Práctica recuperación: Despliegue de un Servidor LDAP con Docker

XiaoLong Ji

Administració i manteniment de Sistemes i aplicacions
Universitat de Lleida - Curs 2025-2026

28 de enero de 2026

Índice

1. Resumen de la Práctica

Este informe documenta el diseño, despliegue y validación de una infraestructura de autenticación centralizada basada en el protocolo LDAP (*Lightweight Directory Access Protocol*).

El proyecto representa un cambio respecto a la práctica anterior realizadas en **AWS EC2**, migrando de un modelo **IaaS** (*Infrastructure as a Service*) basado en máquinas virtuales y configuración manual, a un modelo de **Microservicios** orquestados mediante **Docker**.

El objetivo central ha sido encapsular la lógica de negocio de una autenticación corporativa en contenedores inmutables, garantizando la portabilidad y la persistencia de datos.

2. Comparación y Análisis de Migración (AWS vs Docker)

2.1. Despliegue en AWS (Práctica Anterior)

En la práctica anterior de AWS, el despliegue se basaba en un enfoque **imperativo**. Se provisionaban instancias EC2 con Amazon Linux y se ejecutaban scripts secuenciales (Bash) para:

1. Descargar código fuente en C.
2. Instalar compiladores (`gcc`, `make`) y dependencias del sistema.
3. Compilar e instalar OpenLDAP manualmente.
4. Configurar `systemd` para la gestión del servicio.

2.2. Despliegue con Docker (Práctica Actual)

Para esta práctica, se ha reutilizado el script en Bash diseñado originalmente para el entorno AWS/VM. Durante la fase de análisis del script para generar `bootstrap.ldif`, se ha identificado con **problemas de adaptación**:

- **El Problema:** El script Bash realizaba la instalación desde cero (compilación). Intentar replicar este script línea por línea dentro de un `Dockerfile` es considerado un *anti-patrón* en Docker. Hubiera resultado en una imagen pesada, tiempos de construcción excesivos y una gestión ineficiente de capas.
- **Solución aplicada:** En lugar de usar el script para instalar el software, se analizó el script para extraer únicamente la **lógica de negocio** (elementos como usuarios 'Jordi' y 'Manel', UIDs 4000/4001 y contraseñas hash).
- Se pasó de *construir el software* (AWS) a *configurar una imagen existente* (Docker), aprovechando la imagen `osixia/openldap` que ya contiene los binarios compilados y optimizados.

3. Estructura de Ficheros

La solución se diseñó siguiendo el principio de responsabilidad única, separando la infraestructura en tres servicios definidos en `docker-compose.yml`.

3.1. Orquestación de Servicios (`docker-compose.yml`)

En docker se define la topología de red y volúmenes.

- **Red (`ldap-net`):** Se configuró una red tipo *bridge*. Esto permite **Service Discovery** automático mediante DNS interno. El contenedor LAM puede encontrar al servidor LDAP simplemente resolviendo el hostname `ldap-server`, sin gestionar direcciones IP.
- **Servicio LDAP (Backend):**
  - Expone puertos 389/636.
  - Monta volúmenes persistentes (`ldap_data`) para `/var/lib/ldap`. Esto asegura que si el contenedor se destruye, la base de datos MDB sobrevive.
- **Servicio LAM (Frontend):**
  - Versión fijada a `8.0.1` (buena práctica de poner una versión específica para evitar problemas de compatibilidad).
  - Configurado mediante variables de entorno (`LDAP_SERVER=ldap-server`) para autoconfigurarse al inicio, evitando la configuración manual vía GUI en cada despliegue.

3.2. Datos de usuarios y Persistencia (`bootstrap.ldif`)

Para esta práctica, uno de los objetivos del temario es saber manejar un documento LDIF, por eso, se ha creado unos usuarios iniciales mediante este archivo.

