RED SUMMA
ALIANZA COLOMBIANA DE INSTITUCIONES PÚBLICAS DE EDUCACIÓN SUPERIOR

INSTALACIÓN Y CONFIGURACIÓN DE SERVIDORES EN NUBE PRIVADA

COMPONENTE: IMPLEMENTACIÓN DE ERP - INSTALACIÓN Y CONFIGURACIÓN DE LA PLATAFORMA TECNOLÓGICA BASE

CONTRATO INTERADMINISTRATIVO No. 565 DE 2026 CELEBRADO ENTRE LA SUPERINTENDENCIA DE TRANSPORTE

Fecha: Octubre de 2025
Elaborado por: Equipo Técnico de Implementación

1. Introducción

El presente documento describe la instalación, configuración y administración de los servidores de aplicación que soportan el sistema **Dozzier ERP**, desplegado en el marco del Contrato Interadministrativo No. 565 de 2026 celebrado entre la Superintendencia de Transporte y Dozzier S.A.

Dozzier ERP es un sistema de planificación de recursos empresariales (ERP) multi-tenant construido sobre una arquitectura containerizada que emplea **Docker Compose** como orquestador de servicios. La plataforma está diseñada para manejar aproximadamente **300 usuarios concurrentes** en condiciones normales de operación.

Este documento constituye la guía técnica de referencia para el equipo de infraestructura y operaciones, detallando las especificaciones de hardware, software, configuración de servicios y procedimientos operativos necesarios para el correcto funcionamiento del sistema.

2. Alcance

Este documento cubre:

- Aprovisionamiento y configuración de tres servidores en la nube
- (Producción, Contingencia y Stage/QA). Instalación y configuración del entorno Docker Compose.
- Configuración de los contenedores de aplicación:
- Nginx, PHP-FPM, MySQL y RabbitMQ. Definición de políticas de red, almacenamiento, respaldo y seguridad.
- Procedimientos de despliegue y monitoreo.

No cubre: configuración detallada del motor de base de datos (documento DZR-INF-002), instalación del sistema operativo base (documento DZR-INF-003) ni la instalación inicial del software Dozzier ERP (documento DZR-INF-004).

3. Arquitectura General de Servidores

La infraestructura de Dozzier ERP se compone de **tres servidores** independientes alojados en la plataforma DigitalOcean, cada uno con un rol específico dentro del ciclo de vida del sistema:

Servidor	Rol	Archivo de Composición	Propósito
Producción	Principal	docker-compose.production.yaml	Operación real del sistema, atención a usuarios finales
Contingencia	Failover	docker-compose.production.yaml	Réplica del entorno productivo para continuidad operativa
Stage / QA	Pruebas	docker-compose.qa.yaml	Validación de cambios previo a despliegue en producción

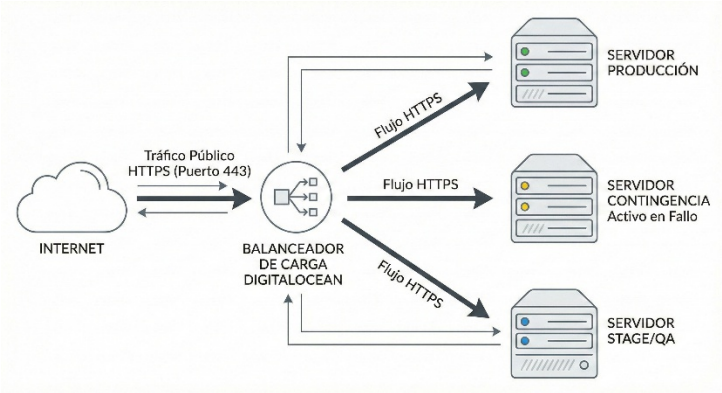
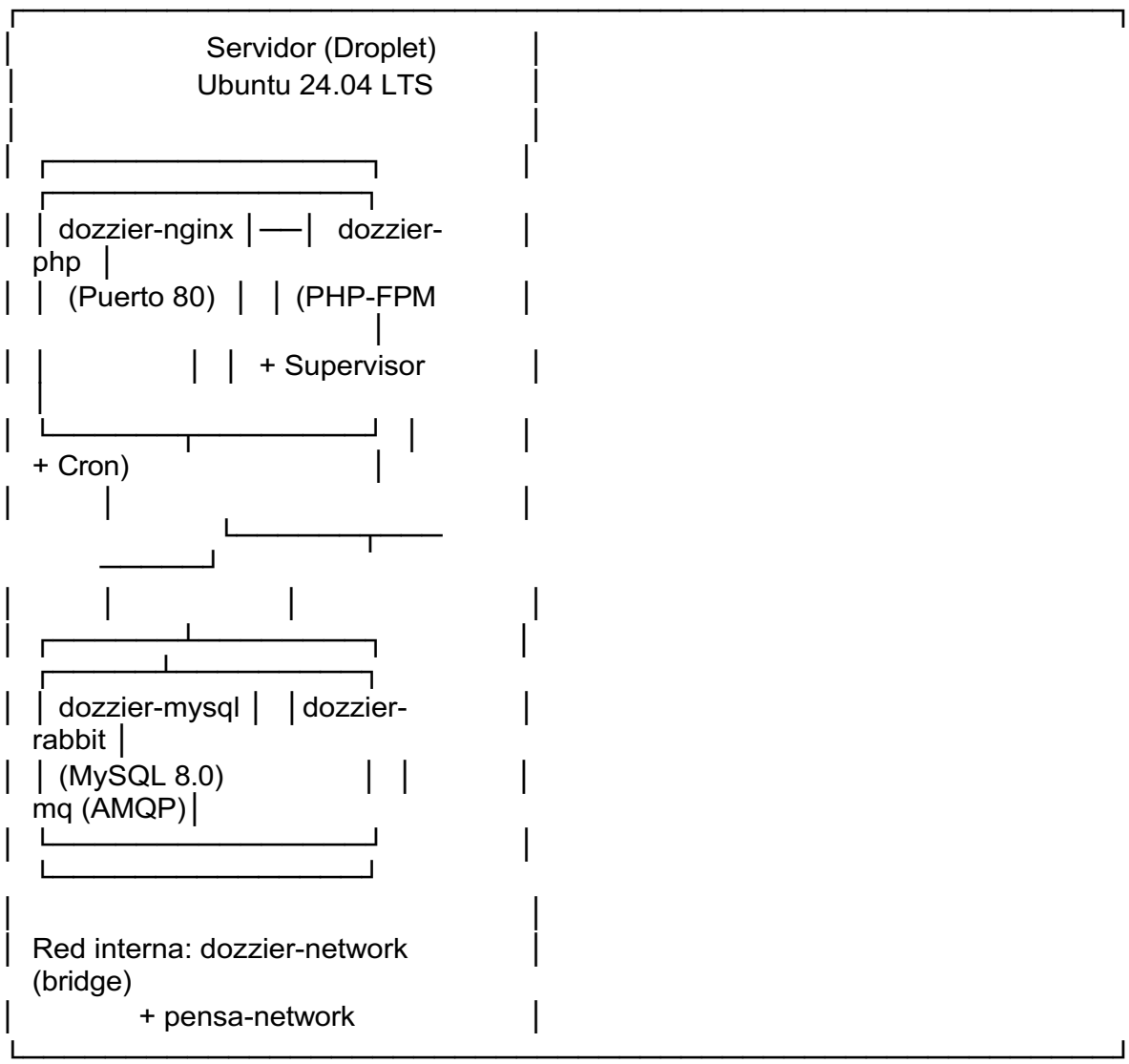


Diagrama lógico de cada servidor

Cada servidor ejecuta la misma pila de contenedores Docker:



4. Proveedor de Infraestructura en la Nube

4.1 DigitalOcean

La infraestructura se despliega sobre **DigitalOcean**, proveedor de servicios en la nube que ofrece máquinas virtuales denominadas *Droplets*, almacenamiento en bloque (*Block Storage Volumes*), redes privadas virtuales (*VPC*) y balanceadores de carga.

