

Despliegue y Configuración de un Servicio Web Autoalojado

Caso de estudio: Immich – Biblioteca de fotos y videos en la nube propia

Informe de instalación y configuración

Alumnos: Franco Marengo, Roldan Joaquin

Materia: Laboratorio de Sistemas Operativos y Redes

Fecha: 16/07/2026

Docente: Jose Luis Di Biase

Índice

1. Descripción del proyecto.....	4
2. Relevamiento de características de la máquina utilizada.....	4
3. Relevamiento de sistema operativo y software utilizado y/o necesario.....	5
3.1 Sistema operativo.....	5
3.2 Software necesario.....	5
4. Instalación y prueba del software.....	5
5. Instructivo paso a paso.....	6
5.1 Despliegue con Docker Compose.....	6
Paso 1 — Instalar Docker y el plugin de Docker Compose.....	6
Paso 2 — Crear el directorio del proyecto.....	6
Paso 3 — Descargar los archivos de configuración.....	6
Paso 4 — Configurar los archivos descargados.....	6
Paso 4.1 — Configurar las variables de entorno (.env).....	6
Paso 4.2 — Editar el docker-compose.yml.....	6
Paso 5 — Levantar los contenedores.....	7
Paso 6 — Verificar el estado de los contenedores.....	7
Paso 7 — Acceder a la interfaz web y crear el usuario administrador.....	7
Paso 8 — Prueba funcional.....	7
5.2 Despliegue bare metal sin Docker (Ubuntu VM).....	7
Paso 1 — Crear la VM y preparar el sistema base.....	7
Paso 2 — Instalar Node.js.....	7
Paso 3 — Instalar y configurar PostgreSQL con extensiones vectoriales.....	8
Paso 4 — Instalar Redis y FFmpeg.....	8
Paso 5 — Instalar Immich mediante immich-native.....	8
Paso 6 — Configurar el archivo de entorno.....	8
Paso 7 — Configurar el servicio systemd.....	9
Paso 8 — Configurar Apache como reverse proxy.....	9
Paso 9 — Verificación y prueba funcional.....	9
6. Requisitos adicionales solicitados.....	9
6.1 Storage template.....	9
6.2 Backup.....	9
6.2.1 Base de datos.....	9
6.2.2 Archivos (Filesystem).....	10
6.2.3 Backup móvil.....	10
6.3 Biblioteca externa.....	10
6.3.1 Crear la carpeta y agregar fotos.....	10
6.3.2 Montar la carpeta en el contenedor.....	10
6.3.3 Configurar la librería en la web.....	10
6.4 Monitoreo.....	11

6.5 Configuración de notificaciones por email en Immich usando SMTP de Gmail.....	12
6.5.1 Requisito previo en Google.....	12
6.5.2 Configuración del SMTP en Immich.....	12
6.5.3 Eventos que disparan notificaciones por email.....	13
6.5.4 Plantillas de email.....	13
6.6 Reverse proxy.....	13
7. Problemas surgidos y cómo se resolvieron.....	14
7.1 PostgreSQL 18 incompatible con VectorChord.....	14
7.2 Fallo de build por falta de espacio en /tmp.....	14
7.3 Immich sólo escuchaba en 127.0.0.1 pese a IMMICH_HOST=0.0.0.0.....	14
7.4 Errores de permisos en carpetas.....	14
8. Uso de Inteligencia Artificial.....	14
9. Conclusión.....	15

1. Descripción del proyecto

El presente trabajo tiene como objetivo levantar y configurar, de forma autónoma, un servicio web ya desarrollado, cumpliendo con la consigna de instalación, puesta en marcha y documentación del proceso. Para ello se seleccionó **Immich**, una plataforma de código abierto autoalojada (self-hosted) para el respaldo, organización y visualización de fotos y videos, pensada como alternativa privada a servicios de nube comercial como Google Photos o iCloud.

Immich permite realizar copias de seguridad automáticas desde dispositivos móviles, cuenta con reconocimiento facial, búsqueda semántica basada en inteligencia artificial, generación automática de álbumes y una interfaz web moderna, todo ejecutándose sobre infraestructura propia y sin depender de terceros para el almacenamiento de los datos.

