Proyecto Intermodular

Ciclo Formativo de Grado Superior en
Administración de Sistemas Informáticos en Red

Plataforma DevOps Autoalojada con CI/CD

Joseph Meneses
Andrés González
Kristina Peleshok

11 de mayo de 2026

Índice

1. Introducción

Las organizaciones que desarrollan software dependen habitualmente de plataformas externas como GitHub o GitLab para gestionar su código fuente. Si bien estas herramientas son ampliamente utilizadas, presentan limitaciones relevantes en contextos empresariales: dependencia de terceros, restricciones en las versiones gratuitas y falta de control sobre los datos propios. Esta dependencia resulta especialmente crítica en entornos donde la confidencialidad del código fuente es un requisito, como pueden ser proyectos con información sensible, entornos educativos con datos de alumnos o pequeñas y medianas empresas que prefieren mantener la soberanía sobre su infraestructura tecnológica.

Este proyecto nace con el objetivo de ofrecer una alternativa autoalojada, segura y económicamente viable a esas plataformas. Para ello se ha diseñado e implementado una plataforma DevOps completa desplegada sobre un servidor VPS, que proporciona control total sobre el código fuente, los datos y el proceso de integración y entrega continua, sin depender de ningún servicio externo de pago ni ceder datos a terceros.

La solución se estructura en torno a cuatro pilares técnicos:

- **Gestión de código y automatización.** El núcleo del sistema es Forgejo, una plataforma de control de versiones Git autoalojada y de código abierto. Se complementa con Forgejo Actions y un runner dockerizado que ejecuta pipelines de integración continua en contenedores aislados, permitiendo automatizar compilaciones, pruebas y despliegues de forma reproducible y segura.
- **Gestión centralizada de identidades.** El control de acceso a todos los servicios se delega en un servidor OpenLDAP, que actúa como directorio único de usuarios. Esto elimina la necesidad de gestionar cuentas de forma independiente en cada herramienta y garantiza que una única fuente de verdad governe la autenticación en toda la plataforma.
- **Seguridad perimetral.** Nginx actúa como proxy inverso frontal con terminación SSL/TLS mediante certificados Let's Encrypt, de modo que ningún servicio interno queda expuesto directamente a internet. La seguridad se refuerza con hardening del acceso SSH al servidor, cabeceras de seguridad HTTP conformes con las recomendaciones de OWASP, y aislamiento de red mediante una subred Docker privada.
- **Infraestructura y persistencia.** Todo el sistema se orquesta con Docker Compose sobre Ubuntu Server 24.04 LTS. Cada servicio dispone de healthchecks automáticos, límites de recursos de CPU y memoria, y volúmenes persistentes en el sistema de ficheros del host, lo que garantiza la continuidad del servicio ante reinicios o recreaciones de contenedores.

La plataforma resultante es accesible a través de URL pública con HTTPS, soporta autenticación centralizada mediante LDAP, ejecuta pipelines de CI/CD de forma automática ante cada cambio en el código.

2. Objetivos

2.1. Objetivo general

Diseñar, implementar y poner en producción una plataforma DevOps autoalojada que proporcione a equipos de desarrollo o entornos educativos un sistema completo de control de versiones, integración continua y gestión de identidades, sin depender de servicios en la nube de terceros y con un nivel de seguridad adecuado para su exposición pública a internet.

2.2. Objetivos específicos

- Desplegar Forgejo como plataforma de control de versiones Git autoalojada, accesible mediante HTTPS y SSH desde cualquier cliente Git estándar.
- Implementar un sistema de integración continua y entrega continua (CI/CD) basado en Forgejo Actions y un runner dockerizado, capaz de ejecutar pipelines automáticos ante eventos de repositorio.
- Configurar OpenLDAP como directorio centralizado de usuarios e integrar la autenticación LDAP en Forgejo, de forma que la gestión de identidades quede unificada en un único servicio.
- Desplegar Nginx como proxy inverso con terminación SSL/TLS, incorporando cabeceras de seguridad HTTP, limitación de tasa de peticiones y configuración de caché, alcanzando un nivel de seguridad acorde con las recomendaciones de OWASP.
- Garantizar la persistencia de todos los datos relevantes mediante volúmenes Docker vinculados al sistema de ficheros del host, de forma que la destrucción o recreación de un contenedor no implique pérdida de información.

3. Requisitos

Requisito funcional
Control de versiones Git mediante interfaz web, HTTPS y SSH.
Ejecución automática de pipelines CI/CD ante eventos de repositorio.
Autenticación con credenciales LDAP y creación automática de perfil en el primer acceso.
Acceso web exclusivamente por HTTPS con certificado de autoridad reconocida.
Persistencia de datos ante reinicios y recreaciones de contenedores.
Autenticación SSH mediante clave pública para operaciones Git.

Requisito no funcional
Reinicio automático de contenedores y healthchecks en todos los servicios.
Ningún servicio interno accesible directamente desde internet; todo el tráfico pasa por Nginx.
Cabeceras de seguridad HTTP conforme a OWASP en todas las respuestas.
Límites de CPU y memoria definidos por contenedor.
Red Docker privada con direccionamiento estático; puertos internos no expuestos innecesariamente.
Parámetros sensibles centralizados en fichero `.env`, sin hardcodear en el código.
Caché de activos estáticos y compresión Gzip en el proxy inverso.

3.1. Requisitos no funcionales

- **Disponibilidad.** Todos los servicios deben configurarse con política de reinicio automático (`restart: unless-stopped`) y healthchecks que permitan a Docker detectar y recuperar contenedores en estado degradado.
- **Seguridad perimetral.** Ningún servicio interno (LDAP, Forgejo HTTP, Portal) debe ser accesible directamente desde internet. Todo el tráfico web externo debe canalizarse a través del proxy inverso Nginx. Los puertos internos de LDAP no deben publicarse en el host.
- **Cabeceras de seguridad HTTP.** El proxy inverso debe incorporar, como mínimo, estas cabeceras en todas las respuestas:
  - `Strict-Transport-Security`,
  - `X-Frame-Options`, `X-Content-Type-Options` y
  - `Content-Security-Policy`
- **Gestión de recursos.** Cada contenedor debe tener definidos límites máximos de memoria y CPU, de acuerdo con los recursos disponibles en el VPS, para evitar que un servicio degrade al conjunto de la plataforma.
- **Aislamiento de red.** Los servicios deben comunicarse entre sí a través de una red Docker privada con direccionamiento estático, sin exponer puertos internos innecesariamente al host ni a internet.
- **Configuración externalizada.** Todos los parámetros sensibles o específicos del entorno (contraseñas, tokens, dominio, claves secretas) deben estar definidos en un fichero `.env` separado del código, de modo que el despliegue en un entorno diferente requiera únicamente modificar dicho fichero.
- **Rendimiento mínimo.** El proxy inverso debe incorporar caché de activos estáticos y compresión Gzip para reducir la carga sobre los servicios backend y mejorar los tiempos de respuesta percibidos por el usuario.

4. Diseño de la Arquitectura

4.1. Componentes del sistema

La tabla siguiente resume los servicios que forman la plataforma, su posición en la red interna y su función:

Servicio	Contenedor	IP interna	Puerto	Función
Nginx	devops-nginx	172.20.0.2	80, 443	Proxy inverso, terminación SSL/TLS
Forgejo	devops-forgejo	172.20.0.3	3000 (HTTP) 2222 (SSH)	Control de versiones Git, CI/CD Operaciones Git por SSH
OpenLDAP	devops-ldap	172.20.0.4	389	Directorio de usuarios (interno)
Portal	devops-portal	172.20.0.5	3001	Interfaz web simplificada
Runner	devops-runner	172.20.0.6	—	Ejecución de pipelines CI/CD

```
root@JAKserver:/srv/devops/compose# docker network ls
NETWORK ID     NAME                 DRIVER    SCOPE
212eecaede6f   bridge               bridge    local
d6311eb747c1   compose_devops-net   bridge    local
33e410ef5ecd   host                 host      local
8c3ab5137739   none                 null      local
root@JAKserver:/srv/devops/compose#
```

Figura 1: Redes Docker del host. La red `compose_devops-net` aloja todos los servicios de la plataforma.

4.2. Diagrama de arquitectura

El diagrama siguiente muestra el flujo del tráfico desde internet hasta cada servicio, así como las comunicaciones internas entre componentes:

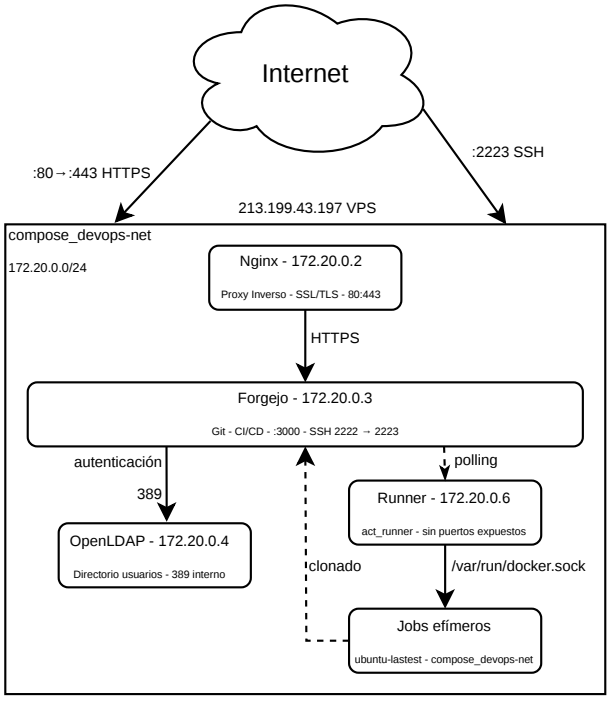


Figura 2: Diagrama del proyecto

4.3. Gestión de puertos SSH

Un aspecto relevante del diseño fue la coexistencia de dos servicios SSH en el mismo servidor: el daemon SSH del sistema operativo, necesario para la administración del VPS, y el servidor SSH de Forgejo, necesario para las operaciones Git. Para

resolverlo se adoptó la siguiente distribución de puertos:

Servicio	Puerto externo	Puerto interno	Uso
SSH del host	2222	—	Administración del VPS
Forgejo SSH	2223	2222	Operaciones Git (`git clone`, `push`)

El puerto 22 del contenedor de Forgejo queda ocupado por el proceso OpenSSH que arranca el entrypoint de la imagen, por lo que Forgejo escucha internamente en el puerto 2222 del contenedor, mapeado al puerto 2223 del host. El daemon SSH del host fue reconfigurado para escuchar en el puerto 2222 antes del despliegue, previa verificación en una sesión paralela para evitar el bloqueo de acceso administrativo.

5. Implementación

5.1. Preparación del servidor

Antes de desplegar los contenedores se realizaron las siguientes tareas sobre el sistema operativo base del VPS:

- Instalación de Docker Engine y Docker Compose en Ubuntu Server 24.04 LTS.
- Creación del árbol de directorios bajo `/srv/devops/`, que aloja tanto los ficheros de configuración como los datos persistentes de todos los servicios.
- Reconfiguración del daemon SSH del host al puerto 2222, liberando el puerto 22 y el 2223 para Forgejo. Esta operación se realizó con una sesión SSH paralela abierta para verificar el acceso antes de cerrar la sesión original.
- Obtención de certificados SSL/TLS mediante Certbot en modo `--standalone`, previo al primer arranque de Nginx, ya que este requiere los certificados para iniciarse correctamente.
- Configuración de los registros DNS para los subdominios `jackdevops.freeddns.org` y `portaljack.freeddns.org`, apuntando a la IP pública del VPS.
- Establecimiento de permisos de propietario en los directorios de datos, adaptados al UID del proceso de cada contenedor: UID 1000 para Forgejo y UID 1001 para OpenLDAP.

