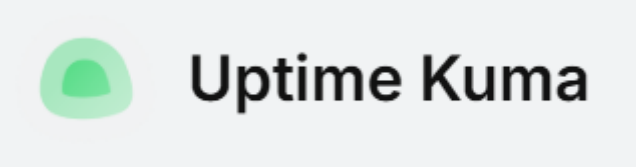
INFORME TRABAJO FINAL



Integrantes:
Polverigiani, Damian
Torres, Nicolas
Uribarri, Gonzalo Damian

Docente: José Luis Di Biase

INTRODUCCIÓN

Uptime Kuma es una herramienta de monitoreo y alertas de tiempo de actividad de código abierto. Si nosotros contamos con diferentes tipos de servicios web y necesitamos llevar un control o simplemente ver el estado de los mismos, esta es la herramienta indicada para dichas funciones.

En esta ocasión vamos a indicar como es la instalación de dicha herramienta monitoreando un par de servicios web de ejemplo que también instalaremos.

CARACTERÍSTICAS DE LA VM UTILIZADA

- Ubuntu LTS 24.04
- **Memoria RAM asignada:** 6420 MB
- **Cantidad de procesadores virtuales:** 4
- **Almacenamiento**: 25GB

REQUISITOS

Vamos a necesitar varias herramientas para poder llevar a cabo todo el proceso, comenzando por las más indispensables que son:

- Node js
- Git
- NGINX
- PM2

Node JS

Para descargar e instalar nvm primero tenemos que verificar que nuestro catálogo de paquetes estén actualizados, esto lo hacemos ejecutando los siguientes dos comandos:

```
sudo apt update
sudo apt upgrade -y
```

Una vez actualizado nuestro catálogo de paquetes lo que necesitamos para instalar NodeJS es ejecutar el siguiente comando:

```
curl -fsSL https://deb.nodesource.com/setup_lts.x | sudo -E bash -
&& sudo apt install -y nodejs
```

Git

Para obtener la última versión estable de su lanzamiento de Debian/Ubuntu ejecutaremos el siguiente comando

```
sudo apt update && sudo apt install -y git
```

NGINX

```
sudo apt install nginx
```

NOTA

En caso de que al ejecutar un comando, el resultado sea un error relacionado a falta de permisos volver a ejecutar el comando indicando la palabra **sudo** delante del mismo, esto le indica a nuestro sistema operativo que se está “cambiando” a un usuario con todos los permisos necesarios momentáneamente solo para la ejecución del comando. Esto nos va a servir para todos los comandos de aquí en adelante del proyecto.

Instalación

Pasos a seguir:

Para poder tener instalado, configurado y funcional nuestra herramienta lo que vamos a necesitar luego de haber instalado todos los prerrequisitos de la sección anterior es saber donde va a quedar hosteado o guardado nuestro servicio por ende lo que vamos a hacer es crear una carpeta o directorio con el nombre y en donde deseemos ejecutando el siguiente comando:

```
mkdir “RUTA_DIRECTORIO”/“NOMBRE_DE_DIRECTORIO”
```

Reemplazar “RUTA_DIRECTORIO”por la ruta donde deseamos crear nuestra carpeta y “NOMBRE_DE_DIRECTORIO” por el nombre que se desea

Una vez creado nuestro directorio donde vamos a guardar todo nuestro proyecto, lo siguiente es acceder al mismo para proseguir con instalación, para eso ejecutaremos el siguiente comando:

```
cd “RUTA_DIRECTORIO”/“NOMBRE_DE_DIRECTORIO”
```

Ahora vamos a clonar el repositorio de nuestra herramienta desde su repositorio remoto hacia nuestro repositorio local que va a ser nuestra carpeta creada recientemente. Acá es donde entra en juego una de nuestras herramientas instaladas con antelación, que es, GIT. Al hacer esto lo que estamos logrando es tener en nuestra computadora todo el código fuente y necesario para poder instalar la herramienta, lo que se debe hacer es ejecutar el siguiente comando:

```
git clone https://github.com/louislam/uptime-kuma.git
```

Esto nos va a crear una nueva carpeta con el nombre del proyecto a la cual debemos acceder con el comando `cd uptime-kuma.`

Al ser un servicio de monitoreo lo que vamos a buscar es que siempre esté funcionando y disponible para validar el estado de nuestros servicios que estemos monitoreando, por ende, vamos a necesitar otra herramienta que nos permita esto ya que si nosotros luego de la instalación ejecutamos la herramienta una la terminal, cuando se cierre la misma, la herramienta se cierra que es lo que no queremos justamente.