El alcance de este proyecto no incluyó el desarrollo del software (que ya existe y es mantenido por la comunidad de Immich), sino las siguientes tareas a cargo de los alumnos:

- Relevar las características del hardware disponible para el despliegue.
- Relevar el sistema operativo y el software de base necesario para ejecutar Immich.
- Instalar las dependencias necesarias (Docker y Docker Compose) y desplegar Immich mediante contenedores.
- Verificar el correcto funcionamiento del servicio, probando el acceso web y la carga de contenido.
- Documentar el proceso paso a paso, junto con los problemas encontrados y su resolución.

2. Relevamiento de características de la máquina utilizada

Antes de instalar el servicio se relevaron las características físicas/virtuales del equipo sobre el que se realizó el despliegue, dado que Immich ejecuta varios servicios en simultáneo (servidor de aplicación, base de datos, caché y un contenedor de machine learning para reconocimiento facial y búsqueda semántica), por lo que tiene requerimientos de hardware no triviales.

Tipo de equipo	Maquina virtual
Modelo / Proveedor	VM en Virtual Box
Procesador (CPU)	Intel Core I5-5300U, 2 nucleos asignados a la VM
Memoria RAM	4GB asignados a la VM
Almacenamiento	16GB asignados a la VM
Arquitectura	x86_64

Como referencia, la documentación oficial de Immich recomienda un mínimo de 6 GB de RAM y 2 núcleos de CPU, teniendo en cuenta la posibilidad de agregar funcionalidades de machine learning. Debido a que esto último se omite en nuestra implementación es que se permite ser más conservador con los requerimientos otorgados a la VM.

<https://docs.immich.app/install/requirements>

3. Relevamiento de sistema operativo y software utilizado y/o necesario

3.1 Sistema operativo

La documentación oficial de Immich recomienda fuertemente el uso de un sistema operativo Linux (o una máquina virtual Linux) para el despliegue mediante Docker, señalando que los sistemas operativos distintos de Linux (por ejemplo Windows sin WSL2) ofrecen una experiencia de Docker más pobre y están desaconsejados para producción. En caso de necesitar usar Windows, se recomienda hacerlo a través de Docker Desktop o de WSL2.

Sistema operativo utilizado	Debian 13 Trixie
Modo de instalación	Maquina virtual

3.2 Software necesario — entorno Docker

Para desplegar Immich se identificó como necesario el siguiente software de base:

- **Docker Engine:** motor de contenedores sobre el cual se ejecutan todos los servicios de Immich.
- **Docker Compose (plugin):** Immich requiere específicamente el comando *docker compose* (con espacio, plugin integrado), no el binario clásico *docker-compose* (con guion), que está deprecado y ya no es soportado.
- **Navegador web:** para acceder a la interfaz de administración y usuario de Immich (puerto 2283 por defecto).
- **Editor de texto / terminal:** para editar los archivos de configuración *docker-compose.yml* y *.env*.

Immich, al levantarse, despliega automáticamente cuatro contenedores/servicios: immich-server (aplicación principal y API), immich-machine-learning (reconocimiento facial y búsqueda inteligente), postgres (base de datos) y redis (cola de trabajos y cache). Esto significa que no fue necesario instalar manualmente una base de datos o un motor de IA: todo se gestiona a través de Docker Compose.

Versión de Docker instalada	v29.6.1
Versión de Docker Compose	v5.3.0
Versión de Immich desplegada	v3

3.3 Software necesario — entorno bare metal

Para el despliegue sin Docker se instaló y configuró manualmente, cada componente que en el entorno Docker viene empaquetado dentro de las imágenes oficiales:

Node.js	v24 — instalado desde el repositorio oficial de NodeSource
PostgreSQL	16.14, escuchando en el puerto 5433 (PostgreSQL 18 resultó incompatible con VectorChord)
pgvector	0.8.5 — instalado desde el repositorio oficial de postgresql.org
VectorChord	1.1.1 — instalado a partir de un paquete .deb para arm64 publicado en GitHub Releases
Redis	instalado vía apt (paquete de sistema), escuchando en el puerto 6379

FFmpeg	8.0.1 — instalado directamente desde los repositorios de Ubuntu 26.04 (no fue necesario compilarlo ni usar Jellyfin-ffmpeg)
Immich	v3.0.2 — instalado mediante el script immich-native, escuchando en el puerto 2283
Apache	2.4.66 — utilizado como proxy reverso en el puerto 80
Prometheus / Grafana	instalados vía apt (Prometheus) y el repositorio oficial de Grafana, para métricas y dashboards

A diferencia del entorno Docker, en el bare metal cada uno de estos componentes corre como proceso independiente del sistema operativo (varios de ellos gestionados con systemd), por lo que se debieron resolver manualmente tareas que en Docker resuelven las imágenes: creación de usuario de sistema para Immich, permisos de carpetas, configuración de la base de datos con extensiones vectoriales, y la definición del propio servicio de arranque.

