CONFIGURACIÓN SERVIDOR PRINCIPAL

PROCESAMIENTO DE DATOS (DEBIAN 12)

Introducción **3**
Modelo estructural 3
Automatización y despliegue bajo demanda 4
Un enfoque DevOps y de infraestructura como código 4
Especificaciones del Sistema **5**
Configuración de Red y Seguridad **5**
Identidad del sistema y configuración básica de red 5
Configuración del firewall con nftables 6
Mitigación de fuerza bruta con Fail2Ban 8
Gestión de actualizaciones de seguridad 8
Configuración del Servidor **9**
Instalación de software base y utilidades 9
Instalación y configuración de Docker 9
Instalación de Nginx (reverse proxy) 10
Instalación de PHP y módulos necesarios 10
Instalación de MariaDB 10
Instalación de WordPress 12
HTTPS (Let's Encrypt) 14
Servicios auxiliares y seguridad 14
Montaje del Almacenamiento iSCSI **16**
Instalación de Docker y Docker Compose **16**
Script de despliegue automatizado **17**
Explicación paso a paso 22
Plantillas Docker Compose **23**
Plantilla para WordPress 23
Plantilla para Nextcloud 24
Plantilla para Moodle 25
Sistema de sustitución de variables 25
Monitorización y Logs **26**
Supervisión del sistema con Netdata 26
Supervisión de contenedores Docker 26
Logs del sistema y servicios 27
Logs personalizados del sistema Clickdeploy 27
Backups y Mantenimiento **28**
Estrategia de copias de seguridad 28
Actualizaciones del sistema y servicios 28
Verificación proactiva del sistema 28
Conclusión **29**

Introducción

El presente anexo detalla la instalación, configuración y puesta en marcha del servidor principal de Clickdeploy, una plataforma automatizada de hosting de aplicaciones web orientada a usuarios sin conocimientos técnicos o experiencia, empresas pequeñas y medianas y centros educativos. Este servidor está diseñado para proporcionar una solución centralizada, segura y escalable para el despliegue inmediato de entornos autogestionados de WordPress, Moodle y Nextcloud (principalmente), combinando tecnologías modernas como contenedores Docker, orquestación mediante scripts automatizados y una arquitectura distribuida que separa las tareas de cómputo de las de almacenamiento.

El servidor principal desempeña las siguientes tareas:

- Nodo de cómputo y orquestación: ejecuta todos los contenedores Docker que constituyen los servicios de los clientes.

- Sistema de gestión centralizada: aloja el sitio web corporativo, desde donde los clientes pueden contratar servicios, y mantiene una base de datos relacional que registra a cada cliente y sus despliegues activos.

- Motor de automatización: contiene scripts en Bash personalizados que automatizan completamente el ciclo de vida de un servicio — desde el registro del cliente, hasta la configuración de la red, puertos, volúmenes y certificados SSL.

Modelo estructural

La arquitectura diseñada se basa en un modelo de diversificación funcional, donde el servidor principal se centra exclusivamente en la ejecución de cargas de trabajo y la lógica de negocio, mientras que el almacenamiento persistente de todos los contenedores se externaliza mediante una conexión iSCSI al servidor secundario (detallado en el Anexo II).

Esto aporta ventajas clave:

- Escalabilidad horizontal: es posible añadir más servidores de cómputo sin tener que duplicar el almacenamiento.

- Seguridad y separación de datos: se reduce la exposición directa de datos al aislar físicamente el backend, es decir, los datos de los clientes no se encuentran físicamente en el servidor principal.

- Rendimiento optimizado: cada servidor se optimiza para su propósito (alto rendimiento en cómputo para el principal, alta capacidad de almacenamiento y redundancia de datos para el secundario).

Automatización y despliegue bajo demanda

Una de las piezas centrales de la infraestructura es el script de despliegue automatizado, desarrollado en Bash, que permite provisionar en tiempo real servicios personalizados para cada cliente. Este script:

- Inserta un nuevo registro en la base de datos.
- Personaliza dinámicamente una plantilla docker-compose.yml según los parámetros solicitados (tipo de servicio, recursos, dominio, credenciales de acceso).
- Crea volúmenes y estructura de carpetas en el almacenamiento iSCSI.
- Configura Nginx como reverse proxy y genera los certificados SSL con Let's Encrypt.
- Asigna puertos exclusivos a cada servicio y lo arranca como contenedor aislado.

Esta automatización no solo reduce la intervención humana, sino que también garantiza coherencia, repetibilidad y escalabilidad en el modelo de servicio. A mayores, este sistema totalmente automatizado garantiza el funcionamiento y disponibilidad para contratar 24/7.

Un enfoque DevOps y de infraestructura como código

Todo el proceso está diseñado siguiendo buenas prácticas de la filosofía DevOps, incluyendo:

- Despliegues reproducibles mediante docker-compose.
- Control de versiones de las configuraciones y scripts.
- Seguridad totalmente restrictiva por defecto (firewall, Fail2Ban).
- Monitorización activa mediante Netdata (en este caso), y respaldo periódico con herramientas como rsync o snapshots LVM.

Especificaciones del Sistema

El servidor principal está compuesto por un hardware de alto rendimiento, orientado a maximizar el procesamiento y la eficiencia en entornos multiusuario:

- Proveedor: Ionos (154,88 €/mes)
- CPU: AMD EPYC™ 7302P – 16 núcleos / 32 hilos a 3,3 GHz
- RAM: 128 GB DDR4 ECC (corrección de errores)
- Almacenamiento local:
    - 960 GB (2x960 GB NVMe SSD) en RAID 1 hardware
    - Usado para el sistema operativo, el sitio web corporativo y datos de gestión
- Almacenamiento remoto:
    - Montaje iSCSI desde servidor TrueNAS (almacenamiento persistente de contenedores)
- Sistema operativo: Debian 12 Server

Configuración de Red y Seguridad

La configuración de red y la aplicación de medidas de seguridad desde las fases iniciales son fundamentales para garantizar la integridad, disponibilidad y confidencialidad del sistema de cómputo Clickdeploy. En este apartado se describen las acciones realizadas para establecer una configuración de red consistente, implementar un cortafuegos restrictivo, mitigar ataques por fuerza bruta y aplicar actualizaciones automáticas.

