

Monitorización con *SNMP*

Nicolás Aguado*

Diseño y Proyectos de Redes — 23 de diciembre de 2024

1. Introducción

Se propone una adaptación del sistema propuesto en la práctica *Aplicaciones escalables y orquestación de servicios* dentro del marco de la asignatura de *Diseño de Proyectos y Redes*.

Se expande el sistema para contener a todo lo relacionado con monitorizar el sistema replicado *API REST*.

2. Descripción General

En la Figura 1. se expone el esquema general (actualizado) del sistema orquestado.

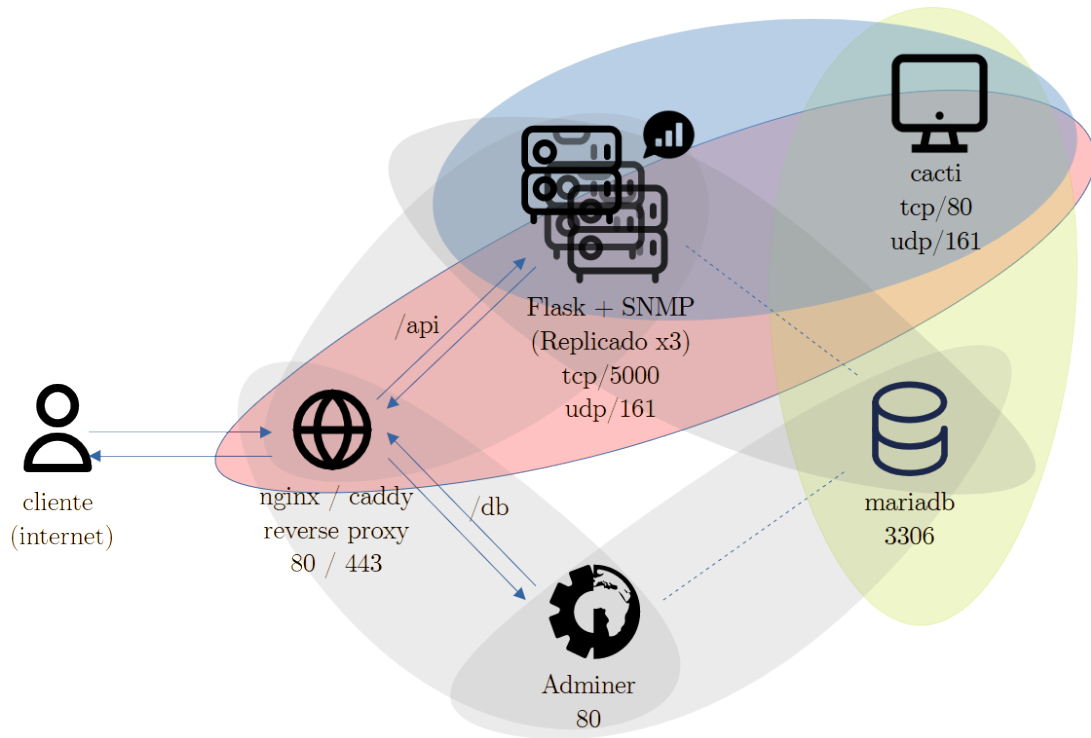


Figura 1: Diagrama del sistema coloreado para mostrar las partes modificadas

El sistema ha sido modificado para instalar un agente que implementa el protocolo **SNMP** [7] en cada réplica del servicio *API REST*. Además, se añade una nuevo contenedor al sistema con

* nico@nico.eus - Facultad de Informática - UPV/EHU

la herramienta **cacti** [6] para la extracción de los datos de cada réplica y para la visualización en un entorno gráfico web de las métricas extraídas.

Se han adaptado los servicios de la base de datos y los servicios del *reverse proxy* para hacer frente a la nueva configuración.