Todos los parámetros sensibles del entorno (contraseñas, tokens, dominio, claves secretas de Forgejo) se centralizaron en el fichero `/srv/devops/compose/.env`:

```
# ================================================================================
# CONFIGURACION DE DOMINIO
# ================================================================================
FORGEJO_DOMAIN=jackdevops.freeddns.org

# ================================================================================
# CONFIGURACION DE FORGEJO
# ================================================================================
# SECRET_KEY: Cadena hexadecimal aleatoria de 32 bytes usada para cifrado de sesion
FORGEJO_SECRET_KEY=1c70edddf39a50a2ba1723b618d4e92e428302cf1b455c90a4a6a84eaaae4d2f

# INTERNAL_TOKEN: Cadena hexadecimal aleatoria de 32 bytes usada internamente por Forgejo
FORGEJO_INTERNAL_TOKEN=ed3fe655f3186b38f640f59ed301f408583c6b68df9c58218b6270820894468e

# ================================================================================
# CONFIGURACION LDAP
# ================================================================================

LDAP_ORGANISATION=devops-org

LDAP_DOMAIN=jak.lab

LDAP_ROOT=dc=jak,dc=lab

LDAP_ADMIN_USERNAME=admin

LDAP_ADMIN_PASSWORD=pr0y3ct02ASIR

# ================================================================================
# CONFIGURACION DEL RUNNER DE FORGEJO
```

```
# ================================================================================
# URL donde el corredor puede alcanzar la instancia de Forgejo.
FORGEJO_INSTANCE_URL=http://172.20.0.3:3000/

FORGEJO_RUNNER_REGISTRATION_TOKEN=CR6Zdvi5bkdLc4WQcC83SMRH3uZlLDJNWDVQIVdf

FORGEJO_ACT_RUNNER_NETWORK=compose_devops-net
#
# Presupuesto total: ~8 GB (deja ~4 GB para SO, kernel, buferes y margen en sistema de 12 GB)
```

Generando los secretos de Forgejo mediante:

```
openssl rand -hex 32   # FORGEJO_SECRET_KEY
openssl rand -hex 32   # FORGEJO_INTERNAL_TOKEN
```

En total, se necesitaron 4 contenedores, uno para cada servicio, que se han juntado en un mismo archivo `docker-compose.yml`. Mantenerlos en un mismo archivo asegura que compartan la misma definición de red, facilitando la comunicación interna sin exponer puertos innecesarios al host. También facilita reiniciar servicios relacionados, revisar logs conjuntos o actualizar versiones ejecutando comandos únicos sin necesidad de gestionar múltiples archivos por separado. Quedando así el fichero en cuestión:

```
services:
  devops-nginx:
    image: nginx:stable-alpine
    container_name: devops-nginx
    restart: unless-stopped
    networks:
      devops-net:
        ipv4_address: 172.20.0.2
    ports:
      - "80:80"
      - "443:443"
    volumes:
      - /srv/devops/configs/nginx/nginx.conf:/etc/nginx/nginx.conf:ro
      - /srv/devops/configs/nginx/conf.d/:/etc/nginx/conf.d/:ro
      - /srv/devops/configs/nginx/ssl/:/etc/nginx/ssl/:ro
      - /srv/devops/data/nginx/logs/:/var/log/nginx/
      - /srv/devops/data/nginx/certbot/:/var/www/certbot/
    deploy:
      resources:
        limits:
          memory: 256m
          cpus: "0.5"
    healthcheck:
      test: ["CMD", "wget", "--quiet", "--tries=1", "--spider", "http://localhost/health"]
      interval: 30s
      timeout: 10s
      retries: 3
      start_period: 5s
    depends_on:
      devops-forgejo:
        condition: service_healthy
      devops-portal:
        condition: service_healthy

  devops-forgejo:
    image: codeberg.org/forgejo/forgejo:8
    container_name: devops-forgejo
    restart: unless-stopped
    networks:
      devops-net:
        ipv4_address: 172.20.0.3
    ports:
        - "3000:3000" # Mapea el puerto HTTP del contenedor al puerto 3000 del host
        - "2223:2222" # Mapea el puerto SSH del contenedor al puerto 2223 del host.
    volumes:
```

```
    - /srv/devops/configs/forgejo/app.ini:/data/gitea/conf/app.ini
    - /srv/devops/data/forgejo/data:/data
    - /srv/devops/data/forgejo/logs:/app/gitea/log
  environment:
    - FORGEJO__database__DB_TYPE=sqlite3
    - FORGEJO__security__SECRET_KEY=${FORGEJO_SECRET_KEY}
    - FORGEJO__security__INTERNAL_TOKEN=${FORGEJO_INTERNAL_TOKEN}
    - FORGEJO__server__ROOT_URL=https://${FORGEJO_DOMAIN}/
    - FORGEJO__server__SSH_DOMAIN=${FORGEJO_DOMAIN}
  deploy:
    resources:
      limits:
        memory: 1536m
        cpus: "1.5"
  healthcheck:
    test: ["CMD", "wget", "--quiet", "--tries=1", "--spider", "http://localhost:3000"]
    interval: 30s
    timeout: 10s
    retries: 3
    start_period: 10s
  depends_on:
    devops-ldap:
      condition: service_healthy

devops-ldap:
  image: osixia/openldap:1.5.0
  container_name: devops-ldap
  restart: unless-stopped
  networks:
    devops-net:
      ipv4_address: 172.20.0.4
  environment:
    - LDAP_ORGANISATION=${LDAP_ORGANISATION}
    - LDAP_DOMAIN=${LDAP_DOMAIN}
    - LDAP_ADMIN_PASSWORD=${LDAP_ADMIN_PASSWORD}
    - LDAP_BASE_DN=${LDAP_ROOT}
    - LDAP_TLS=false
  volumes:
    - /srv/devops/data/ldap/db:/var/lib/ldap
    - /srv/devops/data/ldap/config:/etc/ldap/slapd.d
  deploy:
    resources:
      limits:
        memory: 512m
        cpus: "0.5"
  healthcheck:
    test: ["CMD-SHELL", "ldapsearch -x -H ldap://localhost -D cn=admin,dc=jak,dc=lab -w pr0y3ct02ASIR -b dc=jak,
  dc=lab -s base > /dev/null 2>&1"]
    interval: 30s
    timeout: 10s
    retries: 5
    start_period: 30s

devops-runner:
  image: gitea/act_runner:latest
  container_name: devops-runner
  restart: unless-stopped
  networks:
    devops-net:
      ipv4_address: 172.20.0.6
  dns:
  - 8.8.8.8
  - 1.1.1.1
  volumes:
    - /var/run/docker.sock:/var/run/docker.sock
    - /srv/devops/data/runner/data:/data
```

```
    environment:
      - GITEA_INSTANCE_URL=${FORGEJO\_INSTANCE\_URL}
      - GITEA\_RUNNER\_REGISTRATION\_TOKEN=${FORGEJO_RUNNER_REGISTRATION_TOKEN}
      - CONFIG_FILE=/data/config.yaml
    deploy:
      resources:
        limits:
          memory: 3072m
          cpus: "2.0"
    depends_on:
      devops-forgejo:
        condition: service_healthy
```

5.2. OpenLDAP

OpenLDAP se desplegó mediante la imagen `osixia/openldap:1.5.0`, asignada a la IP interna `172.20.0.4`. El dominio LDAP configurado fue `jak.lab`, con base DN `dc=jak,dc=lab`. El contenedor no expone ningún puerto al host, siendo accesible únicamente desde la red interna Docker.

La estructura del directorio se organizó en dos unidades organizativas, con sus usuarios respectivamente:

```
root@dcf363e58a44:/# ldapsearch -x -H ldap://localhost -D "cn=admin,dc=jak,dc=lab" -w pr0y3ct02ASIR -b "dc=ja
k,dc=lab" -LLL "(objectClass=organizationalUnit)" dn
dn: ou=users,dc=jak,dc=lab

dn: ou=groups,dc=jak,dc=lab

root@dcf363e58a44:/#
```

Figura 3: Consultas de OUs

Consulta usada:

```
ldapsearch -x -H ldap://localhost -D "cn=admin,dc=jak,dc=lab" -w pr0y3ct02ASIR -b "dc=jak,dc=lab" -LLL "(
    objectClass=organizationalUnit)" dn
```

```
root@dcf363e58a44:/# ldapsearch -x -H ldap://localhost -D "cn=admin,dc=jak,dc=lab" -w pr0y3ct02ASIR -b "ou=us
ers,dc=jak,dc=lab" -LLL
dn: ou=users,dc=jak,dc=lab
objectClass: top
objectClass: organizationalUnit
ou: users

dn: uid=alvaro,ou=users,dc=jak,dc=lab
objectClass: top
objectClass: person
objectClass: organizationalPerson
objectClass: inetOrgPerson
objectClass: posixAccount
cn: Alvaro Alvaro
sn: Alvaro
uid: alvaro
mail: alvaro@jak.lab
uidNumber: 10004
gidNumber: 10004
homeDirectory: /home/alvaro
loginShell: /bin/bash
userPassword:: YWx2YXJva3A=

dn: uid=andres,ou=users,dc=jak,dc=lab
objectClass: top
objectClass: person
objectClass: organizationalPerson
objectClass: inetOrgPerson
objectClass: posixAccount
cn: Andres Andres
sn: Andres
uid: andres
mail: andres@jak.lab
uidNumber: 10003
gidNumber: 10003
homeDirectory: /home/andres
loginShell: /bin/bash
userPassword:: YW5kcmVza3A=
```

Figura 4: Consultas de usuarios 1

```
dn: uid=joseph,ou=users,dc=jak,dc=lab
objectClass: top
objectClass: person
objectClass: organizationalPerson
objectClass: inetOrgPerson
objectClass: posixAccount
cn: Joseph Joseph
sn: Joseph
uid: joseph
mail: joseph@jak.lab
uidNumber: 10002
gidNumber: 10002
homeDirectory: /home/joseph
loginShell: /bin/bash
userPassword:: am9zZXBoa3A=

dn: uid=kristina,ou=users,dc=jak,dc=lab
objectClass: top
objectClass: person
objectClass: organizationalPerson
objectClass: inetOrgPerson
objectClass: posixAccount
cn: Kristina Peleshok
sn: Peleshok
uid: kristina
mail: kristina@jak.lab
uidNumber: 10001
gidNumber: 10001
homeDirectory: /home/kristina
loginShell: /bin/bash
userPassword:: a3Jpc3RpbmFrcA==

root@dcf363e58a44:/#
```

Figura 5: Consultas de usuarios 2

```
ldapsearch -x -H ldap://localhost -D "cn=admin,dc=jak,dc=lab" -w pr0y3ct02ASIR -b "ou=users,dc=jak,dc=lab" -LLL "(
    objectClass=organizationalUnit)" dn
```

```
root@dcf363e58a44:/# ldapsearch -x -H ldap://localhost -D "cn=admin,dc=jak,dc=lab" -w pr0y3ct02ASIR -b "ou=gr
oups,dc=jak,dc=lab" -LLL
dn: ou=groups,dc=jak,dc=lab
objectClass: top
objectClass: organizationalUnit
ou: groups

dn: cn=forgejo-users,ou=groups,dc=jak,dc=lab
objectClass: top
objectClass: groupOfNames
cn: forgejo-users
member: uid=kristina,ou=users,dc=jak,dc=lab
member: uid=joseph,ou=users,dc=jak,dc=lab
member: uid=andres,ou=users,dc=jak,dc=lab
member: uid=alvaro,ou=users,dc=jak,dc=lab

dn: cn=forgejo-admins,ou=groups,dc=jak,dc=lab
objectClass: top
objectClass: groupOfNames
cn: forgejo-admins
member: uid=kristina,ou=users,dc=jak,dc=lab
member: uid=joseph,ou=users,dc=jak,dc=lab
member: uid=andres,ou=users,dc=jak,dc=lab
member: uid=alvaro,ou=users,dc=jak,dc=lab

root@dcf363e58a44:/#
```

Figura 6: Consultas de grupos

```
ldapsearch -x -H ldap://localhost -D "cn=admin,dc=jak,dc=lab" -w pr0y3ct02ASIR -b "ou=groups,dc=jak,dc=lab" -LLL
    "(objectClass=organizationalUnit)" dn
```

Para meterse a la consola de LDAP, es necesario ejecutar este comando:

```
docker exec -it devops-ldap bash
```

Los usuarios presentan los atributos `objectClass: inetOrgPerson`; y `objectClass: posixAccount`, con los campos `uid`, `sn` y `mail`. El atributo `givenName` no fue definido, lo que se tuvo en cuenta al configurar la fuente de autenticación en Forgejo.