Identidad del sistema y configuración básica de red

Durante la instalación del sistema operativo se configuró una interfaz de red con IP estática, de modo que el servidor principal dispone de una dirección ip fija en la red pública, ofrecida por Ionos. Posteriormente se editan los archivos para establecer la identidad de host.

/etc/hostname: contiene el nombre del host (clickdeploy).
/etc/hosts: se actualiza para mapear correctamente 127.0.0.1 a localhost y a clickdeploy.

Este paso es clave para garantizar la correcta resolución de nombres local y evitar problemas con servicios dependientes del hostname como Postfix, SSH o Docker.

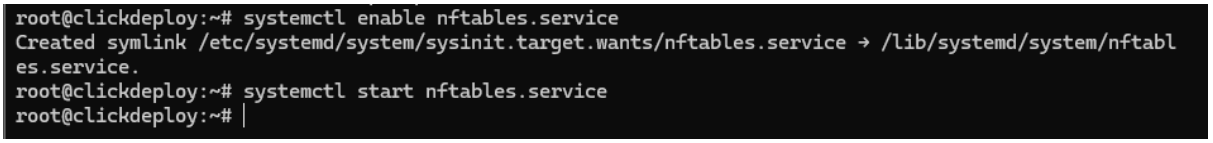
Configuración del firewall con nftables

Como parte de las medidas de protección de red, se ha optado por utilizar nftables, el software moderno para filtrado de paquete, que sustituye progresivamente a iptables y es ahora el backend oficial recomendado en servidores basados en Debian 12.

A diferencia de soluciones como UFW (bastante similar pero más simplificado), nftables ofrece mayor rendimiento y control, permitiendo definir políticas más expresivas, tablas personalizadas y condiciones complejas. Esto resulta especialmente útil en servidores expuestos públicamente y que requieren apertura de puertos para servicios específicos.

Instalación y activación
nftables viene instalado por defecto en Debian 12, en instalaciones más básicas se puede instalar y habilitar con:

```
apt install nftables
systemctl enable nftables
systemctl start nftables
```



Reglas base del cortafuegos

A continuación se detalla una configuración de nftables segura, diseñada específicamente para el servidor principal de este modelo.
Estas reglas:

- Bloquean todo el tráfico entrante por defecto.
- Permiten solo los puertos esenciales (SSH, HTTP, HTTPS).
- Permiten todo el tráfico saliente (para actualizaciones, backups, iSCSI).
- Son compatibles con servicios desplegados en contenedores Docker.

Archivo de configuración base

Editar el archivo **/etc/nftables.conf** con el contenido de la siguiente página.

Una vez guardado el archivo, **aplicar las reglas** con:

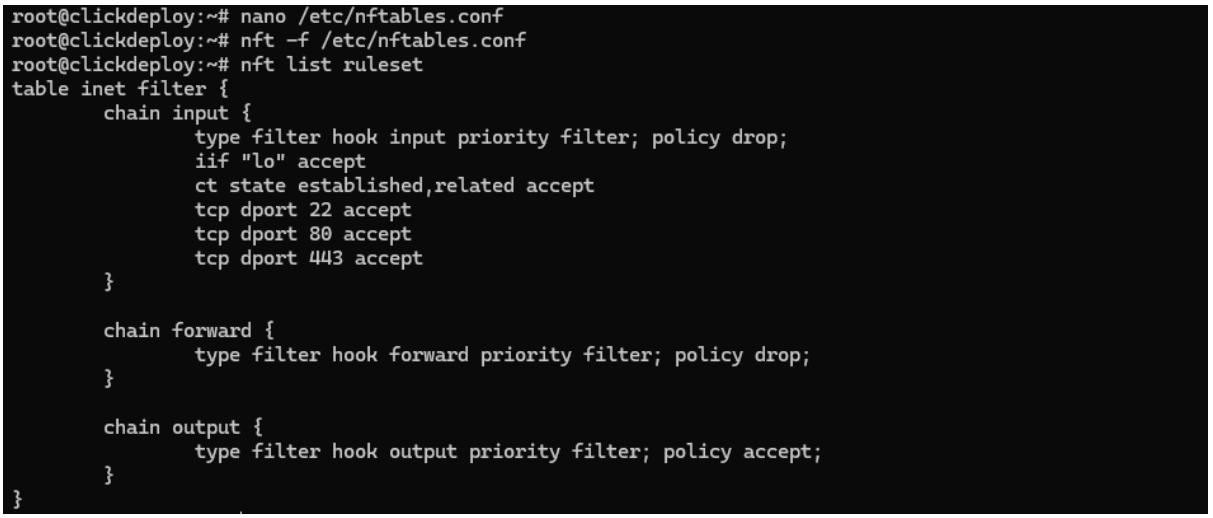
```
nft -f /etc/nftables.conf
```

Verificar el estado con:

```
nft list ruleset
```

```
#!/usr/sbin/nft -f
flush ruleset
table inet filter {
    chain input {
        type filter hook input priority 0;
        policy drop;

        # Aceptar tráfico de loopback
        iif lo accept

        # Permitir conexiones ya establecidas
        ct state established,related accept

        # Permitir SSH
        tcp dport 22 accept

        # Permitir HTTP y HTTPS
        tcp dport 80 accept
        tcp dport 443 accept
    }

    chain forward {
        type filter hook forward priority 0;
        policy drop;
    }

    chain output {
        type filter hook output priority 0;
        policy accept;
    }
}
```

Con esta configuración en nftables, el servidor principal mantiene una postura de seguridad restrictiva por defecto, minimizando la superficie de ataque.



Mitigación de fuerza bruta con Fail2Ban

Para prevenir ataques dirigidos al servicio SSH (ataques de fuerza bruta), se instala Fail2Ban, una herramienta que supervisa los logs del sistema (/var/log/auth.log) y bloquea direcciones IP tras múltiples intentos de acceso fallido.

```
apt install fail2ban
systemctl enable fail2ban --now
```

```
root@clickdeploy:~# systemctl enable fail2ban --now
Synchronizing state of fail2ban.service with SysV service script with /lib/systemd/systemd-sysv-instal
l.
Executing: /lib/systemd/systemd-sysv-install enable fail2ban
root@clickdeploy:~#
```

La configuración predeterminada ya protege el servicio SSH, pero puede personalizarse con filtros más agresivos, notificaciones y listas blancas para evitar bloqueos de IPs legítimas.