4.2 Región y disponibilidad

Parámetro	Valor
Región primaria	NYC3 (New York 3)
Tipo de virtualización	KVM
Red privada (VPC)	Habilitada entre todos los droplets
Monitoreo	DigitalOcean Monitoring habilitado

4.3 Obtención de especificaciones reales del servidor

Para documentar las especificaciones reales de cada servidor, ejecutar los siguientes comandos directamente en cada Droplet:

CPU info

```
root@Dozziererp:~# lscpu | grep -E "^CPU\(s\)|^Model name|^Architecture"
Architecture:                       x86_64
CPU(s):                             8
Model name:                         DO-Premium-Intel
```

RAM

```
root@Dozziererp:~# free -h | head -2
               total        used        free      shared  buff/cache   available
Mem:            15Gi       1.9Gi       3.3Gi        90Mi        10Gi        13Gi
```

Disk

```
root@Dozziererp:~# df -h / && lsblk
Filesystem      Size  Used Avail Use% Mounted on
/dev/vda1       464G   11G  454G   3% /
NAME    MAJ:MIN RM  SIZE RO TYPE MOUNTPOINTS
sda       8:0    0  500G  0 disk /mnt/volume_nyc3_05
vda     253:0    0  480G  0 disk
├─vda1  253:1    0  479G  0 part /
├─vda14 253:14   0    4M  0 part
├─vda15 253:15   0  106M  0 part /boot/efi
└─vda16 259:0    0  913M  0 part /boot
vdb     253:16   0  488K  1 disk
```

OS version

```
root@Dozziererp:~# lsb_release -a
No LSB modules are available.
Distributor ID: Ubuntu
Description:    Ubuntu 24.04.3 LTS
Release:        24.04
Codename:       noble
```

Kernel

```
root@Dozziererp:~# uname -r
6.8.0-71-generic
```

Docker version

```
root@Dozziererp:~# docker --version && docker compose version
Docker version 29.2.1, build a5c7197
Docker Compose version v5.0.2
```

DigitalOcean Droplet metadata (if available)

```
root@Dozziererp:/var/www/html/clientes/      # docker compose --env-file /home/ubuntu/.env ps
NAME                IMAGE                          COMMAND                  SERVICE         CREATED       STATUS                    PORTS
           -mysql-1       mysql:8.0.40                   "docker-entrypoint.s…"   -mysql          24 hours ago   Up 24 hours (healthy)     0.0.0.0:3306->3306/tcp, [::
]:3306->3306/tcp, 33060/tcp
           -nginx-1       nginx:alpine3.20               "/docker-entrypoint.…"   -nginx          24 hours ago   Up 24 hours (unhealthy)   0.0.0.0:8888->80/tcp, [::]:
8888->80/tcp
           ↳-php-1        fjmn2001/      -container:latest   "docker-php-entrypoi…"   -php            24 hours ago   Up 24 hours (healthy)     9000/tcp
           ↳-rabbitmq-1   rabbitmq:3-management          "docker-entrypoint.s…"   -rabbitmq       24 hours ago   Up 24 hours (healthy)     4369/tcp, 5671-5672/tcp, 15
671/tcp, 15691-15692/tcp, 25672/tcp, 0.0.0.0:15672->15672/tcp, [::]:15672->15672/tcp
```

Servidor de Producción

Recurso	Especificación
Tipo de Droplet	[DO-Premium-Intel]
vCPUs	[Por completar]
RAM	[16 gb]
Disco del sistema	[500 gb]
Almacenamiento adicional	[500 gb]
Ancho de banda	[Por completar]
Sistema operativo	[Ubuntu 24.04.3 LTS]

Servidor de Contingencia

Recurso	Especificación
Tipo de Droplet	[DO-Premium-Intel]
vCPUs	[Por completar]
RAM	[16 gb]
Disco del sistema	[500 gb]
Almacenamiento adicional	[500 gb]
Ancho de banda	[Por completar]
Sistema operativo	[Ubuntu 24.04.3 LTS]

Servidor de Stage / QA

Recurso	Especificación
Tipo de Droplet	[DO-Premium-Intel]
vCPUs	[Por completar]
RAM	[16 gb]
Disco del sistema	[500 gb]
Almacenamiento adicional	[500 gb]
Ancho de banda	[Por completar]
Sistema operativo	[Ubuntu 24.04.3 LTS]

5. Servidor de Producción

5.1 Descripción general

El servidor de producción es el entorno principal donde opera Dozzier ERP para los usuarios finales de la Superintendencia de Transporte.
Ejecuta el archivo de composición `docker-compose.production.yaml` que define cuatro servicios containerizados:

1. **dozzier-nginx** — Servidor web y proxy inverso
2. **dozzier-php** — Servidor de aplicación PHP-FPM con Supervisor y Cron
3. **dozzier-mysql** — Motor de base de datos MySQL 8.0.40
4. **dozzier-rabbitmq** — Broker de mensajería RabbitMQ 3

5.2 Prerrequisitos del servidor

Antes de la instalación de los contenedores, el servidor debe contar con:

- Ubuntu 24.04 LTS instalado y actualizado
- Docker Engine 27.x o superior
- Docker Compose v2.x (plugin integrado)
- Acceso SSH configurado con llaves RSA/Ed25519
- Firewall UFW configurado (ver sección 19)
- Volúmenes de DigitalOcean montados (ver sección 15)