La conectividad del servicio se verificó mediante `ldapsearch` desde dentro del contenedor de Forgejo antes de proceder a la integración:

```
docker exec devops-forgejo sh -c "nc -z devops-ldap 389 && echo 'LDAP alcanzable'"
```

Figura 7: Consulta del puerto LDAP

5.2.1. Script de gestión de usuarios

Para simplificar la creación de nuevos usuarios en el directorio se desarrolló el script `add_user.sh`, ubicado en `/srv/devops/config`. El script solicita un nombre de usuario, genera automáticamente la contraseña, el correo y el `uidNumber`, crea el usuario en OpenLDAP mediante `ldapadd` y lo añade a los grupos `forgejo-users` y `forgejo-admins`. Asimismo, actualiza el fichero `users.ldif` para mantener el estado del directorio reproducible:

```
#!/bin/bash
CONTAINER="devops-ldap"
LDAP_ADMIN_DN="cn=admin,dc=jak,dc=lab"
LDAP_PASS="pr0y3ct02ASIR"
BASE_DN="dc=jak,dc=lab"

read -p "Nombre de usuario: " USERNAME
USERNAME=$(echo "$USERNAME" | tr '[:upper:]' '[:lower:]' | tr -d ' ')

# Comprobar si existe
docker exec "$CONTAINER" ldapsearch -x -D "$LDAP_ADMIN_DN" -w "$LDAP_PASS" \
    -b "uid=${USERNAME},ou=users,${BASE_DN}" "(objectClass=*)" >/dev/null 2>&1
if [ $? -eq 0 ]; then echo "ERROR: usuario ya existe."; exit 1; fi

PASSWORD="${USERNAME}kp"
MAIL="${USERNAME}@jak.lab"
LAST_UID=$(grep "^uidNumber:" users.ldif | awk '{print $2}' | sort -n | tail -1)
NEW_UID=$((LAST_UID + 1))

# Crear usuario
TMP_USER="/tmp/${USERNAME}.ldif"
cat > "$TMP_USER" <<EOF
dn: uid=${USERNAME},ou=users,${BASE_DN}
objectClass: inetOrgPerson
objectClass: posixAccount
cn: ${USERNAME}
sn: ${USERNAME}
uid: ${USERNAME}
mail: ${MAIL}
uidNumber: ${NEW_UID}
gidNumber: ${NEW_UID}
homeDirectory: /home/${USERNAME}
loginShell: /bin/bash
userPassword: ${PASSWORD}
EOF

docker cp "$TMP_USER" "$CONTAINER:/tmp/"
docker exec "$CONTAINER" ldapadd -x -D "$LDAP_ADMIN_DN" -w "$LDAP_PASS" -f "/tmp/${USERNAME}.ldif"
if [ $? -ne 0 ]; then echo "ERROR: fallo al crear usuario."; exit 1; fi

# Meter en los grupos
cat > "$TMP_USER" <<EOF
dn: cn=forgejo-users,ou=groups,${BASE_DN}
changetype: modify
add: member
member: uid=${USERNAME},ou=users,${BASE_DN}
```

```
dn: cn=forgejo-admins,ou=groups,${BASE_DN}
changetype: modify
add: member
member: uid=${USERNAME},ou=users,${BASE_DN}
EOF

docker cp "$TMP_USER" "$CONTAINER:/tmp/"
docker exec "$CONTAINER" ldapmodify -x -D "$LDAP_ADMIN_DN" -w "$LDAP_PASS" -f "/tmp/${USERNAME}.ldif"

echo "Usuario '$USERNAME' creado (contrasena: $PASSWORD)"
```

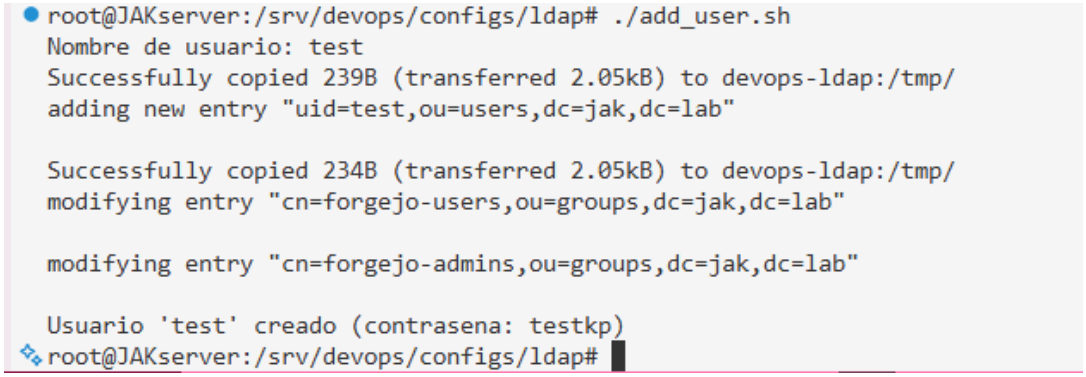


Figura 8: Ejecución del script `add_user.sh` creando el usuario `test` en OpenLDAP y añadiéndolo a los grupos `forgejo-users` y `forgejo-admins`.

5.3. Forgejo

Forgejo se desplegó con la imagen `codeberg.org/forgejo/forgejo:8`, asignada a la IP interna `172.20.0.3`. La configuración del servicio se realizó íntegramente mediante el fichero `app.ini` y variables de entorno inyectadas por Docker Compose, con el asistente de instalación web deshabilitado (`INSTALL_LOCK = true`).

Los parámetros principales del fichero `app.ini` son:

```
[server]
HTTP_PORT        = 3000
START_SSH_SERVER = true
SSH_PORT         = 2223   # Puerto visible por los clientes externos
SSH_LISTEN_PORT  = 2222   # Puerto de escucha dentro del contenedor

[database]
DB_TYPE = sqlite3
PATH = /data/gitea/data/forgejo.db

[security]
INSTALL_LOCK = true

[service]
DISABLE_REGISTRATION = true
ALLOW_ONLY_EXTERNAL_REGISTRATION = true
REQUIRE_SIGNIN_VIEW = true
```

Pero se añadieron unos parámetros más para asegurarse de la funcionalidad correcta del `runner`, dejando el archivo así:

```
APP_NAME = JAK Devops
APP_SLOGAN = El control total sobre tu trabajo.
RUN_USER = git
WORK_PATH = /data/gitea
RUN_MODE = prod

[server]
HTTP_PORT = 3000
START_SSH_SERVER = true
# SSH_PORT: Puerto que ven los CLIENTES EXTERNOS (git@jackdevops.freeddns.org:2223)
```

```
SSH_PORT = 2223
# SSH_LISTEN_PORT: Puerto que escucha DENTRO DEL CONTENEDOR
# Usamos 2222 para evitar colision con OpenSSH del entrypoint (que ocupa el 22)
# Docker Compose mapea este puerto 2222 interno al puerto 2223 del host
SSH_LISTEN_PORT = 2222
# ROOT_URL y SSH_DOMAIN son gestionados por Forgejo desde las variables de entorno
# FORGEJO__server__ROOT_URL y FORGEJO__server__SSH_DOMAIN definidas en docker-compose.yml.
# Para cambiar el dominio edita unicamente FORGEJO_DOMAIN en compose/.env.
ROOT_URL = https://jackdevops.freeddns.org/
SSH_DOMAIN = jackdevops.freeddns.org
DOMAIN = jackdevops.freeddns.org
APP_DATA_PATH = /data/gitea/data
DISABLE_SSH = false
LFS_START_SERVER = true
LFS_JWT_SECRET = TM7-ImNJUklyAcDG7FB7dVomPK2O5E9oBIW0UWVexkU
OFFLINE_MODE = true

[database]
DB_TYPE = sqlite3
# Ruta absoluta para garantizar que la DB quede dentro del volumen persistente /data
# Sin esto, SQLite resuelve la ruta relativa desde el CWD del proceso (/app/gitea)
# y la DB queda fuera del volumen -> se pierde al recrear el contenedor.
PATH = /data/gitea/data/forgejo.db
HOST = 127.0.0.1:5432
NAME = forgejo
USER = forgejo
PASSWD =
SCHEMA =
SSL_MODE = disable
LOG_SQL = false

# SECRET_KEY e INTERNAL_TOKEN se gestionan via variable de entorno en
# docker-compose.yml. No definirlos aqui evita que app.ini sobreescriba
# los valores correctos con el literal ${FORGEJO_SECRET_KEY}.
[security]
INSTALL_LOCK = true
INTERNAL_TOKEN = ed3fe655f3186b38f640f59ed301f408583c6b68df9c58218b6270820894468e
SECRET_KEY = 1c70edddf39a50a2ba1723b618d4e92e428302cf1b455c90a4a6a84eaaae4d2f
PASSWORD_HASH_ALGO = pbkdf2_hi

[service]
# DISABLE_REGISTRATION previene el auto-registro de usuarios.
# Los usuarios se aprovisionan por la fuente de autenticacion LDAP de Forgejo.
DISABLE_REGISTRATION = true
REGISTER_EMAIL_CONFIRM = false
ENABLE_NOTIFY_MAIL = false
ALLOW_ONLY_EXTERNAL_REGISTRATION = true
ENABLE_CAPTCHA = false
REQUIRE_SIGNIN_VIEW = true
DEFAULT_KEEP_EMAIL_PRIVATE = false
DEFAULT_ALLOW_CREATE_ORGANIZATION = true
DEFAULT_ENABLE_TIMETRACKING = true
NO_REPLY_ADDRESS = noreply.jackdevops.freeddns.org

[repository]
ROOT = /data/gitea/data/forgejo-repositories

[lfs]
PATH = /data/gitea/data/lfs

[mailer]
ENABLED = false

[openid]
ENABLE_OPENID_SIGNIN = false
ENABLE_OPENID_SIGNUP = false
```

```
[cron.update_checker]
ENABLED = true

[session]
PROVIDER = file

[log]
MODE = console
LEVEL = info
ROOT_PATH = /data/gitea/log

[repository.pull-request]
DEFAULT_MERGE_STYLE = merge

[repository.signing]
DEFAULT_TRUST_MODEL = committer

[oauth2]
JWT_SECRET = z5oJn1E_3sTxOfP2jjes4BKOFYkGHYhrAECMR9LGXjE

[actions]
ENABLED = true
DEFAULT_ACTIONS_URL = https://gitea.com
```

La directiva `ALLOW_ONLY_EXTERNAL_REGISTRATION = true` garantiza que los usuarios solo pueden acceder mediante credenciales LDAP, sin posibilidad de registro local, manteniendo OpenLDAP como única fuente de identidades.

5.3.1. Problemas encontrados y soluciones

Durante el despliegue de Forgejo se identificaron y resolvieron dos incidencias:

Error de permisos en app.ini. El fichero `app.ini` tenía como propietario al usuario `root`, pero Forgejo se ejecuta dentro del contenedor como el usuario `git` (UID 1000). Esto impedía que el proceso escribiera en su fichero de configuración. La solución fue corregir la propiedad:

```
sudo chown 1000:1000 /srv/devops/configs/forgejo/app.ini
sudo chown -R 1000:1000 /srv/devops/data/forgejo
```

Conflicto de puertos SSH. Al configurar el mapeo `2223:22`, el contenedor entraba en un bucle de reinicios con el error `address already in use`. La causa fue que el entrypoint de la imagen arranca un proceso OpenSSH en el puerto 22 del contenedor antes de lanzar Forgejo, de modo que ambos procesos competían por el mismo puerto. La solución fue configurar Forgejo para escuchar en el puerto 2222 interno y ajustar el mapeo a `2223:2222`.

Base de datos fuera del volumen persistente. La ruta relativa `PATH = data/forgejo.db` hacía que SQLite almacenara la base de datos en el sistema de ficheros del contenedor, perdiéndose al recrearlo. Se corrigió a una ruta absoluta dentro del volumen:

```
PATH = /data/gitea/data/forgejo.db
```

Error de red en el runner CI/CD. Al ejecutar un workflow, el runner lanzaba contenedores efímeros para cada job, pero estos se creaban en una red Docker aislada sin acceso a la red interna `compose_devops-net`. Esto impedía que el contenedor del job pudiera clonar el repositorio desde `172.20.0.3:3000`, fallando con el error *Failed to connect to 172.20.0.3 port 3000*. La solución fue habilitar Actions en `app.ini` y configurar el fichero `config.yaml` del runner para que los contenedores efímeros se conecten a `compose_devops-net`:

```
[actions]
ENABLED = true
DEFAULT_ACTIONS_URL = https://gitea.com
```

Del fichero `config.yaml` se habla y se muestra más adelante.

5.3.2. Creación del usuario administrador

Con el asistente web deshabilitado:

```
INSTALL_LOCK = true
```

El usuario administrador se creó mediante la interfaz de línea de comandos de Forgejo:

```
docker exec -u git devops-forgejo /app/gitea/gitea admin user create \
  --username jakdevops \
  --email jakdevops@jackdevops.freeddns.org \
  --password "pr0y3ct02ASIR" \
  --admin
```

Figura 9: Consulta directa a la base de datos SQLite

- `is_admin=1` declara que es administrador.
- `login_type=0` declara que es un usuario local que no pertenece a LDAP.

Este usuario local se mantiene como cuenta de emergencia para operaciones administrativas en caso de indisponibilidad del servicio LDAP.

