks, and 0 GPUs.

Elaboración propia

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

The screenshot shows the 'Image details' page in Portainer.io. The breadcrumb is 'Images > sha256:05c78963fdac9e8b48688cef021b9a3830452375becb7b93bd81ad03568dc129'. The page title is 'Image details'. Under 'Image tags', there is a tag 'linuxserver/jackett:latest' with upload, download, and delete icons. A note explains the icons. The 'Tag the image' section has a 'Registry' dropdown set to 'DockerHub (anonymous)', an 'Image*' input field with 'docker.io e.g. my-image:my-tag', and a 'Search' button. A warning 'Image name is required.' is shown. An 'Advanced mode' section contains a note about DockerHub pull limits (100 pulls every 6 hours) and a 'Tag' button. At the bottom, the 'Image details' section shows the ID 'sha256:05c78963fdac9e8b48688cef021b9a3830452375becb7b93bd81ad03568dc129' and buttons for 'Delete this image' and 'Export this image'.

Elaboración propia

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

Image details

Image tags

linuxserver/jackett:latest



Note: you can click on the upload icon  to push an image or on the download icon  to pull an image or on the trash icon  to delete a tag.

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

Revision #14

Created 4 February 2023 09:49:08 by Pablo Ruiz

Updated 20 July 2023 17:13:16 by Pablo Ruiz