La herramienta que nos permita dejar Uptime-kuma siempre en funcionamiento y que cuando reiniciemos nuestra computadora se inicie automáticamente va a ser **PM2.**

Nuestra herramienta contiene un script que debemos ejecutar el cual es el que nos brinda la configuración básica necesaria para que se pueda iniciar, para eso vamos a ejecutar el siguiente comando:

```
npm run setup
sudo npm install pm2 -g
```

En este punto puede ser que nos arroje errores de recursos y dependencias deprecadas, si bien nos dice como solucionarlo acá dejamos el comando:

```
npm audit fix –force
```

Una vez ya ejecutado el setup lo que debemos hacer es decirle a pm2 que ejecute nuestra herramienta y la deje funcionando, para esto necesitamos ejecutar los siguientes comandos

```
pm2 start server/server.js --name uptime-kuma
```

```
pm2 save
pm2 startup
```

Cuando dichos comandos son ejecutados, pm2 nos dirá que ejecutemos el siguiente comando:

```
sudo env PATH=$PATH:/usr/bin /usr/lib/node_modules/pm2/bin/pm2
startup systemd -u “NUESTRO_USUARIO” --hp /home/”NUESTRO_USUARIO”
```

Donde en lugar de “NUESTRO_USUARIO” dirá el nombre de usuario del dueño de la máquina donde se esté configurando toda la herramienta.

Básicamente al ejecutar dicho comando lo que estamos haciendo es dejando nuestro servicio como booteable, es decir, que cada vez que encendamos nuestra máquina se va a ejecutar Uptime-kuma de manera automática.

Lo siguiente que debemos instalar es nginx, lo haremos con el siguiente comando:

```
sudo apt install nginx -y
```

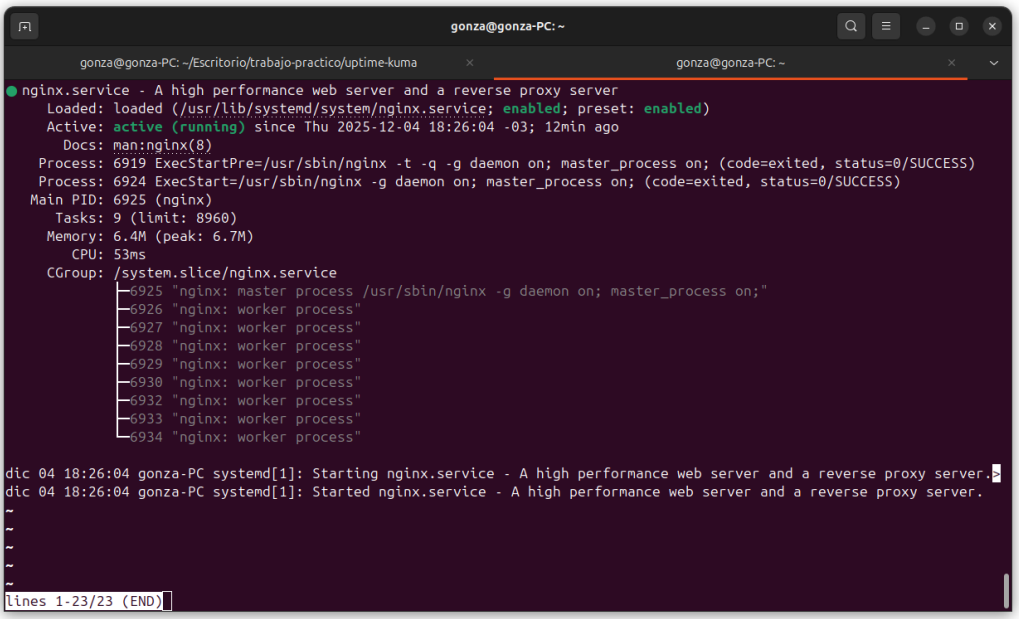
Nginx nos va a ser útil para funcionar como proxy, es decir, cuando le llegue una petición al puerto 80 de node, sabrá que tiene que redireccionarlo hacia el puerto 3001.