4. Instalación y prueba del software

4.1 Entorno Docker

Se instaló Immich siguiendo el método oficial y recomendado por el proyecto para entornos de producción: despliegue mediante Docker Compose, sin clonar el repositorio completo del código fuente, sino descargando únicamente los archivos de configuración necesarios. Además, se optó por omitir la imagen relacionada al machine learning para aliviar la instalación y la ejecución del proyecto.

Una vez levantado el servicio se realizaron las siguientes pruebas de funcionamiento:

- Verificación de que los cuatro contenedores (servidor, base de datos y redis) se encontraban en estado "Up" mediante docker compose ps.
- Acceso a la interfaz web desde el navegador a través de la IP del servidor en el puerto 2283.
- Creación del primer usuario administrador (el primer usuario registrado en una instalación nueva de Immich se convierte automáticamente en administrador).
- Carga de prueba de fotografías/videos desde la interfaz web y verificación de que se procesaron correctamente (miniaturas, metadatos, reconocimiento facial).
- Instalación e inicio de sesión desde la app móvil.

4.2 Entorno bare metal

Sobre la VM se realizaron pruebas equivalentes, adaptadas a una instalación sin contenedores:

- Verificación de que los servicios systemd (immich, postgresql@16-main, redis-server, apache2, prometheus, grafana-server) se encontraban activos mediante systemctl status.
- Chequeo de salud del backend con curl http://localhost:2283/api/server/ping.
- Acceso a la interfaz web desde el navegador a través de http://192.168.64.5, sirviendo el tráfico mediante el reverse proxy de Apache.
- Carga de prueba de fotografías/videos y verificación del procesamiento (miniaturas y metadatos), sin funciones de IA al no estar desplegado el servicio de machine learning.
- Revisión de logs en tiempo real con sudo tail -f /var/log/immich/immich.log durante las pruebas.

5. Instructivo paso a paso

5.1 Despliegue con Docker Compose

A continuación se detalla el procedimiento completo llevado a cabo para instalar y configurar Immich mediante Docker Compose.

Paso 1 — Instalar Docker y el plugin de Docker Compose

Se verificó si Docker ya estaba instalado y, en caso contrario, se instaló siguiendo el método oficial para la distribución utilizada (por ejemplo, en Ubuntu/Debian mediante el script de conveniencia o el repositorio oficial de Docker, evitando el paquete `docker.io` del repositorio de la distribución):

1. Verificar si Docker está instalado: `docker --version`
2. Si no está instalado, instalar Docker Engine siguiendo la guía oficial para el sistema operativo correspondiente.
3. Verificar que el plugin de Compose esté disponible: `docker compose version`
4. (Opcional) Agregar el usuario actual al grupo `docker` para ejecutar comandos sin `sudo`, y cerrar/reabrir la sesión.

Paso 2 — Crear el directorio del proyecto

Se creó una carpeta dedicada para alojar los archivos de configuración de Immich, `/immich-app`, y se ingresó a ella.

Paso 3 — Descargar los archivos de configuración

Desde ese directorio se descargaron los dos archivos oficiales publicados en las releases del proyecto:

- `docker-compose.yml`: define los servicios (servidor, machine learning, base de datos y redis), sus imágenes, puertos y volúmenes.
- `example.env`, que se renombró a `.env`: contiene las variables de entorno de configuración.

Paso 4 — Configurar los archivos descargados

Paso 4.1 — Configurar las variables de entorno (.env)

Se editó el archivo `.env` definiendo, entre otras variables:

- `UPLOAD_LOCATION`: ruta del host donde se almacenan las fotos y videos subidos.
- `DB_DATA_LOCATION`: ruta donde se guardan los datos de la base de datos (debe estar en un sistema de archivos que soporte permisos de usuario/grupo).
- `DB_PASSWORD`: contraseña para la base de datos PostgreSQL, utilizada internamente por Docker Compose.
- `IMMICH_VERSION` (opcional): fijar una versión específica en lugar de `latest` para evitar cambios inesperados al actualizar.

