EBENEZER IT · GUÍAS PASO A PASO

GUÍA 10 · EPISODIOS 19 Y 20

Chat privado para tu empresa con ejabberd

Mensajería tipo WhatsApp en tu propio servidor: chats cifrados de punta a punta, grupos, envío de fotos y archivos, y llamadas y videollamadas que funcionan aunque estéis en redes distintas. Las cuentas son las vuestras (`ana@tudominio.es`) y los mensajes no salen de tu servidor.

~3 horas
la primera vez, en dos sesiones

Requisito
Guías 01, 02 y 06 terminadas

Ubuntu 24.04 LTS
ejabberd · XMPP · STUN/TURN

www.ebenezer.org.es · Alejandro Quevedo · Versión pública 1.0 · noviembre 2026

Índice

Antes de empezar

0 Qué vas a construir y valores de esta guía

Parte A · Cómo funciona y preparación

Paso 01 XMPP: un chat como el correo

Paso 02 Registros DNS

Paso 03 Abrir los puertos del chat

Parte B · Instalar y configurar ejabberd

Paso 04 Instalar ejabberd

Paso 05 El certificado del chat

Paso 06 La configuración de ejabberd

Paso 07 Arrancar y comprobar

Paso 08 Crear las cuentas

Paso 09 Conectar los móviles y el PC

Parte C · Archivos, grupos y llamadas

Paso 10 Enviar fotos y archivos

Paso 11 Grupos

Paso 12 Llamadas y videollamadas

Parte D · Proteger y mantener

Paso 13 El puerto interno de ejabberd, solo en local

Paso 14 fail2ban para el chat

Paso 15 Copias y limpieza automática

Paso 16 Comprobación final e instantánea

Anexos

A Problemas frecuentes

B Puertos al terminar esta guía

C Chuleta de ejabberdctl

Cómo usar esta guía

Tienes aquí **todos los pasos y comandos**, en orden. La explicación de cada paso (el porqué de cada línea) está en el **vídeo del episodio**.

Para copiar los comandos sin errores, usa la versión web: **https://www.ebenezer.org.es/guia-10-chat.html**. Al copiar desde un PDF se pueden romper los saltos de línea y las comillas.

ANTES DE EMPEZAR

Qué vas a construir

Un servicio de mensajería para tu equipo o tu familia, en tu servidor: chats de uno a uno y de grupo, cifrados de punta a punta con OMEMO, fotos y documentos, mensajes que llegan aunque el móvil estuviera apagado, y llamadas de voz y vídeo.

Pieza	Qué hace	Puerto
ejabberd	El servidor de chat: cuentas, contactos, mensajes, grupos	5222 (tus apps), 5269 (otros servidores)
Subida de archivos	Guarda las fotos y documentos que se envían y da un enlace para descargarlos	5443 (HTTPS)
STUN/TURN integrado	Ayuda a que las llamadas conecten aunque los dos móviles estén detrás de un router, y si no pueden hablar directamente, hace de puente	3478 y 49152–65535 (UDP)
Apps	Conversations (Android), Monal (iPhone), Gajim (PC)	—

Un solo servicio para todo

ejabberd trae su propio servidor de llamadas (STUN/TURN). Muchos tutoriales instalan además **coturn**, un programa aparte. Funciona, pero son dos programas, dos configuraciones y una contraseña compartida entre ellos que hay que mantener igual. Con el integrado, ejabberd entrega a cada app unas credenciales temporales que siempre coinciden. Para un servidor pequeño, menos piezas es menos que se pueda romper.

Valores de esta guía

Valor	Qué es
TU_DOMINIO	Tu dominio. Las cuentas serán `usuario@TU_DOMINIO`, igual que el correo
TU_IP	La IP del VPS
TU_USUARIO	Tu cuenta de chat (por ejemplo, tu nombre)

Recordatorio

- Haz una instantánea en IONOS antes de empezar.
- Necesitas dos móviles (o un móvil y el PC) para probar los chats y las llamadas.
- Bloques **verdes** en el servidor, **azules** en PowerShell de tu PC, **morados** son contenido de archivos, **ocres** son clics en un panel.

PARTE A

Cómo funciona y preparación

Entendemos cómo funciona XMPP (el protocolo de ejabberd) y dejamos listos el DNS y los puertos. Corresponde al **episodio 19**.