Para verificar que oficialmente tenemos instalado nginx podemos hacerlo ejecutando el siguiente comando:

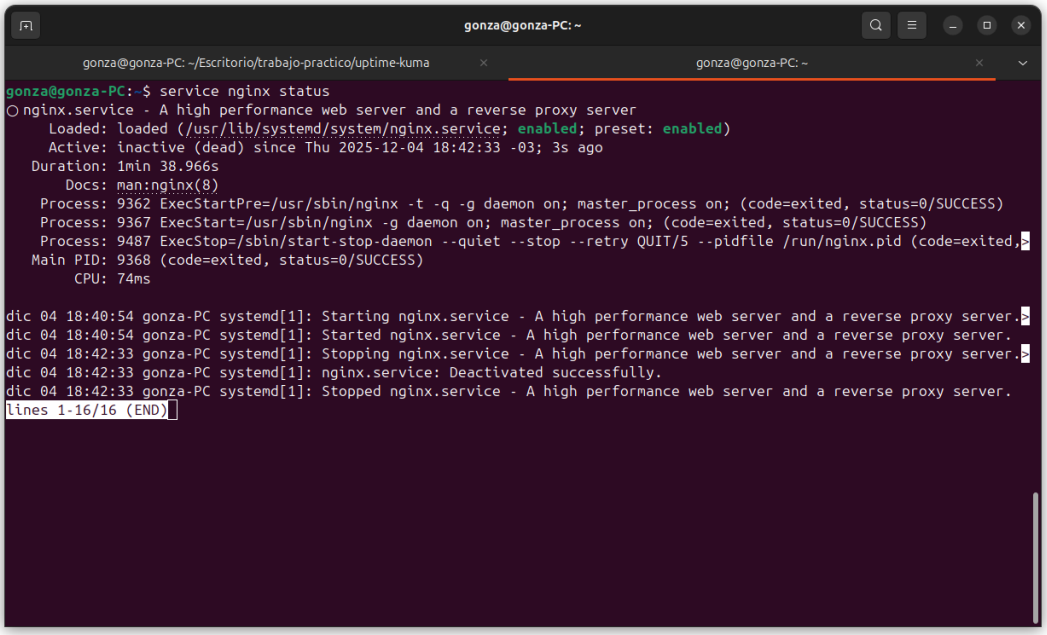
```
nginx -v
```
Si nos arroja algún número de versión es porque efectivamente lo tenemos instalado.

Lo siguiente es validar que tengamos levantado nuestro servidor nginx, eso lo haremos con el siguiente comando:

```
service nginx status
```
El cual nos arrojará un mensaje como este en caso de que esté funcionando



Y en caso contrario uno como este



NOTA

Si tenemos ya instalado en nuestra máquina otro servidor como apache2 no nos permitirá tener ambos servers levantados por conflicto de puertos así que debemos bajar el servidor de apache2 para poder dejar funcionando nuestra herramienta

Una vez tenemos nuestro proxy funcionando lo que vamos a necesitar es configurar nuestro nginx para que efectivamente funcione de proxy, para eso debemos editar el siguiente archivo **`nano /etc/nginx/conf.d/uptime-kuma.conf`** a este archivo agregarle el

siguiente contenido de configuración. Reemplaza «uptime-kuma.tudominio.com» por tu propio nombre de dominio en caso de que tengas uno:

```
server {
    listen 80;
    server_name uptime-kuma.yourdomainname.com;

    location / {
        proxy_pass         http://localhost:3001;
        proxy_http_version 1.1;
        proxy_set_header   Upgrade $http_upgrade;
        proxy_set_header   Connection "upgrade";
        proxy_set_header   Host $host;
        proxy_set_header   X-Real-IP $remote_addr;
        proxy_set_header   X-Forwarded-For
$proxy_add_x_forwarded_for;
        proxy_set_header   X-Forwarded-Proto $scheme;

        # Added WebSocket support
        proxy_set_header   Sec-WebSocket-Key
$http_sec_websocket_key;
        proxy_set_header   Sec-WebSocket-Version
$http_sec_websocket_version;
        proxy_set_header   Sec-WebSocket-Extensions
$http_sec_websocket_extensions;

        # Improve performance of this reverse proxy
        proxy_buffering    off;
    }

    # Redirect HTTP to HTTPS if needed for encryption
    # Uncomment the following lines if you have SSL enabled
    # return 301 https://$host$request_uri;
}
```

Una vez configurado esto, debemos resetear nuestro nginx y probar de que esté funcionando nuestra herramienta, eso lo haremos con el siguiente comando:

```
systemctl restart nginx
```

Una vez instalado y levantado el servicio de Uptime Kuma (normalmente accesible vía URL: *http://<IP_DEL_SERVIDOR>:3001*), se procede con la configuración inicial.

2.1. Selección de idioma

Al ingresar por primera vez, Uptime Kuma solicita elegir el idioma de la interfaz.

- Se selecciona el idioma deseado (en nuestro caso, **Español**).+

2.2. Selección del tipo de Base de Datos

Uptime Kuma ofrece dos opciones de almacenamiento:

A) SQLite

- Base de datos **embebida**, basada en archivos.
- No requiere instalación de un servidor adicional.
- Rendimiento suficiente para cargas bajas o medias.
- Los datos se guardan en un archivo dentro de la carpeta `data/`.