- **Qué es:** Un archivo LDIF (*LDAP Data Interchange Format*) que define el estado inicial del directorio
- **Ventajas:** En ente archivo almacena usuarios iniciales con roles e informaciones ya preestablecidas. Esto permite que al iniciar el contenedor, el servicio LDAP ya tenga los datos necesarios sin necesidad de ejecutar comandos adicionales.
- **Estructura:** El archivo contiene entradas para:
  - **Base DN:** Define la raíz del directorio (`dc=amsa,dc=udl,dc=cat`).
  - **Organizational Units (OU):** Crea unidades organizativas como `users` y `groups`.
  - **Usuarios:** Define los usuarios 'Jordi' y 'Manel' con atributos como `uidNumber`, `gidNumber`, `homeDirectory`, y contraseñas en hash SSHA512.
  - **Grupos:** Crea grupos asociados a los usuarios.
- **Diseño:** Se ha convertido el script Bash original (de AWS) a entradas LDIF estáticas.
  - **Hash SSHA512:** Para ahorrar pasos se ha reutilizado el hash `{SSHA512}...` (password: 1234) del script original para mantener la consistencia de credenciales sin necesidad de instalar herramientas de generación de hash en el contenedor.

- **Mapeo de IDs:** Igual que la práctica anterior, se ha asignado los identificadores UID 4000/4001 y GID 5000/5001. Esto es importante para mantener la coherencia en sistemas que integren LDAP con otros servicios (por ejemplo, NFS).

3.3. Selección de la Imagen Base

Para el servicio OpenLDAP en docker, lo primero que hay que hacer es elegir una imagen base adecuada. Se ha evaluado las principales imágenes disponibles y **oficiales** en Docker Hub:

- **Bitnami OpenLDAP (`bitnami/openldap`):** Enfocada en producción y seguridad, con un sistema de archivos no-root estricto. Es muy popular en entornos Kubernetes pero requiere una configuración inicial más rígida.
- **Osixia OpenLDAP (`osixia/openldap`):** Diseñada para flexibilidad y despliegue rápido en entornos de desarrollo y pruebas, ideal para esta práctica cuyo objetivo es demostrar los conocimientos básicos de LDAP y docker.

Además, se ha elegido **Osixia** por las siguientes ventajas técnicas para este entorno de pràctica:

1. **Bootstrapping simplificado:** Su mecanismo de inicio permite inyectar datos automáticamente mediante el flag `-copy-service`, facilitando la carga del archivo `bootstrap.ldif` sin scripts complejos.
2. **Gestión automática de TLS:** Genera certificados autofirmados automáticamente si no se proporcionan, simplificando la configuración de LDAPS (puerto 636) requerida en esta práctica (usando el enunciado de Lab05/practica3 de referencia).
3. **Verbosidad:** Ofrece logs detallados que facilitan la depuración durante la fase de desarrollo.

3.4. Construcción de Imagen Personalizada (`Dockerfile`)

Una vez elegido la imagen base, como buena práctica, habría que especificar una versión que se usará, en este caso se ha usado la versión `v1.5.0` .

- **Base Image:** `osixia/openldap:1.5.0` (Se evitó el tag `:latest` para asegurar estabilidad y reproducibilidad).
- **Inyección de Configuración:**

```
1 COPY bootstrap.ldif /container/service/slapd/assets/config/bootstrap/ldif/50-
      bootstrap.ldif
```

Listing 1: Inyección del bootstrap en Dockerfile

Esta línea es crítica. Aprovecha los *hooks* de inicio de la imagen base. Al arrancar, el contenedor detecta este archivo y puebla la base de datos automáticamente. Esto elimina la necesidad de ejecutar comandos `ldapadd` manuales post-despliegue, automatizando el aprovisionamiento.

- **Variables de Entorno:** Se definieron `LDAP_DOMAIN` y `LDAP_BASE_DN` dentro del Dockerfile para "quemar"la configuración de dominio (`amsa.udl.cat`) en la imagen.

4. Implementación y Validación

4.1. Despliegue

El despliegue se realizó mediante el comando:

```
docker-compose up -d --build
```

El flag `-build` forzó la lectura del `Dockerfile` local, integrando el archivo `bootstrap.ldif` en la nueva imagen.