3. Cambios Realizados

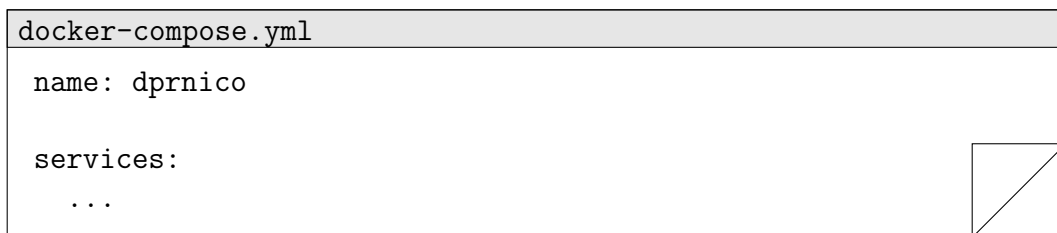
A continuación se expone una descripción más detallada de las mejoras realizadas y de su orquestación dentro del sistema. También se argumentan las decisiones de diseño y se describen los aspectos que se consideran más importantes.

3.1. Configuración Inicial de Red

Por defecto, el sistema de conexiones de **Docker** proporciona un servicio DNS para que los contenedores desplegados se puedan comunicar entre sí. El esquema de nombres para los **contenedores** sigue la estructura de *dir_proyecto-nom_servicio-num_instancia*.

En cambio, de cara a poder facilitar la orquestación y las automatizaciones, resulta conveniente el poder disponer de nombres DNS que no dependan del directorio en el que está contenido el proyecto.

Entonces, al hacer un cambio como el siguiente en la cabecera del fichero *compose*,



```
docker-compose.yml

name: dprnico

services:
  ...
```

se puede lograr una transformación en el esquema de nombres, obteniendo en su lugar la estructura *nombre_canonico-nom_servicio-num_instancia*. Entonces, por ejemplo, en un proyecto con nombre canónico **dprnico**, la tercera réplica del servicio **dprnico-flask** recibirá un nombre DNS *dprnico-dprnico-flask-3*.

Cabe denotar que este cambio no afecta a los nombres DNS de los **servicios** desplegados. Esto es, si hay un servicio replicado de nombre **dprnico-flask**, al acceder al nombre DNS *dprnico-flask* el propio **Docker** sigue haciendo de *load-balancer* y reenvía la petición a una de la réplicas.

3.2. Cacti

La herramienta **cacti** es una plataforma de monitorización que se encarga de la recolección y la representación gráfica de métricas. El objetivo es integrarla mediante el protocolo **SNMP** con los agentes instalados en cada réplica (Sección 3.3) para la monitorización del servicio.

Desde el grupo de desarrolladores de **cacti** no se provee una imagen oficial para el uso de la aplicación dentro de un contenedor **Docker**. Entonces, de cara a no *reinventar la rueda*, se valoran las alternativas propuestas por la comunidad.

En un principio, se valora usar la imagen alojada en el repositorio *Docker Hub* con tag `admindean/cacti` [2], ya que incluye todas las dependencias necesarias dentro de la propia imagen y es una aparente solución inmediata. Desafortunadamente, esta imagen presenta problemas de compatibilidad con ciertos sistemas de ficheros y en ciertas versiones de sistemas operativos [3], así que dada su inestabilidad no se considera adecuada para su uso en un sistema en producción.

Como alternativa, se considera el uso de la imagen `smcline06/cacti` [5]. Este contenedor incluye todos los servicios necesarios para `cacti` excepto el relativo al almacenamiento de las métricas en una base de datos `mysql`. Por suerte, el sistema actual ya contiene una base de datos compatible con `mysql` (`mariadb`) y se puede utilizar este almacén para contener además de los datos actuales también los datos de `cacti`. Por el resto de aspectos relacionados y su extensa documentación y soporte, esta imagen se considera adecuada y es la elegida para su uso dentro del sistema.