B) MySQL / MariaDB

- Requiere un **servidor de base de datos** levantado aparte.
- Útil para:
    - ambientes con alta concurrencia,
    - instancias distribuidas,
    - respaldo avanzado,
    - más de 10.000 monitores o tráfico muy alto.

Teniendo en cuenta estos aspectos mencionados nos pareció conveniente para el proyecto elegir SQLite para esta parte del proyecto

2.3. Creación del Usuario Administrador

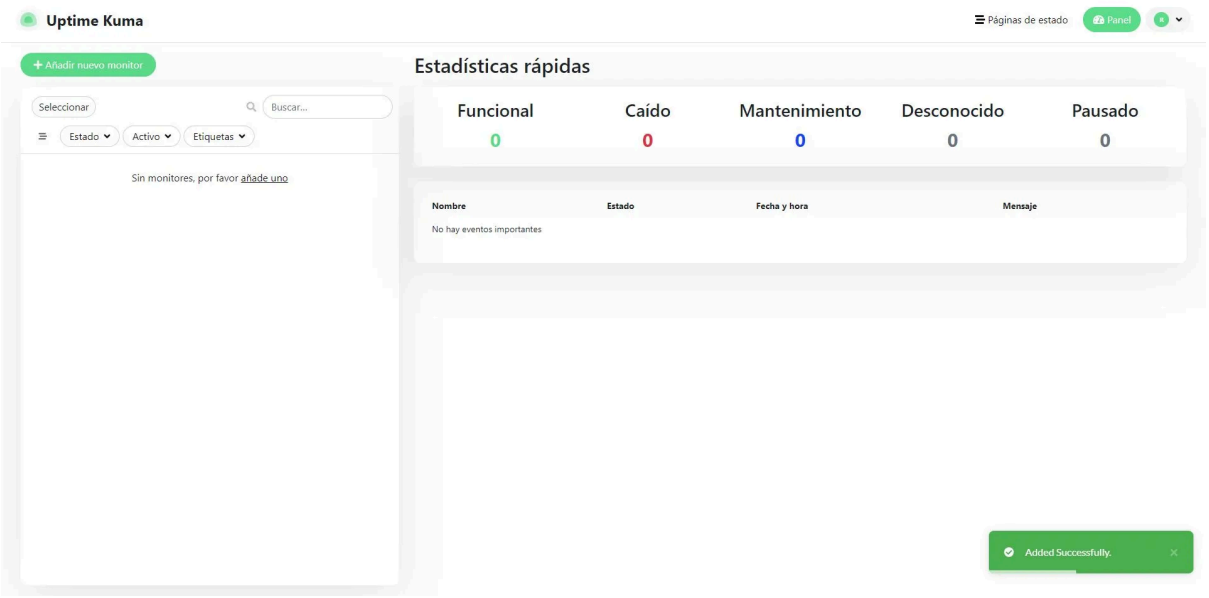
Una vez elegida la base de datos, Uptime Kuma solicita crear la primera cuenta admin.

Se configura Usuario y Contraseña como en cualquier otro tipo de. Este usuario será el administrador principal del panel.

2.4. Acceso al Panel Principal de Uptime Kuma

Una vez completados los pasos anteriores, se ingresa automáticamente al **Dashboard principal**.

La pantalla inicial muestra:



En este punto ya tenemos **Uptime Kuma instalado, configurado y operativo**.