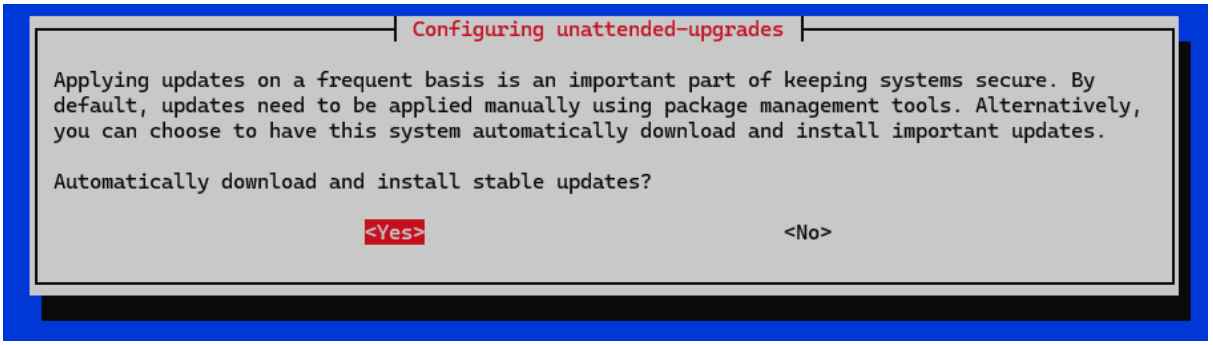
Esto permite defenderse de ataques automatizados comunes, como bots, sin necesidad de intervención manual, bloqueando a nivel de cortafuegos mediante reglas temporales.

Gestión de actualizaciones de seguridad

En un entorno orientado a servicios multiusuario, es importante mantener el sistema operativo y sus dependencias actualizadas frente a vulnerabilidades conocidas. Para ello se instala y activa el paquete unattended-upgrades, que permite aplicar parches de seguridad de forma automática y periódica:

```
apt install unattended-upgrades
dpkg-reconfigure --priority=low unattended-upgrades
```

```
root@clickdeploy:~# dpkg-reconfigure --priority=low unattended-upgrades
root@clickdeploy:~#
```



Este sistema se encarga de descargar e instalar automáticamente actualizaciones críticas del repositorio oficial de Debian, reduciendo la exposición del servidor a amenazas emergentes sin afectar el funcionamiento de los servicios.

Configuración del Servidor

Una vez instalado Debian 12 en el servidor principal, se procede a configurar e instalar todos los servicios y paquetes necesarios para garantizar su funcionalidad como nodo de gestión y despliegue de Clickdeploy. Esta sección cubre la instalación y puesta en marcha de los siguientes componentes clave: Docker, Nginx, PHP, MariaDB, WordPress, Certbot (SSL), Netdata (monitorización), y utilidades auxiliares como git y curl, entre otros.

Actualización general del sistema

Antes de instalar cualquier software, se actualiza el sistema:

```
apt update && apt upgrade -y
```

Instalación de software base y utilidades

Estas herramientas son necesarias para operaciones generales, despliegues y scripts:

```
apt install -y \
  curl wget unzip git gnupg lsb-release ca-certificates \
  software-properties-common nano apt-transport-https
```

Instalación y configuración de Docker

Docker es el motor de contenedores que permite desplegar servicios aislados por cliente:

```
# Repositorio oficial
mkdir -p /etc/apt/keyrings
curl -fsSL https://download.docker.com/linux/debian/gpg | gpg --dearmor -o
/etc/apt/keyrings/docker.gpg

echo "deb [arch=$(dpkg --print-architecture)
signed-by=/etc/apt/keyrings/docker.gpg]
https://download.docker.com/linux/debian $(lsb_release -cs) stable" | tee
/etc/apt/sources.list.d/docker.list

apt update
apt install -y docker-ce docker-ce-cli containerd.io docker-compose-plugin
systemctl enable docker --now
```

```
root@clickdeploy:~# systemctl enable docker --now
Synchronizing state of docker.service with SysV service script with /lib/systemd/systemd-sysv-install.
Executing: /lib/systemd/systemd-sysv-install enable docker
root@clickdeploy:~#
```

Instalación de Nginx (reverse proxy)

Nginx se utiliza como proxy inverso para exponer servicios bajo dominios personalizados:

```
apt install -y nginx
systemctl enable nginx --now
```

```
root@clickdeploy:~# systemctl enable nginx --now
Synchronizing state of nginx.service with SysV service script with /lib/systemd/systemd-sysv-install.
Executing: /lib/systemd/systemd-sysv-install enable nginx
root@clickdeploy:~#
```

El archivo /etc/nginx/sites-available/default se puede dejar desactivado, ya que cada dominio será configurado automáticamente desde el script de automatización.

Instalación de PHP y módulos necesarios

Aunque los servicios de los clientes se despliegan en contenedores, el sitio corporativo está alojado directamente en el servidor, utilizando el CMS Wordpress, que requiere un entorno LAMP, en este caso se utiliza Nginx + MariaDB + PHP.

```
apt install -y php php-fpm php-mysql php-curl \
       php-xml php-gd php-mbstring php-zip php-intl
systemctl enable php8.2-fpm --now
```

```
root@clickdeploy:~# systemctl enable php8.2-fpm --now
Synchronizing state of php8.2-fpm.service with SysV service script with /lib/systemd/systemd-sysv-inst
all.
Executing: /lib/systemd/systemd-sysv-install enable php8.2-fpm
root@clickdeploy:~#
```

Instalación de MariaDB

Para gestionar los datos del sitio corporativo y la base de clientes:

```
apt install -y mariadb-server mariadb-client
systemctl enable mariadb --now
```

```
root@clickdeploy:~# systemctl enable mariadb --now
Synchronizing state of mariadb.service with SysV service script with /lib/systemd/systemd-sysv-install
.
Executing: /lib/systemd/systemd-sysv-install enable mariadb
root@clickdeploy:~#
```