5.3 Estructura de directorios en producción

```
/var/www/html/clientes/dozzier/
├── docker-compose.production.yaml
├── .env
├── etc/
│   ├── nginx/prod/default.conf
│   ├── supervisor/supervisord.conf
│   ├── cron/crontab
│   └── php-fpm/www.conf
├── public/
│   └── uploads/     # Archivos de usuarios
├── src/             # Código DDD
├── application/     # Código legado
├── apps/            # Frontend y API
└── ...

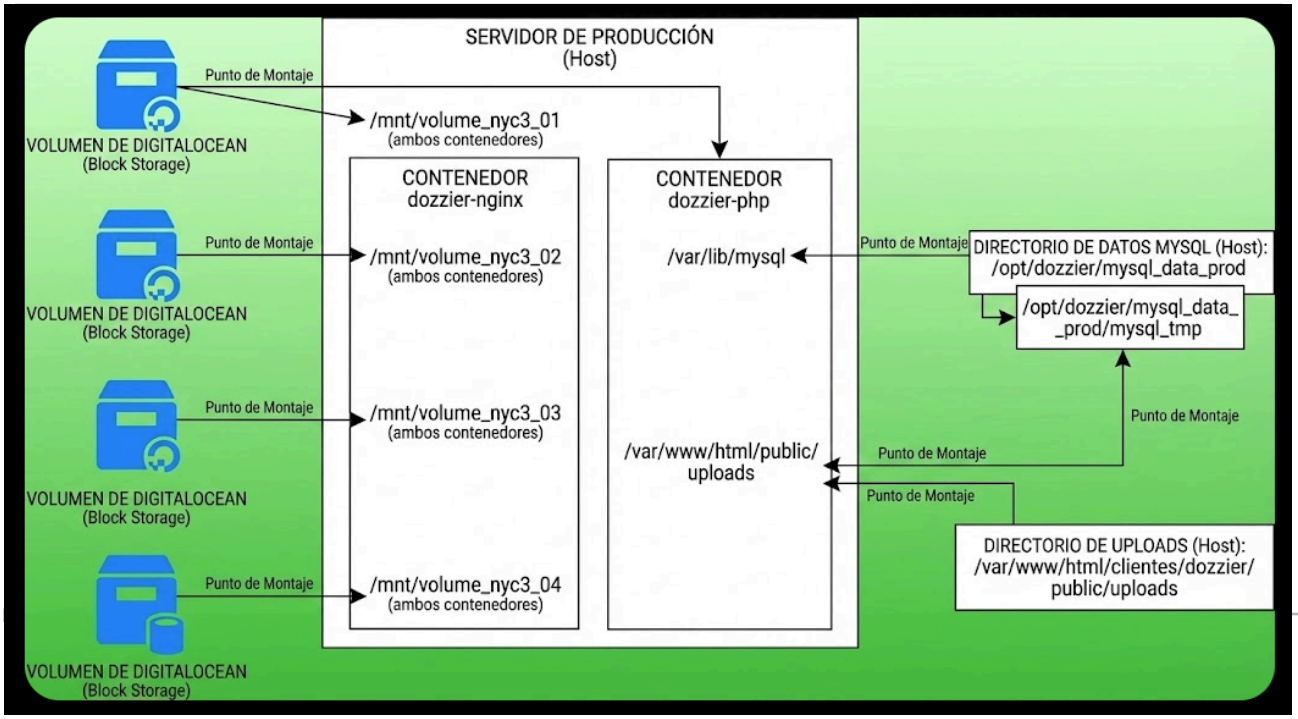
/opt/dozzier/
├── mysql_data_prod/     # Datos persistentes MySQL
│   └── mysql_tmp/
└── backups/         # Respaldos de base de datos

/mnt/
├── volume_nyc3_01/     # Volumen de almacenamiento 1
├── volume_nyc3_02/
├── volume_nyc3_03/
└── volume_nyc3_04/
```

5.4 Variables de entorno

El archivo en producción contiene las variables de configuración sensibles. A continuación se listan las categorías de variables (los valores reales se gestionan de forma confidencial):

Categoría	Variables	Descripción
Base de datos	HOSTDB , DB , USERDB , PASSDB , DOCKER_PASSDB	Credenciales de conexión a MySQL
Aplicación	APP_URL , APP_BRAND , ENC_KEY , ENC_PATCH	URL base, marca (dozzier), clave de cifrado
Mensajería	AMQP_HOST	Host del broker RabbitMQ
Correo SMTP	SMTP_PROTOCOL , SMTP_HOST , _NAME , SMTP_PORT , SMTP_USER , SMTP_PASS , SMTP SMTP_CRYP , SMTP_AUTH	Configuración de correo saliente
Azure AD	AZURE_TENANT_ID , AZURE_CLIENT_ID , AZURE_CLIENT_SECRET , AZURE_*_REDIRECT_URI	Integración con Azure Active Directory para SSO
Monitoreo	SENTRY_DSN , SENTRY_ENVIRONMENT , SENTRY_SEND_PII	Integración con Sentry para rastreo de errores
Sistema	UID , GID	Usuario y grupo del sistema para permisos de archivos



6. Servidor de Contingencia

6.1 Propósito

El servidor de contingencia es una réplica del entorno de producción que garantiza la **continuidad operativa** del sistema ante fallos en el servidor principal. Utiliza el mismo archivo de composición (`docker-compose.production.yaml`) y la misma configuración.

Estrategia de failover

Aspecto	Detalle
Configuración	Idéntica al servidor de producción
Base de datos	Réplica sincronizada mediante mysqldump diario o replicación MySQL
Activación	Manual: redireccionamiento DNS o cambio en el balanceador de carga
RTO estimado	< 30 minutos (activación manual)
RPO estimado	< 24 horas (basado en frecuencia de respaldo)

6.2 Procedimiento de activación

1. Verificar que el respaldo más reciente de la base de datos ha sido restaurado en el servidor de contingencia.
2. Asegurar que el código fuente esté sincronizado con la última versión desplegada en producción.
3. Iniciar los contenedores Docker:

```
cd /var/www/html/clientes/dozzier
docker compose -f docker-compose.production.yaml up -d
```

4. Verificar la salud de los servicios:

```
docker compose -f docker-compose.production.yaml ps
```

5. Redirigir el tráfico actualizando el registro DNS o el balanceador de carga de DigitalOcean.
6. Monitorear los logs durante las primeras horas de operación.

6.3 Sincronización periódica

Para mantener el servidor de contingencia actualizado:

- **Base de datos:** restauración diaria del respaldo nocturno de producción.
- **Código fuente:** sincronización mediante `git pull` o `rsync` desde el repositorio.
- **Volúmenes de almacenamiento:** sincronización periódica de archivos de usuario.

7. Servidor de Stage / QA

7.1 Propósito

El servidor de Stage / QA es el entorno de pruebas donde se validan todas las modificaciones al sistema antes de ser desplegadas en producción. Utiliza el archivo de composición `docker-compose.qa.yaml`.

7.2 Diferencias con producción

Aspecto	Producción	Stage / QA
Archivo de composición	`docker-compose.production.yaml`	`docker-compose.qa.yaml`
Imagen Nginx	`nginx:alpine3.20`	`nginx:stable-alpine`
Imagen MySQL	`mysql:8.0.40` (versión fija)	`mysql:8.0` (tag flotante)
Tuning MySQL	Optimizado (500 conexiones, 1G paquete)	Valores predeterminados
Volúmenes externos	4 volúmenes DigitalOcean	No requeridos
Datos MySQL	`/opt/dozzier/mysql_data_prod`	`/opt/dozzier/mysql_data`
Healthchecks	Habilitados con reinicio automático	No configurados
Política de reinicio	`restart: unless-stopped`	Predeterminada
Nombres de contenedor	Asignados por Docker Compose	Fijos (`dozzier-nginx` , `dozzier-php` , etc.)