Paso 4.2 — Editar el `docker-compose.yml`

Se editó el archivo `docker-compose.yml` omitiendo la imagen de machine learning, de esta forma se quitan las funcionalidades de IA dentro de la app, la cuales exceden el alcance del tp pero además removerlo implica una optimización significativa para la máquina donde se ejecute el servicio.

- imagen a remover: `immich-machine-learning`

Paso 5 — Levantar los contenedores

Desde el directorio del proyecto se ejecutó el comando que descarga las imágenes necesarias y levanta todos los servicios en segundo plano:

```
docker compose up -d
```

La primera ejecución demoró varios minutos debido a la descarga de las imágenes de Docker.

Paso 6 — Verificar el estado de los contenedores

Se verificó que los cuatro contenedores estuvieran activos (estado "Up") con:

```
docker compose ps
```

Paso 7 — Acceder a la interfaz web y crear el usuario administrador

Se abrió el navegador y se accedió a la dirección del servidor en el puerto 2283 (por ejemplo, <http://localhost:2283> si se accede desde la misma máquina, o <http://<ip-del-servidor>:2283> desde otro equipo de la red). El primer usuario registrado quedó automáticamente configurado como administrador del sistema.

Paso 8 — Prueba funcional

Se subieron fotos/videos de prueba desde la interfaz web para confirmar que el servicio procesaba correctamente el contenido (miniaturas, metadatos y reconocimiento facial), validando así el correcto funcionamiento del despliegue.

Referencia: <https://docs.immich.app/install/docker-compose>

5.2 Despliegue bare metal sin Docker (Ubuntu VM)

Para el segundo entorno se creó una máquina virtual Ubuntu 26.04 LTS ARM64 con UTM y se instaló Immich de forma nativa, sin contenedores, siguiendo el método immich-native de la comunidad.

Paso 1 — Crear la VM y preparar el sistema base

- 1. Crear una VM Ubuntu 26.04 LTS (ARM64) en UTM, asignando al menos 3 GB de RAM y ~60 GB de disco.
- 2. Actualizar el sistema (`sudo apt update && sudo apt upgrade`) e instalar utilidades básicas (`curl`, `git`, `build-essential`).
- 3. Crear un usuario y grupo de sistema dedicados (`immich`) para ejecutar el servicio con permisos acotados.

Paso 2 — Instalar Node.js

Se instaló Node.js v24 desde el repositorio oficial de NodeSource, requerido para ejecutar el backend y el build de Immich.

Paso 3 — Instalar y configurar PostgreSQL con extensiones vectoriales

- 1. Instalar PostgreSQL 16 desde el repositorio oficial de postgresql.org (PostgreSQL 18, la versión más nueva disponible en los repos de Ubuntu 26.04, resultó incompatible con VectorChord).
- 2. Configurar el cluster para escuchar en el puerto 5433.
- 3. Instalar la extensión pgvector 0.8.5 desde el mismo repositorio de postgresql.org.
- 4. Instalar VectorChord 1.1.1 a partir del paquete .deb para arm64 publicado en las releases de GitHub del proyecto.
- 5. Crear la base de datos y el usuario immich, y habilitar las extensiones necesarias (vector, vectorchord) sobre esa base.

Paso 4 — Instalar Redis y FFmpeg

Se instaló Redis como paquete de sistema vía apt (puerto 6379 por defecto) y FFmpeg 8.0.1 directamente desde los repositorios de Ubuntu 26.04, sin necesidad de compilarlo ni de depender de Jellyfin-ffmpeg.

Paso 5 — Instalar Immich mediante immich-native

- 1. Descargar/clonar el instalador immich-native correspondiente a la versión v3.0.2 dentro de /var/lib/immich/
- 2. Ejecutar el script de instalación, que compila el backend y el frontend de Immich dentro de /var/lib/immich/app/.
- 3. Ampliar temporalmente el tamaño de /tmp (que por defecto es tmpfs limitado a la RAM de la VM) con sudo mount -o remount,size=4G /tmp, y definir TMPDIR=/var/lib/immich/tmp, ya que el build fallaba por falta de espacio.

