Despliegue de OpenNebula en Dos Servidores: Frontend y Backend

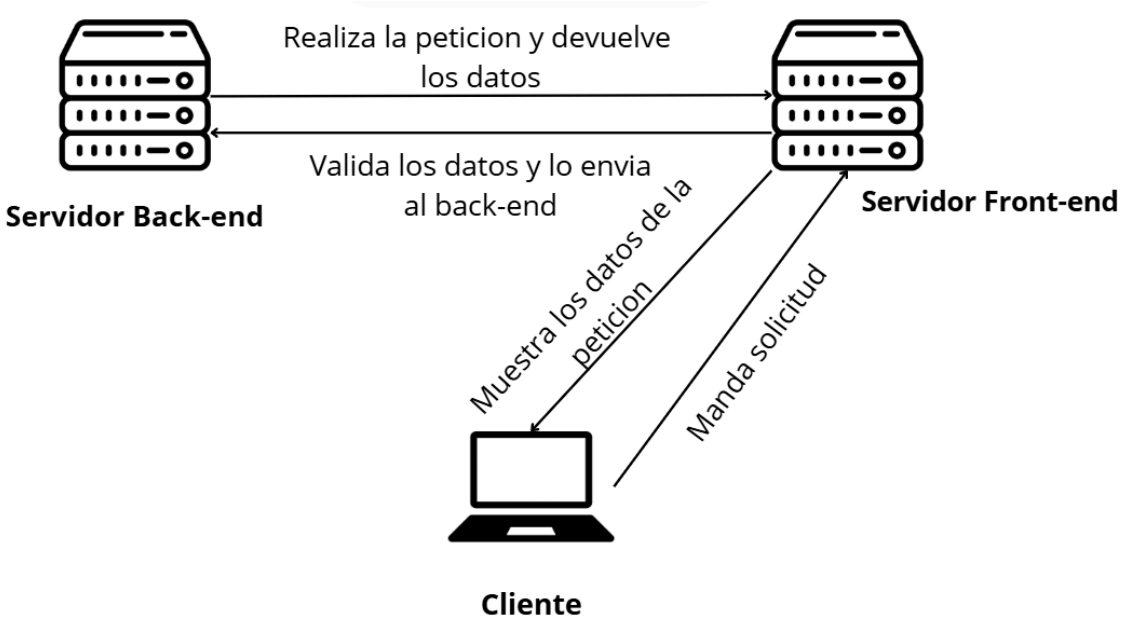
Introducción

En este documento se describe el proceso de despliegue de OpenNebula en dos servidores distintos, separando el frontend y el backend. Esta arquitectura permite mejorar la escalabilidad, la seguridad y el rendimiento del sistema. A lo largo del documento, se explicará la razón de esta elección, los pasos realizados y cómo funciona el sistema en esta configuración.

Arquitectura del Despliegue

El sistema se ha dividido en dos servidores:

- **Servidor 1: Frontend** → Alojando la interfaz de OpenNebula Sunstone.
- **Servidor 2: Backend** → Encargado de la gestión de las máquinas virtuales y almacenamiento.



Justificación de la Separación

La decisión de dividir OpenNebula en dos servidores se basa en los siguientes factores:

1. **Escalabilidad**: Separa la carga de trabajo entre la gestión y la interfaz gráfica.
2. **Seguridad**: Reduce la exposición del backend a accesos directos desde Internet.
3. **Mantenimiento**: Permite actualizaciones independientes del frontend y backend sin afectar a todo el sistema.

Implementación con Debian 12 y Docker

Para realizar el despliegue, se utilizó **Debian 12** como sistema operativo base en ambos servidores. Además, se empleó **Docker** para gestionar la ejecución de las aplicaciones, facilitando la portabilidad y el aislamiento de los servicios.

1. Instalamos Docker en ambos servidores
2. Creamos contenedores para cada componente asegurando la correcta comunicación entre ellos.
3. Configuramos las redes y volúmenes en Docker para la persistencia de datos y la comunicación entre los contenedores.

DockerFile y docker-compose servidor Back-end

Dockerfile:

```
# Usa la imagen base oficial de PHP con Apache y PHP 8.2
FROM php:8.2-apache

# Instala extensiones necesarias para Laravel y cron
RUN apt-get update && apt-get install -y \
    libzip-dev \
    unzip \
    git \
    cron \
    && docker-php-ext-install zip pdo pdo_mysql

# Instala Composer
COPY --from=composer:2.6 /usr/bin/composer /usr/bin/composer

# Habilita el módulo de reescritura de Apache
RUN a2enmod rewrite
RUN sed -i 's!/var/www/html!/var/www/html/public!g' /etc/apache2/sites-available/000-default.conf

# Establece el directorio de trabajo
WORKDIR /var/www/html

# Copia los archivos del proyecto Laravel
COPY . /var/www/html

# Configura permisos
RUN chown -R www-data:www-data /var/www/html

# Expone el puerto 80
EXPOSE 80

# Comando para iniciar cron y Apache
CMD ["sh", "-c", "cron && apache2-foreground"]
```

docker-compose.yml:

```
version: '3.8'

services:
  laravel-app:
    build:
      context: .
      dockerfile: Dockerfile
    ports:
      - "8080:80"
    volumes:
      - ./proyecto-backend:/var/www/html
    environment:
      APP_ENV: local
      APP_DEBUG: true
      APP_KEY: "base64:rhKHr2xitJBkC3MBj+PtBC3dMs9D4pCDOPz+tuVGtlk="
      DB_CONNECTION: mysql
      DB_HOST: "172.20.227.241"
      DB_PORT: 3306
      DB_DATABASE: "grupo4_2425"
      DB_USERNAME: "grupo4_2425"
      DB_PASSWORD: "UMq7.E-[5Lt_@oMA"
```

DockerFile, docker-compose y nginx para el servidor Front-end

Dockerfile:

```
# Usa una imagen base oficial de Node.js para construir la aplicación
FROM node:18 as build-stage

# Establece el directorio de trabajo
WORKDIR /app

# Copia los archivos package.json y package-lock.json
COPY package*.json ./

# Instala las dependencias
RUN npm install

# Copia el resto del código de la aplicación
COPY . .

# Construye la aplicación para producción
RUN npm run build

# Usa una imagen ligera de Nginx para servir los archivos estáticos
FROM nginx:1.23 as production-stage

# Copia los archivos de la build de Vue al directorio de Nginx
COPY --from=build-stage /app/dist /usr/share/nginx/html

# Copia una configuración personalizada de Nginx (opcional)
COPY nginx.conf /etc/nginx/conf.d/default.conf


# Inicia Nginx
CMD ["nginx", "-g", "daemon off;"]
```

docker-compose.yml:

```
services:
  vue-app:
    build:
      context: .
      dockerfile: Dockerfile
    ports:
      - "80:80"
    restart: always
```

nginx:

```
server {
    listen 80;
    server_name localhost;

    root /usr/share/nginx/html;
    index index.html;

    location / {
        try_files $uri /index.html;
    }

    error_page 404 /index.html;
}
```