7.3 Uso del entorno QA

- Validación de nuevas funcionalidades antes de producción.
- Pruebas de regresión tras actualizaciones del sistema.
- Pruebas de carga y rendimiento en condiciones controladas.
- Capacitación de usuarios en un entorno seguro.
- Reproducción de incidentes reportados en producción.

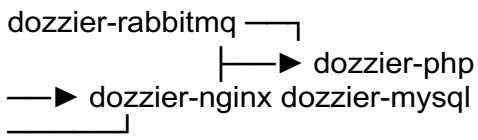
8. Configuración de Contenedores Docker

8.1 Resumen de contenedores

Contenedor	Imagen	Versión	Puerto interno	Puerto expuesto	Función
dozzier-nginx	nginx:alpine3.20	Alpine 3.20	80	8888	Servidor web y proxy inverso
dozzier-php	dozzier-php:latest (imagen personalizada)	PHP 8.2+	9000 (FastCGI)	—	Servidor de aplicación
dozzier-mysql	mysql:8.0.40	8.0.40	3306	3306	Motor de base de datos
dozzier-rabbitmq	rabbitmq:3-management	3.x	5672, 15672	15672	Broker de mensajería

8.2 Dependencias entre contenedores

El orden de inicio se define mediante directivas depends_on:



1. **dozzier-rabbitmq** y **dozzier-mysql** inician primero (sin dependencias).
2. **dozzier-php** inicia después de que RabbitMQ y MySQL están disponibles.
3. **dozzier-nginx** inicia después de que PHP-FPM está disponible.

8.3 Redes Docker

Red	Tipo	Propósito
dozzier-pensa-network	Bridge (interna)	Comunicación entre contenedores de la aplicación
	Externa	Comunicación con servicios compartidos de la plataforma Dozzier

Todos los contenedores están conectados a ambas redes, lo que permite la comunicación interna entre servicios y la interacción con servicios externos de la plataforma.

8.4 Comando de inicio

Para iniciar todos los servicios en producción:

```
cd /var/www/html/clientes/dozzier
docker compose -f docker-compose.production.yaml up -d
```

Para verificar el estado de los contenedores:

```
docker compose -f docker-compose.production.yaml ps
```

Para detener todos los servicios:

```
docker compose -f docker-compose.production.yaml down
```



9. Configuración del Servidor Web (Nginx)

9.1 Rol del servicio

El contenedor `dozzier-nginx` actúa como servidor web frontal y proxy inverso. Recibe las peticiones HTTP en el puerto 80 del contenedor
(expuesto como puerto 8888 en el host) y las redirige al servidor de aplicación PHP-FPM a través del protocolo FastCGI.

9.2 Configuración principal

El archivo de configuración se monta como volumen de solo lectura desde etc/nginx/prod/default.conf .

Parámetros generales

Parámetro	Valor	Descripción
listen	80	Puerto de escucha interno del contenedor
root	/var/www/html	Directorio raíz de la aplicación
index	index.php index.html	Archivos de índice
client_max_body_size	30 MB	Tamaño máximo de carga de archivos
client_body_timeout	3600s	Tiempo máximo para recibir el cuerpo de la petición

Timeouts de proxy

Parámetro	Valor
proxy_connect_timeout	300s
proxy_send_timeout	3600s
proxy_read_timeout	3600s

Timeouts de FastCGI

Parámetro	Valor
fastcgi_read_timeout	3600s
fastcgi_send_timeout	3600s
fastcgi_connect_timeout	300s

Buffers de FastCGI

Parámetro	Valor
fastcgi_buffers	16 x 32 KB
fastcgi_buffer_size	64 KB
fastcgi_busy_buffers_size	64 KB
fastcgi_temp_file_write_size	64 KB

9.3 Cabeceras de seguridad

Nginx inyecta cabeceras de seguridad en todas las respuestas para prevenir ataques de clickjacking y controlar el embebido de frames:

Cabecera	Valor	Propósito
X-Frame-Options	SAMEORIGIN	Permite embebido solo desde el mismo dominio

9.4 Reglas de bloqueo

El servidor web bloquea el acceso a recursos sensibles:

Patrón	Acción	Propósito
*.7z , *.old , *.bak , *.backup , *.tmp	HTTP 404	Impedir descarga de archivos de respaldo
backup	HTTP 404	Bloquear rutas con nombre de respaldo
/.git/*	Deny + 404	Impedir acceso al repositorio Git

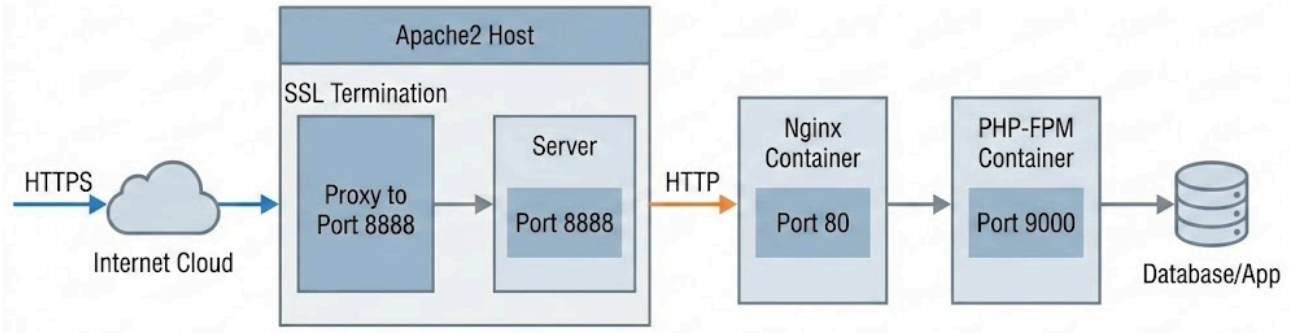
9.5 Caché de archivos estáticos

Los archivos estáticos (PDF, imágenes, CSS, JS) se sirven con un encabezado de expiración de **365 días**, lo que permite una caché agresiva en el navegador del usuario y reduce la carga en el servidor.

