Universidad Europea
CENTRO PROFESIONAL

- Ciclo Formativo: Técnico Superior en Administración de Sistemas Informáticos en Red – Modalidad a distancia.
- Titulo: "Optimización y Despliegue de Infraestructura Tecnológica para el Servicio Técnico de RPJ S.L. mediante Entorno LEMP con Docker en Proxmox VE".
- Director(a)/tutor(a) del proyecto:
    - Sara Gonzalez Gonzalez
- Nombre y apellidos del/los autor/es.
    - Jose Angel Perez Fausto
    - Pablo Quevedo Pacín
    - Rommel Rafael Rojas Vargas
- Curso academico 2023 - 2024.

Universidad Europea
CENTRO PROFESIONAL

Índice

Resumen/Abstract

El presente proyecto tiene como objetivo desplegar una aplicación web de ticketing on-premise como herramienta para el servicio técnico de la empresa RPJ S.L. Para ello, desarrollamos e implementamos un entorno LEMP (Linux, Nginx, MariaDB, PHP) con Docker Compose sobre un contenedor Debian LXC alojado en un servidor local Proxmox VE. El entorno debe ser accesible mediante VPN, disponer de un sistema de backup externo open source y considerar la redundancia de datos y la alta disponibilidad.

This project aims to deploy a ticketing web application on-premise as a tool for technical service of the company RPJ S.L. To do so, we develop and implement a LEMP environment (Linux, Nginx, MariaDB, PHP) with Docker Compose on a Debian LXC container hosted on a local Proxmox VE server. The environment must be accessible via VPN, have an external open source backup system and consider data redundancy and high availability.

Justificación del Proyecto

La ejecución de este proyecto surge de una cuidadosa evaluación de las necesidades operativas de RPJ S.L., centrándose en la mejora de la gestión del personal y las intervenciones del dpto. de Servicio Técnico mediante la implementación de una herramienta de ticketing. A continuación, se detallan las razones específicas que respaldan la decisión de llevar a cabo este proyecto con un determinado *stack* tecnológico:

1. Optimización del Servicio Técnico: con la adopción de una herramienta web para gestionar las intervenciones del personal técnico de RPJ S.L. La empresa obtendrá un control y seguimiento más eficiente de incidencias, asegurando una atención oportuna y una resolución efectiva de problemas.
2. Tecnologías Modernas y Eficiencia Operativa: la puesta en marcha de un entorno LEMP mediante Docker Compose en un contenedor LXC virtualizado en Proxmox supone la adopción de tecnologías modernas y open source que garantizan la eficiencia operativa. Este enfoque modular

facilita la administración y actualización de componentes de forma ágil y sostenible.

3. Escalabilidad y Flexibilidad: la modularidad ofrecida por Docker permite una escalabilidad flexible según las necesidades del servicio técnico. Esto optimiza los recursos existentes y brinda la capacidad de adaptarse a futuras expansiones y cambios en los requerimientos operativos.
4. Acceso Seguro desde Ubicaciones Externas: la implementación de una VPN proporciona un acceso seguro a la aplicación web desde ubicaciones externas, lo cual es esencial para todos los colaboradores remotos y facilita la conectividad segura en un entorno empresarial distribuido.
5. Respaldo de Datos Confiable: utilizar herramientas de respaldo propias del software Proxmox y sistemas de almacenamiento en red como Proxmox Backup Server, nos permiten configurar copias de seguridad de forma local y externa, por lo que podemos asegurar una protección fiable de los datos, minimizando el riesgo de pérdida de información crítica y preservando la integridad de los datos empresariales.

Objetivos

Objetivo general

Desplegar una aplicación web que ayudará a gestionar las intervenciones del personal del dpto. Servicio Técnico de la empresa RPJ S.L, utilizando un grupo de elementos de software conocido como entorno LEMP (Linux, Nginx, MariaDB, PHP) mediante Docker Compose en un contenedor LXC alojado on-premise en un servidor Proxmox VE.

Objetivos específicos

1. Diseñar y desplegar la infraestructura tecnológica con el software de virtualización Proxmox VE, el cual nos permitirá virtualizar los contenedores LXC.
2. Establecer un servicio de VPN mediante Wireguard en un contenedor LXC para facilitar un acceso remoto seguro a la web y al propio servidor.

3. Desplegar una aplicación web en un contenedor LXC en Proxmox con Docker Compose gestionando los servicios Nginx, Php-fpm y MariaDB, desarrollando el código de forma coordinada en un repositorio de Github.
4. Establecer dos sistemas de respaldo, uno de forma local con la herramienta ofrecida por Proxmox VE y uno de forma externa utilizando un equipo a mayores con Proxmox Backup Server, garantizando la integridad de los datos.

Producto Mínimo Viable

El Producto Mínimo Viable (PMV) en el contexto de este proyecto, se centra en lograr un entorno operativo básico que incluya las características fundamentales para el despliegue de la herramienta de ticketing. A continuación, se detallan los requisitos y las user stories correlacionadas con los módulos implicados en el proyecto:

1. Módulo: Servidor Proxmox VE *on-premise*

- Requisitos:
    - Instalar Proxmox VE en el servidor de producción.
    - Establecer la configuración de red y asignación de IP estática.
    - Identificar los discos físicos y volúmenes que se van a usar como almacenamiento.
- User Story:
    - Como administradores del servidor, necesitamos preparar el entorno donde vamos desplegar y gestionar los contenedores LXC que alojarán la aplicación web.

2. Módulo: Contenedores Docker para la Aplicación Web:

- Requisitos:
    - Crear contenedores Docker para Nginx, MySQL/MariaDB y PHP-FPM.
    - Configurar volúmenes persistentes para almacenar datos críticos.
    - Desplegar la aplicación web de ticketing dentro de los contenedores.
- User Story:

- Como administradores del sistema, queremos usar una herramienta moderna como Docker Compose para desplegar contenedores, que faciliten la gestión de la aplicación web y que nos permite portabilidad y flexibilidad.

3. Módulo: Acceso Seguro mediante VPN:

- Requisitos:
  - Configurar un contenedor LXC para el servicio Wireguard.
  - Establecer una conexión VPN segura al servidor Proxmox VE desde ubicaciones externas.
- User Story:
  - Como usuarios remotos, necesitamos acceder de forma segura al entorno de gestión de tickets desde ubicaciones externas.

4. Módulo: Copias de Seguridad y Respaldo:

- Requisitos:
  - Implementar la herramienta de backup de Proxmox, la cual ofrece configurar copias de seguridad que apunten a directorios tanto locales como externos.
  - Configurar tareas automatizadas para ejecutar copias de seguridad
- User Story:
  - Como administradores de sistemas, queremos garantizar que se realicen copias de seguridad periódicas del contenedor LXC que alberga la aplicación web, utilizando Backup de Proxmox salvaguardando así la integridad de los datos para minimizar el riesgo de pérdida de información.

Tecnologías Empleadas

Para la implementación de nuestro proyecto, utilizamos el hardware del que ya disponíamos, el cual fue suficiente para satisfacer las necesidades de rendimiento y capacidad de almacenamiento requeridas para el desarrollo del entorno donde se ejecutaría la herramienta web.

Material Empleado

- **Hardware**:
  - Laboratorio 1 (Proxmox VE): HP Proliant DL380p Gen.8
    - Procesadores: 2x Intel Xeon E5 2650v2
    - Memoria Ram: 64 Gb ECC
    - Almacenamiento: 4x HDD SAS SFF 600Gb
    - Controladora de discos: HP Smart Array P420i
  - Laboratorio 2 (Proxmox Backup Server):
    - Procesador: Intel i3-8100
    - Memoria Ram: 12 Gb
    - Almacenamiento: 3x HDD 20Gb
- **Sistemas Operativos**:
  - Proxmox VE: Utilizado como el hipervisor anfitrión para alojar contenedores LXC y gestionar los recursos de hardware.
  - Debian: Seleccionado por su ligereza para la puesta en marcha, usando en los contenedores LXC.
  - Proxmox Backup Server: Es una solución de respaldo de datos de código abierto diseñada para entornos de virtualización y contenedores. Permite respaldar y restaurar máquinas virtuales, contenedores y almacenamiento de datos de manera eficiente, escalable y segura. Ofrece una interfaz web intuitiva, y la programación de respaldos automáticos.
- **Stack LEMP**:
  - Se optó por Linux y Docker Compose como plataforma/arquitectura.
  - Para el servicio web, se utilizaron Nginx y Php-fpm.
  - Se emplea MariaDB como el sistema gestor de base de datos.
- **Software**:
  - Docker Compose: Empleado para la orquestación y gestión del contenedor LXC que compone nuestro entorno LEMP.