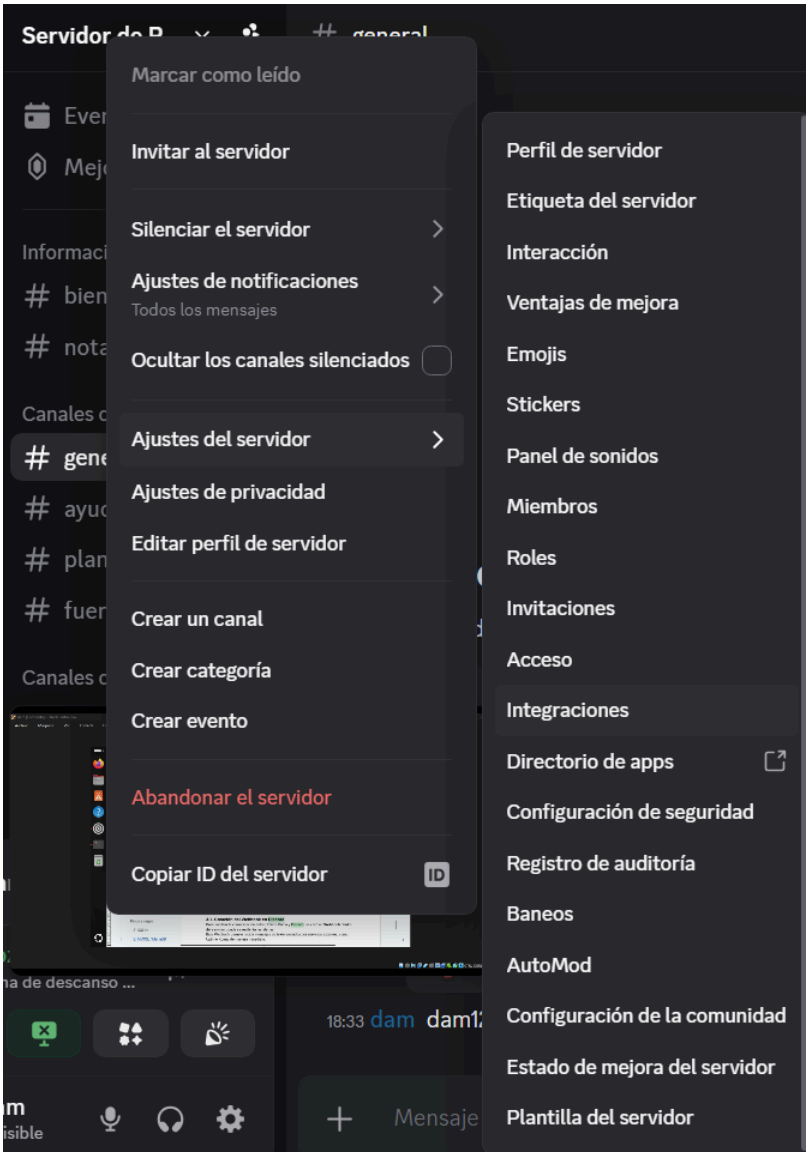
3. Configuración de Alertas y Notificaciones mediante Discord

El objetivo de esta etapa fue configurar un sistema de notificaciones externas para Uptime Kuma, de forma tal que el servicio informe automáticamente caídas, recuperaciones o cambios de estado de los monitores configurados.
Para este trabajo se seleccionó Discord, aprovechando su sistema de Webhooks, que permite recibir alertas en tiempo real sin necesidad de bots complejos.

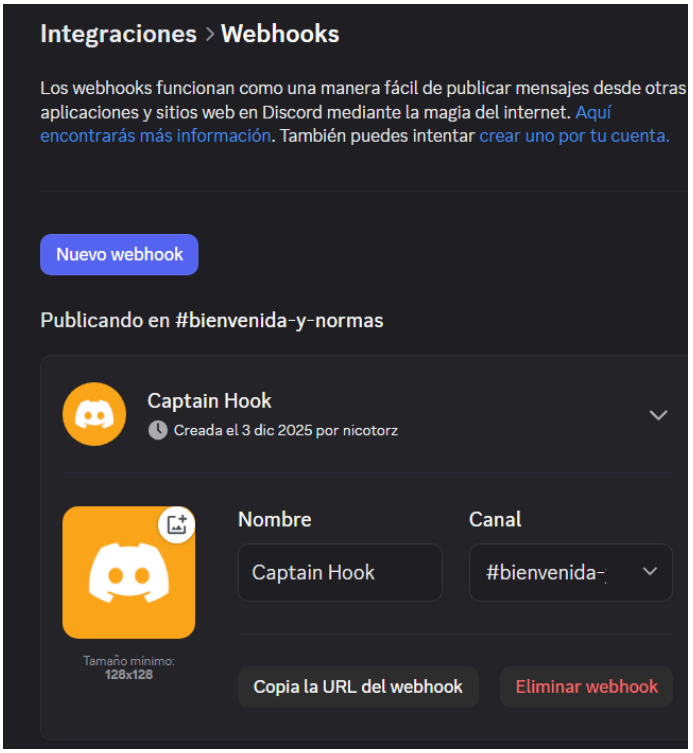
3.1. Creación del Webhook en Discord

Para habilitar la comunicación entre Uptime Kuma y Discord, se creó un **Webhook** dentro del servidor donde se recibirán las alertas.
Este Webhook permite recibir mensajes de texto enviados por servicios externos, como Uptime Kuma, de manera inmediata.

Para esto vamos a ir al Servidor -> Ajustes del Servidor -> Integraciones



En el menú Webhooks creamos un nuevo webhook y copiamos la URL



3.2. Configuración de la notificación en Uptime Kuma

Una vez generado el Webhook, se procedió a integrarlo en Uptime Kuma.