5.3.3. Integración con OpenLDAP

La integración LDAP se configuró desde el panel de administración de Forgejo (*Administración del Sitio → Identidad y Acceso → Fuentes de Autenticación*), seleccionando el tipo **LDAP (via BindDN)**. Los parámetros configurados fueron:

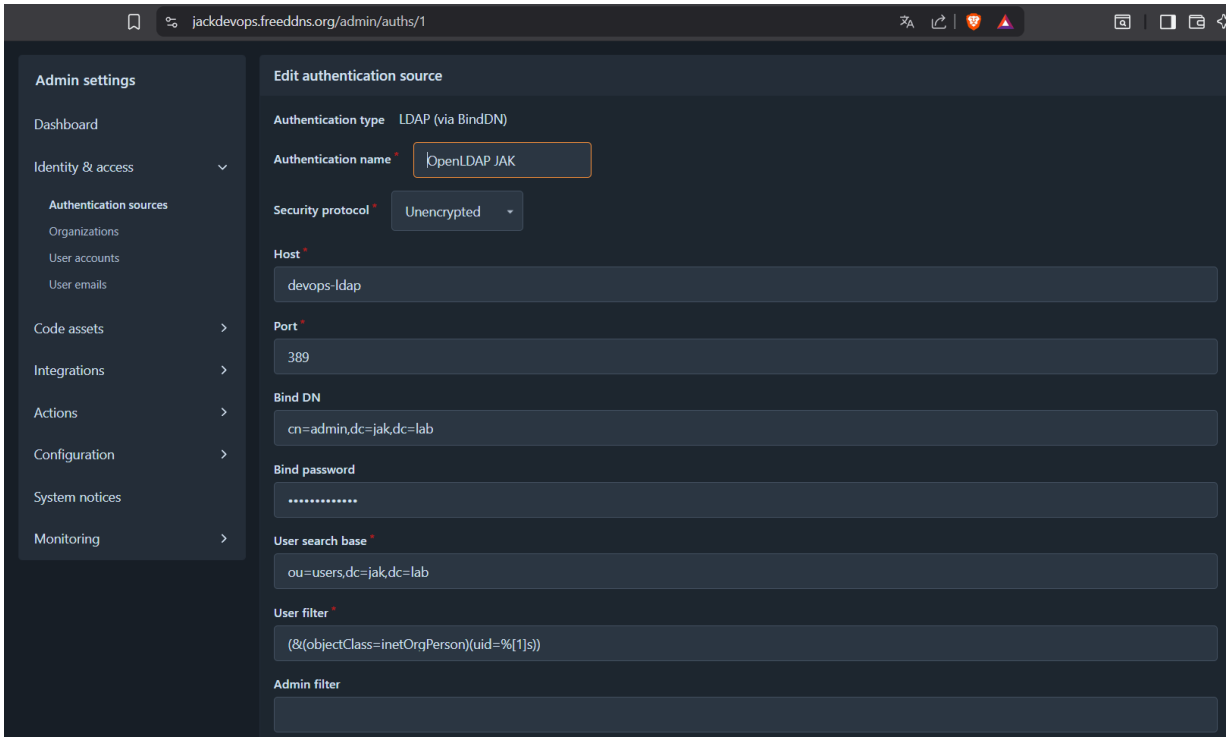


Figura 10: Parametros configurados.

Figura 11: Parametros configurados.

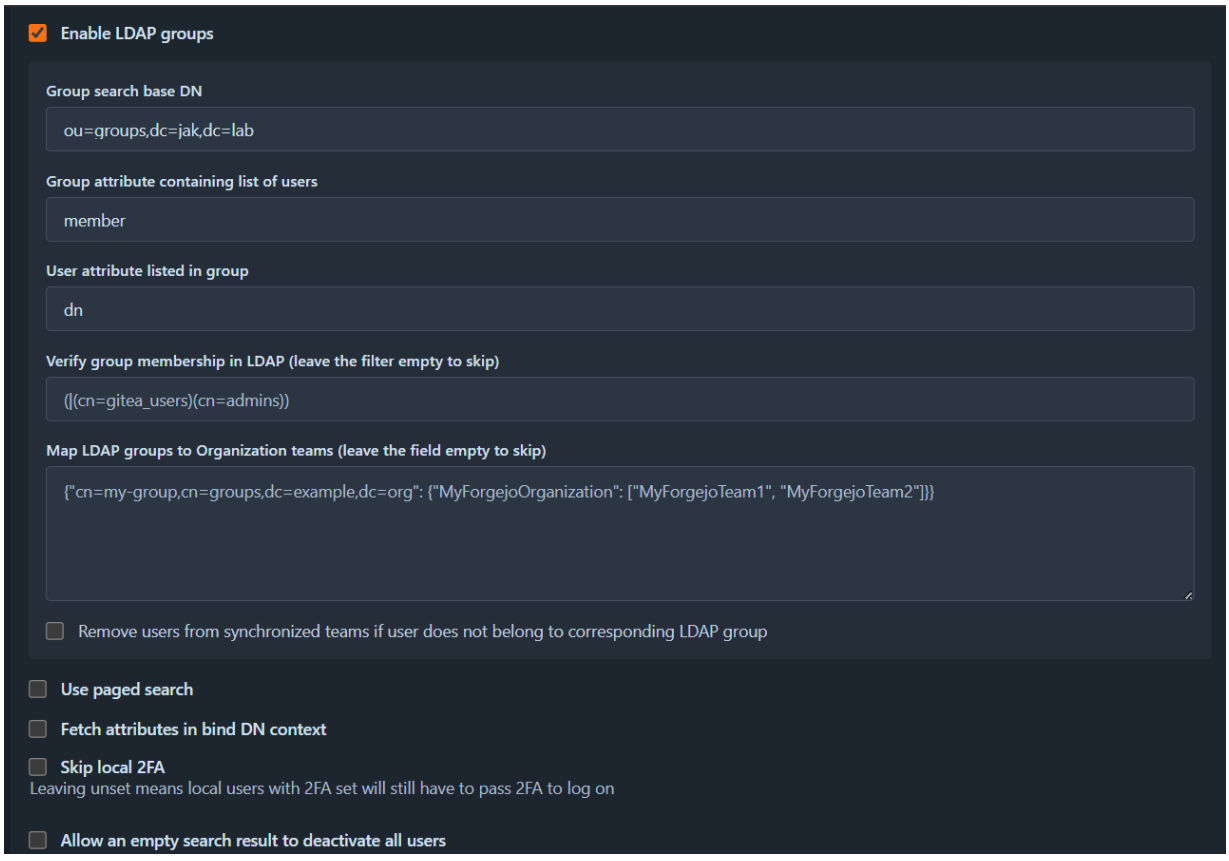


Figura 12: Parametros configurados.

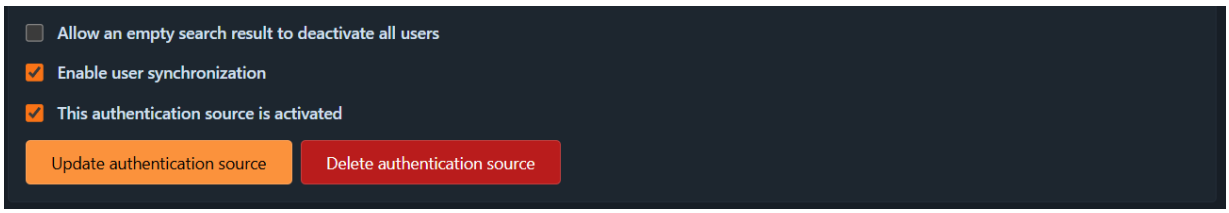


Figura 13: Parametros configurados.

Con esta configuración, cuando un usuario LDAP inicia sesión por primera vez, Forgejo realiza un bind con la cuenta de servicio, busca el usuario en el directorio, verifica las credenciales mediante un segundo bind y crea automáticamente el perfil en su base de datos local.

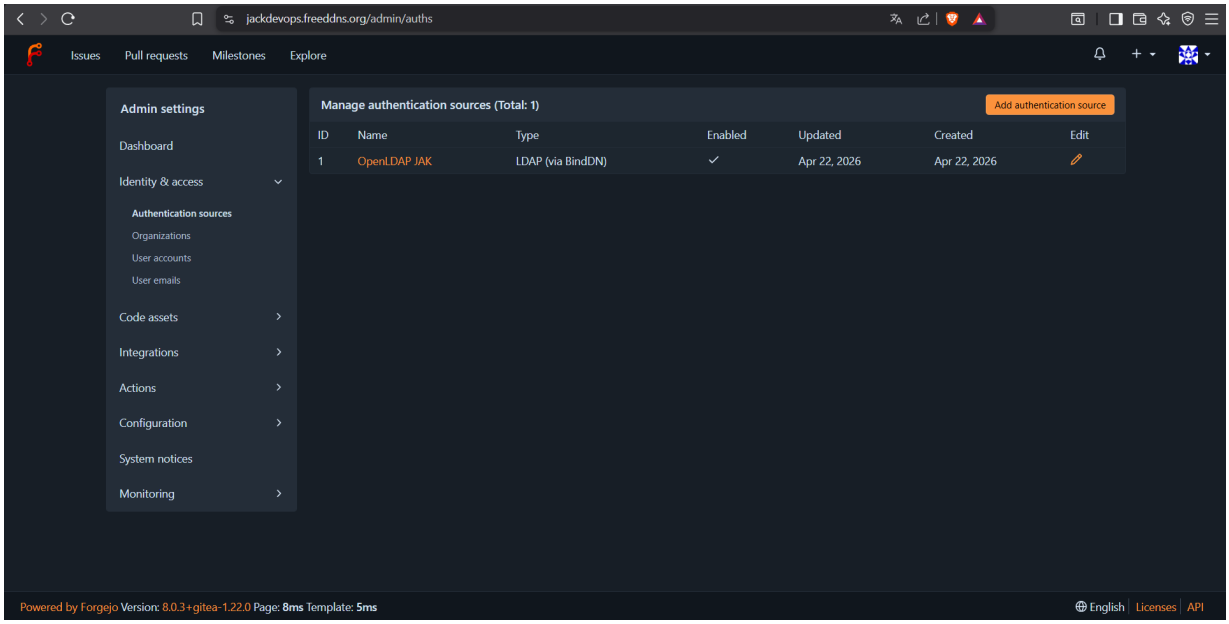


Figura 14: Panel de administración de Forgejo mostrando la fuente de autenticación `OpenLDAP JAK` activa y habilitada.

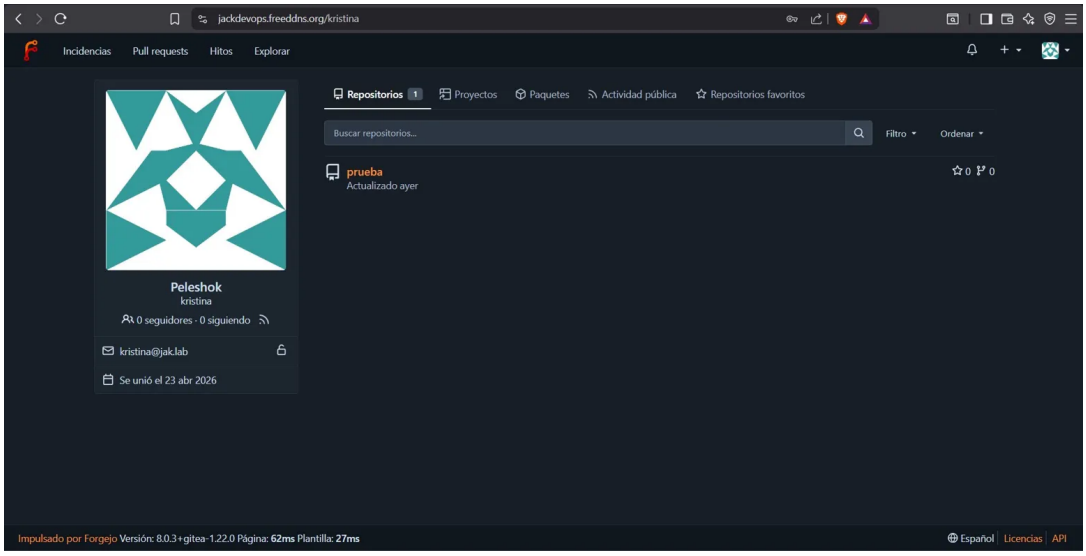


Figura 15: Perfil del usuario LDAP `kristina` en Forgejo, creado automáticamente en el primer inicio de sesión.

5.4. Nginx

Nginx se desplegó con la imagen `nginx:stable-alpine`, asignada a la IP interna `172.20.0.2` y con los puertos 80 y 443 publicados en el host. Actúa como único punto de entrada desde internet, realizando la terminación SSL/TLS y enrutando el tráfico hacia Forgejo o el Portal según el dominio de la petición.

La configuración se estructura en dos ficheros: `nginx.conf` con los parámetros globales, y `forgejo.conf` en el directorio `conf.d/` con la definición del virtualhost.