- Backup Proxmox (vzdump): Implementado para realizar copias de seguridad de los contenedores LXC, asegurando la integridad y disponibilidad de los datos.
- Git: Utilizado para el control de versiones del código fuente y la colaboración en el desarrollo de la infraestructura.
- Visual Studio Code: Herramienta de desarrollo utilizada para escribir y editar el código de la aplicación web.
- Wireguard: Instalado en uno de nuestros contenedores LXC y empleado para establecer una VPN privada que permita el acceso seguro a la aplicación web desde ubicaciones externas.

Esta combinación de hardware y software proporcionó la base necesaria para el diseño, implementación y despliegue efectivo de la infraestructura tecnológica que respalda la aplicación web de gestión de tickets en el entorno LEMP con Docker sobre Proxmox VE.

Desarrollo

El desarrollo del proyecto se ha estructurado en varias etapas, comenzando desde la preparación del hardware hasta el despliegue efectivo de la aplicación en el entorno LEMP con Docker Compose en un contenedor LXC sobre Proxmox VE. A continuación, mencionamos y describimos todas las fases.

Puesta en marcha del servidor

Este módulo se centra en la preparación inicial del servidor lo que incluyó, la verificación física de todos sus componentes, chequeo de sus capacidades y recursos, la configuración del almacenamiento mediante RAID 1+0 y los pasos para la instalación y configuración del OS.

RAID a nivel BIOS

Es importante mencionar que para la puesta a punto del servidor nombrado como Laboratorio 1 (HP Proliant DL380 Gen8), se ha configurado un RAID 1+0 mediante la herramienta de gestión del almacenamiento que este trae incorporada (Smart Storage Administrator) accediendo a través de la BIOS.

Como ya hemos mencionado, este software viene integrado en la BIOS del servidor, el cual se puede utilizar para configurar y gestionar como controladora raid en los servidores HP Proliant. Esta tarea se pudo realizar mediante interfaz gráfica. El hecho de que pudiéramos realizar una configuración RAID 1+0 (RAID 10), nos permitió:

- Proporcionar un rendimiento más elevado de lectura de todas las configuraciones de tolerancia a fallos.
- No perder datos cuando una unidad falla, siempre que ninguna unidad con fallos esté duplicada en otra unidad con fallos.
- Tener en el servidor una tolerancia a fallos de hasta la mitad de las unidades físicas.

En el Anexo I pueden encontrar un breve paso a paso de cómo realizar la configuración del RAID.

Proxmox VE

Una vez que tuvimos preparado el equipo pasamos al siguiente paso que fue el de la instalación del sistema operativo "Proxmox PVE versión 8". Elegimos Proxmox VE [1] debido a su naturaleza de hipervisor de tipo 1, también conocido como bare-metal, lo que significa que se ejecuta directamente sobre los recursos físicos del servidor, sin necesidad de un sistema operativo host adicional.

Este software de virtualización nos permite una mayor flexibilidad, escalabilidad y agilidad de todas la máquinas o contenedores virtualizados por su interfaz gráfica de gestión la cual nos brinda una visión completa de la arquitectura desplegada. Gracias a esto obtenemos ciertas ventajas y beneficios:

- Reducción de costes y gastos operativos.
- Minimiza/elimina los tiempos de inactividad.
- Acelera y simplifica el despliegue de recursos y aplicaciones.
- Respalda la continuidad operativa y la recuperación ante desastres.
- Permite una gestión centralizada.

Para la configuración del sistema y con el servidor (Laboratorio 1) en marcha, procedemos a utilizar nuestros ordenadores personales, los cuales deben estar conectado a la misma red que el servidor. Nos dirigimos entonces a el navegador web y establecemos la conexión utilizando la dirección IP que configuramos en el servidor, seguida del puerto 8006, viéndose de esta manera: http://192.168.1.150:8006. Seguidamente el navegador nos indica que se ha establecido una conexión, pero que esta no era segura. Esto sucede porque el certificado SSL no está configurado ni validado. Permitimos la conexión e iniciamos sesión en la interfaz web de Proxmox con nuestro usuario root y contraseña.

Proxmox VE es un desarrollo de software libre (Open Source), que permite a gran parte de la comunidad de administradores de sistemas acceder a un Hipervisor de tipo 1 excelente para entornos de desarrollo con capacidades equiparables a sistemas de virtualización de pago. Sin embargo, Proxmox también ofrece en el mercado versiones en modalidad de suscripción. Esto hace que, por defecto, el sistema se descargue con repositorios (/apt/*) que apuntan a la versión "pve-enterprise" en vez de a la versión "pve-no-subscription". Es importante cambiar los repositorios a la versión de no suscripción para reducir costos, obtener flexibilidad en la gestión, asegurar actualizaciones de seguridad y mantener transparencia y control sobre el sistema.

- Procedemos a desactivar los repositorios "pve-enterprise" y los sustituimos con los repositorios "pve-no-subscription", con los siguientes comandos.

```
sed -i '/enterprise/s/^/# /' /etc/apt/sources.list.d/ceph.list
sed -i '/enterprise/s/^/# /' /etc/apt/sources.list.d/pve-enterprise.list
echo 'deb http://download.proxmox.com/debian/pve bookworm pve-no-subscription'
>> /etc/apt/sources.list
```

- Verificamos que los repositorios se han actualizado correctamente en el archivo de configuración.

```
root@pve:~# cat /etc/apt/sources.list
deb http://ftp.es.debian.org/debian bookworm main contrib
```

```
deb http://ftp.es.debian.org/debian bookworm-updates main contrib
# security updates
deb http://security.debian.org bookworm-security main contrib
deb http://download.proxmox.com/debian/pve bookworm pve-no-subscription
```

Con estos pasos nos aseguramos de que nuestro servidor pueda mantener sus paquetes actualizados a las últimas versiones, evitando así descargas erróneas, desactualizadas u obsoletas.

Contenedores LXC

En comparación con las máquinas virtuales, los contenedores LXC [3]ofrecen una mayor ligereza al momento de realizar el despliegue por la particularidad de compartir el kernel con el host en el que se ejecutan, en lugar de emular un sistema operativo (SO) completo. Esto quiere decir que no hay una capa de abstracción entre el contenedor LXC y el hardware, algo que es poco usual en entornos de desarrollo porque implica una "brecha" de seguridad ante un ciberataque. Pero nosotros al no abrir el servidor a red pública, no nos supone un riesgo como tal.

Proxmox VE ofrece este tipo de virtualización y están perfectamente integrado, esto quiere decir que utilizan los mismos recursos de red y de almacenamiento que las máquinas virtuales por lo que es compatible con la configuración de nuestra propuesta. La gestión de dichos contenedores se simplifica al proporcionar la interfaz gráfica del propio Proxmox permitiendo así realizar tareas complejas de manera fácil y rápida.

Los contenedores LXC para nosotros vienen a ser la opción más idónea, ya que son perfectos para instalar y ejecutar aplicaciones como Docker, brindándonos beneficios de usar una máquina virtual sin el peso que ello conlleva. Esto significa que los contenedores LXC en Proxmox se pueden clasificar como "contenedores de sistema", en lugar de "contenedores de aplicaciones".

Contenedor/host LXC VPN con WireGuard

Entre todas las aplicaciones con protocolo de VPN de código abierto que existen, hemos elegido WireGuard [9] por su ligereza, rapidez y seguridad. Esta

herramienta combina operaciones criptográficas avanzadas para cifrar los datos intercambiados.