Paso 1 Episodio 19

XMPP: un chat como el correo

Por qué: XMPP funciona como el correo, no como WhatsApp: cada empresa puede tener su servidor, las cuentas llevan el dominio, y los servidores se hablan entre ellos.

Correo	XMPP (chat)
`ana@TU_DOMINIO`	`ana@TU_DOMINIO` (se llama **JID**)
Tu programa de correo → tu servidor (587/993)	Tu app de chat → tu servidor (**5222**)
Tu servidor → el de otra empresa (25)	Tu servidor → el de otra empresa (**5269**, «federación»)
Registro MX	Registros **SRV**

Cifrado de punta a punta: OMEMO

El canal entre la app y el servidor siempre va cifrado (TLS). Además, las apps modernas cifran cada mensaje con **OMEMO** (el mismo sistema que Signal): solo el móvil del destinatario puede leerlo. Tu servidor guarda y reparte los mensajes, pero no puede leer su contenido. Ni tú, que eres el administrador.

¿Y si no quiero hablar con otros servidores?

La federación (5269) permite que tus usuarios hablen con cuentas de otros servidores XMPP. Si tu chat es solo interno, puedes no abrir el 5269: todo funcionará igual dentro de tu dominio.

Comprueba antes de seguir

Sabes explicar qué es un JID, para qué sirve el 5222 y para qué el 5269, y por qué el administrador no puede leer los mensajes cifrados con OMEMO.

Paso 2 Episodio 19

Registros DNS

Por qué: las apps y los otros servidores buscan dónde está tu chat con registros **SRV**.

Tipo	Nombre	Valor	Para qué
SRV	Servicio `_xmpp-client`, protocolo `_tcp`	Prioridad 0, peso 5, puerto **5222**, destino `TU_DOMINIO`	Las apps encuentran tu servidor
SRV	Servicio `_xmpp-server`, protocolo `_tcp`	Prioridad 0, peso 5, puerto **5269**, destino `TU_DOMINIO`	Otros servidores encuentran el tuyo

Tipo	Nombre	Valor	Para qué
A	`conference`	TU_IP	Los grupos (`sala@conference.TU_DOMINIO`)
A	`upload`	TU_IP	El servicio de subida de archivos

En IONOS, los SRV se crean con el tipo **SRV** y unos campos separados (servicio, protocolo, prioridad, peso, puerto, destino). Espera unos minutos y comprueba:

EN TU PC · POWERSHELL

```
Resolve-DnsName _xmpp-client._tcp.TU_DOMINIO -Type SRV -Server 1.1.1.1
Resolve-DnsName _xmpp-server._tcp.TU_DOMINIO -Type SRV -Server 1.1.1.1
Resolve-DnsName conference.TU_DOMINIO -Type A -Server 1.1.1.1
```

Comprueba antes de seguir

Los dos SRV responden con el puerto y el destino correctos, y `conference` y `upload` apuntan a TU_IP.

Paso 3 Episodio 19

Abrir los puertos del chat

Por qué: cinco reglas, cada una con su motivo.

Puerto	Protocolo	Para qué
5222	TCP	Tus apps de chat
5269	TCP	Otros servidores XMPP (opcional, paso 1)
5443	TCP	Subir y descargar fotos y archivos (HTTPS)
3478	UDP y TCP	STUN/TURN: el «ayudante» de las llamadas
49152–65535	UDP	El audio y el vídeo de las llamadas que pasan por tu servidor

En IONOS, añade esas reglas de entrada (TCP 5222, 5269, 5443 y 3478; UDP 3478 y el rango 49152–65535). En el servidor:

EN EL SERVIDOR

```
sudo ufw allow 5222/tcp comment 'XMPP: apps de chat'
sudo ufw allow 5269/tcp comment 'XMPP: otros servidores'
sudo ufw allow 5443/tcp comment 'XMPP: archivos'
sudo ufw allow 3478 comment 'STUN/TURN: llamadas'
sudo ufw allow 49152:65535/udp comment 'TURN: audio y video'
sudo ufw status | grep -E '5222|5269|5443|3478|49152'
```

¿No es mucho abrir 16 000 puertos?