En general, la orquestación de este tipo de imágenes relacionadas con `cacti` requiere de (a parte de la configuración `Compose` pertinente habitual) una inicialización con una interfaz web en la que se configuran todos los parámetros relativos a la base de datos seleccionada, los sistemas que se desean monitorizar, los gráficos que se desean ver, etc.

Gracias a que los nombres DNS de los contenedores son siempre los mismos (Sección 3.1), se puede proveer una instalación desatendida realizando unos pequeños ajustes para cargar previamente la configuración necesaria en la base de datos (dispositivos a monitorizar, gráficos, usuarios). Estos ajustes se encuentran descritos en la Sección 3.4.2 y en el Anexo A.

Además, una vez se tienen las configuraciones ya cargadas, para omitir la instalación web inicial es suficiente con utilizar el siguiente *Dockerfile* a la hora de realizar el despliegue.

```
cacti-init/Dockerfile-cacti
```

```
FROM smcline06/cacti:latest
```

```
# Crear archivo install.lock para saltar la instalación
```

```
RUN touch /cacti/install.lock
```

Finalmente, con la argumentación provista y las configuraciones habituales en aislamiento de redes y configuraciones de volúmenes, la parte relativa a la orquestación de `cacti` en el fichero *docker-compose* queda de la siguiente manera.

`docker-compose.yml`

```
dprnico-cacti:
  build:
    dockerfile: cacti-init/Dockerfile-cacti
  restart: unless-stopped
  privileged: true
  depends_on:
    - dprnico-flask
    - dprnico-mysql
  environment:
    - DB_HOST=dprnico-mysql
    - DB_NAME=cactidb
    - DB_PASS=P4ssw0rd
    - DB_USER=cactiwriter
    - TZ=Europe/Madrid
    - PHP_MEMORY_LIMIT=256M
  volumes:
    - ./cacti-files/cacti-data:/cacti
    - ./cacti-files/cacti-spine:/spine
    - ./cacti-files/cacti-backups:/backups
  networks:
    - monitor-cacti
    - cacti-nginx
    - cacti-db
```

De esta manera, es suficiente con acceder a la interfaz web en el apartado *Graphs* en modo *Preview* (menú superior, primero a la izquierda y luego a la derecha) para visualizar los gráficos preconfigurados relativos a la monitorización de las tres réplicas del servicio *API REST*. (Dirección *HTTP* `/cacti/graph_view.php?action=preview`)

3.3. Agente SNMP en Servicio API

Para poder conseguir una visión completa, es necesario instalar un agente en cada contenedor que se desee monitorizar. De esta manera, se facilita la recolección de métricas por parte del servicio monitorizador (en este caso, *Cacti*).

De los distintos protocolos que soporta *Cacti* para la recolección de métricas, **SNMP** [7] es un protocolo suficientemente ligero y suficientemente completo para las necesidades de vigilancia del sistema. En cuanto al agente, dado que los contenedores destino son sistemas *UNIX-like*, se considera adecuado el uso del *daemon net-snmp* [1] (también conocido como *snmpd*).

En cuanto a la configuración de *net-snmp*, se provee una configuración mínima suficiente para el objeto de la monitorización.

`snmpd.conf`

```
agentAddress udp:161
rocommunity public 0.0.0.0/0
sysLocation "Flask Application"
sysContact "dprnico@nico.eus"
```

En cuanto a la orquestación del sistema, se ha de tener en cuenta que en los contenedores relativos a la *API REST* ahora se tienen que ejecutar dos procesos a la vez (**snmpd** y el servicio API en sí, **gunicorn**). Para lograr este propósito, se modifica ligeramente el fichero *Dockerfile-flask* para la construcción de la imagen y se provee un nuevo script *entrypoint.sh* para servir de comando de entrada para la ejecución de ambos procesos.