Este protocolo utiliza una criptografía denominada enrutamiento de criptoclaves, que funciona asociando claves públicas a una lista de direcciones IP utilizadas explícitamente para túneles, esto quiere decir que los equipos que se quieren interconectar utilizarán claves públicas cifradas para autenticarse entre sí.

Debido a la gran seguridad y fiabilidad que otorga, viendo la imperiosa necesidad de tener un acceso remoto al servidor sin exponerlo de manera directa, tanto para la administración en caso de tener que realizar mantenimientos o reparaciones, como la opción de poder trabajar con la aplicación de tickets en remoto. Hemos decidido implementar este servicio en nuestro servidor.

Es compatible prácticamente con cualquier sistema operativo. Esto nos facilita la utilización por ejemplo de dispositivos portátiles como tabletas. Wireguard dispone de una aplicación para Android, la cual se vincula con el escaneo de un código qr. Es generado por consola mediante un simple comando.

Ventajas de usar Wireguard:

- Es una herramienta eficiente y rápida utilizando algoritmos optimizados permitiendo un gran rendimiento incluso en redes congestionadas.
- Es seguro y robusto ya que utiliza algoritmos criptográficos modernos con menos código y al mismo tiempo menos complejos, pero, diseñados para resistir diversos tipos de ataques.
- Fácil de usar y fácil de configurar, su simplicidad lo hace ideal para nuestro proyecto ya que nuestro entorno requería una implementación rápida y que fuera compatible con las tecnologías que usamos para el despliegue de la aplicación web.

El proceso de instalación y configuración del servicio se encuentra redactado en el Anexo II.

Contenedor/host LXC con Aplicación Web Ticketing

Una vez hemos preparado Proxmox para alojar servicios y Wireguard para permitir conexiones seguras, es hora de desplegar la aplicación de Ticketing.

Cabe destacar que nosotros no hemos diseñado la aplicación, sino que lo ha hecho otro equipo. Nosotros hemos tomado los archivos de la aplicación y los hemos incluido en nuestro repositorio proyecto lemp compose junto con los archivos de configuración de Nginx, MariaDB y Docker Compose, entre otros.

A continuación, describiremos el proceso de creación y configuración del contenedor LXC, así como la administración del despliegue de la aplicación web.

Instalación y configuración del contenedor LXC

Inicialmente habíamos hecho pruebas creando contenedores de forma manual a través de las plantillas de Proxmox pero finalmente nos decantamos por emplear estos scripts [2] que automatizan el proceso.

Introduciendo el siguiente comando en la consola de Proxmox, se ejecutará un script que resultará en un nuevo contenedor LXC. Al aplicar la configuración predeterminada, se le asignarán 2GB de RAM, 4GB de almacenamiento y 2vCPU, lo suficiente para nuestra aplicación web de uso corporativo.

```
bash -c "$(wget -qLO - https://github.com/tteck/Proxmox/raw/main/ct/docker.sh)"
```

Si bien el comando anterior crea el contenedor (al que se le asigna la dirección IP 192.168.1.50 en la red local corporativa) y automáticamente instala Docker y Docker Compose, decidimos configurar más a fondo este LXC para su futura administración.

Una vez que podemos iniciar sesión en el nuevo LXC, introducimos esto:

```
bash -c "$(curl –fsSL
https://raw.githubusercontent.com/pabloqpacin/proyecto_lemp_compose/main/scripts/
lxc-base.sh)"
```

Ese comando ejecuta un script (Anexo IV) que forma parte de nuestro repositorio, y que hemos creado para automatizar la configuración inicial del

sistema, instalación de paquetes, etc. Sin entrar en detalle, estas son las acciones llevadas a cabo por nuestro script:

- Instalar las siguientes herramientas de terminal: bat (lector de archivos con resaltado sintáctico), fzf (buscador de archivos "fuzzy" recursivo), grc (colorizador de texto), jq (procesador de JSON), lf (explorador de archivos), mycli (cliente para bases de datos MySQL y MariaDB), nmap (herramienta de análisis de red), neofetch (presenta información del sistema), tmux (multiplexador de terminal), tree (lista archivos recursivamente) y vim (editor de texto).
- Clonar un repositorio dotfiles con varios archivos de configuración y hacer symlinks de los archivos relevantes (destacando aliases para comandos).
- De manera opcional pero recomendada, instalar zsh, oh-my-zsh y un par de plugins para tener resaltado sintáctico en la terminal y sugerencias según el histórico de comandos.
- Clonar proyecto lemp compose y poner en funcionamiento los servicios.

Despliegue de servicios para la aplicación web

Ya está, dos comandos bastan para crear de forma automatizada un contenedor LXC Debian en Proxmox, configurar el sistema y desplegar los servicios de la aplicación web con Docker Compose.

Pero ¿cómo funciona este despliegue? Como ya hemos anticipado, usamos Docker Compose, por lo que no necesitamos instalar los servidores web y de base de datos en el host (en este caso un LXC). En cambio, definimos la operativa de los servicios en el archivo 'compose.yaml' (Anexo III).

Básicamente el proceso de despliegue es el siguiente:

1. Se clona en el LXC nuestro repositorio remoto proyecto lemp compose, concretamente la rama main, que incorpora los últimos cambios estables. Esta es la relación de archivos del repositorio y su papel:

. ├── helpdesk-core-php │ ├── database/ │ ├── scss/ │ └── *.php ├── nginx │ └── default.conf ├── php-fpm │ └── Dockerfile ├── scripts │ ├── exec-mysqldump.sh │ └── lxc-base.sh ├── LICENSE ├── README.md ├── .gitignore ├── compose.yaml ├── mysql.properties └── phpmyadmin.properties	EL REPOSITORIO Y SUS ARCHIVOS - La aplicación web, base de datos, estilos - Configuración del servidor web Nginx - Configuración de PHP (se instala mysqli) - Scripts para la administración del LXC - Licencia Open Source (MIT) - Información sobre el repo - Definición de los servicios - Configuración de MariaDB

2. Ya sea manualmente o desde el script 'lxc-base.sh' (Anexo IV), se accede al repositorio local y se ejecuta el comando 'docker compose up –d', con lo que Docker Compose seguirá las instrucciones del archivo 'compose.yaml' (Anexo III).
3. Lo primero que hará Docker Compose es descargar las imágenes Docker. En nuestro entorno empleamos nginx:1.25.4-alpine para el servicio **www**, mariadb:11.2 para el servicio **db** y, en el caso del servicio **php-fpm**, construiremos nuestra propia imagen ("proyecto-php-fpm") a partir de php:8.1.27-fpm-bullseye, instalando la extensión mysqli haciendo uso de un Dockerfile.
4. En segundo lugar, Docker crea el volumen [4] 'proyecto_mysql_data', que vinculamos al directorio '/var/lib/mysql' del contenedor MariaDB para dar persistencia a nuestra base de datos.
5. Ahora Docker Compose crea la red [5] interna 'proyecto_default' para facilitar la integración de los servicios de los contenedores Docker.

6. Finalmente, se crean los propios contenedores Docker, en nuestro caso 3. Describamos por encima cada uno de ellos:

php-fpm

Este contenedor proporciona el intérprete PHP, responsable de procesar los archivos .php de la aplicación web y de hacer consultas a la base de datos.

```
php-fpm:
  build: ./php-fpm
  volumes:
    - ./helpdesk-core-php:/var/www/html
  restart:
    always
```

Aparte de instalar mysqli para facilitar el intercambio de información entre la base de datos y el intérprete de PHP, realizamos un bind mount [6] que montará los archivos locales de la aplicación web en el directorio relevante del contenedor. Los archivos permanecen en el LXC. La política de rebotes [7] definida es *always*, que también aplicaremos al resto de los contenedores.

www

Si bien es cierto que inicialmente habíamos usado una misma imagen que proporcionaba ambos servicios PHP y servicio web (php:7.4-apache), finalmente decidimos separar estos servicios.

Nginx como servidor web de alto rendimiento estable, y con un consumo de recursos muy bajo, es el compañero ideal de PHP-FPM. Nginx tiene una arquitectura asíncrona que es mucho más escalable, basada en eventos. Además, al usar Nginx con PHP-FPM se mejora la eficiencia a nivel de consumo de memoria.