Paso 6 — Configurar el archivo de entorno

Se creó el archivo /var/lib/immich/env (permisos 600) con la configuración de conexión a la base de datos, Redis y el directorio de subida:

```
DB_URL=postgresql://immich:examplePassword@localhost:5433/immich
UPLOAD_LOCATION=/var/lib/immich/upload
IMMICH_TELEMETRY_INCLUDE=all
REDIS_URL=redis://localhost:6379
REDIS_HOSTNAME=127.0.0.1
REDIS_PORT=6379
IMMICH_HOST=0.0.0.0
```

Nota: IMMICH_HOST=0.0.0.0 es una variable con un bug conocido en Immich v3.0.2 — se ignora, y el backend queda escuchando solo en 127.0.0.1. Por eso el acceso externo se resuelve con Apache como reverse proxy (ver Paso 8).

Paso 7 — Configurar el servicio systemd

Se creó el archivo `/usr/lib/systemd/system/immich.service`, con dependencias sobre `redis-server` y `postgresql@16-main`, y un script de arranque `start.sh` que carga las variables del archivo `env` y ejecuta el proceso de Node:

```
ExecStart=/var/lib/immich/app/start.sh
```

Luego se habilitó e inició el servicio:

```
sudo systemctl enable --now immich
```

Paso 8 — Configurar Apache como reverse proxy

Dado que Immich sólo escucha en `127.0.0.1:2283`, se configuró Apache en el puerto 80 para exponer el servicio en la red local (ver detalle completo en la sección 6.6).

Paso 9 — Verificación y prueba funcional

- Verificar el estado de todos los servicios: `sudo systemctl status immich postgresql@16-main redis-server apache2`.
- Revisar los logs en caso de error: `sudo tail -f /var/log/immich/immich.log`.
- Acceder desde el navegador a `http://192.168.64.5` y crear el usuario administrador.
- Subir fotos de prueba y verificar que se procesan correctamente.

6. Requisitos adicionales solicitados

6.1 Storage template

El storage template define cómo Immich organiza los archivos de medios en el sistema de archivos. Se configuró desde la interfaz web en Administración → Storage Template, habilitando la funcionalidad y definiendo el patrón de organización:

```
{{y}}/{{MMMM}}/{{filename}}
```

Este template organiza las fotos en carpetas por año y mes, por ejemplo: `2024/January/foto.jpg`. Esta configuración se almacena en la base de datos PostgreSQL, por lo que persiste entre reinicios del stack y no requiere reconfiguración.

Variables disponibles: `{{y}}`, `{{MMMM}}`, `{{MM}}`, `{{dd}}`, `{{filename}}`, `{{ext}}`, `{{album}}`.

Referencia: <https://docs.immich.app/administration/storage-template/>

6.2 Backup

6.2.1 Base de datos

Se implementó una estrategia de backup en dos niveles siguiendo las recomendaciones oficiales de Immich:

Backup de base de datos: Immich genera automáticamente backups de la base de datos PostgreSQL en el directorio `UPLOAD_LOCATION/backups`. La retención y el horario son configurables desde Administración → Settings → Backup. Por defecto se conservan los últimos 14 backups con ejecución diaria a las 2:00 AM.

Para disparar un backup manual: Administración → Job Queues → Create Database Dump.

6.2.2 Archivos (Filesystem)

Backup del filesystem: Se desarrolló un script de backup utilizando Borg, una herramienta de backup deduplicante. El script lee automáticamente las variables del archivo .env para obtener las rutas de UPLOAD_LOCATION, realizando snapshots incrementales y excluyendo thumbnails y videos codificados (que pueden regenerarse). La retención configurada es de 4 semanas y 3 meses.

Instalar Borg:

```
sudo apt install borgbackup
```

Inicializar el repositorio Borg (solo la primera vez):

```
mkdir -p ./backups/local  
borg init --encryption=none ./backups/local/immich-borg
```

Ejecutar el script de backup manualmente:

```
./backup.sh
```

Programar ejecución diaria via cron:

```
crontab -e
```

Agregar:

```
0 2 * * * /ruta/completa/al/proyecto/backup.sh >>  
/ruta/completa/al/proyecto/backups/backup.log 2>&1
```

El script excluye thumbnails y videos codificados (regenerables) para ahorrar espacio.