Inicialización segura:

```
mysql_secure_installation
```

Sigue las indicaciones para establecer la contraseña root, eliminar usuarios anónimos, deshabilitar acceso remoto root y borrar la base de datos de test.

```
root@clickdeploy:~# mysql_secure_installation

NOTE: RUNNING ALL PARTS OF THIS SCRIPT IS RECOMMENDED FOR ALL MariaDB
      SERVERS IN PRODUCTION USE!  PLEASE READ EACH STEP CAREFULLY!

In order to log into MariaDB to secure it, we'll need the current
password for the root user. If you've just installed MariaDB, and
haven't set the root password yet, you should just press enter here.

Enter current password for root (enter for none):
OK, successfully used password, moving on...

Setting the root password or using the unix_socket ensures that nobody
can log into the MariaDB root user without the proper authorisation.

You already have your root account protected, so you can safely answer 'n'.

Switch to unix_socket authentication [Y/n] n
 ... skipping.

You already have your root account protected, so you can safely answer 'n'.

Change the root password? [Y/n] y
New password:
Re-enter new password:
Password updated successfully!
Reloading privilege tables..
 ... Success!


By default, a MariaDB installation has an anonymous user, allowing anyone
to log into MariaDB without having to have a user account created for
them.  This is intended only for testing, and to make the installation
go a bit smoother.  You should remove them before moving into a
production environment.

Remove anonymous users? [Y/n] y
 ... Success!

Normally, root should only be allowed to connect from 'localhost'.  This
ensures that someone cannot guess at the root password from the network.

Disallow root login remotely? [Y/n] y
 ... Success!

By default, MariaDB comes with a database named 'test' that anyone can
access.  This is also intended only for testing, and should be removed
before moving into a production environment.
```

```
Remove test database and access to it? [Y/n] y
 - Dropping test database...
 ... Success!
 - Removing privileges on test database...
 ... Success!

Reloading the privilege tables will ensure that all changes made so far
will take effect immediately.

Reload privilege tables now? [Y/n] y
 ... Success!

Cleaning up...

All done!  If you've completed all of the above steps, your MariaDB
installation should now be secure.

Thanks for using MariaDB!
root@clickdeploy:~#
```

Instalación de WordPress

Se desplegará WordPress en /var/www/clickdeploy, gestionado directamente por Nginx:

```
cd /var/www/
wget https://wordpress.org/latest.tar.gz
tar -xzf latest.tar.gz
mv wordpress clickdeploy.dev
chown -R www-data:www-data /var/www/clickdeploy
```

```
root@clickdeploy:/var/www# ls
html  latest.tar.gz  wordpress
root@clickdeploy:/var/www# mkdir clickdeploy
root@clickdeploy:/var/www# mv wordpress/* clickdeploy/
root@clickdeploy:/var/www# chown -R www-data:www-data /var/www/clickdeploy
root@clickdeploy:/var/www#
```

Crear base de datos y usuario para WordPress:

```
mysql -u root -p -e "
CREATE DATABASE clickdeploy;
CREATE USER 'wpuser'@'localhost' IDENTIFIED BY 'Abc123..';
GRANT ALL PRIVILEGES ON clickdeploy.* TO 'wpuser'@'localhost';
FLUSH PRIVILEGES;"
```

```
root@clickdeploy:~# mysql -u root -p -e "
CREATE DATABASE clickdeploy;
CREATE USER 'wp_user'@'localhost' IDENTIFIED BY 'Abc123..';
GRANT ALL PRIVILEGES ON clickdeploy.* TO 'wp_user'@'localhost';
FLUSH PRIVILEGES;"
Enter password:
root@clickdeploy:~#
```

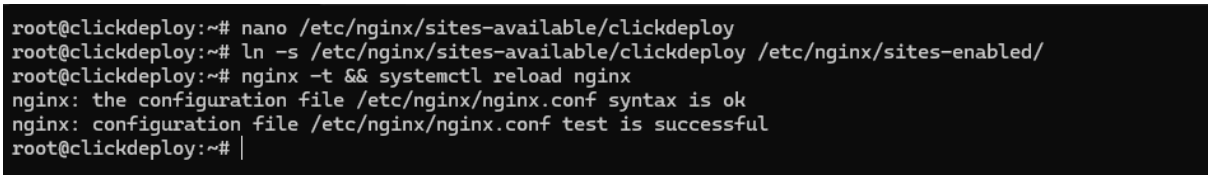
Configurar Nginx para asociar el WordPress al dominio:

EDITAR EL ARCHIVO nano /etc/nginx/sites-available/clickdeploy

```
server {
    listen 80;
    server_name clickdeploy.dev www.clickdeploy.dev;

    root /var/www/clickdeploy.dev;
    index index.php index.html;

    location / {
        try_files \$uri \$uri/ /index.php?\$args;
    }

    location ~ \.php$ {
        include snippets/fastcgi-php.conf;
        fastcgi_pass unix:/run/php/php8.2-fpm.sock;
    }

    location ~ /\.ht {
        deny all;
    }
}
```

Activar y reiniciar Nginx:

```
ln -s /etc/nginx/sites-available/clickdeploy /etc/nginx/sites-enabled/
nginx -t && systemctl reload nginx
```



HTTPS (Let’s Encrypt)