Dockerfile-flask

```
...
COPY ./flask-files/snmpd.conf /etc/snmp/snmpd.conf
COPY ./flask-files/entrypoint.sh /entrypoint.sh
RUN chmod +x /entrypoint.sh
...
EXPOSE 161/udp
...
ENTRYPOINT ["/entrypoint.sh"]
```

entrypoint.sh

```
#!/bin/sh

# Ejecutar SNMPd en segundo plano (&)
snmpd -C -c /etc/snmp/snmpd.conf &

# Ejecutar gunicorn para FastAPI en primer plano
exec gunicorn -b 0.0.0.0:5000 app:app
```

3.4. Expansión de Base de Datos

De cara a proveer de almacenamiento persistente a la aplicación **cacti** para las métricas y las configuraciones, se han realizado dos modificaciones en el sistema **mariadb**.

3.4.1. Creación de Espacio

Es una buena práctica el mantener aislados por distintos usuarios y distintas bases de datos a los procesos que operan en el almacenamiento persistente. Se provee un script para inicializar este nuevo espacio de la base de datos, y se muestra a continuación un extracto de éste.

db-init/startcactidb1.sh

```
# Crear db con nombre pasado por variable de entorno
CREATE DATABASE IF NOT EXISTS \'$CACTI_DATABASE\'

# Crear usuario con user y pass por variable de entorno
CREATE USER IF NOT EXISTS \'$CACTI_USER\'@\'%\' IDENTIFIED
    BY \'$CACTI_PASSWORD\' ;
GRANT ALL PRIVILEGES ON \'$CACTI_DATABASE\'.*
    TO \'$CACTI_USER\'@\'%\' ;
GRANT SELECT ON mysql.time_zone_name TO \'$CACTI_USER\'@\'%\' ;
FLUSH PRIVILEGES;
```

3.4.2. Incorporación de Datos

Se considera una buena práctica proveer un sistema plenamente configurado para su despliegue en producción. En cuanto a la aplicación **cacti**, hay una serie de configuraciones a mencionar para lograr este propósito.

De cara a evitar la configuración inicial (instalador web), la configuración de los dispositivos a monitorizar y la creación de los gráficos correspondientes, se ha exportado la base de datos una vez se han realizado los pasos de configuración inicial (descritos en el Anexo A) y se ha generado un archivo (mediante la herramienta **adminer**) con el objetivo de insertar directamente todas estas opciones ya configuradas en la base de datos la primera vez que se ejecuta el sistema.

El fichero correspondiente a esta exportación se encuentra en `db-init/startcactidb2.sql`.

Ambos ficheros inicializadores, tanto el de tipo *bash* (adaptado para que soporte variables de entorno) `db-init/startcactidb1.sh` como el fichero de tipo *sql* `db-init/startcactidb2.sql` se encuentran montados dentro de la carpeta `initdb.d` de `mariadb` para su ejecución en la primera creación de la base de datos. Véase un extracto del *docker-compose.yml*

```
docker-compose.yml
...
dprnico-mysql:
  image: mariadb:latest
  volumes:
    - ./dbdata:/var/lib/mysql
    # inicializar bd con tablas iniciales
    - ./db-init/startcactidb1.sh:
        /docker-entrypoint-initdb.d/startcactidb1.sh
    - ./db-init/startcactidb2.sql:
        /docker-entrypoint-initdb.d/startcactidb2.sql
  ...
```

3.5. Modificación *Reverse Proxy*

De cara a poder acceder de manera sencilla a la plataforma de monitorización **cacti**, se añade la entrada correspondiente al fichero de configuración de **caddy** (`/caddy-init/Caddyfile`) y al fichero de configuración de **nginx** (`/nginx-files/nginx.conf`).