Dentro del panel principal de Uptime Kuma, acceder a:

Añadir nuevo monitor → Configurar notificación.

1. Seleccionar el tipo de notificación: **Discord**.
2. Completar los campos requeridos:
    - **Webhook URL**: se pegó la URL obtenida desde Discord (https://discord.com/api/webhooks/1445944276021416010/zvSvN1R62pwEC-OqpoNPi4amOU9_ACKzc1sJ_TOUnfpvwrAvyQfNTPmdeSBDKuG8Hox2)
    - **Nombre de la notificación**.
3. Guardar la configuración.

Con esto, Uptime Kuma queda habilitado para enviar alertas al servidor de Discord configurado. Para corroborar esto Uptime Kuma nos da la opción de realizar una Test de la notificación y en nuestro discord debemos visualizar algo como esto:

3.3 Configuración de un Monitor SSH y Asociación de Notificaciones

Para que Uptime Kuma pueda no solo monitorear un servicio sino también **enviar alertas al canal de Discord**, es necesario agregar un monitor y asociar la notificación previamente configurada.

En esta práctica, se preparó un **servidor SSH funcional** para ser monitoreado.

3.3.1 Creación del Monitor SSH

1. En el panel izquierdo de Uptime Kuma, seleccionar **Añadir nuevo monitor**.
2. Completar los parámetros del monitor:
    - **Tipo de monitor:** *TCP Port*
    - **Nombre del monitor:** por ejemplo *Servidor SSH Laboratorio*
    - **Host / IP:** dirección del servidor SSH previamente levantado
    - **Puerto:** 22 (por defecto, salvo configuración distinta)
    - **Método de autenticación:**
        - *Password* (para pruebas)
        - *Private Key* (más seguro)
3. Configurar el intervalo de chequeo.
4. Guardar.

En este punto, Uptime Kuma ya está monitoreando activamente el servicio SSH, verificando conectividad, tiempo de respuesta y disponibilidad.

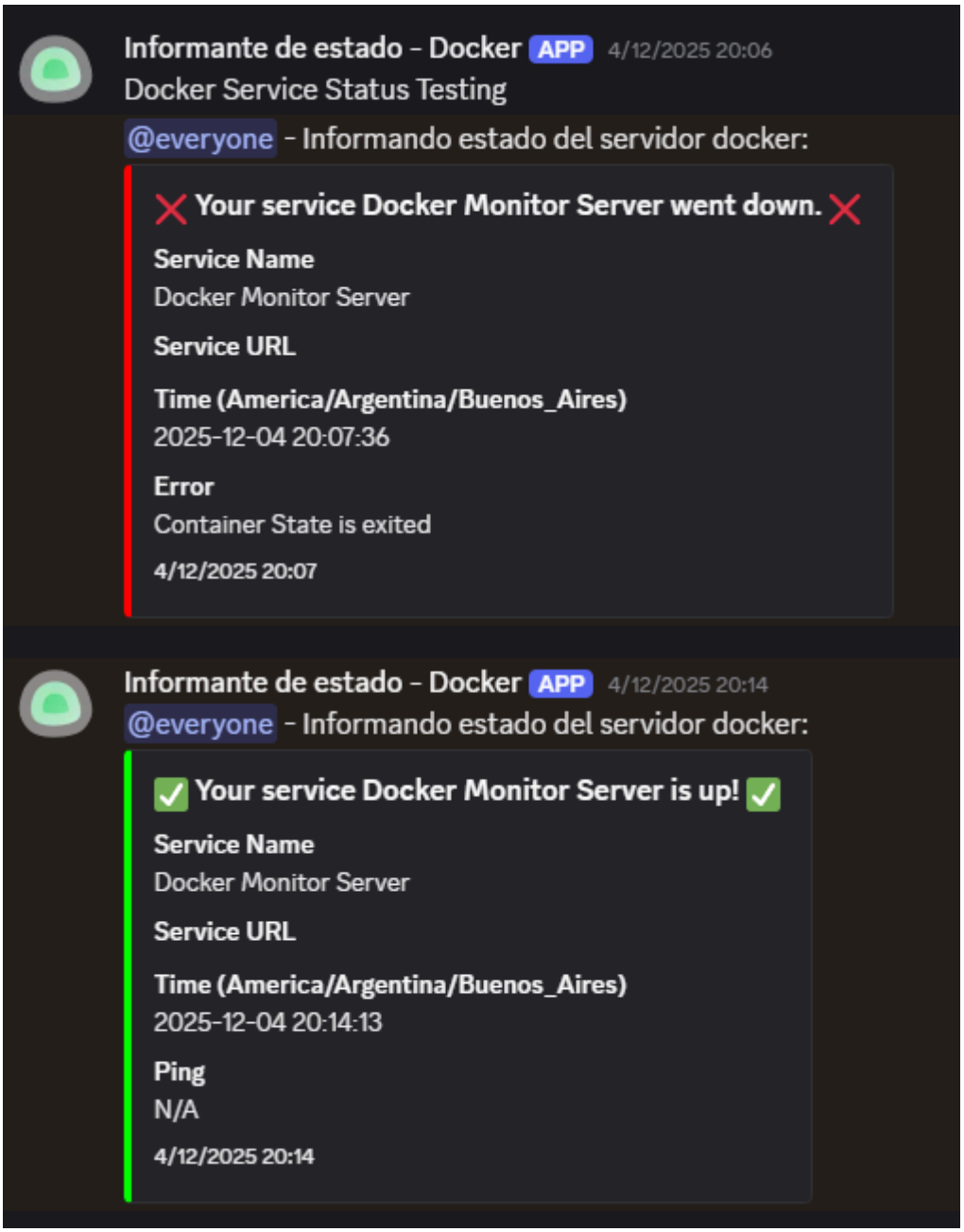
3.3.2 Asociación del Monitor a la Notificación de Discord