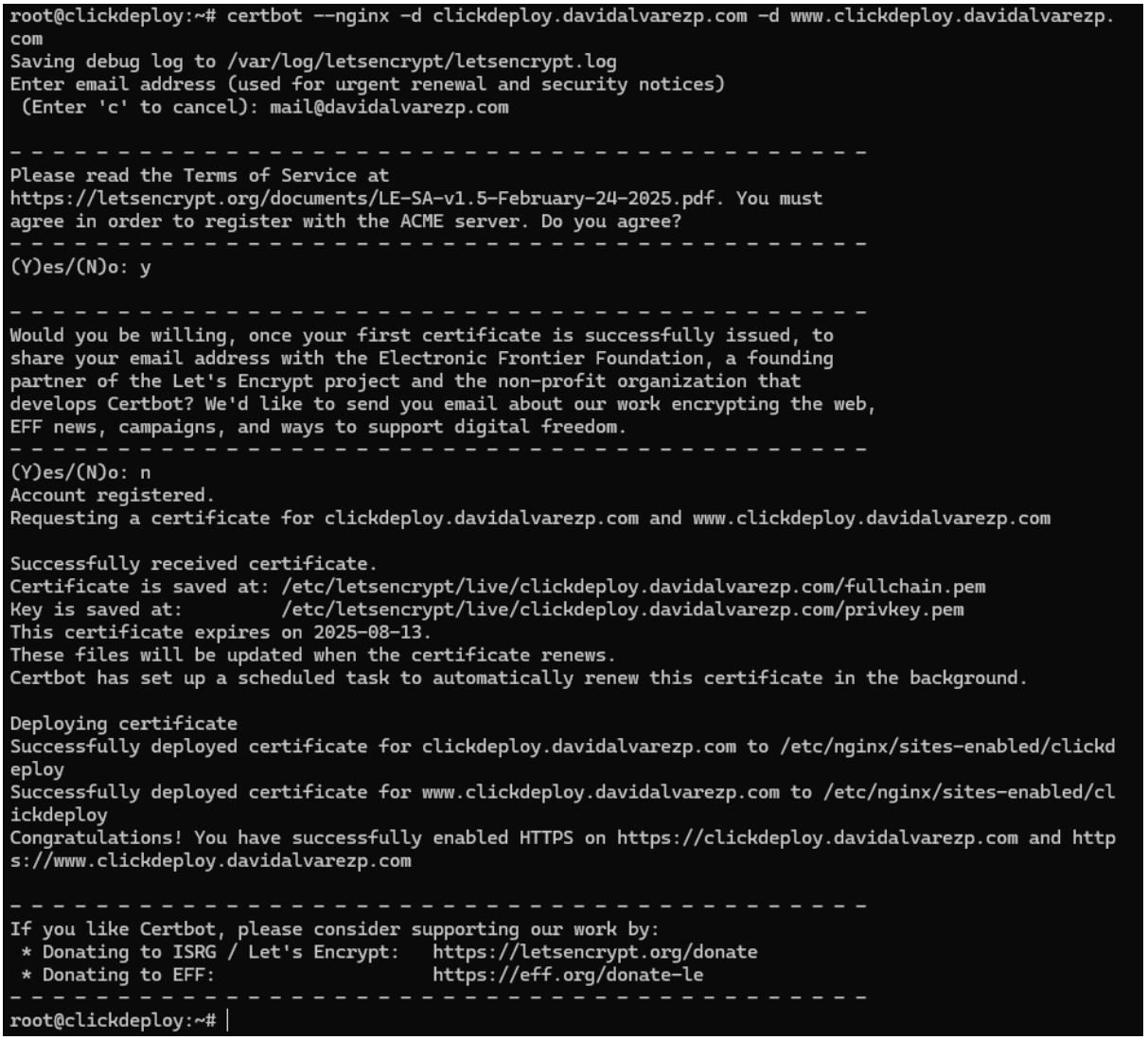
Para proteger todos los sitios con certificados SSL:

```
apt install -y certbot python3-certbot-nginx
```

Obtener certificados SSL:

```
certbot --nginx -d clickdeploy.dev -d www.clickdeploy.dev
```

Esto también funcionará automáticamente para los servicios de clientes, ya que el script de despliegue llama a certbot para cada nuevo dominio, y genera un certificado ssl.



Servicios auxiliares y seguridad

Además de los servicios principales, se instalan:

- Fail2Ban: para protección contra ataques por fuerza bruta.
- nftables: como cortafuegos (detallado en configuración del firewall).
- Unattended-upgrades: para aplicar automáticamente actualizaciones críticas de seguridad.
- Open-iscsi: para conectar con el almacenamiento remoto TrueNAS.

```
apt install -y fail2ban nftables unattended-upgrades open-iscsi
systemctl enable fail2ban nftables
```

Con esta configuración, el servidor queda completamente preparado para:

- Alojar el sitio web corporativo con WordPress.
- Ejecutar despliegues automáticos de contenedores Docker.
- Ofrecer HTTPS en todos los servicios.
- Monitorizar el estado del sistema.
- Protegerse contra amenazas externas.
- Integrarse con almacenamiento remoto y realizar backups periódicos.

Montaje del Almacenamiento iSCSI

Utilizado para almacenar todos los datos de los contenedores docker.

Instalar herramienta:

```
apt install open-iscsi
```

Descubrir el destino iSCSI desde TrueNAS:

```
iscsiadm -m discovery -t sendtargets -p (ip.del.servidor.secundario)
iscsiadm -m node --login
```

Formatear, montar y enlazar el volumen:

```
mkfs.ext4 /dev/sdX
mkdir /mnt/containers
mount /dev/sdX /mnt/containers
echo '/dev/sdX /mnt/containers ext4 _netdev 0 0' >> /etc/fstab
```

Redirigir /var/lib/docker:

```
systemctl stop docker
mv /var/lib/docker /mnt/containers/docker
ln -s /mnt/containers/docker /var/lib/docker
systemctl start docker
```

Instalación de Docker y Docker Compose

Instalación completa de Docker CE y Docker Compose plugin:

```
apt update && apt install -y ca-certificates curl gnupg lsb-release
mkdir -p /etc/apt/keyrings
curl -fsSL https://download.docker.com/linux/debian/gpg | gpg --dearmor -o
/etc/apt/keyrings/docker.gpg
echo "deb [arch=$(dpkg --print-architecture)
signed-by=/etc/apt/keyrings/docker.gpg] https://download.docker.com/linux/debian
$(lsb_release -cs) stable" | tee /etc/apt/sources.list.d/docker.list > /dev/null
apt update && apt install -y docker-ce docker-ce-cli containerd.io
docker-compose-plugin
systemctl enable docker --now
```

Script de despliegue automatizado

1. Comprobación de argumentos

- Comprueba que el script recibe exactamente 7 argumentos.
- Si no es así, muestra un mensaje de uso y termina la ejecución con error.

```
#!/bin/bash

# ——————————————————————————————————————————
# 1. Comprobación de argumentos
# ——————————————————————————————————————————
if [ "$#" -ne 7 ]; then
    echo "Uso: $0 SERVICIO CPU RAM DISCO USUARIO PASSWORD DOMINIO"
    exit 1
fi
```

2. Variables de entrada

- Asigna los argumentos recibidos a variables para facilitar el uso en el resto del script.

```
# ——————————————————————————————————————————
# 2. Variables de entrada
# ——————————————————————————————————————————
SERVICIO=$1
CPU=$2
RAM=$3
DISCO=$4
USUARIO=$5
PASSWORD=$6
DOMINIO=$7
```

3. Configuración general

- Define variables de configuración general:
    - Credenciales de la base de datos.
    - Rutas de trabajo (plantillas, despliegues, logs).