6.2.3 Backup móvil

Backup móvil: La app móvil de Immich realiza backup automático de las fotos del dispositivo al servidor. Se configura desde la app en Perfil → Backup, con opciones de backup solo por WiFi y selección de álbumes.

Referencia: <https://docs.immich.app/administration/backup-and-restore/>

6.3 Biblioteca externa

Las librerías externas permiten indexar fotos que ya existen en el disco sin importarlas al servidor. Se configuró de la siguiente manera:

6.3.1 Crear la carpeta y agregar fotos

```
mkdir -p ./external-library  
cp /ruta/a/tus/fotos/* ./external-library/
```

6.3.2 Montar la carpeta en el contenedor

En docker-compose.yml, bajo immich-server → volumes:

```
- ./external-library:/usr/src/app/external-library:ro
```

Reiniciar el stack:

```
docker compose up -d
```

6.3.3 Configurar la librería en la web

1. Ir a Administration → External Libraries
2. Crear una nueva librería
3. En Folders, agregar /usr/src/app/external-library
4. Guardar y hacer click en Scan

6.3.4 Configurar sincronización periódica

1. Ir a Administration → Settings → External Library:
2. Habilitar Enable periodic library scanning
3. Configurar el cron (ej. 0 0 * * * para escaneo diario a medianoche)
4. Guardar

Referencia: <https://docs.immich.app/features/libraries/>

6.4 Monitoreo

Se habilitó el sistema de monitoreo incorporado de Immich, que expone métricas en formato Prometheus. El proceso involucró los siguientes pasos:

- Para activar la exposición de métricas en los puertos 8081 (API) y 8082 (microservicios), Se agregó la siguiente variable al archivo .env:

```
IMMICH_TELEMETRY_INCLUDE=all
```

- Agregar en el docker-compose.yml

```
immich-prometheus:
  container_name: immich_prometheus
  ports:
    # this exposes the default port for Prometheus so you can interact with
    it
    - 9090:9090
  image: prom/prometheus
  volumes:
    # the Prometheus configuration file - a barebones one is provided to get
    started
    - ./prometheus.yml:/etc/prometheus/prometheus.yml
    # a named volume defined in the bottom of the Compose file; it can also
    be a mounted folder
    - prometheus-data:/prometheus

immich-grafana:
  container_name: immich_grafana
  command: ['./run.sh', '-disable-reporting'] # this is to disable Grafana's
  telemetry
  ports:
    - 3000:3000
  image: grafana/grafana
  volumes:
    # stores your pretty dashboards and panels
    - grafana-data:/var/lib/grafana
```

y agregar los volúmenes:

```
volumes:  
  
  model-cache:  
  
  prometheus-data:  
  
  grafana-data:
```

- Descargar el archivo de [configuración](#) y ubicarlo en la misma carpeta que el archivo compose. Este archivo define las fuentes a las que Prometheus debe apuntar.
- Se configuró el aprovisionamiento automático de Grafana mediante archivos en grafana/provisioning/, incluyendo la definición del datasource de Prometheus y un dashboard personalizado con métricas del sistema.
- Se presentó una dificultad con el dashboard importado (ID 21493) que apuntaba a un endpoint incorrecto (ntp:9123). La solución fue corregir la variable "host" del dashboard para que apuntara al contenedor correcto de Prometheus.
- Las métricas disponibles incluyen: uso de CPU y memoria del proceso, operaciones de la base de datos, latencia de lectura/escritura de archivos, emisión de eventos, trabajos en cola, y estadísticas de red.

Referencia: <https://docs.immich.app/features/monitoring/>

6.5 Configuración de notificaciones por email en Immich usando SMTP de Gmail

6.5.1 Requisito previo en Google

Antes de tocar Immich hay que generar una contraseña de aplicación en la cuenta de Gmail que se va a usar para enviar los correos.

Activar la verificación en dos pasos en la cuenta de Google (es obligatorio, sin esto no se pueden crear contraseñas de aplicación). Se hace desde la configuración de seguridad de la cuenta de Google.

Una vez activada, ir a la sección de contraseñas de aplicaciones de Google (myaccount.google.com/apppasswords) y crear una nueva.