- En el fichero `nginx.conf` se encuentra:

```
user nginx;
worker_processes auto;
worker_rlimit_nofile 65535;
error_log /var/log/nginx/error.log warn;
pid /var/run/nginx.pid;

events {
    worker_connections 2048;
    use epoll;
    multi_accept on;
}

http {
    include /etc/nginx/mime.types;
    default_type application/octet-stream;

    # ========================================
    # LOGGING AVANZADO
    # ========================================
    log_format main '$remote_addr - $remote_user [$time_local] "$request" '
                    '$status $body_bytes_sent "$http_referer" '
                    '"$http_user_agent" "$http_x_forwarded_for"';

    log_format detailed '$remote_addr - $remote_user [$time_local] '
                        '"$request" $status $body_bytes_sent '
                        '"$http_referer" "$http_user_agent" '
                        'rt=$request_time uct="$upstream_connect_time" '
                        'uht="$upstream_header_time" urt="$upstream_response_time" '
                        'cs=$upstream_cache_status';

    access_log /var/log/nginx/access.log main;
    access_log /var/log/nginx/detailed.log detailed buffer=32k flush=5s;

    # ========================================
    # OPTIMIZACION DE RENDIMIENTO
    # ========================================
    sendfile on;
    tcp_nopush on;
    tcp_nodelay on;
    keepalive_timeout 65;
    keepalive_requests 100;
    types_hash_max_size 2048;
    client_max_body_size 100m;
    client_body_buffer_size 1m;
    client_header_buffer_size 1k;
    large_client_header_buffers 4 16k;

    # ========================================
    # SEGURIDAD
    # ========================================
    server_tokens off;
    disable_symlinks on;

    # Hide Nginx version in error pages
    map $status $status_text {
        400 'Bad Request';
```

```
        401 'Unauthorized';
        403 'Forbidden';
        404 'Not Found';
        405 'Method Not Allowed';
        408 'Request Timeout';
        429 'Too Many Requests';
        500 'Internal Server Error';
        502 'Bad Gateway';
        503 'Service Unavailable';
        504 'Gateway Timeout';
        default '';
    }

    # ========================================
    # COMPRESION GZIP
    # ========================================
    gzip on;
    gzip_vary on;
    gzip_min_length 1024;
    gzip_proxied any;
    gzip_comp_level 6;
    gzip_types
        text/plain
        text/css
        text/xml
        text/javascript
        application/json
        application/javascript
        application/xml+rss
        application/atom+xml
        image/svg+xml;
    gzip_disable "msie6";

    # ========================================
    # CACHE
    # ========================================
    proxy_cache_path /var/cache/nginx/static levels=1:2 keys_zone=static:10m max_size=100m inactive=60m
    use_temp_path=off;
    proxy_cache_path /var/cache/nginx/api levels=1:2 keys_zone=api:10m max_size=50m inactive=10m use_temp_path=off
    ;
    proxy_cache_lock on;
    proxy_cache_lock_timeout 5s;

    # ========================================
    # RATE LIMITING: Zonas de limite
    # ========================================
    limit_req_zone $binary_remote_addr zone=general:10m rate=10r/s;
    limit_req_zone $binary_remote_addr zone=api_limit:10m rate=30r/s;
    limit_req_zone $binary_remote_addr zone=strict:10m rate=2r/s;
    limit_req_status 429;

    # ========================================
    # UPSTREAM - HEALTH CHECKS
    # ========================================
    upstream forgejo_backend {
        least_conn;
        server 172.20.0.3:3000 max_fails=3 fail_timeout=30s;
        keepalive 32;
        keepalive_requests 100;
        keepalive_timeout 60s;
    }

    # Punto de verificacion de salud usado por verificaciones de salud de Docker
    server {
        listen 80;
        listen [::]:80;
```

```
        server_name _;

        location /health {
            access_log off;
            return 200 "ok\n";
            add_header Content-Type text/plain;
        }

        # Negar acceso a otros paths cuando se accede directamente
        location / {
            return 444;
        }
    }

    include /etc/nginx/conf.d/*.conf;
}
```

- En el fichero `forgejo.conf` se encuentra:

```
# ========================================
# FORGEJO GIT PLATFORM
# Domain: jackdevops.freeddns.org
# ========================================
# SSH Git operations on port 22 bypass this nginx config entirely
# They connect directly to devops-forgejo container (172.20.0.3:22)

# ========================================
# HTTP -> HTTPS REDIRECT
# ========================================
server {
    listen 80;
    listen [::]:80;
    server_name jackdevops.freeddns.org;

    # Allow ACME validation for certificate renewal
    location /.well-known/acme-challenge/ {
        default_type text/plain;
        return 200 "Empty\n";
    }

    # Redirect all HTTP to HTTPS
    location / {
        return 301 https://$server_name$request_uri;
    }
}

# ========================================
# HTTPS MAIN SERVER
# ========================================
server {
    listen 443 ssl http2;
    listen [::]:443 ssl http2;
    server_name jackdevops.freeddns.org;

    # ========================================
    # SSL/TLS CONFIGURATION (ENTERPRISE)
    # ========================================
    ssl_certificate /etc/nginx/ssl/fullchain.pem;
    ssl_certificate_key /etc/nginx/ssl/privkey.pem;
    ssl_protocols TLSv1.2 TLSv1.3;
    ssl_ciphers 'ECDHE-ECDSA-AES128-GCM-SHA256:ECDHE-RSA-AES128-GCM-SHA256:ECDHE-ECDSA-AES256-GCM-SHA384:ECDHE-RSA
    -AES256-GCM-SHA384:ECDHE-ECDSA-CHACHA20-POLY1305:ECDHE-RSA-CHACHA20-POLY1305:DHE-RSA-AES128-GCM-SHA256:DHE-RSA
    -AES256-GCM-SHA384';
    ssl_prefer_server_ciphers on;
    ssl_session_cache shared:SSL:50m;
    ssl_session_timeout 1d;
```

```
    ssl_session_tickets off;
    ssl_stapling on;
    ssl_stapling_verify on;
    ssl_stapling_responder http://ocsp.isrg.x1.letsencrypt.org;

    # ========================================
    # CLIENT LIMITS & BUFFERS
    # ========================================
    client_max_body_size 100m;
    client_body_buffer_size 1m;
    client_header_buffer_size 1k;
    large_client_header_buffers 4 16k;

    # ========================================
    # RATE LIMITING
    # ========================================
    limit_req zone=general burst=20 nodelay;
    limit_req_status 429;

    # ========================================
    # ACCESS LOGGING
    # ========================================
    access_log /var/log/nginx/forgejo_access.log detailed;
    error_log /var/log/nginx/forgejo_error.log warn;

    # ========================================
    # ROOT LOCATION - PROXY TO FORGEJO
    # ========================================
    location / {
        # Rate limiting stricter for APIs
        limit_req zone=api_limit burst=50 nodelay;

        # ========================================
        # SECURITY HEADERS (MUST BE HERE AFTER PROXY_PASS)
        # ========================================
        add_header Strict-Transport-Security "max-age=63072000; includeSubDomains; preload" always;
        add_header X-Frame-Options "SAMEORIGIN" always;
        add_header X-Content-Type-Options "nosniff" always;
        add_header X-XSS-Protection "1; mode=block" always;
        add_header Referrer-Policy "strict-origin-when-cross-origin" always;
        add_header Permissions-Policy "geolocation=(), microphone=(), camera=()" always;
        add_header Content-Security-Policy "default-src 'self' https://jackdevops.freeddns.org; script-src 'self'
    'unsafe-inline' 'unsafe-eval'; style-src 'self' 'unsafe-inline'; img-src 'self' data: https:; font-src 'self'
    data:; connect-src 'self' https://jackdevops.freeddns.org;" always;

        # Cache configuration
        proxy_cache api;
        proxy_cache_key "$scheme$proxy_host$request_uri";
        proxy_cache_valid 200 30s;
        proxy_cache_valid 404 1m;
        proxy_cache_bypass $http_pragma $http_authorization;
        add_header X-Cache-Status $upstream_cache_status;

        # Proxy pass to upstream
        proxy_pass http://devops-forgejo:3000;

        # ========================================
        # PROXY HEADERS - SECURITY & TRANSPARENCY
        # ========================================
        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;
        proxy_set_header X-Forwarded-Host $server_name;
        proxy_set_header X-Forwarded-By $hostname;
        proxy_set_header X-Request-ID $request_id;
```

```
        # ========================================
        # PROXY BUFFERING - OPTIMIZED
        # ========================================
        proxy_buffering on;
        proxy_buffer_size 4k;
        proxy_buffers 16 4k;
        proxy_busy_buffers_size 16k;
        proxy_max_temp_file_size 2m;
        proxy_temp_file_write_size 32k;

        # ========================================
        # TIMEOUTS - ENTERPRISE
        # ========================================
        proxy_connect_timeout 60s;
        proxy_send_timeout 300s;
        proxy_read_timeout 300s;

        # ========================================
        # KEEP-ALIVE UPSTREAM
        # ========================================
        proxy_http_version 1.1;
        proxy_set_header Connection "";
    }

    # ========================================
    # LOCATION: /api - STRICTER RATE LIMITING
    # ========================================
    location /api {
        limit_req zone=strict burst=5 nodelay;

        proxy_pass http://devops-forgejo:3000;
        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;
        proxy_http_version 1.1;
        proxy_set_header Connection "";
    }

    # ========================================
    # LOCATION: Static files - CACHE 30 DAYS
    # ========================================
    location ~* \.(js|css|png|jpg|jpeg|gif|ico|svg|woff|woff2|ttf|eot)$ {
        proxy_cache static;
        proxy_cache_valid 200 30d;
        add_header Cache-Control "public, max-age=2592000, immutable";
        add_header X-Cache-Status $upstream_cache_status;

        proxy_pass http://devops-forgejo:3000;
        proxy_set_header Host $host;
        proxy_http_version 1.1;
        proxy_set_header Connection "";
    }
}
```

5.4.1. Proxy inverso y enrutamiento

Para el virtualhost `jackdevops.freeddns.org` se define un bloque de redirección HTTP a HTTPS que fuerza el uso de conexiones cifradas, y un bloque HTTPS con `proxy_pass` hacia Forgejo en la red interna:

```
# jackdevops.freeddns.org -> Forgejo
proxy_pass http://172.20.0.3:3000;
```

El backend se define mediante un bloque `upstream` con detección pasiva de fallos (`max_fails=3 fail_timeout=30s`) y

reutilización de conexiones TCP mediante `keepalive 32`, mejorando la disponibilidad y el rendimiento del proxy.

5.4.2. Cabeceras de seguridad HTTP

Se implementaron siete cabeceras de seguridad en todos los bloques `location`, posicionadas después de `proxy_pass` para garantizar su aplicación también en respuestas de error:

```
add_header Strict-Transport-Security
           "max-age=63072000; includeSubDomains; preload" always;
add_header X-Frame-Options          "SAMEORIGIN"   always;
add_header X-Content-Type-Options   "nosniff"      always;
add_header X-XSS-Protection         "1; mode=block" always;
add_header Content-Security-Policy  "default-src 'self'; ..." always;
add_header Referrer-Policy          "strict-origin-when-cross-origin" always;
add_header Permissions-Policy       "geolocation=(), microphone=(), camera=()" always;
```

Estas cabeceras protegen contra ataques comunes:

- **HSTS**: Fuerza HTTPS durante 2 años y habilita preload en navegadores
- **X-Frame-Options**: Previene ataques de clickjacking permitiendo solo frames del mismo origen
- **X-Content-Type-Options**: Evita que el navegador interprete archivos con MIME type incorrecto
- **X-XSS-Protection**: Activa el filtro anti-XSS del navegador
- **CSP**: Restringe las fuentes de contenido permitidas
- **Referrer-Policy**: Controla qué información de referencia se envía en peticiones
- **Permissions-Policy**: Deshabilita APIs sensibles como geolocalización y cámara

5.4.3. Problema de healthcheck con IPv6

Durante el despliegue, el healthcheck definido en Docker Compose ejecutaba `wget` contra `http://localhost/health`. En sistemas con IPv6 activo, `wget` intentaba conectar primero a `[::1]:80` (localhost IPv6) antes de `127.0.0.1:80` (localhost IPv4), pero Nginx solo escuchaba en IPv4, provocando que el contenedor apareciera como *unhealthy*.

```
server {
    listen 80;
    listen [::]:80;
    location /health {
        access_log off;
        return 200 "ok\n";
    }
}
```

5.5. Runner de CI/CD

El runner de Forgejo Actions se desplegó mediante la imagen `gitea/act_runner:latest`, asignada a la IP interna 172.20.0.6. Este componente ejecuta los workflows definidos en los repositorios, creando contenedores efímeros para cada job mediante Docker-in-Docker.

5.5.1. Configuración en Docker Compose

El runner se auto-registra al arrancar mediante variables de entorno y lee su configuración desde un archivo YAML montado como volumen:

```
  devops-runner:
    image: gitea/act_runner:latest
    container_name: devops-runner
    restart: unless-stopped
    networks:
      devops-net:
        ipv4_address: 172.20.0.6
    dns:
    - 8.8.8.8
```

```
    - 1.1.1.1
    volumes:
      - /var/run/docker.sock:/var/run/docker.sock
      - /srv/devops/data/runner/data:/data
    environment:
      - GITEA_INSTANCE_URL=${FORGEJO_INSTANCE_URL}
      - GITEA_RUNNER_REGISTRATION_TOKEN=${FORGEJO_RUNNER_REGISTRATION_TOKEN}
      - CONFIG_FILE=/data/config.yaml
    deploy:
      resources:
        limits:
          memory: 3072m
          cpus: "2.0"
    depends_on:
      devops-forgejo:
        condition: service_healthy
```

El montaje de `/var/run/docker.sock` permite al runner crear contenedores Docker-in-Docker para ejecutar los jobs. La variable `CONFIG_FILE` indica la ruta del archivo de configuración personalizado, mientras que `depends_on` asegura que Forgejo esté operativo antes de iniciar el runner.