Una vez creado el monitor, se debe vincular con el sistema de notificaciones:

1. Entrar al monitor recién creado.
2. Ir a la opción **Notificaciones** dentro del monitor.
3. Seleccionar la notificación creada en el paso anterior (Discord).
4. Guardar cambios.

A partir de este momento:

- Si el servicio SSH **cae**, Uptime Kuma enviará una alerta automática a Discord.
- Si el servicio **se recupera**, también generará una notificación.
- Si hay problemas de latencia, tiempo de respuesta o conexión, también notificará.



Otras de las cosas que vamos a monitorear es un container de Docker el cual dentro va a tener un servidor NGINX y así podremos ver cuando dicho servicio está caído o no.

Para esto vamos a necesitar tener Docker instalado en nuestra computadora, el cual lo instalaremos ejecutando el siguiente comando:

```
sudo apt install docker.io -y
```

Una vez instalado debemos como a cualquier otro servicio iniciarlo, esto se hace ejecutando los siguientes comandos:

```
sudo systemctl start docker
sudo systemctl enable docker
```

Una vez con nuestro servicio funcionando lo que debemos hacer es simplemente crear y ejecutar nuestro contenedor con nuestro servidor dentro, para eso vamos Docker es super amigable y nos va a permitir hacer todos estos pasos con la simple ejecución de un comando, el siguiente:

```
sudo docker run -d --name mi-web-docker -p 8081:80 nginx:alpine
```

No nos vamos a detener en explicar que es Docker porque no es el fin de este trabajo pero detallaremos brevemente el comando anterior.

- **`docker run`** con este comando le estamos indicando a Docker que ejecute nuestro container con la parametrización que le demos.
- **`-d`** con este comando lo que le indicamos a Docker es que ejecute dicho container en background para que nos permita seguir utilizando la terminal
- **`--name`** con este le indicamos que nombre va a tener nuestro contendor de Docker.
- **`-p`** con este comando le indicamos que puerto vamos a disponibilizar de nuestro container para que escuche contra nuestra computadora.
- **`nginx:alpine`** con estos datos le estamos indicando que servicio queremos instalar sobre qué distribución

Una vez ejecutado dicho comando ya vamos a tener nuestro servidor funcionando en nuestro container, para certificar esto podemos ejecutar el siguiente comando:

```
docker ps
```

El cual si nuestro container está funcionando nos arrojará un listado con los datos de nuestro container.