Al finalizar, Google muestra una contraseña de 16 caracteres. Guardarla en un lugar seguro, porque se muestra una sola vez y es la que se va a usar en Immich.

6.5.2 Configuración del SMTP en Immich

- Ingresar al panel de administración de Immich.
- Ir a Administration, luego Settings, luego Notification Settings.
- Dentro de la sección Email, completar los siguientes campos:

Host: smtp.gmail.com

Port: 587

Username: la dirección de Gmail completa

Password: la contraseña de aplicación de 16 caracteres generada en el paso 1 (no la contraseña normal de la cuenta)

From address: la dirección de Gmail (o la que se quiera mostrar como remitente)

Ignore certificate errors: dejar desactivado, salvo que haya problemas de conexión

- Activar la opción Enable Email Notifications.

- Usar el botón "Send test email and save" para validar que la conexión SMTP funciona correctamente. Este botón sólo prueba la conectividad, no usa las plantillas personalizadas.
- Si la prueba es exitosa, guardar la configuración.

6.5.3 Eventos que disparan notificaciones por email

Immich envía notificaciones automáticas por email en tres situaciones:

- Cuando se crea un usuario nuevo (email de bienvenida).
- Cuando se agrega un usuario como colaborador a un álbum compartido (email de invitación).
- Cuando se agregan nuevos archivos a un álbum compartido (email de actualización).

Cada usuario puede activar o desactivar estas notificaciones individualmente desde su propia configuración de cuenta.

6.5.4 Plantillas de email

Immich permite reemplazar el texto por defecto de cada uno de esos tres eventos con una plantilla HTML propia. Esto se configura en la misma pantalla de Notification Settings, donde hay un campo de texto separado para cada tipo de plantilla: la de bienvenida, la de invitación a álbum y la de actualización de álbum. Cada campo tiene su propia vista previa para chequear el resultado antes de guardar. Las plantillas admiten variables dinámicas que Immich reemplaza automáticamente por los datos reales (nombre de usuario, contraseña temporal, nombre del álbum, enlace al servidor, etc.) al momento de enviar el correo.

Referencia: <https://docs.immich.app/administration/email-notification/>

6.6 Reverse proxy

Se configuró Apache como reverse proxy para exponer Immich en el puerto 80, ocultando el puerto interno 2283.

6.6.1 Habilitar los módulos necesarios

```
sudo a2enmod proxy proxy_http proxy_wstunnel
```

6.6.2 Crear el archivo de configuración immich.conf con el siguiente contenido:

```
<VirtualHost *:80>
    ServerName <snip>
    ProxyRequests Off
    # set timeout in seconds
    ProxyPass / http://127.0.0.1:2283/ timeout=600 upgrade=websocket
    ProxyPassReverse / http://127.0.0.1:2283/
    ProxyPreserveHost On
</VirtualHost>
```

El archivo de configuración del VirtualHost para Immich (immich.conf) debe moverse a la carpeta de sitios disponibles de Apache:

```
sudo cp immich.conf /etc/apache2/sites-available/immich.conf
```

6.6.3 Activar el sitio

```
sudo a2ensite immich.conf
```

6.6.4 Recargar Apache

```
sudo systemctl reload apache2
```

6.6.5 Verificación

Acceder desde el navegador a:

```
http://<server_ip>/
```

Debería cargar la interfaz web de Immich, sirviendo el tráfico a través de Apache en el puerto 80, mientras Immich sigue escuchando internamente en 127.0.0.1:2283.

Referencia: <https://docs.immich.app/administration/reverse-proxy/>

7. Problemas surgidos y cómo se resolvieron

7.1 PostgreSQL 18 incompatible con VectorChord

En la VM bare metal, la versión de PostgreSQL disponible en los repositorios de Ubuntu 26.04 (PG18) no tenía soporte estable de VectorChord.

Solución: Se instaló PostgreSQL 16 desde el repositorio oficial de postgresql.org en lugar de usar la versión del sistema.

7.2 Fallo de build por falta de espacio en /tmp

Durante la instalación con immich-native, el build fallaba porque /tmp es un tmpfs limitado a la RAM de la VM (1.7 GB).

Solución: Se remontó /tmp con más tamaño (sudo mount -o remount,size=4G /tmp) y se definió TMPDIR=/var/lib/immich/tmp para el proceso de build.