El concepto Docker-in-Docker (DinD) se refiere a la capacidad de un contenedor de crear y gestionar otros contenedores. En este caso, el runner no ejecuta los workflows directamente en su propio contenedor, sino que crea contenedores efímeros e independientes para cada job. Al montar el socket de Docker del host (`/var/run/docker.sock`), el runner obtiene acceso al daemon de Docker y puede lanzar contenedores hermanos en el mismo nivel que él. Cuando un workflow se dispara, el runner lee las especificaciones del job (imagen base, comandos a ejecutar, etc.), solicita al daemon de Docker la creación de un nuevo contenedor con esos parámetros, ejecuta los pasos definidos y, al finalizar, destruye automáticamente el contenedor. Este enfoque garantiza aislamiento entre jobs y permite usar diferentes entornos de ejecución sin contaminar el estado del runner.

5.5.2. Archivo de configuración del runner

El archivo `config.yaml` define la red Docker que usarán los contenedores de los jobs y las imágenes disponibles para ejecutar workflows:

```
container:
  network: compose_devops-net
  privileged: false
  valid_volumes:
    - '**'

runner:
  labels:
    - ubuntu-latest:docker://catthehacker/ubuntu:act-latest
    - ubuntu-22.04:docker://catthehacker/ubuntu:act-22.04
    - ubuntu-24.04:docker://catthehacker/ubuntu:act-24.04
```

La directiva `network` es crítica: garantiza que los contenedores de los jobs se creen en la misma red Docker que Forgejo, permitiendo el acceso directo por IP interna sin necesidad de exponer puertos adicionales. El parámetro `privileged: false` restringe las capacidades de los contenedores por seguridad, mientras que `valid_volumes` permite el montaje de volúmenes necesarios para ciertos workflows.

5.5.3. Resolución de problema: Aislamiento de red en jobs

Durante las pruebas iniciales, los workflows fallaban sistemáticamente al intentar clonar repositorios con `actions/checkout@v3`, mostrando errores de conexión rechazada. Al inspeccionar los contenedores en ejecución se observó que los jobs se creaban en redes aisladas generadas automáticamente (con nombres como `GITEA-ACTIONS-TASK-X-...`), sin acceso a la red `compose_devops-net` donde reside Forgejo.

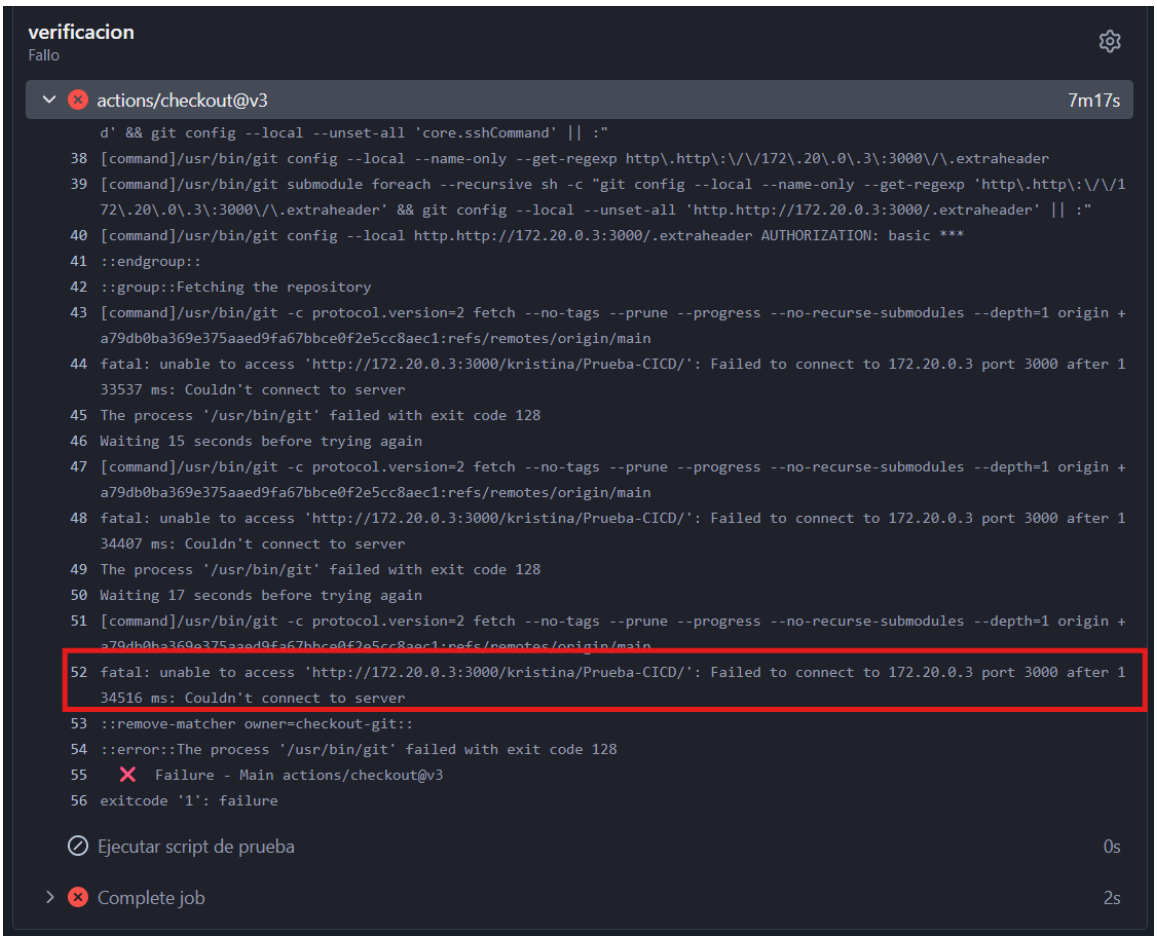


Figura 16: Error de conexión.

La causa del problema fue que el runner ignoraba el archivo `.runner.yaml` montado. Durante el auto-registro inicial, el runner genera automáticamente un archivo `.runner` con configuración por defecto, y este archivo tiene prioridad sobre el `.runner.yaml` personalizado.

Para resolver el problema se eliminó el archivo `.runner` y se reinició el contenedor, forzando un nuevo registro que leyera correctamente la configuración personalizada. También se añadió un parámetro más en `docker-compose.yml`, del que se habló anteriormente:

```
docker compose stop devops-runner
rm /srv/devops/data/runner/data/.runner
docker compose up -d devops-runner
```

```
    environment:
      - GITEA_INSTANCE_URL=${FORGEJO_INSTANCE_URL}
      - GITEA_RUNNER_REGISTRATION_TOKEN=${FORGEJO_RUNNER_REGISTRATION_TOKEN}
      - CONFIG_FILE=/data/config.yaml # -> obliga a usar ese fichero de configuracion
```

Tras el nuevo registro, los contenedores de los jobs se crearon correctamente en `compose_devops-net`, permitiendo el acceso directo a Forgejo y solucionando los fallos de clonado:

```
root@JAKserver:/srv/devops/configs/ldap# docker ps --format "table {{.Names}}\t{{.Networks}}"
NAMES                                                          NETWORKS
GITEA-ACTIONS-TASK-22_WORKFLOW-Pipeline-de-prueba_JOB-verificacion   compose_devops-net
devops-nginx                                                   compose_devops-net
devops-runner                                                  compose_devops-net
devops-forgejo                                                 compose_devops-net
devops-portal                                                  compose_devops-net
devops-ldap                                                    compose_devops-net
portainer                                                      bridge
root@JAKserver:/srv/devops/configs/ldap#
```

Figura 17: Verificación de los contenedores de los jobs conectados a la red compartida.

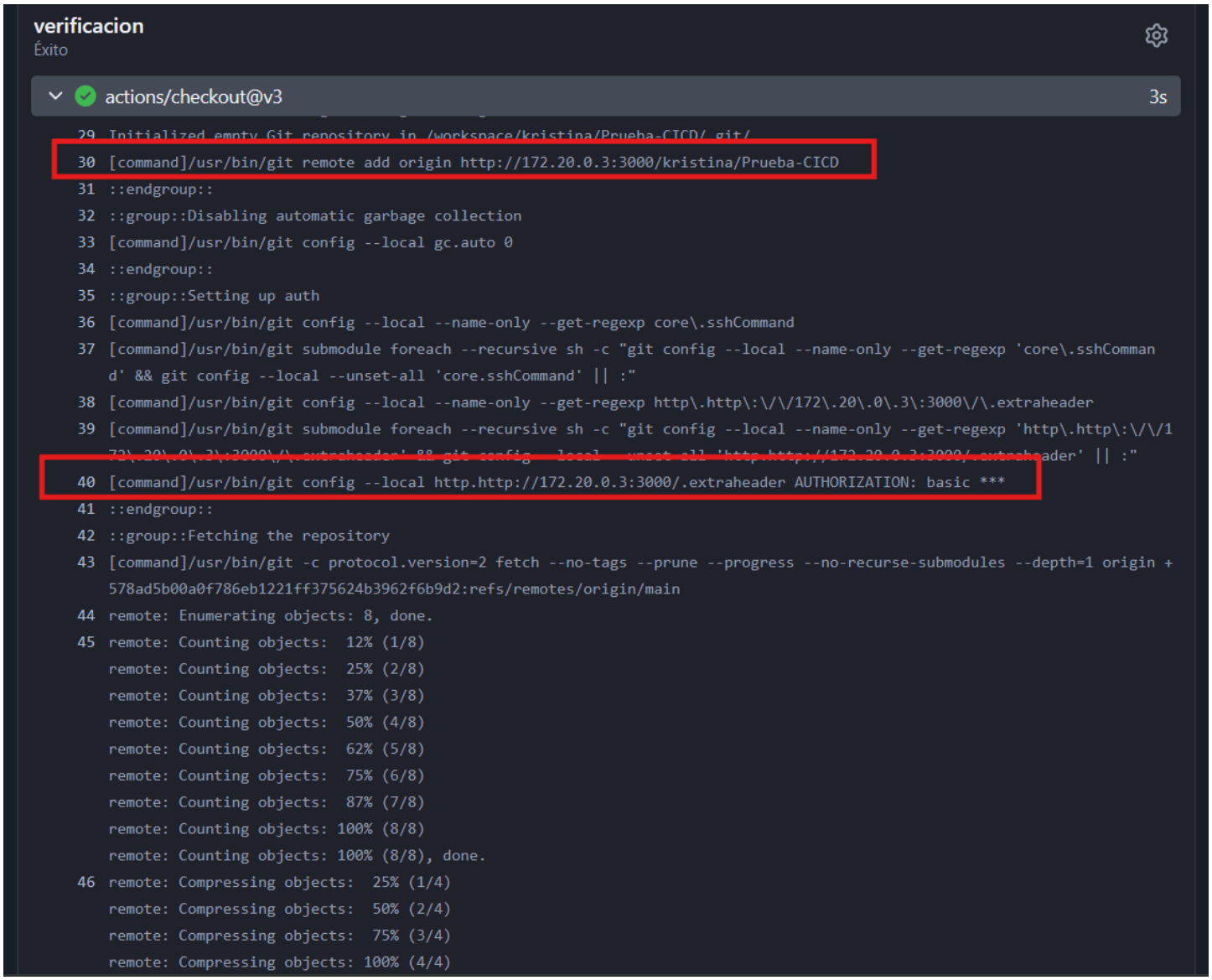


Figura 18: Se puede observar como la verificación se realiza correctamente.

5.6. Verificación del stack completo

Una vez desplegados todos los servicios, se verificó el estado del stack mediante:

```
docker compose ps
```

Figura 19: Estado del stack completo tras el despliegue. Todos los contenedores reportan estado *healthy*.

La conectividad entre servicios se validó comprobando que Nginx obtenía respuesta HTTP 200 de Forgejo y que Forgejo alcanzaba el puerto 389 de OpenLDAP. El acceso SSH de Git se verificó con un ciclo completo de `clone`, `commit` y `push` desde una máquina cliente externa, obteniendo respuesta satisfactoria del servidor en el puerto 2223.

6. Seguridad

La seguridad de la plataforma se aborda en tres niveles: el perímetro de red, la capa de transporte y la configuración de cada servicio. Esta sección describe las medidas implementadas en cada uno de ellos.