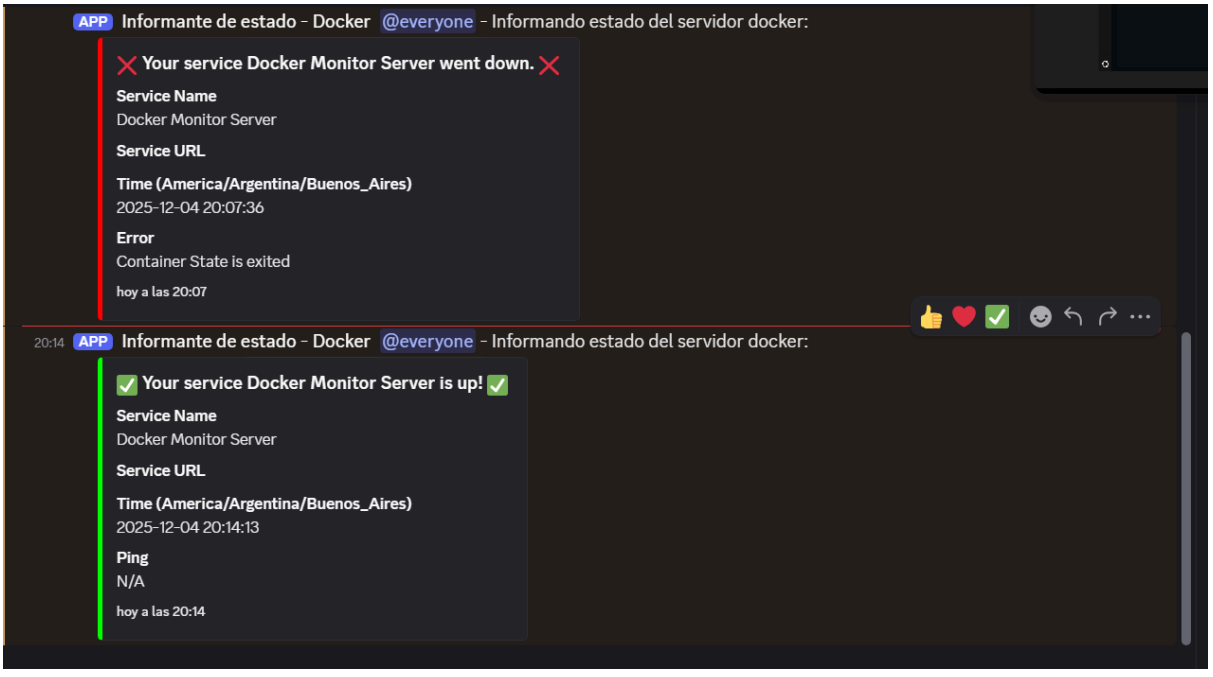
3.3.3 Creación de monitor container Docker

1. En el panel izquierdo de Uptime Kuma, seleccionar **Añadir nuevo monitor**.

2. Completar los parámetros del monitor:

    - **Tipo de monitor:** *Contenedor de Docker.*
    - **Nombre sencillo:** *Un nombre que nos guste.*
    - **Nombre/ID de Contenedor:** *El id o nombre de contenedor que nos aparece al ejecutar el comando de Docker ps.*
    - **Host Docker:** *Es este campo tenemos que agregar un host con el signo "+" que nos aparece y lo vamos a configurar de la siguiente manera:*
        - **Nombre Sencillo:** *Un nombre que nos guste.*
        - **Tipo de Conexión:** *Socket.*
        - **Dominio Docker:** */var/run/docker.sock*

Notificación docker:



4. Creación de una Status Page

Las *Status Pages* son páginas públicas (o privadas) generadas por Uptime Kuma para mostrar el estado de los servicios monitoreados.

Son utilizadas en entornos reales para informar a usuarios o clientes sobre incidentes, mantenimientos o disponibilidad general del sistema..

4.1 Acceso al Módulo de Status Pages

Dentro del panel principal de Uptime Kuma:

1. En el menú seleccionar Status Pages.
2. Hacer clic en Crear nueva Status Page.

Esto abre un formulario de configuración inicial.

4.2 Configuración Inicial de la Status Page

En la creación de la página se completaron los siguientes campos:

- Nombre de la Status Page
- Descripción (opcional): para indicar propósito o alcance.
- Modo de acceso:
    - Pública → accesible sin autenticación
    - Privada → requiere password o login. En este caso, se configuró Pública para facilitar la visualización.

Una vez confirmada esta información, la Status Page queda creada.

4.3 Personalización de la página

Uptime Kuma permite personalizar la apariencia de la Status Page:

- Logo (opcional)
- Colores y estilo general
- Título principal
- Mensaje introductorio
- URL pública que generará Kuma para compartir

4.4 Agregar Monitores a la Status Page

Una Status Page no muestra información hasta que se vinculan los monitores.

Para agregar Monitores:

1. Abrir la Status Page creada.
2. Ir a la opción Añadir servicio o Add Monitor.
3. Seleccionar el monitor previamente configurado (por ejemplo, *SSH, MySQL, Docker, etc)*
4. Opcional: asignar un “grupo” para ordenar los monitores, como:
    - Backend
    - Bases de datos
    - Infraestructura
5. Guardar cambios.

El monitor ahora aparece públicamente en la Status Page con su estado actual, histórico y uptime.

4.5 Visualización y funcionamiento de la Status Page

La Status Page muestra:

- Estado actual de cada monitor (Up, Down, Degraded)
- Porcentaje de uptime
- Historial de incidentes
- Línea temporal de disponibilidad
- Fecha y hora del último chequeo