9.6 Proxy inverso externo

En el entorno de producción, **Apache2** actúa como proxy inverso a nivel del host, gestionando la terminación SSL y redirigiendo el tráfico al puerto 8888 del servidor donde escucha el contenedor dozzier-nginx :

Internet → Apache2 (:443 HTTPS, SSL Termination)
↓ ProxyPass /clientes/dozzier → localhost:8888
→ dozzier-nginx:80
↓ (FastCGI)
→ dozzier-php:9000



10. Configuración del Servidor de Aplicación (PHP-FPM)

10.1 Imagen Docker

El servidor de aplicación utiliza la imagen personalizada dozzier-php:latest , que incluye:

- PHP 8.2+ con
- extensiones requeridas
- PHP-FPM (FastCGI Process Manager) Supervisor (gestor de procesos)
- Cron (programador de tareas)
- Composer (gestor de
- dependencias PHP) Herramientas de línea de comandos necesarias

10.2 Parámetros PHP

Los siguientes parámetros PHP se configuran a través de Nginx mediante directivas fastcgi_param PHP_VALUE :

Parámetro	Valor	Descripción
upload_max_filesize	20 MB	Tamaño máximo de archivos individuales
post_max_size	30 MB	Tamaño máximo del cuerpo de la petición POST
max_input_vars	100 000	Número máximo de variables de entrada
max_execution_time	3600s	Tiempo máximo de ejecución de scripts
max_input_time	3600s	Tiempo máximo para parsear datos de entrada
memory_limit	2048 MB	Memoria máxima por proceso PHP
allow_url_fopen	On	Permite abrir URLs como archivos

10.3 PHP-FPM Pool

La configuración del pool de PHP-FPM se monta desde `etc/php-fpm/www.conf` y define los parámetros de gestión de procesos worker que atienden las peticiones FastCGI.

10.4 Permisos de archivo

El contenedor PHP se ejecuta con un usuario y grupo específicos definidos por las variables y del archivo .env . Esto garantiza que

UID GID

los archivos creados por la aplicación tengan los permisos correctos tanto dentro del contenedor como en los volúmenes montados del host.

10.5 Volúmenes montados (producción)

Ruta del host	Ruta del contenedor	Propósito
. (código fuente)	/var/www/html	Código de la aplicación
etc/php-fpm/www.conf	/usr/local/etc/php-fpm.d/zz-www.co nf	Configuración de PHP-FPM
/var/www/html/clientes/dozzier/public/uploads	/var/www/html/public/uploads	Archivos subidos por usuarios
/mnt/volume_nyc3_01 a _04	/mnt/volume_nyc3_01 a _04	Volúmenes de almacenamiento externo
etc/cron/crontab	/etc/crontabs/root	Configuración de Cron

11. Configuración del Motor de Mensajería (RabbitMQ)

11.1 Rol del servicio

El contenedor dozzier-rabbitmq implementa un broker de mensajería basado en el protocolo AMQP (Advanced Message Queuing Protocol). Este servicio permite la ejecución asíncrona de tareas pesadas, desacoplando procesos como facturación, cobros y despachos del flujo principal de la aplicación.

11.2 Imagen y versión

Parámetro	Valor
Imagen	rabbitmq:3-management
Versión	RabbitMQ 3.x con plugin de administración
Protocolo	AMQP 0-9-1

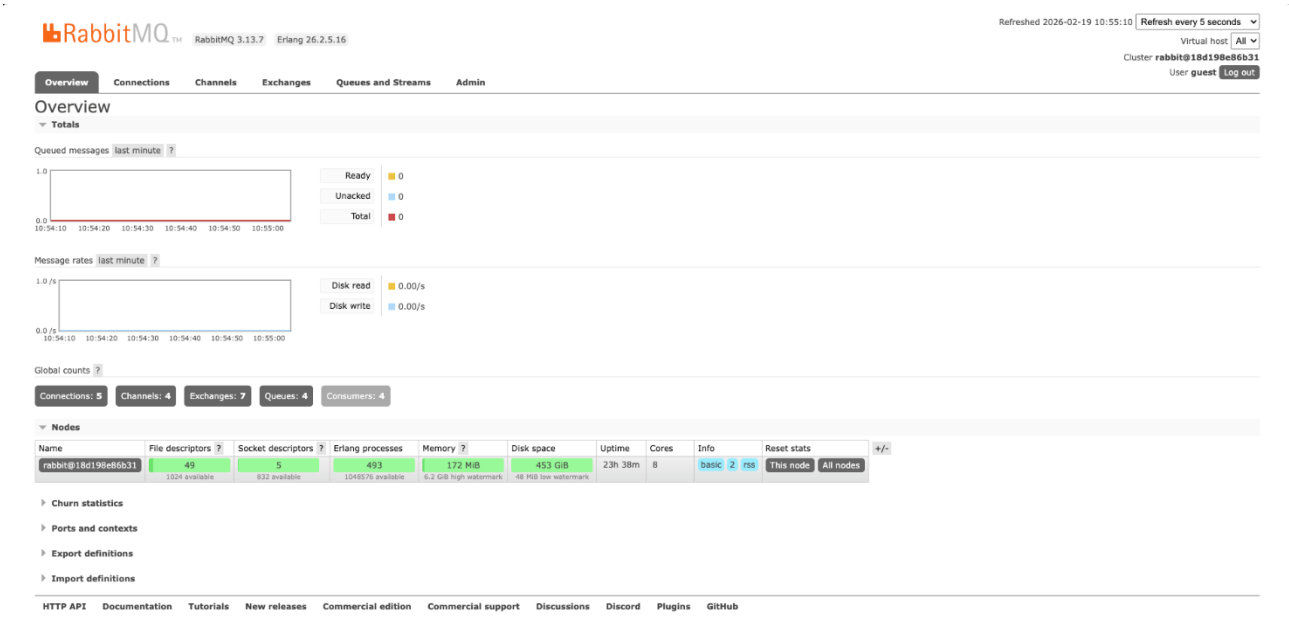
11.3 Puertos

Puerto	Protocolo	Propósito
5672	AMQP	Comunicación de mensajería (solo red interna)
15672	HTTP	Interfaz web de administración

11.4 Interfaz de administración

RabbitMQ incluye una interfaz web de administración accesible en el puerto **15672**. Esta interfaz permite:

Monitorear colas, intercambios y conexiones activas. Visualizar tasas de publicación y consumo de mensajes. Gestionar usuarios y permisos.
Inspeccionar mensajes en cola.



11.5 Colas de procesamiento

El sistema utiliza las siguientes colas de mensajería para el procesamiento asíncrono:

Cola	Descripción
consumer_cobros	Procesamiento de operaciones de cobros
consumer_despachos	Procesamiento de operaciones de despachos
consumer_facturas	Procesamiento de operaciones de facturación
`consumer_sancionados_reverse_scan`	Escaneo inverso de registros de sancionados

11.6 Healthcheck

En producción, RabbitMQ cuenta con una verificación de salud configurada:

```
healthcheck:
  test: ["CMD", "rabbitmqctl",
  "status"] interval: 30s
  timeout:
  10s
  retries: 3
```

12. Gestión de Procesos con Supervisor

12.1 Rol de Supervisor

Supervisor es un sistema de control de procesos para sistemas operativos tipo UNIX. En el contexto de Dozzier ERP, Supervisor ejecuta dentro del contenedor `dozzier-php` y gestiona múltiples procesos de manera simultánea:

12.2 Procesos gestionados

Programa	Comando	Reinicio automático	Logs
php-fpm	php-fpm	Sí	/var/log/php-fpm.log
consumer_cobros	php /var/www/html/index.php jobs consumer_cobros	Sí	/var/log/consumer_cobros.log
consumer_despachos	php /var/www/html/index.php jobs consumer_despachos	Sí	/var/log/consumer_despachos.log
consumer_facturas	php /var/www/html/index.php jobs consumer_facturas	Sí	/var/log/consumer_facturas.log
consumer_sancionados_reverse_scan	php /var/www/html/index.php jobs consumer_sancionados_reverse_scan	Sí	/var/log/consumer_sancionados_reverse_

12.3 Parámetros de resiliencia

Cada proceso consumidor está configurado con:

- `startsecs=10`: El proceso debe ejecutarse al menos 10 segundos para considerarse iniciado exitosamente.
- `startretries=10`: Hasta 10 intentos de reinicio antes de marcarse como fallido.
- `autorestart=true`: Reinicio automático ante cualquier terminación inesperada.