7.3 Immich sólo escuchaba en 127.0.0.1 pese a IMMICH_HOST=0.0.0.0

Se trata de un bug conocido de Immich v3.0.2: la variable IMMICH_HOST se ignora y el proceso siempre queda escuchando en localhost.

Solución: Se resolvió el acceso externo colocando Apache como reverse proxy delante del backend, en lugar de depender de que Immich escuche en todas las interfaces.

7.4 Errores de permisos en carpetas

Tanto Prometheus como PostgreSQL fallaron al iniciar (permission denied) porque los directorios montados no pertenecían al usuario interno que usa cada contenedor. Se resolvió ajustando el propietario de las carpetas con chown (UID 999 para PostgreSQL) y los permisos del archivo de configuración de Prometheus con chmod.

8. Uso de Inteligencia Artificial

Se utilizó un modelo de inteligencia artificial como apoyo durante la elaboración de este informe y durante el proceso de instalación. A continuación se detalla su uso conforme lo solicitado:

Modelo utilizado	Claude Sonnet 5
Herramienta / interfaz	claude.ai

Prompts utilizados y finalidad de cada uno:

- **Prompt:** estoy haciendo un proyecto que se basa en levantar un servicio web, el servicio ya está desarrollado pero lo que se me evalúa es que lo levante y lo configure por mi cuenta, para esto elegimos immich, una biblioteca de fotos en la nube propia. quiero que redactes un informe que cuente con los siguientes requerimientos:

Descripción del proyecto

- * Relevamiento de características de la maquina utilizada
- * Relevamiento de sistema operativo y software utilizado y/o necesario.
- * Lograr instalar y probar software
- * Hacer un instructivo paso a paso de lo ocurrido
- * Enunciar los problemas que surgieron y como se resolvieron
- * Si se uso IA, especificar: Modelo usado, prompts y para que se utilizo

los datos que no los sepas, ej mi os y hardware, dejalos como para que lo complete yo pero redacta la oracion o sección completa

— **Finalidad:** Obtener la estructura base del informe, con índices, títulos y formato, para luego completar y especificar las particularidades de nuestro proceso por nuestra cuenta.

- **Prompt:** immich permite html custom para 3 eventos: bienvenida de usuario nuevo (variables: {username}, {password}, {displayName}), invitacion a album compartido ({senderName}, {recipientName}, {albumId}, {albumName}) y updates en el album({recipientName}, {albumId}, {albumName}). generame una plantilla html simple para cada evento con estilos inline

— **Finalidad:** generar plantillas genéricas para los emails.

En todos los casos, la información sugerida por el modelo de IA fue revisada, contrastada con la documentación oficial de Immich y verificada empíricamente en el propio despliegue antes de darla por válida, siendo el alumno responsable final de la instalación, la configuración y el contenido de este informe.

9. Conclusión

La implementación de Immich demostró ser un proyecto técnicamente enriquecedor que involucró la integración de múltiples tecnologías y la resolución de problemas reales en entornos de producción.

La modalidad Docker simplificó enormemente la gestión de dependencias y la configuración inicial, siendo ideal para entornos de desarrollo y prototipado. La integración de Prometheus y Grafana para monitoreo, junto con el sistema de backups automatizado con Borg, conforman una solución robusta y production-ready.

La implementación bare metal, aunque significativamente más compleja, ofrece mayor control sobre cada componente del stack y es representativa de un despliegue en infraestructura propia. Los principales desafíos fueron la compatibilidad entre versiones de PostgreSQL y las extensiones vectoriales, y las limitaciones de recursos de la VM.

Ambas implementaciones cumplieron con todos los requisitos establecidos por la cátedra:

Storage template personalizado configurado y activo

App web accesible y app móvil conectada via Tailscale

Estrategia de backup en tres niveles: base de datos (automático), filesystem (Borg) y móvil (app)

Librería externa con escaneo periódico diario configurado

Monitoreo con Prometheus scrapeando métricas y dashboard personalizado en Grafana

Reverse proxy Apache en producción exponiendo el servicio en el puerto 80

La experiencia reforzó la importancia de comprender en profundidad el stack tecnológico subyacente, especialmente al trabajar en entornos fuera de los configuraciones soportadas oficialmente, como fue el caso de Ubuntu 26.04 LTS con componentes diseñados para versiones anteriores.