4.2. Pruebas de Autenticación

Se validó el funcionamiento utilizando un contenedor cliente (`ubuntu`) dentro de la misma red Docker, simulando un entorno de producción donde el cliente y el servidor están aislados de internet pero conectados entre sí.

- **Acceder al contenedor cliente:**
  Se accedió al contenedor cliente Ubuntu para ejecutar comandos LDAP contra el servidor.

  ```
  docker exec -it ldap-client bash
  ```

- **Prueba de Consulta (Search Request):**
  Se verificó la existencia del usuario 'Jordi' definido en el script Bash original.

  ```
  ldapsearch -x -H ldap://ldap-server \
  -b "dc=amsa,dc=udl,dc=cat" \
  -D "cn=admin,dc=amsa,dc=udl,dc=cat" \
  -w adminpassword "(uid=jordi)"
  ```

```
root@fc3bb3500c79:/# ldapsearch -x -H ldap://ldap-server -b "dc=amsa,dc=udl,dc=cat" -D "cn=admin,dc=amsa,dc=udl,dc=cat" -w adminpassword "(uid=jordi)"
# extended LDIF
#
# LDAPv3
# base <dc=amsa,dc=udl,dc=cat> with scope subtree
# filter: (uid=jordi)
# requesting: ALL
#

# jordi, users, amsa.udl.cat
dn: uid=jordi,ou=users,dc=amsa,dc=udl,dc=cat
objectClass: inetOrgPerson
objectClass: posixAccount
objectClass: shadowAccount
cn: Jordi
sn: Mateo
uid: jordi
uidNumber: 4000
gidNumber: 5000
homeDirectory: /home/jordi
loginShell: /bin/bash
userPassword:: e1NTSEE1MTJ9Q0JWYVVkUUM5bVZ2QWkrME85MkozaEErYVBkaVdVcWY0bFZyNmJ
 HUkFVc0ZKWDVhRk9FYisxcFNzWThQUXdXMVVLdXVDR08yKzE2MEhvdG5malhJYVJLbHJ5VmVrTG51

# search result
search: 2
result: 0 Success

# numResponses: 2
# numEntries: 1
```

Figura 1: Captura consulta LDAP Jordi (Datos importados correctamente)

- **Prueba de Autenticación (Bind Request):**
  Se verificó la existencia del usuario 'Manel' y su capacidad de autenticación password.

```
ldapwhoami -x \
-H ldap://ldap-server \
-D "uid=manel,ou=users,dc=amsa,dc=udl,dc=cat" \
-w 1234
```

```
root@fc3bb3500c79:/# ldapwhoami -x -H ldap://ldap-server -D "uid=manel,ou=users,dc=amsa,dc=udl,dc=cat" -w 1234
dn:uid=manel,ou=users,dc=amsa,dc=udl,dc=cat
```

Figura 2: Captura de autenticación LDAP Manel (Autenticación exitosa)

- **Resultado:**
  El servidor retornó exitosamente el objeto DN `uid=jordi,ou=users...` con `uidNumber: 4000`. Esto confirma que:

  1. El contenedor levantó correctamente el servicio `slapd`.
  2. El proceso de *bootstrapping* leyó e importó el archivo LDIF correctamente.
  3. La autenticación (Bind) del administrador funcionó.
  4. El usuario 'jordi' existe con los atributos esperados.
  5. El usuario 'manel' pudo autenticarse con la contraseña definida (1234).

4.3. Evidencia Gráfica (LAM)

Se verificó visualmente la disponibilidad del servicio y la correcta creación de usuarios y grupos a través de la interfaz web LAM en `localhost:8080`.