12.4 Administración de Supervisor

Para verificar el estado de los procesos:

```
docker compose -f docker-compose.production.yaml exec dozzier-php supervisorctl status
```

Para reiniciar un proceso específico:

```
docker compose -f docker-compose.production.yaml exec dozzier-php supervisorctl restart consumer_facturas
```

Para reiniciar todos los procesos:

```
docker compose -f docker-compose.production.yaml exec dozzier-php supervisorctl restart all
```

13. Tareas Programadas (Cron)

13.1 Configuración

El sistema de tareas programadas (Cron) se ejecuta dentro del contenedor `dozzier-php`. La tabla de tareas se monta desde `etc/cron/crontab` al contenedor.

13.2 Tareas configuradas

Frecuencia	Comando	Propósito
Cada minuto	`cd /var/www/html/ && php vendor/bin/crunz schedule:run tasks`	Ejecutar el programador de tareas Crunz
Cada minuto	`cd /var/www/html/ && chmod 777 public/logs/*.log`	Asegurar permisos de escritura en logs
Cada minuto	`echo "CRON OK $(date)"`	Verificación de funcionamiento del Cron

13.3 Crunz Scheduler

Crunz es un framework de programación de tareas para PHP que permite definir tareas programadas directamente en código PHP. El comando `crunz schedule:run tasks` evalúa cada minuto si hay tareas definidas en el directorio `tasks/` que deban ejecutarse.

14. Configuración de Red y Puertos

14.1 Mapa de puertos

Puertos expuestos en el host

Puerto del host	Puerto del contenedor	Servicio	Acceso
8888	80	dozzier-nginx	Proxy inverso / Balanceador de carga
3306	3306	dozzier-mysql	Acceso directo a BD (restringir por firewall)
15672	15672	dozzier-rabbitmq	Interfaz de administración (restringir por firewall)

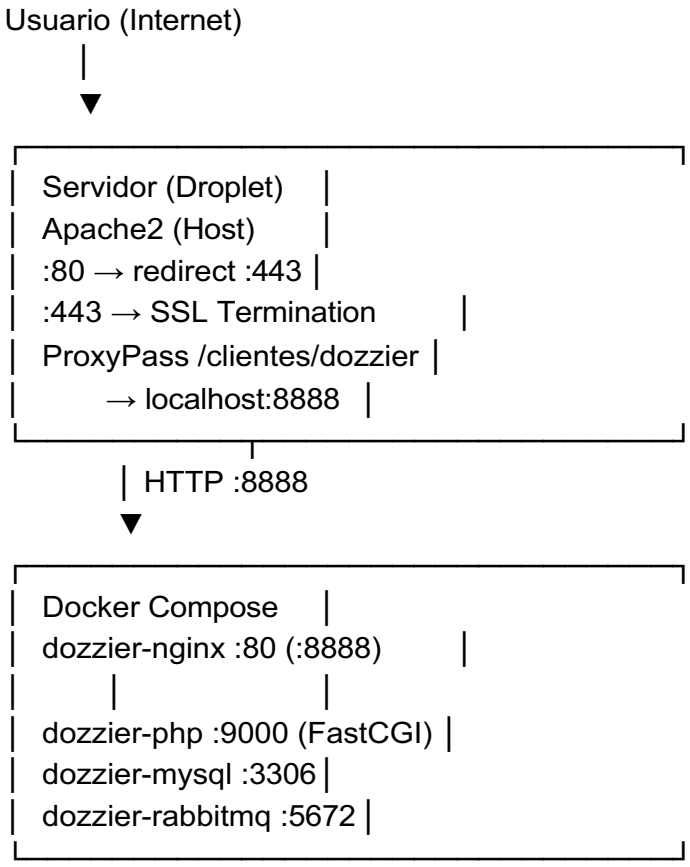
Puertos internos (solo red Docker)

Puerto	Servicio	Protocolo
9000	dozzier-php	FastCGI
5672	dozzier-rabbitmq	AMQP

14.2 Puertos externos (balanceador / proxy)

Puerto externo	Protocolo	Destino
80	HTTP	Redirige a 443 (HTTPS)
443		

14.3 Diagrama de flujo de red



Configuración de Apache2 como proxy inverso

Apache2 se configura en el host como proxy inverso para gestionar la terminación SSL y redirigir el tráfico al entorno Docker:

```
# /etc/apache2/sites-available/dozzier.conf
<VirtualHost *:443>
    ServerName dozzier.example.com
```

```
    SSLEngine on
    SSLCertificateFile
    /path/to/cert.pem
    SSLCertificateKeyFile
    /path/to/key.pem

    ProxyPreserveHost On
    RequestHeader set X-Forwarded-Proto
    "https" RequestHeader set X-Forwarded-
    For "%{REMOTE_ADDR}s"

    ProxyPass /clientes/dozzier
    http://localhost:8888 ProxyPassReverse
    /clientes/dozzier http://localhost:8888
</VirtualHost>

<VirtualHost *:80>
    ServerName dozzier.example.com
    Redirect permanent / https://dozzier.example.com/clientes/dozzier
</VirtualHost>
```

15. Almacenamiento y Volúmenes

15.1 Volúmenes de DigitalOcean (producción)

El dimensionamiento de los volúmenes de almacenamiento depende directamente de la **tasa de carga de archivos por parte de los usuarios** (documentos, imágenes, adjuntos). A mayor volumen de cargas, mayor capacidad de almacenamiento requerida. Se recomienda monitorear el uso periódicamente y escalar los volúmenes según la demanda.

El servidor de producción utiliza cuatro volúmenes de Block Storage de DigitalOcean para el almacenamiento de archivos de usuarios y documentos generados por el sistema:

Volumen	Punto de montaje	Propósito
volume_nyc3_01	/mnt/volume_nyc3_01	Almacenamiento de documentos primario
volume_nyc3_02	/mnt/volume_nyc3_02	Almacenamiento de documentos secundario
volume_nyc3_03	/mnt/volume_nyc3_03	Almacenamiento de archivos adjuntos
volume_nyc3_04	/mnt/volume_nyc3_04	Almacenamiento adicional / respaldos

Estos volúmenes se montan en `dozzier-nginx` y `dozzier-php` para que el servidor web pueda servir los archivos y la aplicación pueda leerlos y escribirlos.