6.1. Aislamiento de red

Todos los servicios se comunican a través de la red Docker privada `devops-net` (172.20.0.0/24), con direcciones IP estáticas asignadas a cada contenedor. Ningún servicio interno —OpenLDAP, Forgejo HTTP— tiene puertos publicados en el host salvo los estrictamente necesarios. En particular, el puerto 389 de LDAP nunca se expone al exterior: las credenciales de usuario viajan sin cifrar dentro de la red interna, lo cual es aceptable dado que el tráfico no abandona el host, pero sería inaceptable si el puerto estuviese accesible desde internet.

El único punto de entrada desde el exterior es Nginx, que actúa como frontera entre internet y la red interna.

6.2. SSL/TLS

Nginx realiza la terminación SSL/TLS con certificados emitidos por Let's Encrypt mediante Certbot. La configuración restringe los protocolos a TLS 1.2 y TLS 1.3, descartando versiones anteriores con vulnerabilidades conocidas, y limita los conjuntos de cifrado a suites basadas en ECDHE, que garantizan confidencialidad hacia el futuro (*Perfect Forward Secrecy*):

```
ssl_protocols TLSv1.2 TLSv1.3;
ssl_ciphers   'ECDHE-RSA-AES128-GCM-SHA256:ECDHE-RSA-AES256-GCM-SHA384';
ssl_prefer_server_ciphers on;
ssl_session_tickets off;
```

La directiva `ssl_session_tickets off` desactiva los tickets de sesión TLS para eliminar el riesgo de que una filtración futura de la clave del servidor permita descifrar tráfico capturado anteriormente.

Todo el tráfico HTTP entrante se redirige automáticamente a HTTPS mediante un bloque `server` en Nginx, de modo que ninguna petición puede llegar a los backends sin cifrado.

6.3. Cabeceras de seguridad HTTP

Se implementaron siete cabeceras de seguridad en todas las respuestas, alineadas con las recomendaciones de OWASP:

Cabecera	Función
`Strict-Transport-Security`	Fuerza HTTPS durante 730 días e incluye el dominio en la preload list de los navegadores.
`X-Frame-Options`	Impide que la página sea incrustada en un iframe desde otro origen, previniendo ataques de clickjacking.
`X-Content-Type-Options`	Desactiva el MIME sniffing del navegador, evitando que ficheros subidos con tipo incorrecto se ejecuten como scripts.
`X-XSS-Protection`	Activa el filtro XSS nativo de navegadores legacy.
`Content-Security-Policy`	Define la lista blanca de orígenes desde los que el navegador puede cargar recursos.
`Referrer-Policy`	Limita la información de referencia enviada al navegar a otros dominios.
`Permissions-Policy`	Desactiva el acceso a APIs sensibles del navegador (geolocalización, micrófono, cámara).

Estas cabeceras se configuran en el bloque `server` de Nginx y se aplican a todas las respuestas servidas por la plataforma.

6.4. Hardening SSH

El daemon SSH del sistema operativo fue reconfigurado al puerto 2222, siguiendo las buenas prácticas de reducción de superficie de ataque. Se deshabilitó la autenticación por contraseña, de modo que el acceso administrativo al servidor requiere exclusivamente clave pública.

6.5. Limitación de tasa de peticiones

Nginx incorpora tres zonas de limitación de tasa para proteger la plataforma frente a ataques de fuerza bruta y denegación de servicio:

- **Zona general** (10 req/s): aplicada al tráfico web ordinario.
- **Zona API** (30 req/s): para endpoints de la API de Forgejo con mayor throughput legítimo.
- **Zona estricta** (2 req/s): aplicada a endpoints críticos como autenticación, donde un volumen elevado de peticiones podría indicar un intento de fuerza bruta sobre credenciales.

Las peticiones que superan el límite reciben una respuesta HTTP 429 (**Too Many Requests**).

6.6. Consideraciones de seguridad del runner

El runner de CI/CD monta el socket del daemon Docker del host (`/var/run/docker.sock`), lo que le otorga acceso equivalente a privilegios de administrador sobre el sistema. Este nivel de acceso es aceptable en el contexto del proyecto, dado que el entorno es de un único inquilino, los usuarios que definen los workflows son de confianza y el historial Git permite auditar cualquier cambio en las definiciones de los pipelines. En un entorno de producción con usuarios no confiables se recomendaría desplegar el runner en un nodo separado o evaluar alternativas como Docker-in-Docker con contenedores privilegiados aislados.

7. Pruebas y Validación

7.1. Verificación de servicios

Tras el despliegue completo se verificó el estado de cada contenedor mediante `docker compose ps`, comprobando que todos reportaban estado *healthy*. La conectividad entre servicios se validó con las siguientes comprobaciones:

```
# Nginx -> Forgejo (debe devolver HTTP 200)
docker exec devops-nginx curl -s -o /dev/null -w "%{http_code}" http://devops-forgejo:3000

# Forgejo -> LDAP (debe imprimir "OK")
docker exec devops-forgejo sh -c "nc -z devops-ldap 389 && echo OK"
```

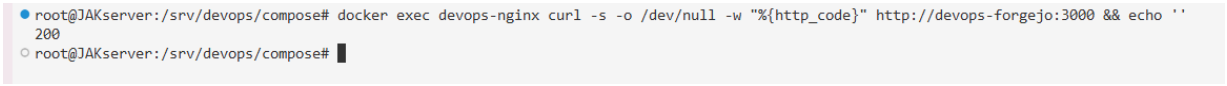


Figura 20: Prueba de conectividad Nginx-Forgejo (HTTP 200).

Figura 21: Prueba de conectividad Forgejo-LDAP (puerto 389).

Ambas comprobaciones devolvieron el resultado esperado.

7.2. Verificación SSL/TLS

Se comprobó que los certificados Let's Encrypt estaban activos y correctamente servidos mediante:

```
curl -I https://jackdevops.freeddns.org/
```

La respuesta confirmó protocolo HTTP/2, certificado válido emitido por Let's Encrypt y presencia de todas las cabeceras de seguridad implementadas.

Figura 22: Respuesta de `curl -I` mostrando las cabeceras de seguridad HTTP implementadas en Nginx.

7.3. Verificación de autenticación LDAP

Se realizó un inicio de sesión en Forgejo con el usuario LDAP `kristina` y su contraseña correspondiente. Forgejo consultó el directorio OpenLDAP, verificó las credenciales y creó automáticamente el perfil del usuario en su base de datos local. El acceso fue satisfactorio, confirmando el correcto funcionamiento de la integración LDAP.

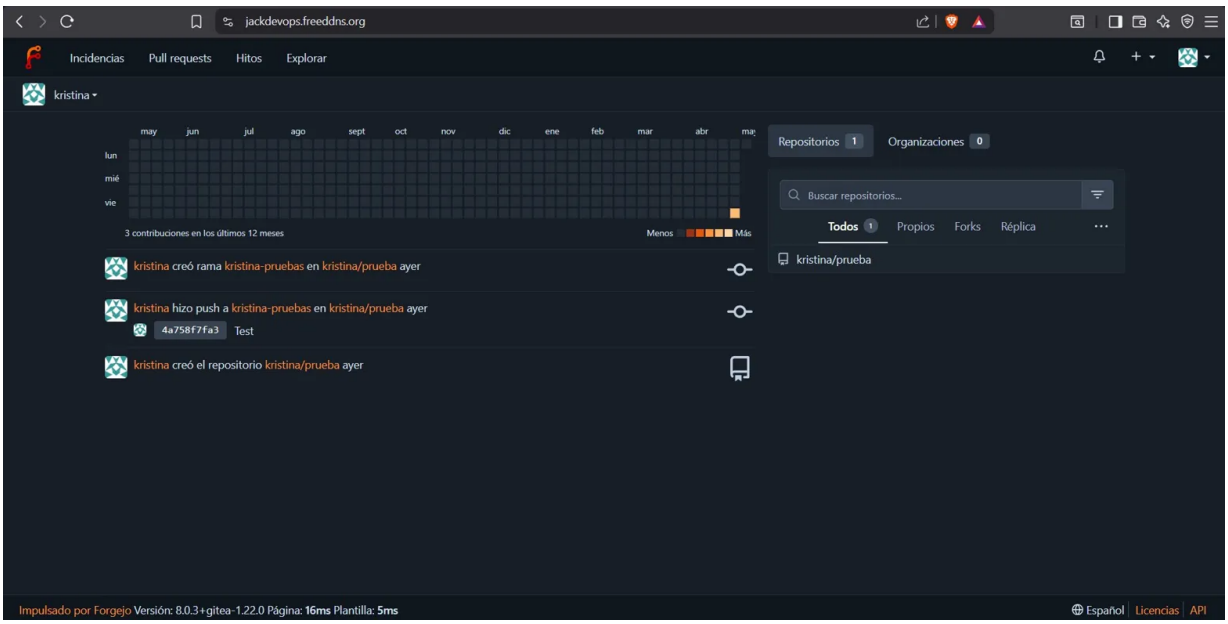


Figura 23: Dashboard del usuario LDAP `kristina` tras iniciar sesión, con su actividad reciente y repositorios visibles.

7.4. Verificación del pipeline CI/CD

Se creó un repositorio de prueba con el siguiente workflow en `.gitea/workflows/ci.yaml`:

```
name: Pipeline de prueba
on:
  push:
    branches:
      - main

jobs:
  verificacion:
    runs-on: ubuntu-latest
    steps:
      - uses: actions/checkout@v3
      - name: Ejecutar script de prueba
        run: bash test.sh
```

El script `test.sh` contiene lo siguiente:

```
#!/bin/bash
echo "Fecha"
date
echo "Directorios"
pwd
echo "Archivos"
ls -l
echo "Funciona correctamente"
```

Tras realizar un `git push` a la rama principal, el runner detectó el evento, descargó el código, levantó el contenedor de ejecución y completó el trabajo con éxito. El resultado quedó registrado en la interfaz web de Forgejo bajo la pestaña *Actions* del repositorio. Así como se muestra en la siguiente captura:

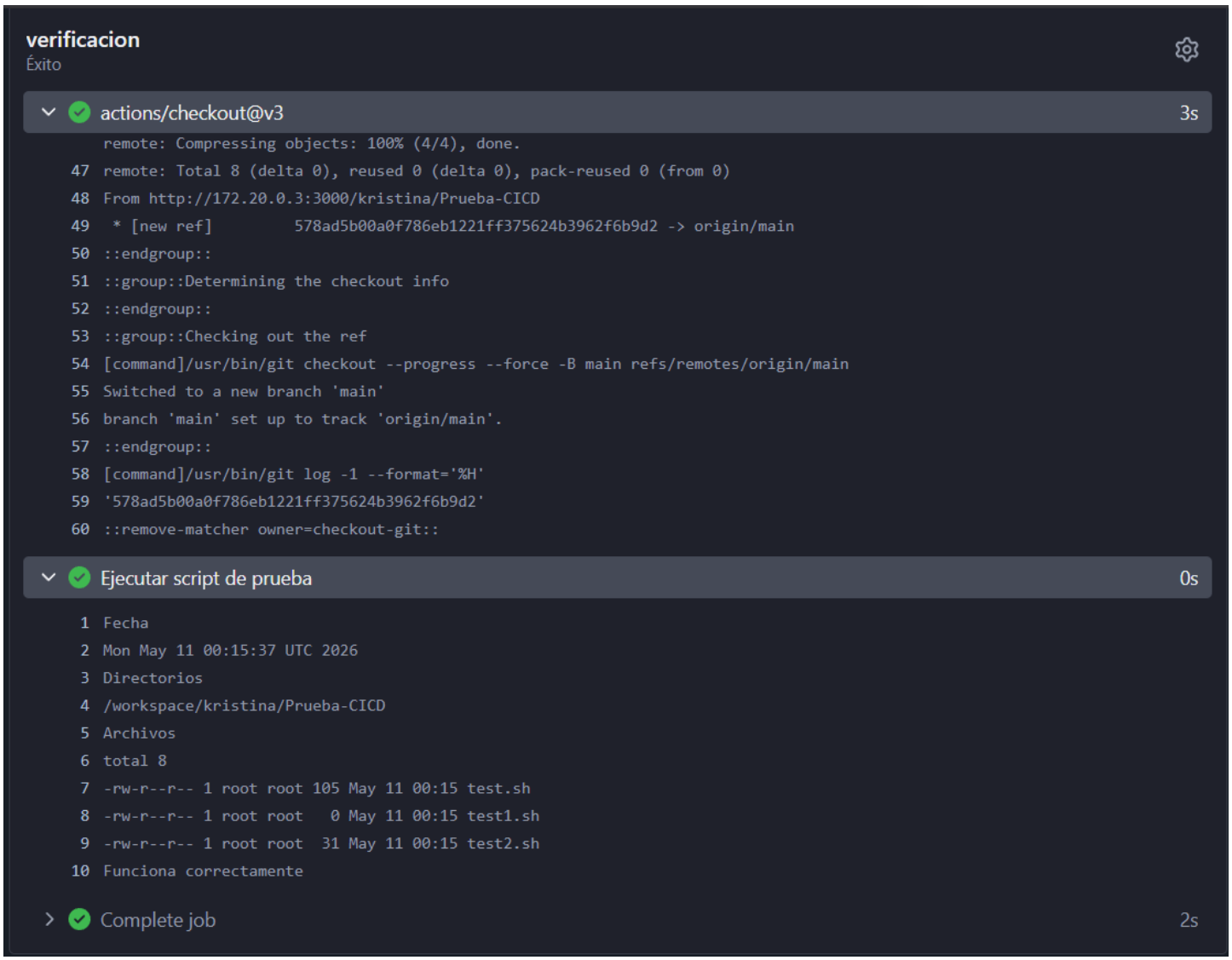


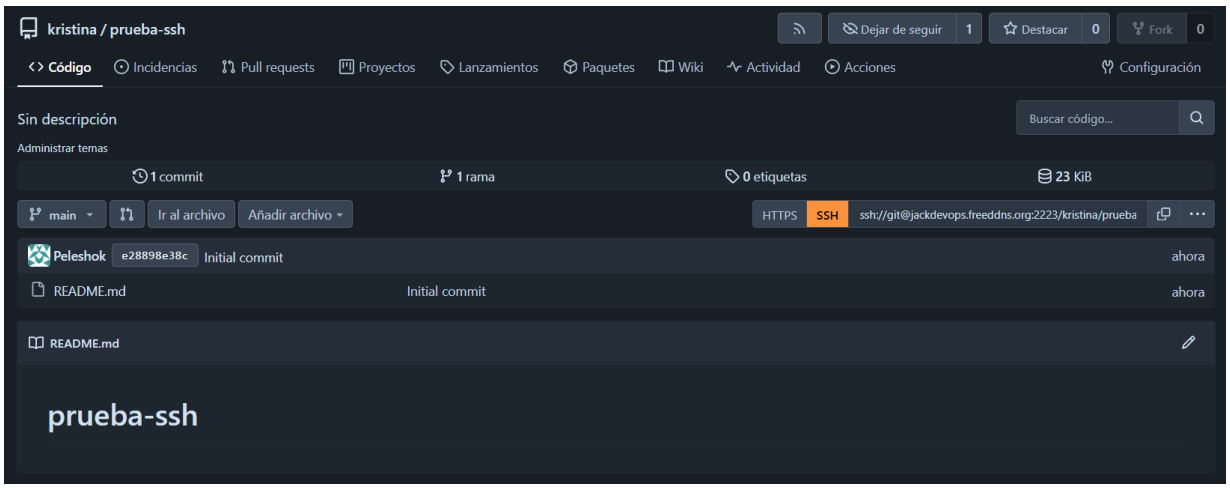
Figura 24: Verificación del workflow ejecutado correctamente.

7.5. Verificación de operaciones Git por SSH

Desde una máquina cliente externa se realizó un ciclo completo de operaciones Git sobre el puerto SSH 2223:

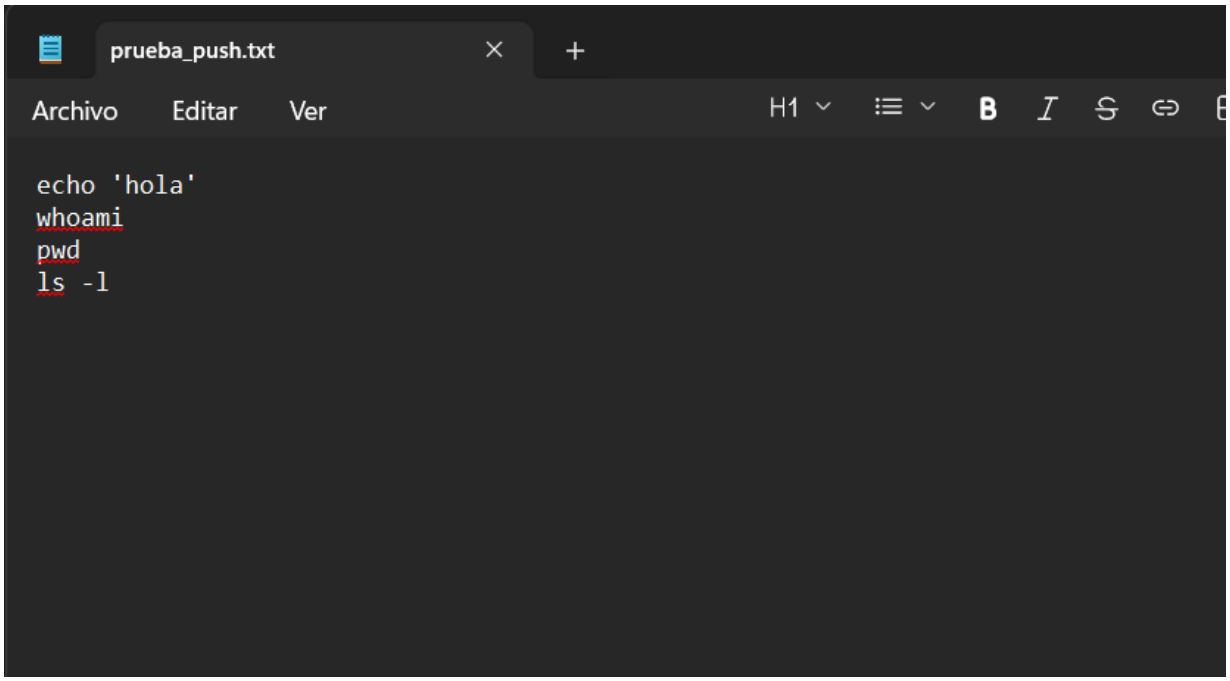
```
git clone git@jackdevops.freeddns.org:joseph/prueba-ssh.git
cd prueba-ssh
notepad prueba_push.txt
git add prueba_push.txt
git commit -m "prueba push desde cliente externo"
git push
```

El push se completó sin errores, confirmando el funcionamiento del servidor SSH de Forgejo y la autenticación mediante clave pública.




```
C:\Users\krist\Proyecto\cicd>git clone ssh://git@jackdevops.freeddns.org:222
3/kristina/prueba-ssh.git
Cloning into 'prueba-ssh'...
** WARNING: connection is not using a post-quantum key exchange algorithm.
** This session may be vulnerable to "store now, decrypt later" attacks.
** The server may need to be upgraded. See https://openssh.com/pq.html
Enter passphrase for key 'C:\Users\krist\.ssh\ssh-clave-proyecto':
remote: Enumerating objects: 3, done.
remote: Counting objects: 100% (3/3), done.
remote: Total 3 (delta 0), reused 0 (delta 0), pack-reused 0 (from 0)
Receiving objects: 100% (3/3), done.

C:\Users\krist\Proyecto\cicd>
```



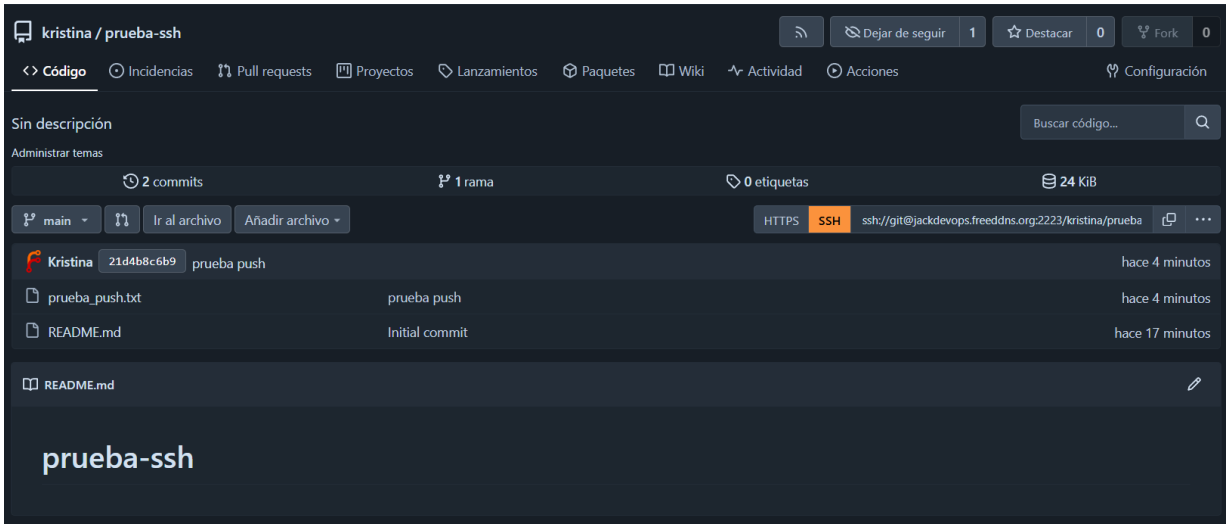
```
C:\Users\krist\Proyecto\cicd>cd prueba-ssh

C:\Users\krist\Proyecto\cicd\prueba-ssh>notepad prueba_push.txt

C:\Users\krist\Proyecto\cicd\prueba-ssh>git add prueba_push.txt
warning: in the working copy of 'prueba_push.txt', CRLF will be replaced by
LF the next time Git touches it

C:\Users\krist\Proyecto\cicd\prueba-ssh>git commit -m "prueba push"
[main 21d4b8c] prueba push
 1 file changed, 4 insertions(+)
 create mode 100644 prueba_push.txt

C:\Users\krist\Proyecto\cicd\prueba-ssh>git push
** WARNING: connection is not using a post-quantum key exchange algorithm.
** This session may be vulnerable to "store now, decrypt later" attacks.
** The server may need to be upgraded. See https://openssh.com/pq.html
Enter passphrase for key 'C:\Users\krist\.ssh\ssh-clave-proyecto':
Enumerating objects: 4, done.
Counting objects: 100% (4/4), done.
Delta compression using up to 16 threads
Compressing objects: 100% (2/2), done.
Writing objects: 100% (3/3), 316 bytes | 316.00 KiB/s, done.
Total 3 (delta 0), reused 0 (delta 0), pack-reused 0 (from 0)
To ssh://jackdevops.freeddns.org:2223/kristina/prueba-ssh.git
   37098a8..21d4b8c  main -> main
```



8. Conclusiones

El proyecto ha cumplido con los objetivos planteados. Se ha diseñado, implementado y puesto en producción una plataforma DevOps autoalojada completamente funcional, accesible de forma pública y segura, que resuelve los problemas de dependencia de servicios externos identificados en la motivación del proyecto.

Desde el punto de vista técnico, el proyecto ha permitido aplicar de forma integrada competencias de distintos módulos del ciclo: administración de sistemas en entornos Linux, contenedorización con Docker, gestión de identidades con LDAP, seguridad en redes y sistemas, y desarrollo de scripts y automatizaciones. La naturaleza colaborativa del trabajo, con distribución de tareas por áreas de especialización, ha reproducido fielmente la dinámica de un equipo de operaciones real.

Entre los aspectos más relevantes del proceso destacan la resolución de los conflictos de puertos SSH, que requirió un análisis detallado del comportamiento del entrypoint de la imagen de Forgejo; la corrección de la persistencia de la base de datos SQLite, que inicialmente se almacenaba fuera del volumen persistente; el diagnóstico y resolución del aislamiento de red del runner de CI/CD, que impedía a los jobs acceder a Forgejo hasta identificar el conflicto de prioridad entre archivos

de configuración; y la auditoría de seguridad de Nginx, que partió de una configuración mínima y alcanzó un nivel de conformidad con OWASP adecuado para un entorno de producción.

Como líneas de trabajo futuro se identifican la migración de SQLite a PostgreSQL para soportar mayor concurrencia, la implementación de un sistema de copias de seguridad automatizadas con herramientas como `rclone`, la activación de LDAPS (puerto 636) para cifrar el tráfico entre Forgejo y OpenLDAP en escenarios multi-servidor, y el despliegue del runner en un nodo separado para mejorar el aislamiento de seguridad de los pipelines CI/CD.

Referencias

- Forgejo — Documentación oficial: https://forgejo.org/docs/latest/
- Forgejo Actions — Anuncio oficial: https://forgejo.org/2023-02-27-forgejo-actions/
- OpenLDAP — Documentación oficial: https://www.openldap.org/doc/
- Nginx — Documentación oficial: https://nginx.org/en/docs/
- Docker — Documentación oficial: https://docs.docker.com/
- Let's Encrypt / Certbot: https://certbot.eff.org/
- OWASP Secure Headers Project: https://owasp.org/www-project-secure-headers/
- act_runner (Gitea): https://gitea.com/gitea/act_runner
- Claude IA: https://claude.ai
- Perplexity IA: https://perplexity.ai

Uso de inteligencia artificial. Se ha usado Claude y Perplexity para consultar dudas técnicas sobre temas muy específicos y puntuales, como la configuración avanzada de Nginx y el diagnóstico de problemas de conectividad entre servicios. En esencia, se ha usado como una herramienta de consulta técnica complementaria a la documentación oficial.