Por mostrar uno de ellos, la configuración añadida para el *reverse proxy* **nginx** es la siguiente.

```
nginx-files/nginx.conf

location /cacti {
  proxy_pass http://dprnico-cacti:80;
  proxy_set_header Host $host;
  proxy_set_header X-Real-IP $remote_addr;
  proxy_set_header X-Forwarded-For $proxy_add_x_forwarded_for;
}
```

Esta entrada añadida es conceptualmente idéntica al resto de configuraciones del sistema, tanto en el fichero para el servicio **nginx** como en el fichero para el servicio **caddy**.

4. Desplegado en Local

Para desplegar todos los servicios descritos en una máquina local es suficiente con extraer los contenidos del fichero *zip* adjunto y ejecutar el siguiente comando en una terminal *UNIX-like*

```
bash
$ sudo docker compose up -d
```

Por defecto, el *reverse-proxy* está escuchando en todas las interfaces de red en el puerto 80, de modo que es suficiente con acceder a <http://localhost/cacti/> para interactuar con el servicio de monitorización **cacti**. Es recomendable revisar la sección de *Posibles Problemas y Soluciones* en caso de experimentar algún tipo de fallo (5.2)

El usuario importado mediante el volcado de datos iniciales para el servicio **cacti** para el panel de administración es **admin** y su contraseña es **1Sobresaliente!**

5. Extra

A parte de todas las consideraciones y todos los detalles que se han mencionado anteriormente, se desea exponer una serie de detalles presentes en las mejoras propuestas al sistema.

5.1. Desplegado en la Nube

De cara al despliegue en un entorno real del sistema, se ha optado por integrarlo dentro de una infraestructura ya existente. Además, ya que en la infraestructura actual se encuentra la iteración previa del sistema, se ha optado por sustituirla e integrar la nueva versión en su lugar.

Como se menciona en iteraciones pasadas, la máquina utilizada está alquilada a *Netcup Gmbh* [4] en Núremberg, Alemania, con 4 CPUs y 4 GB de RAM, que ya ejecuta varios contenedores **docker** con **caddy** como "fachada". Se han realizado los siguientes pasos para la sustitución del sistema.

- Se han quitado del *reverse proxy* las redes correspondientes a la instalación antigua.
- Se ha localizado la iteración pasada del proyecto y se ha ejecutado el comando (desde la carpeta contenedora) `sudo docker compose down --volumes --remove-orphans` para la limpieza de cualquier traza relacionada con el proyecto. Además, ya que hay imágenes y *Dockerfiles* que han cambiado y **Docker** hay veces que no detecta este tipo de cambios, se ha ejecutado el comando `sudo docker builder prune --all`.
- Se han incorporado las redes internas descritas en el fichero `docker-compose.yml` y las configuraciones de *reverse-proxy* descritas en `caddy-init/Caddyfile` para la instalación ya existente de **caddy**. Se ha reiniciado el servicio para aplicar los cambios.
- Se ha modificado el fichero `docker-compose.yml`, eliminando la parte relativa al *reverse proxy*.
- Se han copiado los ficheros del sistema a la máquina destino y se ha lanzado el despliegue con `docker-compose up -d`.

Así, el sistema (mejorado) ha sido desplegado en la dirección <https://dpr.nico.eus>

5.2. Posibles Problemas y Soluciones

A continuación se detallan algunos problemas que pueden llegar a surgir al configurar el sistema y las posibles soluciones propuestas en cada caso.

- Como se menciona en el apartado 4, las credenciales para acceder a la interfaz web de **cacti** se refieren al usuario **admin** y a la contraseña **1Sobresaliente!**. Estas credenciales también son válidas para la instancia desplegada en la nube (5.1).
- Como se menciona en las conclusiones del apartado 3.2, para visualizar los gráficos ya configurados es suficiente con acceder a la ruta `/cacti/graph_view.php?action=preview`
- Si ya se tiene un sistema **Docker Compose** con nombre canónico **dprnico** ejecutándose en la máquina anfitriona, se ha de parar el sistema previo primero antes de iniciar el sistema provisto.
- No se puede cambiar el nombre canónico del sistema (**dprnico**) ni ninguno de los nombres de los servicios (**dprnico-***) ni los nombres de las redes que conectan a los servicios. Las configuraciones en los ficheros *SQL* y en los diversos ficheros de configuración se encuentran ya establecidas para trabajar con estos nombres y no es posible proveer un sistema de variables de entorno para cambiar estos nombres de forma dinámica.
- Al acceder a las distintas rutas implementadas en la aplicación, se ha de considerar incluir la url completa con la barra al final. Por ejemplo, el acceder a la ruta `/cacti` puede dar problemas; y se deberá de acceder usando `/cacti/` en su lugar.