15.2 Almacenamiento de datos MySQL

Entorno	Ruta del host	Ruta del contenedor
Producción	/opt/dozzier/mysql_data_prod	/var/lib/mysql
Stage / QA	/opt/dozzier/mysql_data	/var/lib/mysql

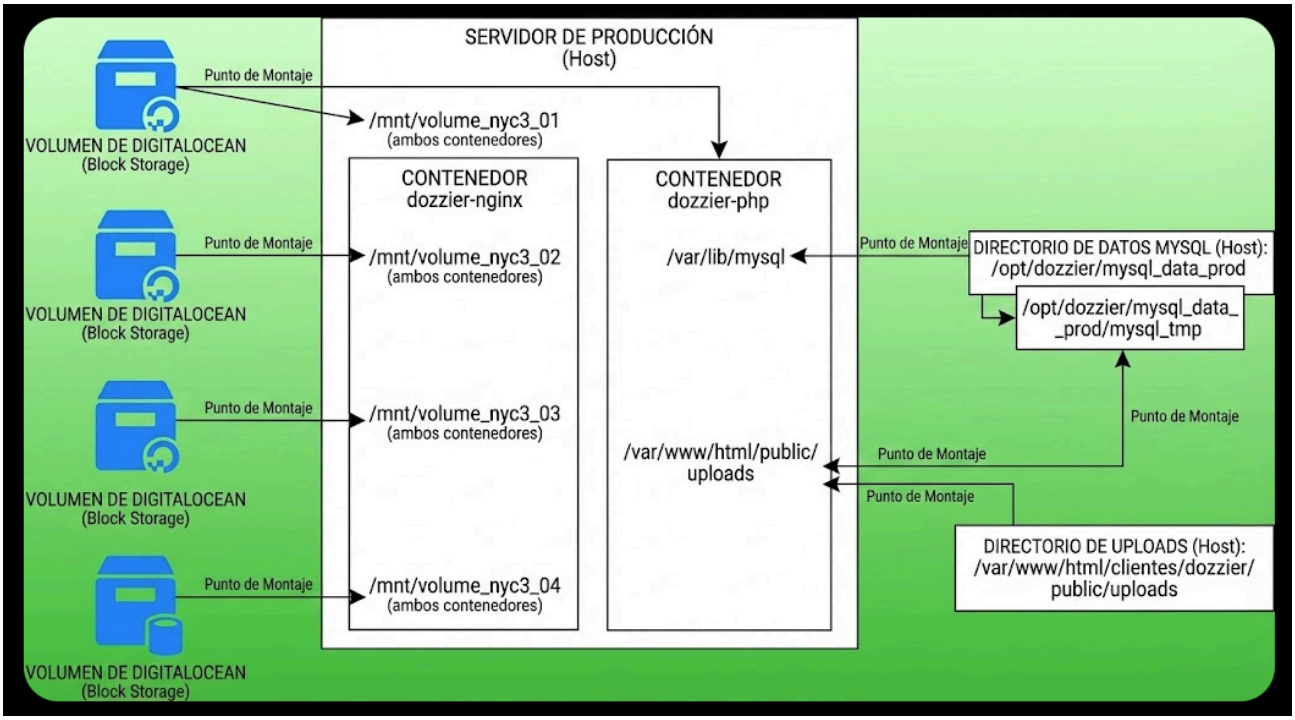
El directorio temporal de MySQL se monta por separado en ordenamiento y unión que requieren archivos temporales. /opt/dozzier/mysql_data_prod/mysql_tmp para optimizar las operaciones de

15.3 Almacenamiento de archivos de usuario

Los archivos subidos por los usuarios (documentos, imágenes, adjuntos) se almacenan en:

/var/www/html/clientes/dozzier/public/uploads → /var/www/html/public/uploads (dentro del contenedor)

Este directorio se monta como un volumen para persistir los datos fuera del contenedor.



16. Esquema de Respaldos

16.1 Estrategia general

El esquema de respaldos de Dozzier ERP se basa en tres pilares:

1. **Respaldo de base de datos** — Exportación diaria mediante.
2. **Respaldo de archivos** — Sincronización de volúmenes de almacenamiento.
3. **Redundancia de servidores** — Servidor de contingencia como réplica.

16.2 Respaldo de base de datos

Parámetro	Valor
Herramienta	`mysqldump`
Frecuencia	Diaria (nocturna)
Hora estimada	02:00 AM (hora local Colombia, UTC-5)
Destino	Volumen de almacenamiento separado
Retención	30 días (rotación automática)
Formato	SQL comprimido (`.sql.gz`)

Comando de respaldo

```
docker compose -f docker-compose.production.yaml
  exec dozzier-mysql \ mysqldump -u root -
  p"${DOCKER_PASSDB}" \
  --single-transaction --routines --triggers --events \
  dozzier | gzip > /mnt/volume_nyc3_04/backups/dozzier_$(date
  +%Y%m%d_%H%M%S).sql.gz
```

16.3 Retención y rotación

Tipo	Retención	Almacenamiento
Diario	30 días	Volumen DigitalOcean
Semanal	12 semanas	Volumen DigitalOcean
Mensual	12 meses	Volumen DigitalOcean (archivado)

16.4 Verificación de respaldos

Se recomienda realizar una prueba de restauración mensual en el servidor de Stage/QA para verificar la integridad de los respaldos:

```
# Descomprimir y restaurar en QA
gunzip < /ruta/al/respaldo/dozzier_YYYYMMDD.sql.gz | \
  docker compose -f docker-compose.qa.yaml
  exec -T dozzier-mysql \ mysql -u root -
  p"${DOCKER_PASSDB}" dozzier
```

17. Balanceo de Carga y Escalabilidad

17.1 Balanceador de carga de DigitalOcean

Para el entorno de producción, se utiliza (o se puede activar) un **DigitalOcean Load Balancer** que distribuye el tráfico HTTPS entrante hacia el servidor de producción:

Parámetro	Valor
Tipo	DigitalOcean Load Balancer
Protocolo de entrada	HTTPS (puerto 443)
Protocolo de salida	HTTP (puerto 8888)
Terminación SSL	En el balanceador
Certificado SSL	Let's Encrypt (gestionado por DigitalOcean)
Healthcheck	HTTP GET en puerto 8888, ruta /
Algoritmo	Round Robin

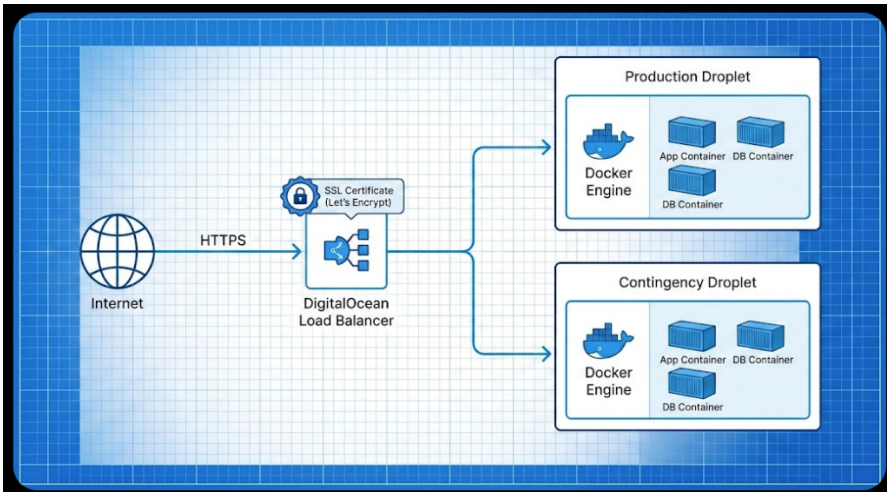
17.2 Escalabilidad horizontal

La arquitectura containerizada permite escalar horizontalmente añadiendo servidores adicionales al pool del balanceador de carga. El procedimiento consiste en:

1. Aprovisionar un nuevo Droplet con la misma configuración.
2. Desplegar la aplicación con `docker-compose.production.yaml` .
3. Configurar la conexión a la base de datos centralizada.
4. Agregar el Droplet al balanceador de carga.