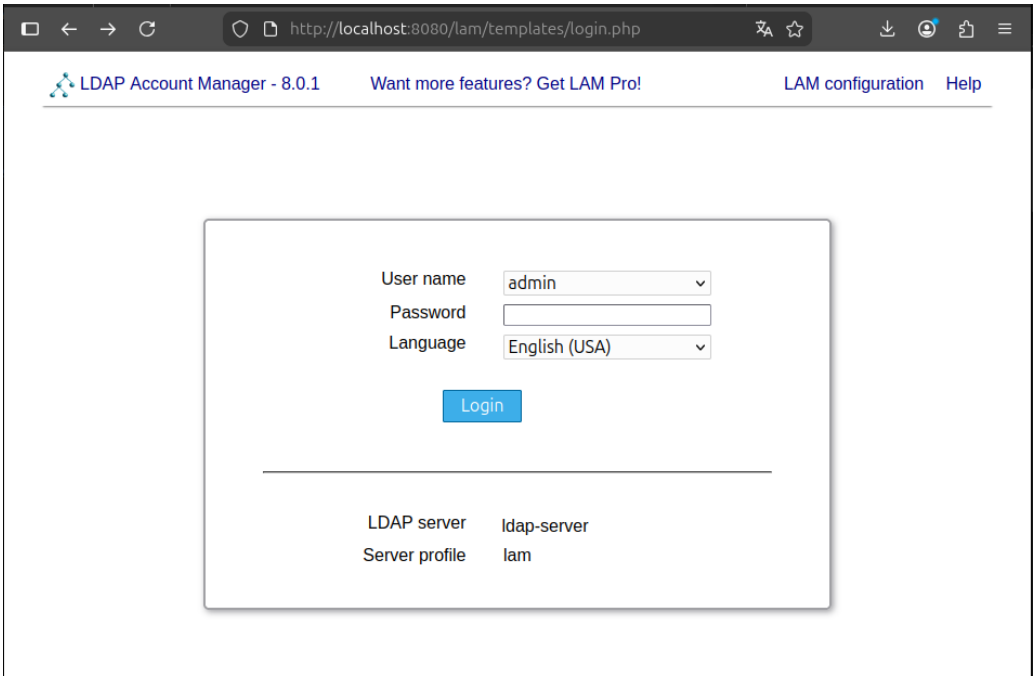

Figura 3: Pantalla de inicio de sesión de LAM (Evidencia de servicio activo)

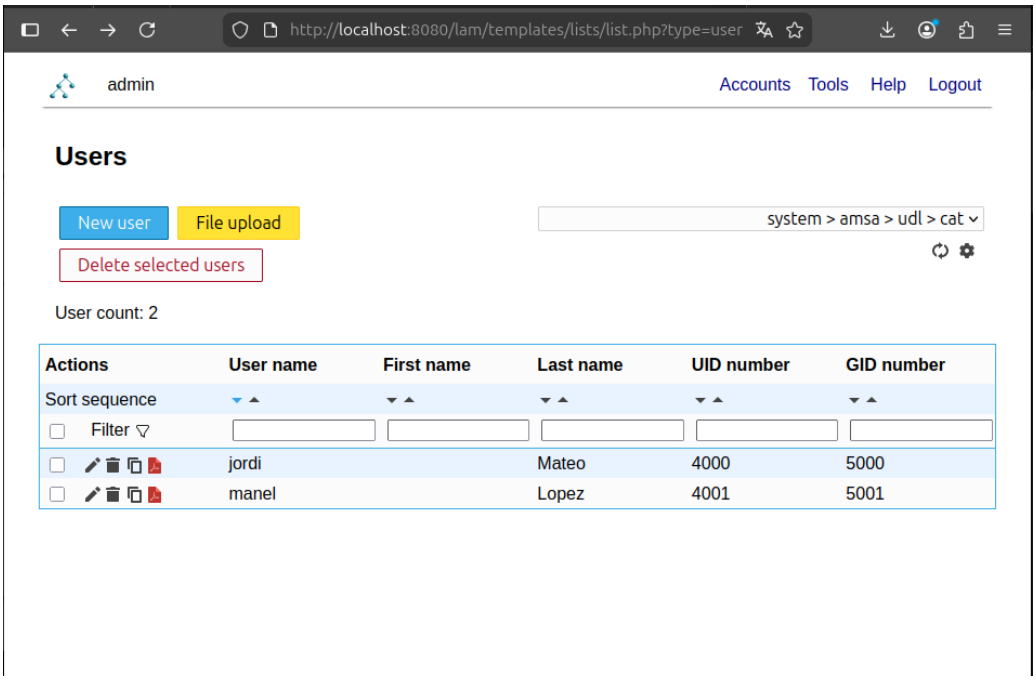

Figura 4: Pantalla de gestión de usuarios (Jordi y Manel)