- Crea el directorio de logs (/var/log/clickdeploy) si no existe.

- Crea el archivo de log y asigna permisos restrictivos.

- Define una función log() que permite registrar eventos con fecha y hora en el log.

```
# ——————————————————————————————————————
# 3. Configuración general
# ——————————————————————————————————————
DB_USER="root"
DB_PASS="your_db_root_password"  # Cambiar por la real en producción
BASE_DIR="/opt/clickdeploy"
TEMPLATE_DIR="$BASE_DIR/Servicios"
DEPLOYMENTS_DIR="/mnt/containers/deployments"  # NUEVO PATH para el iSCSI
CLIENT_DEPLOY_DIR="$DEPLOYMENTS_DIR/$USUARIO-$SERVICIO"
LOG_DIR="/var/log/clickdeploy"
LOG_FILE="$LOG_DIR/clickdeploy.log"

mkdir -p "$LOG_DIR"
touch "$LOG_FILE"
chown www-data:www-data "$LOG_FILE"
chmod 640 "$LOG_FILE"

log() {
    echo "$(date '+%Y-%m-%d %H:%M:%S') - $1" >> "$LOG_FILE"
}
```

4. Registro en la base de datos

- Inserta el cliente en la base de datos con los datos recibidos.
- Recupera el CLIENT_ID autogenerado tras la inserción.
- Si falla, registra el error y detiene el script.
- Calcula dos puertos únicos para el cliente sumando su id al valor base.
- Registra en log el usuario y puertos asignados.

```
# ——————————————————————————————————————
# 4. Inserción del cliente en la base de datos
# ——————————————————————————————————————
CLIENT_ID=$(mysql -u "$DB_USER" -p"$DB_PASS" -N -e \
"INSERT INTO clickdeploy.clientes (usuario, servicio, password, dominio)
VALUES ('$USUARIO', '$SERVICIO', '$PASSWORD', '$DOMINIO');
SELECT LAST_INSERT_ID();")

if [ -z "$CLIENT_ID" ]; then
    log "Error al insertar cliente en la base de datos"
    exit 1
fi

WEB_PORT=$((10000 + CLIENT_ID))
DB_PORT=$((30000 + CLIENT_ID))

log "Usuario $USUARIO registrado con ID $CLIENT_ID | Puertos asignados:
WEB=$WEB_PORT, DB=$DB_PORT"
```

5. Preparación del entorno de despliegue

- Crea el directorio de despliegue específico para el cliente.
- Copia los archivos base del servicio solicitado al nuevo directorio.
- Ajusta permisos y propiedad.
- Cambia al directorio de despliegue (si falla, registra y sale con error).

```
# ——————————————————————————————————————————
# 5. Preparación del entorno de despliegue
# ——————————————————————————————————————————
mkdir -p "$CLIENT_DEPLOY_DIR"
cp -r "$TEMPLATE_DIR/$SERVICIO"/* "$CLIENT_DEPLOY_DIR/"
chown -R www-data:www-data "$CLIENT_DEPLOY_DIR"
chmod -R 775 "$CLIENT_DEPLOY_DIR"
cd "$CLIENT_DEPLOY_DIR" || { log "No se pudo acceder al directorio
$CLIENT_DEPLOY_DIR"; exit 1; }
```

6. Generación del archivo .env personalizado

- Crea un archivo .env con las credenciales y datos específicos del cliente.
- Asigna permisos y propietario correctos al archivo para seguridad.

```
# ——————————————————————————————————————————
# 6. Generación del archivo .env personalizado
# ——————————————————————————————————————————
cat > .env <<EOL
WORDPRESS_DB_HOST=db
WORDPRESS_DB_USER=$USUARIO
WORDPRESS_DB_PASSWORD=$PASSWORD
WORDPRESS_DB_NAME=$USUARIO
MYSQL_ROOT_PASSWORD=$PASSWORD
MYSQL_DATABASE=$USUARIO
MYSQL_USER=$USUARIO
MYSQL_PASSWORD=$PASSWORD
EOL

chown www-data:www-data .env
chmod 640 .env
```

7. Sustitución de variables en docker-compose.yml

- Realiza reemplazos en el archivo docker-compose.yml:
- Usuario, CPU, RAM, puertos, disco.
- Permite que cada despliegue tenga configuraciones personalizadas según el cliente.

```
# ——————————————————————————————————————————
# 7. Sustitución de variables en docker-compose.yml
# ——————————————————————————————————————————
sed -i "s|\${USUARIO}|$USUARIO|g" docker-compose.yml
sed -i "s|\${CPU}|$CPU|g" docker-compose.yml
sed -i "s|\${RAM}|${RAM}G|g" docker-compose.yml
sed -i "s|\${WEB_PORT}|$WEB_PORT|g" docker-compose.yml
sed -i "s|\${DB_PORT}|$DB_PORT|g" docker-compose.yml
sed -i "s|\${DISCO}|$DISCO|g" docker-compose.yml
```

8. Despliegue con Docker Compose

- Ejecuta el despliegue de contenedores en modo detached.
- Guarda la salida en el log.
- Si hay error en el despliegue, lo registra y detiene el script.

```
# ——————————————————————————————————————————
# 8. Despliegue con Docker Compose
# ——————————————————————————————————————————
log "Desplegando contenedores con Docker Compose"
docker compose up -d 2>&1 | tee -a "$LOG_FILE"

if [ $? -ne 0 ]; then
    log "Error durante el despliegue con Docker"
    exit 1
fi
```

9. Configuración de Nginx como Reverse Proxy

- Crea una configuración de Nginx específica para el dominio del cliente.
- Asigna el proxy_pass al puerto web asignado.
- Activa la configuración creando el enlace simbólico en sites-enabled.
- Verifica la configuración y recarga Nginx.
- Registra en log el resultado.

```
# ——————————————————————————————————————————
# 9. Configuración de Nginx como Reverse Proxy
# ——————————————————————————————————————————
NGINX_CONF="/etc/nginx/sites-available/$DOMINIO"
NGINX_LINK="/etc/nginx/sites-enabled/$DOMINIO"

cat > "$NGINX_CONF" <<EOF
server {
    listen 80;
    server_name $DOMINIO;
```

```
    location / {
        proxy_pass http://127.0.0.1:$WEB_PORT;
        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;
    }
}
EOF

ln -sf "$NGINX_CONF" "$NGINX_LINK"
nginx -t && systemctl reload nginx

if [ $? -ne 0 ]; then
    log "Error en la configuración de Nginx"
    exit 1
else
    log "Nginx recargado correctamente para $DOMINIO"
fi
```