17.3 Capacidad estimada

Métrica	Valor estimado
Usuarios concurrentes por servidor	~300
Peticiones por segundo	~150 (estimado)
Tiempo de respuesta promedio	< 500ms



18. Monitoreo y Salud de Servicios

18.1 Healthchecks de Docker

En producción, cada contenedor cuenta con una verificación de salud (healthcheck) que Docker evalúa periódicamente:

Contenedor	Comando de verificación	Intervalo	Timeout	Reintentos
dozzier-nginx	wget --spider -q http://localhost	60s	10s	3
dozzier-php	php -v	60s	10s	3
dozzier-mysql	mysqladmin ping -h localhost	60s	10s	3
dozzier-rabbitmq	rabbitmqctl status	30s	10s	3

18.2 Política de reinicio

Todos los contenedores en producción están configurados con la política `restart: unless-stopped`, lo que garantiza que Docker reiniciará automáticamente cualquier contenedor que termine de forma inesperada, excepto si fue detenido manualmente.

18.3 Integración con Sentry

La aplicación está integrada con **Sentry** para el monitoreo de errores en tiempo real. Las variables de configuración incluyen:

- `SENTRY_DSN` — Endpoint de reporte de errores.
- `SENTRY_ENVIRONMENT` — Identificador del entorno (producción, QA).
- `SENTRY_SEND_PII` — Control de envío de información personal identificable.

18.4 Monitoreo de DigitalOcean

Se recomienda habilitar **DigitalOcean Monitoring** en todos los Droplets para obtener métricas de:

- Uso de CPU, memoria y disco.
- Tráfico de red entrante y saliente.
- Alertas configurables por umbral.

18.5 Logs de la aplicación

Tipo de log	Ubicación	Descripción
PHP / Aplicación	/var/www/html/public/logs/log-YYYY-MM-DD.log	Logs de la aplicación PHP
Nginx acceso	/var/log/nginx/access.log (dentro del contenedor)	Peticiones HTTP recibidas
Nginx errores	/var/log/nginx/error.log (dentro del contenedor)	Errores del servidor web
Cron	/var/log/cron.log (dentro del contenedor)	Errores de tareas programadas
PHP-FPM	/var/log/php-fpm.log (dentro del contenedor)	Logs del proceso PHP-FPM
Consumidores	/var/log/consumer_*.log (dentro del contenedor)	Logs de procesos asíncronos

19. Consideraciones de Seguridad

19.1 Firewall (UFW)

Se recomienda configurar el firewall UFW en cada servidor para limitar el acceso a los puertos expuestos:

```
# Permitir
SSH ufw
allow 22/tcp

# Permitir tráfico web (desde balanceador)
ufw allow from <IP_BALANCEADOR> to any port 8888

# Permitir MySQL solo desde IPs autorizadas
ufw allow from <IP_CONTINGENCIA> to any port 3306

# Denegar todo lo demás
por defecto ufw default
deny incoming
ufw default allow outgoing# Activar firewall ufw enable
```

19.2 Acceso SSH

Parámetro	Recomendación
Autenticación	Solo por llave pública (deshabilitar contraseña)
Puerto	22 (considerar puerto no estándar)
Protocolo	SSH-2
Root login	Deshabilitar (PermitRootLogin no)
Usuarios autorizados	Definir con AllowUsers

19.3 Seguridad de la aplicación

Medida	Implementación
Cabeceras anti-clickjacking	`X-Frame-Options: SAMEORIGIN`
Content Security Policy	`frame-ancestors 'self'`
Bloqueo de archivos sensibles	Reglas Nginx para `.git` , `*.bak` , `*.backup`
Cifrado de datos	Clave de cifrado vía variable `ENC_KEY`
Autenticación SSO	Integración con Azure Active Directory
Variables sensibles	Almacenadas en `.env` fuera del repositorio

19.4 Seguridad de base de datos

- El puerto 3306 debe estar restringido por firewall solo a IPs autorizadas.
- La contraseña root de MySQL se gestiona mediante la variable de entorno `DOCKER_PASSPR` .
- La autenticación utiliza el plugin `mysql_native_password` .

19.5 Actualizaciones de seguridad

Se recomienda establecer un calendario de actualizaciones:

Componente	Frecuencia	Procedimiento
Sistema operativo (Ubuntu)	Mensual	`apt update && apt upgrade`
Imágenes Docker	Según disponibilidad	Actualizar tags en compose, reconstruir
Dependencias PHP	Según necesidad	`composer update` en entorno QA primero
Certificados SSL	Automático (Let's Encrypt)	Renovación automática cada 90 días

20. Procedimiento de Despliegue

20.1 Despliegue en producción

El procedimiento estándar de despliegue en producción es:

1. **Validar en QA**: Asegurar que los cambios han sido probados y aprobados en el servidor de Stage/QA.
2. **Conectar al servidor**:

   ssh usuario@<IP_PRODUCCION>

3. **Actualizar código fuente**:

   cd
   /var/www/html/clientes/d
   ozzier git pull origin
   master

4. **Actualizar dependencias** (si es necesario):

   docker compose -f docker-compose.production.yaml exec dozzier-php composer install --no-dev --optimize-autoloader

5. **Ejecutar migraciones** (si es necesario):

   docker compose -f docker-compose.production.yaml exec dozzier-php php artisan migrate

6. **Limpiar caché**:

```
docker compose -f docker-compose.production.yaml exec dozzier-php rm -rf apps/dozzier/backend/var/cache/dev
```

7. **Reiniciar consumidores**:

```
docker compose -f docker-compose.production.yaml exec dozzier-php supervisorctl restart all
```

8. **Verificar servicios**:

```
docker compose -f docker-compose.production.yaml ps
```

20.2 Rollback

En caso de que un despliegue presente problemas:

1. Revertir el código al commit anterior:

```
git checkout <commit_anterior>
```

2. Reiniciar los servicios:

```
docker compose -f docker-compose.production.yaml restart
```

3. Verificar la operación normal del sistema.
4. Documentar el incidente y analizar la causa raíz.