Es lo que necesita TURN, y en ese rango solo escucha ejabberd cuando hay una llamada en marcha, con credenciales temporales que entrega solo a usuarios con sesión iniciada. Fuera de una llamada, no hay nada escuchando ahí.

Comprueba antes de seguir

En UFW y en IONOS están las cinco reglas, y no hay ninguna para el 5280, el 5223 ni el 1883.

PARTE B

Instalar y configurar ejabberd

Instalamos ejabberd del repositorio de Ubuntu, le damos su certificado y una configuración completa y comentada. Al final, tus primeras cuentas y tus apps conectadas. Corresponde al **episodio 19**.

Paso 4 Episodio 19

Instalar ejabberd

Por qué: el paquete de Ubuntu se actualiza con el sistema (y con Ubuntu Pro recibe los parches de seguridad).

EN EL SERVIDOR

```
sudo apt install ejabberd
sudo systemctl stop ejabberd
id ejabberd
```

Pregunta	Respuesta
Nombre del servidor (hostname)	`TU_DOMINIO`
Usuario administrador	`admin`, con una contraseña larga de tu gestor

Comprueba antes de seguir

`id ejabberd` muestra el usuario y su grupo, y `systemctl is-active ejabberd` responde `inactive`.

Paso 5 Episodio 19

El certificado del chat

Por qué: un certificado de Let's Encrypt para `TU_DOMINIO`, `conference.` y `upload.`.

5.1 · Nginx, para que Let’s Encrypt compruebe los subdominios

EN EL SERVIDOR

```
sudo tee /etc/nginx/sites-available/chat-acme > /dev/null <<'EOF'
# Solo para Let's Encrypt: conference. y upload. no tienen web
server {
    listen 80;
    listen [::]:80;
    server_name conference.TU_DOMINIO upload.TU_DOMINIO;

    location ^~ /.well-known/acme-challenge/ {
        root /var/www/letsencrypt;
    }

    location / {
        return 444;
    }
}
EOF
sudo ln -s /etc/nginx/sites-available/chat-acme /etc/nginx/sites-enabled/
sudo nginx -t && sudo systemctl reload nginx
```

5.2 · El hook y el certificado

El *hook* se crea **antes** de pedir el certificado: Certbot también lo ejecuta la primera vez que lo obtiene.

EN EL SERVIDOR

```
sudo tee /etc/letsencrypt/renewal-hooks/deploy/ejabberd-certificado.sh > /dev/null <<'EOF'
#!/bin/sh
# Copia el certificado del chat a ejabberd (que no puede leer los de root) y lo recarga
case "$RENEWED_LINEAGE" in
    */chat.TU_DOMINIO) ;;
    *) exit 0 ;;
esac
umask 027
install -d -o root -g ejabberd -m 750 /etc/ejabberd/certs
cat "$RENEWED_LINEAGE/fullchain.pem" "$RENEWED_LINEAGE/privkey.pem" >
/etc/ejabberd/certs/chat.pem
chown root:ejabberd /etc/ejabberd/certs/chat.pem
chmod 640 /etc/ejabberd/certs/chat.pem
systemctl is-active --quiet ejabberd && systemctl reload ejabberd
exit 0
EOF
sudo chmod +x /etc/letsencrypt/renewal-hooks/deploy/ejabberd-certificado.sh
```

EN EL SERVIDOR

```
sudo certbot certonly --webroot -w /var/www/letsencrypt \
  -d TU_DOMINIO -d conference.TU_DOMINIO -d upload.TU_DOMINIO \
  --cert-name chat.TU_DOMINIO
sudo ls -l /etc/ejabberd/certs/
```

SALIDA ESPERADA

```
-rw-r----- 1 root ejabberd 5321 ... chat.pem
```

Comprueba antes de seguir

Existe `/etc/ejabberd/certs/chat.pem` con permisos `-rw-r----- root ejabberd`.

Paso 6 Episodio 19

La configuración de ejabberd

Por qué: sustituimos la configuración de ejemplo por una completa, sin nada que no usemos.