PHP funciona como un servicio separado al usar PHP-FPM. Al usar esta versión de PHP como intérprete del lenguaje, las peticiones se procesan a través de un socket TCP/IP; de modo que el servidor web Nginx solo maneja las peticiones HTTP y PHP-FPM interpreta el código PHP. El hecho de tener

dos servicios separados es clave para ganar en eficiencia. (Stackscale, 2022) [8]

```
  www:
    image: nginx:1.25.4-alpine
    ports:
      - "80:80"
    volumes:
      - ./helpdesk-core-php:/var/www/html
      - ./nginx/default.conf:/etc/nginx/conf.d/default.conf
    depends_on:
      - php-fpm
      - db
    restart:
      always
```

Así, nuestro servicio web lo proporciona Nginx. Mapeamos el puerto 80 del contenedor con el del LXC anfitrión para facilitar las conexiones a la web, y montamos en el contenedor tanto los archivos de la aplicación como el archivo de configuración que define el comportamiento de Nginx (Anexo V). También se define la relación de dependencia del servidor web con respecto a los otros servicios.

db

Este tercer y último contenedor proporciona el servicio de base de datos.

```
services:
  db:
    image: mariadb:10.6
    volumes:
      - ./helpdesk-core-php/database:/docker-entrypoint-initdb.d
      - mysql_data:/var/lib/mysql
    command: --default-authentication-plugin=caching_sha2_password
    env_file: mysql.properties
    restart:
      always
```

```
volumes:
  mysql_data:
```

Usamos MariaDB por ser totalmente open source y compatible con MySQL. Podríamos mapear el puerto 3306 del contenedor al mismo del LXC para permitir conexiones con mycli pero, superada la fase de desarrollo, decidimos no llevar a cabo este mapeo en producción ya que si fuera necesario aún podríamos conectarnos con 'docker exec -it proyecto-db-1 mariadb -u admin –ppassword'.

Para inicializar nuestra base de datos montamos nuestro dump.sql en la ubicación '/docker-entrypoint-initdb.d' del contenedor. Usaremos el volumen 'mysql_data' para facilitar la administración del servicio, y cargaremos las variables de entorno más importantes a través del archivo mysql.properties (Anexo VI): MYSQL_DATABASE, MYSQL_ROOT_PASSWORD, MYSQL_USER, etc.

Uso y administración de la aplicación web

Podemos verificar que los contenedores están corriendo con 'docker ps', y revisar los eventos de los distintos servicios con 'docker compose logs'. Al completarse con éxito el despliegue, y teniendo configurada la conexión VPN, ya es posible conectarse remotamente a nuestra página web (en la dirección IP asignada al contenedor LXC) y hacer login con nuestro usuario.

En el dump.sql se definen una serie de usuarios. Al loguearnos podemos consultar los tickets existentes y crear nuevos, según las necesidades de la organización y del departamento al que pertenezca el empleado de RPJ S.L., tal y como se muestra en el Anexo VII.

Configuración del Respaldo del Servidor

Uno de los objetivos específicos de este proyecto es la automatización de copias de seguridad tanto locales como externas, asegurando la integridad y disponibilidad de los datos en todo momento. Para lograr esto, hemos implementado la herramienta de backup del sistema Proxmox, que permite realizar snapshots de los contenedores LXC en producción sin necesidad de detener su

funcionamiento. Las copias de seguridad se generan en un formato de archivo estándar, lo que facilita su almacenamiento tanto en dispositivos locales como en red.

Configuración de respaldos en Local

Para programar las tareas de backup local, inicialmente se creó un directorio específico, /mnt/backuplocal, que se utiliza como destino de los respaldos. Esto nos permite como administradores del sistema acceder y restaurar rápidamente las copias de seguridad en caso de fallos o errores en el servidor.

Configuración de respaldos externos

Para este cometido como mencionamos a lo largo de la memoria, vamos a utilizar Proxmox Backup Server. Es una variante de proxmox creada y sumunistrada por la misma empresa.

Para este apartado utilizaremos una máquina dedicada en exclusiva para este cometido (laboratorio 2), la instalación no dista de un sistema operativo cualquiera, o del propio Proxmox VE. Al hacer la instalación se le asigna una ip estática para luego poder acceder a él mediante interfaz web (192.168.1.165:8007). El proceso de instalación y configuración se encuentra redactado en el Anexo VIII.

Conclusiones

Este proyecto lo hemos llevado a cabo con el objetivo de desplegar una aplicación web para gestionar las intervenciones del personal del departamento de Servicio Técnico de la empresa RPJ S.L. Utilizando un conjunto de elementos de software integrados o lo que es igual a un entorno LEMP (Linux, Nginx, MariaDB, PHP) mediante Docker Compose en contenedores LXC alojados on-premise en un servidor Proxmox VE.

Podemos confirmar que hemos logrado satisfactoriamente todos los objetivos planteados en este proyecto. Desde el diseño e implementación de la infraestructura tecnológica hasta el despliegue de la aplicación web, cada paso ha sido completado con éxito, permitiendo un funcionamiento confiable del servicio.

Aunque no hemos realizado contribuciones significativas al campo, nuestro proyecto destaca por su aplicación práctica y utilidad para empresas del sector. La utilización de sistemas y software open source demuestra nuestra capacidad para implementar soluciones eficientes y económicas.

Las limitaciones en su mayoría podemos decir que fueron económicas, durante la realización del proyecto, al depender en su mayoría de recursos propios y software open source. Sin embargo, identificamos como área de mejora la implementación de un clúster de tres nodos Proxmox para garantizar una alta disponibilidad del servicio y redundancia de los datos, lo cual podría ser considerado en futuras investigaciones o desarrollos.

El trabajo realizado es relevante para empresas consultoras de servicios informáticos que buscan optimizar la gestión de su personal técnico. La puesta en marcha de una aplicación web como la que hemos elegido para este proyecto, puede ser implementada en la práctica real, proporcionando un mejor control y seguimiento todos de las intervenciones realizadas por los técnicos.

Durante la realización de este proyecto, hemos adquirido conocimientos y habilidades fundamentales en el manejo de sistemas y software como Proxmox VE, Docker, Proxmox Backup Server, Contenedores LXC Linux, entre otros, enfrentando desafíos diversos que hemos superado mediante investigación y documentación. La experiencia obtenida constituye un valioso aprendizaje para futuros proyectos y el desarrollo profesional en el campo de la Administración de Sistemas Informáticos en Red.

Este proyecto representa como se pueden usar profesionalmente herramientas y aplicaciones tecnológicas e innovadoras open source para mejorar la gestión de recursos y servicios en el ámbito empresarial, demostrando que no se necesita una gran cantidad de recursos o inversión inicial, obteniendo como resultado una solución práctica y prácticamente libre en el campo de la Administración de sistemas informáticos.

Webgrafía. Bibliografía

1. Software Hipervisor: Proxmox VE. [En línea]. Disponible en: https://www.proxmox.com/en/proxmox-virtual-environment/features. [Accedido: 23-abril-2024].
2. Proxmox VE Helper-Scripts: Scripts for Streamlining Your Homelab with Proxmox VE. [En línea]. Disponible en: https://tteck.github.io/Proxmox/. [Accedido: 08-may-2024].
3. Contenedores Linux LXC. [En línea]. Disponible en: https://pve.proxmox.com/wiki/Linux_Container#_technology_overview. [Accedido: 23-abril-2024].
4. Docker: Volumes. [En línea]. Disponible en: https://docs.docker.com/storage/volumes. [Accedido: 09-may-2024].
5. Docker: Networking in Compose. [En línea]. Disponible en: https://docs.docker.com/compose/networking. [Accedido: 09-may-2024].
6. Docker: Bind mounts. [En línea]. Disponible en: https://docs.docker.com/storage/bind-mounts. [Accedido: 09-may-2024].
7. Docker: Política de rebotes. [En línea]. Disponible en: https://github.com/compose-spec/compose-spec/blob/master/spec.md#restart . [Accedido: 09-may-2024].
8. Stackscale: ¿Qué es PHP-FPM? Un PHP para webs de alto tráfico. [En línea]. Disponible en: https://www.stackscale.com/es/blog/php-fpm-php-webs-alto-trafico/#Nginx_y_PHP-FPM_los_companeros_ideales . [Accedido: 09-may-2024].
9. Acceso Remoto VPN: WireGuard. [En línea]. Disponible en: https://www.wireguard.com/quickstart/. [Accedido 23-abril-2024].