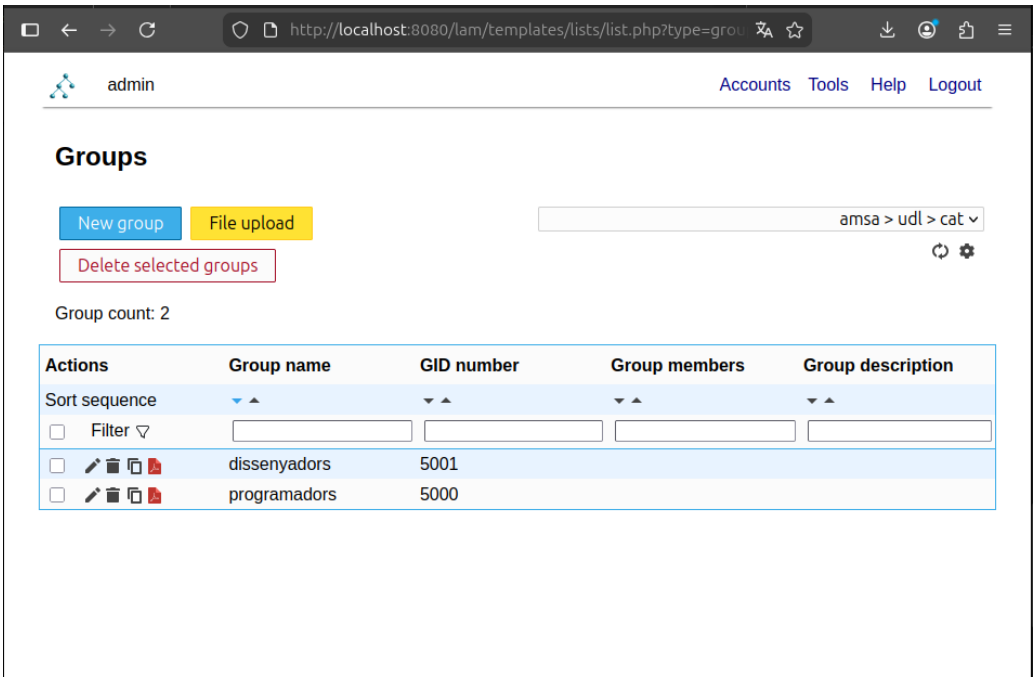

Figura 5: Pantalla de gestión de grupos

5. Conclusiones

La transición de la práctica de AWS a Docker ha demostrado las ventajas del modelo declarativo. A continuación se presenta una comparativa de impacto entre ambos modelos:

5.1. Tabla Comparativa

Criterio	Script Bash (Legacy)	Docker (Moderno)
Tiempo de Despliegue	∼20 minutos	∼1 minuto
Configuración	Manual / Imperativa	Automática / Declarativa
Gestión SSL	Manual	Automática
Reproducibilidad	Poco reproducible	Totalmente reproducible

Tabla 1: Comparativa AWS vs Docker

Además de las mejoras cuantitativas reflejadas en la Tabla 1, se destacan las siguientes ventajas cualitativas:

1. **Reducción de Complejidad:** Se eliminaron más de 50 líneas de comandos de instalación del script Bash original, reemplazándolas por 5 líneas en un `Dockerfile`.
2. **Idempotencia:** A diferencia del script Bash, que podía fallar si se ejecutaba dos veces (intentando crear usuarios ya existentes), el entorno Docker se reinicia siempre en un estado conocido y limpio.
3. **Aislamiento:** Las dependencias de OpenLDAP no contaminan el sistema operativo anfitrión, residiendo únicamente dentro de la imagen del contenedor.

Este laboratorio confirma la adquisición de competencias en la orquestación de servicios seguros y la capacidad de traducir requisitos de infraestructura tradicional a arquitecturas modernas de contenedores.

Firma: XiaoLong Ji

Repositorio del Proyecto: https://github.com/xji650/practica-ldap.git