Referencias

- [1] Net-SNMP Project. <http://www.net-snmp.org/>, [Online; consultado 23-Dic-2024]
- [2] admindean: Cacti 1.2.19 with weathermap,thold and monitor plugins, All in one container. <https://hub.docker.com/r/admindean/cacti>, [Online; consultado 23-Dic-2024]
- [3] FriFra: Re: Run Cacti 1.2.20 @ Weathermap on Docker. <https://forums.cacti.net/viewtopic.php?p=294075#p294075>, [Online; consultado 23-Dic-2024]
- [4] netcup: netcup Gmbh. Virtual Server (VPS) Offers. <https://www.netcup.com/en/server/vps>, [Online; consultado 23-Dic-2024]
- [5] smcline06: Cacti v1+ in docker. Feedback and pull requests are always welcome! <https://hub.docker.com/r/smcline06/cacti>, [Online; consultado 23-Dic-2024]
- [6] TheCactiGroup: Cacti - The Complete RRDTool-based Graphing Solution. <https://cacti.net>, [Online; consultado 23-Dic-2024]
- [7] Wikipedia: Simple Network Management Protocol. https://en.wikipedia.org/wiki/Simple_Network_Management_Protocol, [Online; consultado 23-Dic-2024]

Anexo

A. Configuración Inicial de Cacti

Los pasos que se han seguido para configurar la instalación de `cacti` (para luego exportar las configuraciones) han sido los siguientes.

A.1. Instalación

Una vez se dispone de un contenedor correcto en ejecución y de una ruta para acceder a la herramienta de monitorización, se puede comenzar con la instalación.

1. Al acceder por primera vez a `cacti`, la pantalla de login es presentada. Las credenciales de inicio son `admin` y `admin`
2. Se cambia la contraseña por una que cumpla con los requisitos. Por ejemplo, `1Sobresaliente!`
3. Se acepta la licencia y se hace clic en *Begin*
4. *Next > Next > Next > Next > I have read this statement > Next > Next*
5. Desmarcar todas las casillas excepto *NetSNMP* (para ahorro de recursos)
6. *Next > Next > Confirm Installation > Install*
7. Clic en *Get Started*

A.2. Dispositivos y Gráficos

Una vez se dispone de una instalación correcta de `cacti`, los pasos para añadir los dispositivos a monitorizar y gráficos de las métricas son triviales.

1. Se accede al menú principal de `cacti` con el usuario `admin` y la contraseña creada en el paso anterior (A.1)
2. En el menú lateral, se hace clic en *Create > New Device*
3. Se añade un nombre relevante en el apartado *description*, por ejemplo `api-1`
4. Se pone el hostname donde se encuentra el agente `snmp`, por ejemplo, `dprnico-dprnico-flask-1`.
5. En *Device Template*, se indica *Net-SNMP Device*.
6. En *Downed Device Detection* se indica *Ping and SNMP Uptime*
7. Clic en *Create*.
8. Se repite el proceso desde el paso 2. con el hostname para cada dispositivo.
9. Al entrar en *Graphs > Preview* (en el menú superior, primero a la izquierda y luego a la derecha) se pueden visualizar todos los gráficos con estadísticas relevantes para los dispositivos creados. También existe la posibilidad de hacer un filtrado.