Anexos

Anexo I

Pasos para configurar un RAID 1+0 o 10:

1. Inicie el servidor y presione la tecla F5 durante el arranque para abrir la interfaz gráfica HP Smart Storage Administrator.
2. Seleccione el controlador desde la lista en la ventana GRUB y presione Enter.
3. En la interfaz de HP Smart Storage Administrator, vaya a "Controlador" y luego seleccione "Configure" y "Create Array".
4. Si hay un RAID existente, elimínelo seleccionando "Delete Array" en la sección "Logical Devices".
5. Si es necesario, limpie las unidades anteriores marcándolas y seleccionando "Erase Drive" en la sección "Unassigned Drive".
6. Cree un nuevo array seleccionando las unidades deseadas y haciendo clic en "Create Array".
7. Seleccione las unidades para el nuevo array y haga clic en "Create Array".
8. Especifique el nivel RAID, tamaño de bloques y otros parámetros deseados.
9. Haga clic en "Create logical Drive", verifique los parámetros y haga clic en "Finish" para completar el proceso.

Anexo II

Instalación y configuración Wireguard:

- Para ello hemos utilizado y ejecutado un script que nos ha facilitado la instalación de dicho contenedor. Los pasos a seguir son muy sencillos, solo debemos ejecutar el siguiente comando en la shell de nuestro nodo proxmox.

```
bash -c "$(wget -qLO - https://github.com/tteck/Proxmox/raw/main/ct/wireguard.sh)"
```

- Tras su ejecución nos ha dado la opción de dejar los ajustes por defecto o hacer una configuración mas personalizada. Tras varias pruebas hemos concluido que los valores por defecto están bien configurados y son suficientes para nuestro propósito.El contenedor viene por defecto con estos valores:
    - 512MiB de memoria RAM
    - 2GB de almacenamiento
    - 1vCPU
        - Estos valores se pueden modificar al momento de la instalación, o posteriormente sin ningún problema.

- Lo siguiente para dejar establecida la conexión con entre el servidor y los clientes/usuarios que van a acceder a este servicio sería el de editar el archivo de configuración del servidor. Ejecutamos lo siguiente:

```
nano /etc/pivpn/wireguard/setupVars.conf
```

- La parte a configurar serían, los DNS que queremos que use la VPN, el puerto de escucha y añadir nuestra ip publica, o en su defecto como es nuestro caso ponerle un DDNS. Hemos usado los servicios de no-ip para crear el DDNS, da 3 DDNS gratuitos renovables cada mes de por vida.

```
PLAT=Debian
OSCN=bullseye
USING_UFW=0
pivpnforceipv6route=1
IPv4dev=eth0
install_user=root
install_home=/root
VPN=wireguard
pivpnPORT=51820
pivpnDNS1=1.1.1.1
pivpnDNS2=8.8.8.8
pivpnHOST=josea991.ddns.net
INPUT_CHAIN_EDITED=0
FORWARD_CHAIN_EDITED=0
INPUT_CHAIN_EDITEDv6=
FORWARD_CHAIN_EDITEDv6=
pivpnPROTO=udp
pivpnMTU=1420
```

- Para la creación de los usuarios ejecutamos el siguiente comando:

```
pivpn add
```

- Al introducir el comando nos pedirá el nombre de usuario que queremos añadir. Posteriormente se creará el archivo de configuración de ese usuario automaticamente.
    - En nuestro caso vamos a suponer que queremos añadir este cliente a un ordenador para trabajar en remoto, y no queremos hacer uso de la app que tiene para movil. Si quisieramos usar un dispositivo movil, solo haría falta escanear el qr.

```
pivpn -qr
```

    - Seleccionariamos el usuario que queremos y nos generaría un qr en consola.

- Para usarlo en un ordenador habria que acceder a la ruta de los archivos de configurción de cada usuario dentro del servidor.

```
cd configs
```

  - Aquí se encuentran todas las carpetas de configuracion de cada usuario y podemos ver la del usuario prueba:

```
[Interface]
PrivateKey = yODODIPJweQgYNLLGdU6wKXq9soCys9jsOEGUWkKFFM=
Address = 10.144.146.9/24
DNS = 1.1.1.1, 8.8.8.8

[Peer]
PublicKey = Zj7UU3UQ97/msrZH51nqePa/ykrHm5Gc34oZhhMrWgI=
PresharedKey = 2zTzWD1EksmwdcdMmTGZtgIIxrqLRJui1cUSgYFb/9U=
Endpoint = josea991.ddns.net:51820
AllowedIPs = 0.0.0.0/0, ::0/0
PersistentKeepalive = 25
```

- Estos datos son los que habria que pegar en el programa de wireguard dentro del ordenador que queramos usar en remoto.

Iinstalación en dispositivo remoto:

- Dentro del programa le daríamos a añadir túnel vacío, le pondríamos un nombre y pegariamos los datos anteriormente mencionados. Se guardaría, y estaría listo para usarse.

Con estos pasos, logramos implementar con éxito un contenedor LXC con WireGuard VPN para garantizar un acceso seguro y privado a nuestra aplicación web de gestión de tickets desde ubicaciones externas, utilizando claves generadas para establecer conexiones seguras. Una vez dentro de la red, los usuarios deben autenticarse en la interfaz de la herramienta de ticketing a través de su navegador, accediendo a la dirección IP: 192.168.1.50.

Anexo III

Archivo docker-compose.yaml del repositorio:

```
name: proyecto
 services:

  php-fpm:
    build: ./php-fpm
    volumes:
      - ./helpdesk-core-php:/var/www/html
    restart:
      always

  www:
    image: nginx:1.25.4-alpine
    ports:
      - "80:80"
    volumes:
      - ./helpdesk-core-php:/var/www/html
      - ./nginx/default.conf:/etc/nginx/conf.d/default.conf
    depends_on:
      - php-fpm
      - db
    restart:
      always

  db:
    image: mariadb:10.6
    # ports:
    #   - "3306:3306"
    volumes:
      - ./helpdesk-core-php/database:/docker-entrypoint-initdb.d
      - mysql_data:/var/lib/mysql
    command: --default-authentication-plugin=caching_sha2_password
env_file: mysql.properties
    restart:
      always

  # phpmyadmin:
  #   image: phpmyadmin:5-apache
  #   ports:
  #     - "8080:80"
  #   env_file:
  #     - phpmyadmin.properties
  #   depends_on:
  #     - db
 volumes:
  mysql_data:
  # /var/lib/docker/volumes/proyecto_mysql_data/
```