10. Generación del certificado SSL

- Comprueba si certbot está instalado.
- Si lo está, solicita un certificado SSL automático usando el modo Nginx.
- Si la generación falla o no existe certbot, lo registra en el log.
- Finalmente, registra que el despliegue ha finalizado con éxito.

```
# ——————————————————————————————————————————
# 10. Generación del certificado SSL
# ——————————————————————————————————————————
if command -v certbot &> /dev/null; then
    certbot --nginx --non-interactive --agree-tos --register-unsafely-without-email
-d "$DOMINIO"
    if [ $? -eq 0 ]; then
        log "Certificado SSL generado correctamente para $DOMINIO"
    else
        log "Error al generar certificado SSL"
    fi
else
    log "Certbot no está instalado. SSL omitido."
fi

log "Despliegue finalizado para $USUARIO - $DOMINIO"
exit 0
```

Explicación paso a paso

1. **Comprobación de argumentos:** se aseguran los 7 parámetros obligatorios

2. **Inicialización de variables:** se definen las rutas base del proyecto (/opt/clickdeploy), rutas de plantillas, directorios de despliegue en almacenamiento iSCSI (/mnt/containers/deployments), y archivo de log.

3. **Creación del log:** se asegura que exista un archivo de log con permisos adecuados para www-data.

4. **Inserción en base de datos:** se ejecuta una consulta SQL que registra al nuevo cliente y obtiene su ID único (auto-incremental), usado para asignar puertos personalizados.

5. **Creación del entorno de despliegue:** se copian los archivos del servicio desde su plantilla correspondiente, y se prepara un directorio exclusivo para el cliente en el almacenamiento remoto.

6. **Generación del archivo .env:** se crean las variables de entorno necesarias para el despliegue del servicio, incluyendo credenciales de bases de datos.

7. **Sustitución de variables** en docker-compose.yml: se personaliza el fichero del contenedor para reflejar usuario, recursos, puertos y dominio.

8. **Despliegue con Docker:** se lanza el servicio en segundo plano usando docker compose up -d.

9. **Configuración de Nginx:** se genera dinámicamente un bloque de servidor (server block) para exponer el servicio bajo el dominio del cliente, redirigiendo tráfico HTTP al puerto asignado localmente al contenedor.

10. **Generación de certificado SSL** (Let's Encrypt): se utiliza certbot para emitir un certificado y proteger el servicio con HTTPS.

Plantillas Docker Compose

Cada aplicación ofrecida se basa en una **plantilla docker-compose.yml predefinida**, diseñada para ser modular, segura y fácilmente parametrizable. Estas plantillas son procesadas y adaptadas automáticamente por el script de despliegue, que sustituye las variables necesarias según los datos del cliente: usuario, recursos contratados, puertos, credenciales y dominio.

Las plantillas están ubicadas en el directorio base del sistema:
/opt/clickdeploy/Servicios/…/docker-compose.yml

Plantilla para WordPress

Esta configuración despliega un contenedor de WordPress junto a una base de datos MySQL. Se asignan volúmenes persistentes montados en el servidor secundario.

```
version: '3.7'

services:
  wordpress:
    image: wordpress:latest
    restart: always
    ports:
      - "${WEB_PORT}:80"
    environment:
      WORDPRESS_DB_HOST: db
      WORDPRESS_DB_USER: ${USUARIO}
      WORDPRESS_DB_PASSWORD: ${PASSWORD}
      WORDPRESS_DB_NAME: ${USUARIO}
    volumes:
      - /mnt/containers/${USUARIO}-wordpress/html:/var/www/html
    deploy:
      resources:
        limits:
          cpus: '${CPU}'
          memory: '${RAM}'
  db:
    image: mysql:5.7
    restart: always
    environment:
      MYSQL_DATABASE: ${USUARIO}
      MYSQL_USER: ${USUARIO}
      MYSQL_PASSWORD: ${PASSWORD}
      MYSQL_ROOT_PASSWORD: ${PASSWORD}
    ports:
      - "${DB_PORT}:3306"
    volumes:
      - /mnt/containers/${USUARIO}-wordpress/db:/var/lib/mysql
```

Plantilla para Nextcloud

Este despliegue consta de una aplicación Nextcloud y una base de datos MariaDB, preparada para manejar múltiples sesiones concurrentes y almacenamiento compartido.

```
version: '3'

services:
  db:
    image: mariadb
    restart: always
    command: --transaction-isolation=READ-COMMITTED --log-bin=binlog
--binlog-format=ROW
    environment:
      MYSQL_ROOT_PASSWORD: ${PASSWORD}
      MYSQL_PASSWORD: ${PASSWORD}
      MYSQL_DATABASE: ${USUARIO}
      MYSQL_USER: ${USUARIO}
    volumes:
      - /mnt/containers/${USUARIO}-nextcloud/db:/var/lib/mysql
    ports:
      - "${DB_PORT}:3306"

  app:
    image: nextcloud
    restart: always
    ports:
      - "${WEB_PORT}:80"
    depends_on:
      - db
    environment:
      MYSQL_PASSWORD: ${PASSWORD}
    volumes:
      - /mnt/containers/${USUARIO}-nextcloud/data:/var/www/html
```

Características clave:

- Configuración pensada para la sincronización de archivos.
- Requiere puertos expuestos para web y base de datos (internamente aislados).
- Los volúmenes se montan directamente en el directorio remoto persistente.

Plantilla para Moodle

Moodle requiere un entorno más complejo, por lo que se opta por una imagen de Bitnami preconfigurada, con soporte para HTTPS interno, múltiples servicios y volúmenes.

```
version: '3'

services:
  moodle:
    image: bitnami/moodle:latest
    restart: always
    ports:
      - "${WEB_PORT}:8080"
      - "44${CLIENT_ID}:8443"
    environment:
      - MOODLE_USERNAME=${USUARIO}
      - MOODLE_PASSWORD=${PASSWORD}
      - MOODLE_EMAIL=admin@${DOMINIO}
    volumes:
      - /mnt/containers/${USUARIO}-moodle/data:/bitnami/moodle
      - /mnt/containers/${USUARIO}-moodle/mariadb:/bitnami/mariadb
```

Características clave:

- Utiliza imagen de Bitnami con configuración simplificada.
- Expone puertos HTTP y HTTPS (interno), listos para certbot.
- Permite la creación de entornos de formación completos por cliente.

Sistema de sustitución de variables

Todas las plantillas utilizan variables con el formato **${VARIABLE}**, que son reemplazadas automáticamente en el despliegue por el script de automatización mediante **sed**. Esto permite:

- Uso de plantillas únicas reutilizables.
- Personalización completa para cada cliente.
- Despliegues coherentes y controlados.

Este enfoque modular permite desplegar **instancias aisladas**, **seguras** y **configuradas** a medida para cada cliente, manteniendo una arquitectura limpia, reproducible y **preparada para escalar**.

Monitorización y Logs

La supervisión del servidor principal es esencial para garantizar la continuidad del servicio, detectar anomalías y anticipar problemas de rendimiento o seguridad.

Supervisión del sistema con Netdata

Para monitorizar en tiempo real el sistema y los servicios, se ha desplegado Netdata, una solución ligera pero extremadamente completa que permite observar el estado del servidor con métricas de:

- Uso de CPU, RAM, red y disco.
- Estado de los servicios Docker.
- Carga del sistema.
- Conexiones activas.
- Tiempos de respuesta HTTP.
- Tráfico por contenedor y proceso.

Instalación de Netdata

```
bash <(curl -Ss https://my-netdata.io/kickstart.sh)
```

Una vez instalado, Netdata se inicia como servicio y expone su interfaz por defecto en http://localhost:19999. Se recomienda accede a través de un túnel SSH para evitar la exposición pública.

Configuraciones adicionales recomendadas:

- Definir umbrales de alerta personalizados.
- Configurar notificaciones por correo o Telegram.
- Limitar el almacenamiento histórico de métricas para optimizar rendimiento.

Supervisión de contenedores Docker

La propia herramienta Netdata ofrece integración con Docker para visualizar:

- Contenedores activos.
- Uso de recursos por contenedor.
- Métricas de red y volumen.

Esto permite identificar rápidamente cuellos de botella en servicios individuales, posibles fugas de memoria o saturación de CPU por contenedores específicos.

Logs del sistema y servicios

Los logs del sistema se gestionan con journald y rsyslog, mientras que los contenedores Docker almacenan logs por defecto en formato JSON (/var/lib/docker/containers/*/*.log). Para mantener el sistema limpio y evitar consumo excesivo de disco:

Configuración de Docker Logging:
En el archivo /etc/docker/daemon.json:

```
{
  "log-driver": "json-file",
  "log-opts": {
    "max-size": "10m",
    "max-file": "5"
  }
}
```

Esto limita cada log a 10MB y conserva 5 rotaciones por contenedor.

Logs personalizados del sistema Clickdeploy

El sistema de despliegue automático mantiene su propio archivo de log:

- Ruta: /var/log/clickdeploy/clickdeploy.log
- Propietario: www-data
- Permisos: 640
- Acceso controlado por el script de automatización.

Cada acción del script genera una entrada con timestamp, lo que permite registrar incidentes, errores o tareas programadas.

Backups y Mantenimiento

La planificación de respaldos y tareas de mantenimiento periódicas es un componente clave para el sistema y la persistencia de los datos.

Estrategia de copias de seguridad

Se aplican diferentes mecanismos de respaldo según la naturaleza del dato:

Tipo de dato	Método de backup	Frecuencia	Destino
Datos persistentes de contenedores	rsync desde /mnt/containers/*	Semanal (cron)	Servidor TrueNas
Base de datos de clientes	mysqldump con cifrado opcional	Diario (cron)	almacenamiento remoto
Archivos del sistema Clickdeploy	Snapshots LVM	Semanal	Almacenamiento local y remoto

Actualizaciones del sistema y servicios

Como parte del mantenimiento periódico se ejecutan:

- Actualizaciones del sistema operativo mediante unattended-upgrades.
- Actualización de contenedores mediante docker pull y docker-compose up -d.
- Verificación de estado de certificados SSL y renovación automática con Certbot.

Verificación proactiva del sistema

Una vez al mes, se realiza una auditoría manual de:

- Integridad de backups.
- Espacio en disco (df -h).
- Estado del almacenamiento iSCSI.
- Logs de errores en /var/log/syslog, /var/log/clickdeploy/, Netdata y Nginx.

Con esta estructura de monitorización y mantenimiento en el servidor principal, se garantiza estabilidad operativa a largo plazo, capacidad de recuperación ante posibles incidentes y trazabilidad de las incidencias mediante logs, manteniendo la alta disponibilidad y calidad.

Conclusión

El servidor principal constituye el núcleo operativo de una plataforma automatizada para el despliegue de aplicaciones web, diseñada para ofrecer rendimiento en el cómputo de datos, aislamiento, escalabilidad y seguridad, apoyándose en tecnologías sólidas extendidas en el mercado.. Su arquitectura modular delega el almacenamiento persistente en un servidor secundario (detallado en el Anexo II), lo que mejora el rendimiento, refuerza la seguridad y simplifica desde el mantenimiento hasta la recuperación de datos ante fallos.

Capacidades alcanzadas

- Despliegue automático de contenedores personalizados por cliente.
- Servicios simultáneos: WordPress, Moodle y Nextcloud, aislados y escalables.
- Almacenamiento remoto iSCSI: persistente, redundante y seguro.
- Seguridad avanzada: firewall nftables, Fail2Ban.
- Monitorización en tiempo real y gestión estructurada de logs.
- Backups automatizados, con soporte para copias externas y snapshots LVM.
- Arquitectura modular, preparada para escalar horizontal o verticalmente.
- Proyección futura: interfaz web de gestión, integración con LDAP/OAuth, y posibilidad de migrar a Kubernetes o alguna otra tecnología futura.