Anexo IV

Script de configuración inicial y despliegue de la aplicación:

```
#!/usr/bin/env bash

# # En los LXC o VMs (testeado siendo root en Debian LXC):
# bash -c "$(curl -fsSL
https://github.com/pabloqpacin/proyecto_lemp_compose/raw/main/scripts/lxc-base.sh)"

apt_install(){
    if [ ! -e "/etc/apt/apt.conf.d/99show-versions" ]; then
        echo 'APT::Get::Show-Versions "true";' > /etc/apt/apt.conf.d/99show-versions
    fi

    apt update && apt upgrade -y && apt autoremove -y && apt autoclean

    apt install -y bat fzf grc git jq lf mycli nmap tmux tree vim
    apt install -y --no-install-recommends neofetch

    if ! command -v bat && command -v batcat; then
        mv $(which batcat) /usr/bin/bat
    fi

    read -p "Install tldr [y/N]? " opt_tldr
    if [[ $opt_tldr == 'y' ]]; then
        apt install tldr
        if [ ! -d ~/.local/share ]; then
            mkdir ~/.local/share
        fi
        tldr --update
    fi
}

check_disk(){
    disk_usage=$(df -h | grep '/$' | grep % | awk '{print $5}' | tr -d '%')
    if [[ $disk_usage > 50 ]]; then
        echo "Disk is $disk_usage% full, watch out"
    fi
}

clone_dotfiles() {
    if [ ! -d ~/dotfiles ]; then
        git clone --depth 1 https://github.com/pabloqpacin/dotfiles ~/dotfiles
    fi
    if [ ! -d ~/.config ]; then
        mkdir ~/.config &>/dev/null
    fi
    if [ ! -L ~/.myclirc ]; then
        ln -s ~/dotfiles/.myclirc ~/
    fi
```

```
    if [ ! -L ~/.config/lf ]; then
        ln -s ~/dotfiles/.config/lf ~/.config
    fi
    if [ ! -L ~/.config/bat ]; then
        ln -s ~/dotfiles/.config/bat ~/.config
        sed -i 's/OneHalfDark/Coldark-Dark/' ~/.config/bat/config
    fi
    if [ ! -L ~/.vimrc ]; then
        if [ -e ~/.vimrc ]; then mv ~/.vimrc{,.bak}; fi
        sed -i 's/nvim/vim/g' ~/dotfiles/.zshrc
        ln -s ~/dotfiles/.vimrc ~/
    fi
    # if [ ! -L ~/.config/tmux ]; then
    #     ln -s ~/dotfiles/.config/tmux ~/.config       # Problemas para seleccionar/copiar texto
    # fi
}

setup_zsh(){
    apt update && apt install -y zsh

    if [ ! -d ~/.oh-my-zsh ]; then
        yes | sh -c "$(curl -fsSL
https://raw.githubusercontent.com/ohmyzsh/ohmyzsh/master/tools/install.sh)"
        bash $HOME/dotfiles/scripts/setup/omz-msg_random_theme.sh
    fi

    chsh -s $(which zsh) $USER

    if [ ! -L ~/.zshrc ]; then
        mv ~/.zshrc{,.bak} &&
        ln -s ~/dotfiles/.zshrc ~/
    fi

    if [[ ! -d ~/dotfiles/zsh/plugins/zsh-autosuggestions || ! -d ~/dotfiles/zsh/plugins/zsh-syntax-
highlighting ]]; then
        bash $HOME/dotfiles/zsh/plugins/clone-em.sh
    fi
}

# setup_docker(){
#     sh <(curl -sSL https://get.docker.com)
# }


start_compose(){
    if [ ! -d $HOME/PROYECTO ]; then
        git clone https://github.com/pabloqpacin/proyecto_lemp_compose $HOME/PROYECTO
    fi

    if ! docker ps -a --format '{{.Names}}' | grep -q 'proyecto'; then
```

```
        cd $HOME/PROYECTO
        docker compose up -d
        sleep 7 && ./scripts/exec-mysqldump.sh
        cd $HOME
    fi
}


# ---

if true; then
    apt_install
    clone_dotfiles

    opt_zsh=''
    while [[ $opt_zsh != 'y' && $opt_zsh != 'n' ]]; do
        read -p "Establecer zsh [y/n]? " opt_zsh
    done
    if [[ $opt_zsh == 'y' ]]; then
        setup_zsh
    fi

    start_compose
fi

echo "" && neofetch && grc docker ps -a && echo "" &&
df -h | grep -B1 '/$' && echo -e "\nReinicia el contenedor.\n"
```

Anexo V

Archivo de configuración de Nginx:

```
server {
  index index.php index.html;
  server_name phpfpm.local;
  error_log  /var/log/nginx/error.log;
  access_log /var/log/nginx/access.log;
  root /var/www/html;

  location ~ \.php$ {
    try_files $uri =404;
    fastcgi_split_path_info ^(.+\.php)(/.+)$;
    fastcgi_pass php-fpm:9000;
    fastcgi_index index.php;
    include fastcgi_params;
    fastcgi_param SCRIPT_FILENAME $document_root$fastcgi_script_name;
    fastcgi_param PATH_INFO $fastcgi_path_info;
  }
}
```

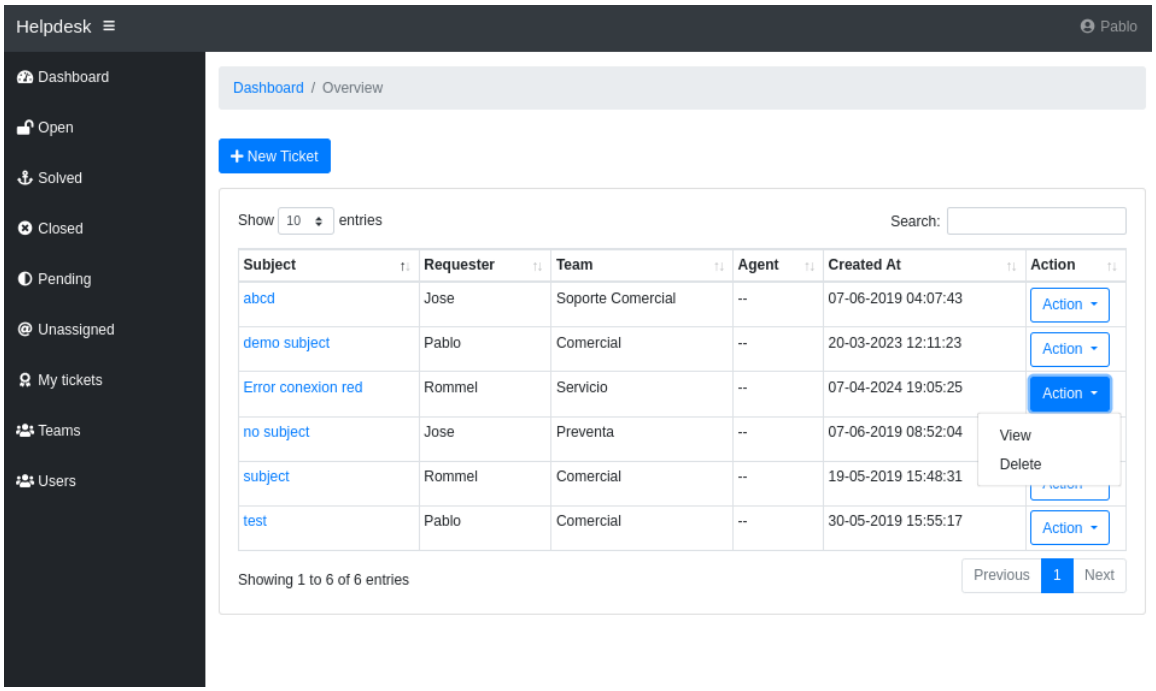
Anexo VI

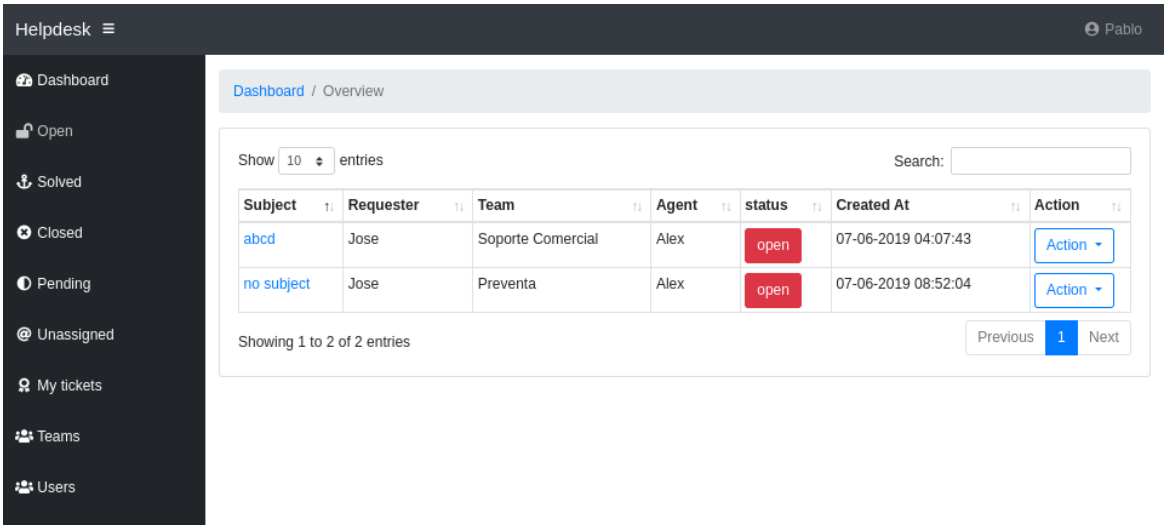
Archivo de configuración de MariaDB:

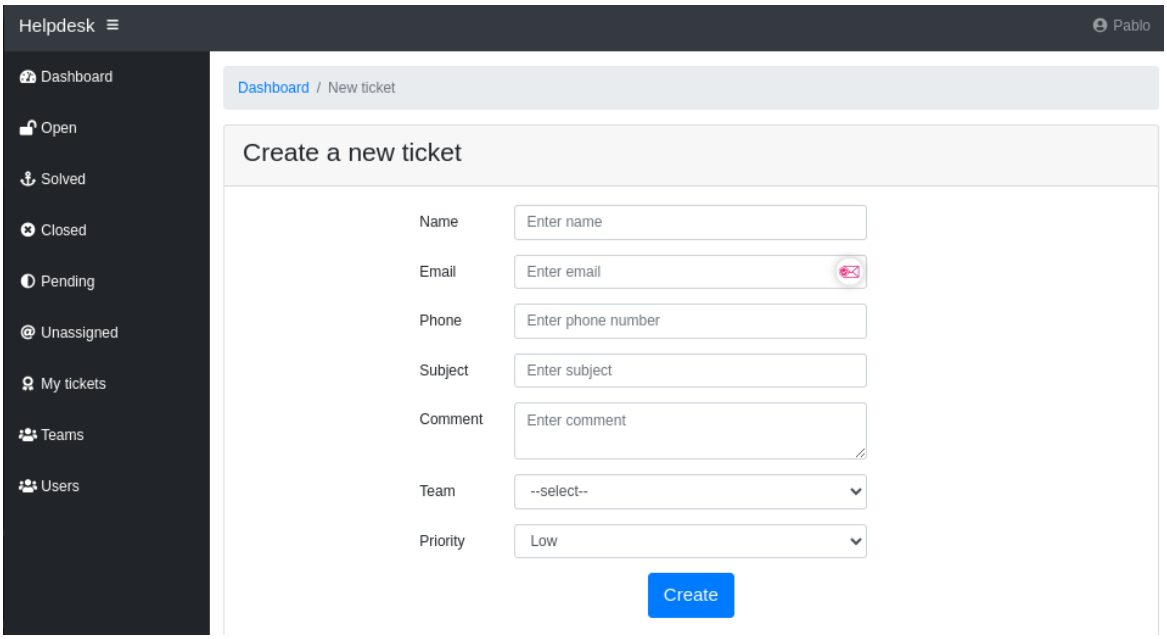
```
MYSQL_USER='admin'
MYSQL_PASSWORD='password'
MYSQL_ROOT_PASSWORD='password'
MYSQL_DATABASE='helpdesk_core_php'
MYSQL_ROOT_HOST='%'
TZ='Europe/Madrid'
```

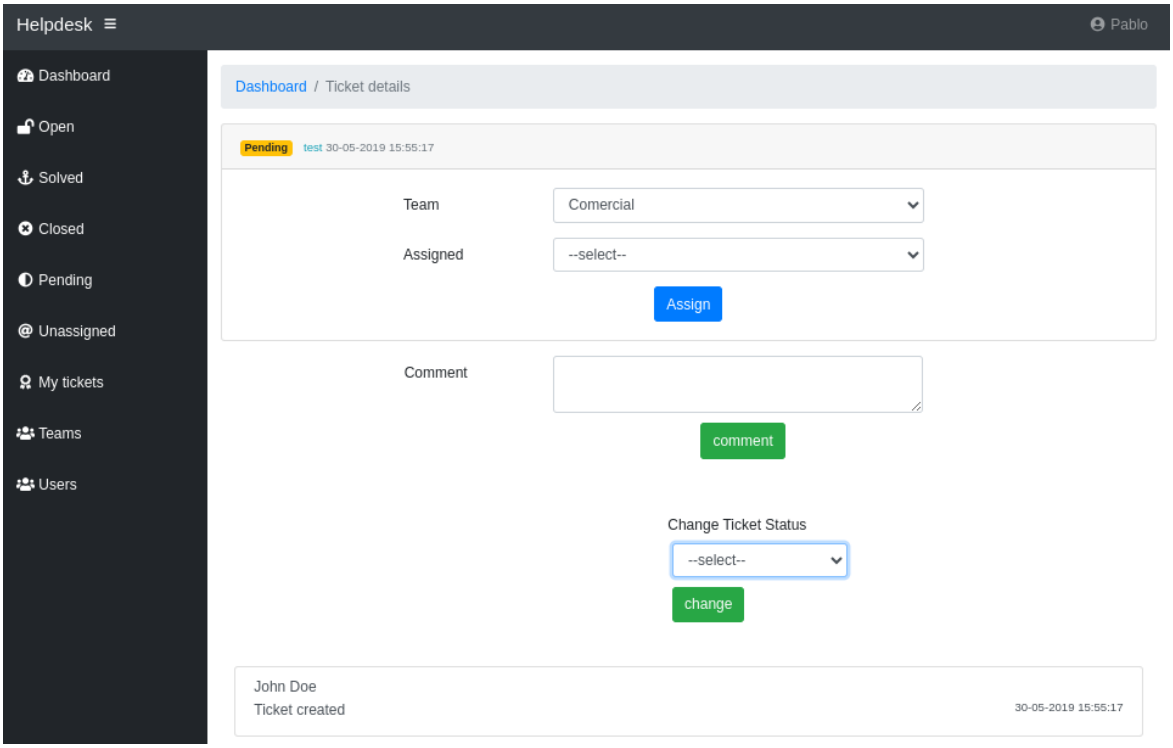
Anexo VII

Uso de la aplicación web de ticketing:






Helpdesk
Pablo
Dashboard
Open
Solved
Closed
Pending
Unassigned
My tickets
Teams
Users
Dashboard / New ticket
Create a new ticket
Name
Enter name
Email
Enter email
Phone
Enter phone number
Subject
Enter subject
Comment
Enter comment
Team
--select--
Priority
Low
Create

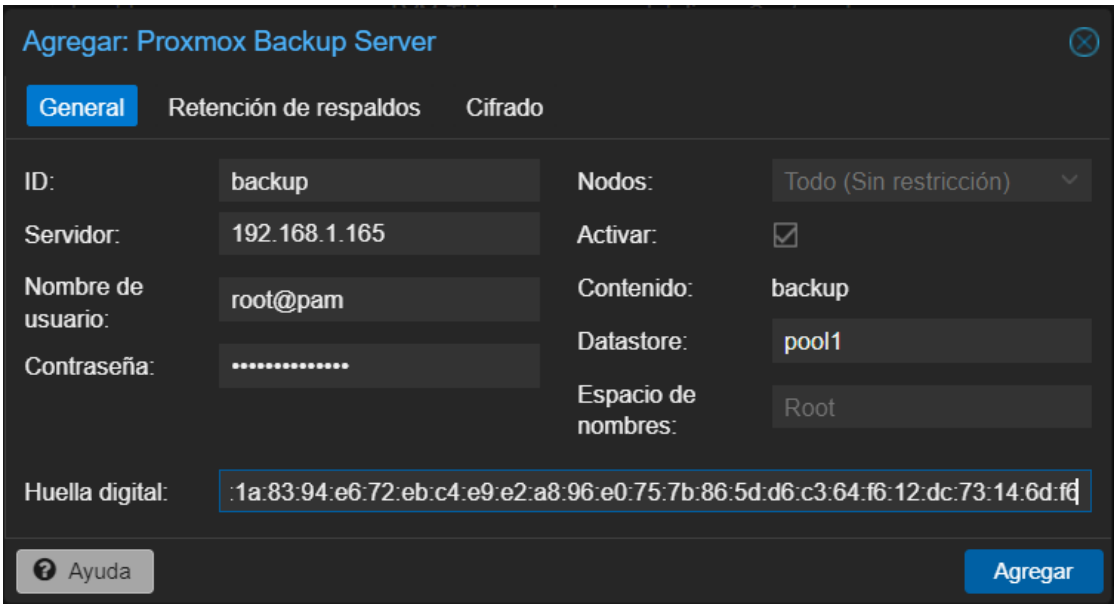

Helpdesk
Pablo
Dashboard
Open
Solved
Closed
Pending
Unassigned
My tickets
Teams
Users
Dashboard / Ticket details
Pending
test 30-05-2019 15:55:17
Team
Comercial
Assigned
--select--
Assign
Comment
comment
Change Ticket Status
--select--
change
John Doe
Ticket created
30-05-2019 15:55:17

Anexo VIII

Instalación y configuración de Proxmox backup Server:

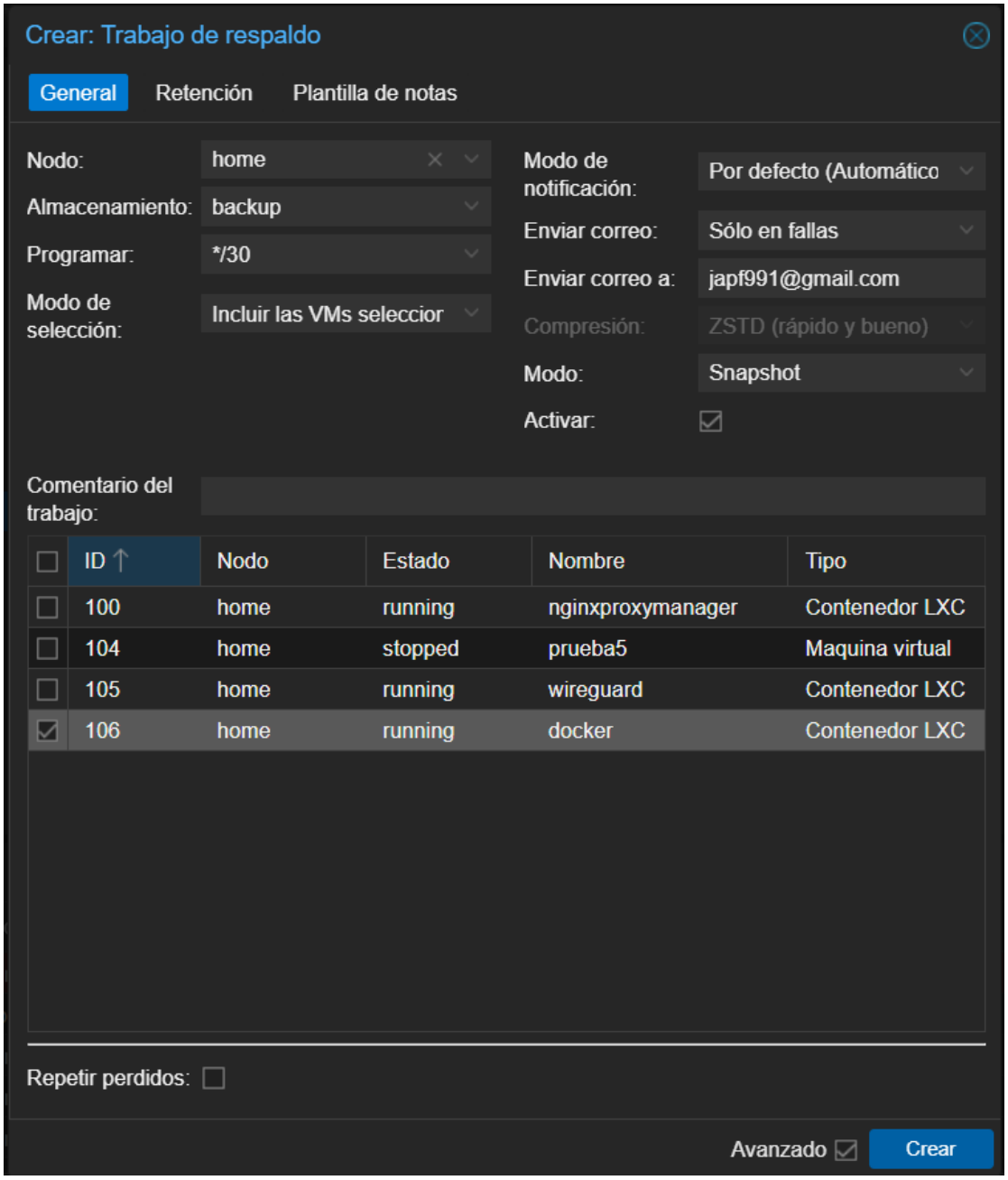
- Instalación de Proxmox Backup Server, la IP asignada al hacer la instalación es 192.168.1.161/24. La cambiamos mas adelante por motivos de disponibilidad a la 192.168.1.165/24 en el archivo nano /etc/network/interfaces.
- Con los 3 discos que tenemos en el servidor, creamos una pool con ZFS, en este caso un raidz. Capacidad de error de 1 disco sin pérdida de información.
- Una vez hecho todo esto nos vamos a nuestro servidor de Proxmox VE, en el apartado de almacenamiento, agregamos un nuevo almacenamiento seleccionando la opción de Proxmox Backup Server.

Y completamos los campos que nos pide:

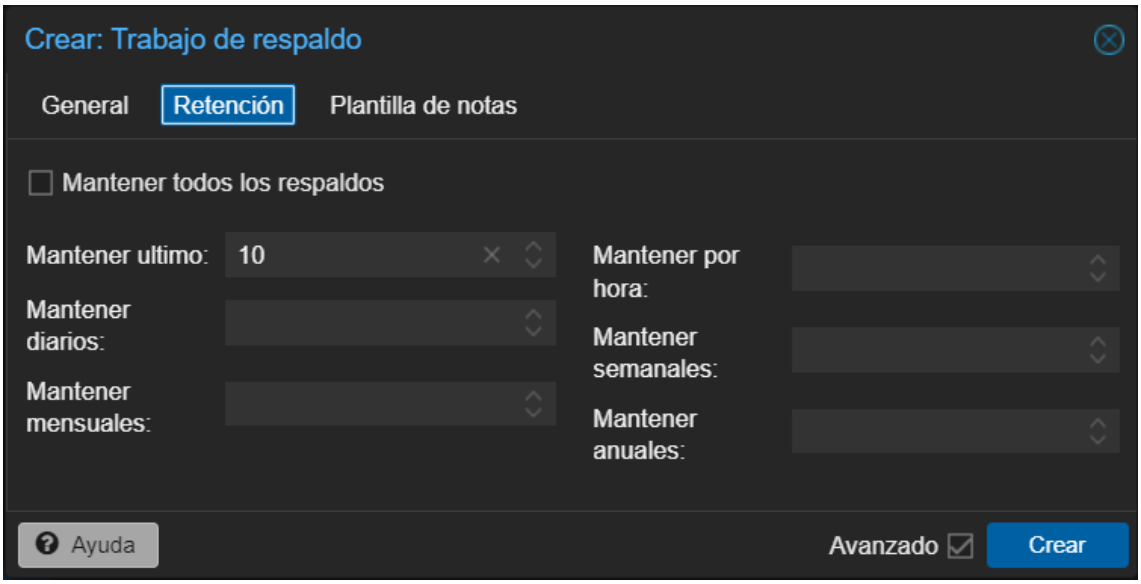


  - Importante, la huella digital se obtiene del servidor Proxmox Backup Server.
- Para configurar los respaldos nos vamos al apartado de respaldos dentro del nodo, agregamos una instrucción nueva seleccionamos el almacenamiento anteriormente añadido al sistema, el tiempo entre copias, el contenedor de

donde hacer la copia de seguridad y el tipo de modo de la copia, en nuestro caso un snapshot para no tener que parar el contenedor en ningún momento.



- En el apartado de retención pondremos las copias de seguridad que se quieren guardar sin borrar.



- Una vez hecho el backup, verificamos que se ha guardado correctamente.