EN EL SERVIDOR

```
sudo cp /etc/ejabberd/ejabberd.yml /etc/ejabberd/ejabberd.yml.orig
sudo nano /etc/ejabberd/ejabberd.yml
```

Borra todo su contenido (en nano: Alt+T, o mantén Ctrl+K) y pega:

CONTENIDO DEL ARCHIVO · /ETC/EJABBERD/EJABBERD.YML

```
###  Ebenezer IT · ejabberd (Guía 10)
hosts:
  - TU_DOMINIO

loglevel: info
log_rotate_count: 5

certfiles:
  - "/etc/ejabberd/certs/*.pem"

define_macro:
  TLS_CIPHERS: "HIGH:!aNULL:!eNULL:!3DES:@STRENGTH"
  TLS_OPTIONS:
    - "no_sslv3"
    - "no_tlsv1"
    - "no_tlsv1_1"
    - "cipher_server_preference"
    - "no_compression"

c2s_ciphers: TLS_CIPHERS
s2s_ciphers: TLS_CIPHERS
c2s_protocol_options: TLS_OPTIONS
s2s_protocol_options: TLS_OPTIONS
s2s_use_starttls: required

listen:
  -
    port: 5222
    ip: "::"
    module: ejabberd_c2s
    max_stanza_size: 262144
    shaper: c2s_shaper
    access: c2s
    starttls_required: true
  -
    port: 5269
    ip: "::"
    module: ejabberd_s2s_in
```

```
    max_stanza_size: 524288
    shaper: s2s_shaper
  -
    port: 5443
    ip: "::"
    module: ejabberd_http
    tls: true
    request_handlers:
      /upload: mod_http_upload
      /ws: ejabberd_http_ws
      /bosh: mod_bosh
  -
    port: 5280
    ip: "127.0.0.1"
    module: ejabberd_http
    request_handlers:
      /admin: ejabberd_web_admin
  -
    port: 3478
    ip: "0.0.0.0"
    transport: udp
    module: ejabberd_stun
    use_turn: true
    turn_ipv4_address: "TU_IP"
    turn_min_port: 49152
    turn_max_port: 65535
  -
    port: 3478
    ip: "0.0.0.0"
    transport: tcp
    module: ejabberd_stun
    use_turn: true
    turn_ipv4_address: "TU_IP"
    turn_min_port: 49152
    turn_max_port: 65535

auth_method: internal
auth_password_format: scram
disable_sasl_mechanisms:
  - "digest-md5"
  - "X-OAUTH2"

acl:
  admin:
    user:
      - "admin@TU_DOMINIO"
  local:
    user_regexp: ""
  loopback:
    ip:
      - 127.0.0.0/8
      - ::1/128

access_rules:
  local:
    allow: local
  c2s:
```

```
    deny: blocked
    allow: all
  announce:
    allow: admin
  configure:
    allow: admin
  muc_create:
    allow: local
  pubsub_createnode:
    allow: local
  trusted_network:
    allow: loopback

api_permissions:
  "console commands":
    from: ejabberd_ctl
    who: all
    what: "*"
  "admin access":
    who:
      access:
        allow:
          - acl: loopback
          - acl: admin
    what:
      - "*"
      - "!stop"
      - "!start"

shaper:
  normal:
    rate: 50000
    burst_size: 100000
  fast: 200000

shaper_rules:
  max_user_sessions: 10
  max_user_offline_messages:
    5000: admin
    1000: all
  c2s_shaper:
    none: admin
    normal: all
  s2s_shaper: fast
  soft_upload_quota:
    1000: all
  hard_upload_quota:
    1100: all

modules:
  mod_adhoc: {}
  mod_admin_extra: {}
  mod_announce:
    access: announce
  mod_avatar: {}
  mod_blocking: {}
  mod_bosh: {}
```

```
  mod_caps: {}
  mod_carboncopy: {}
  mod_client_state: {}
  mod_configure: {}
  mod_disco: {}
  mod_fail2ban: {}
  mod_http_upload:
    put_url: "https://@HOST@:5443/upload"
    docroot: "/var/lib/ejabberd/upload"
    max_size: 104857600
    thumbnail: false
  mod_http_upload_quota:
    max_days: 90
  mod_last: {}
  mod_mam:
    assume_mam_usage: true
    default: always
  mod_muc:
    access:
      - allow
    access_admin:
      - allow: admin
    access_create: muc_create
    access_persistent: muc_create
    access_mam:
      - allow
    default_room_options:
      mam: true
  mod_muc_admin: {}
  mod_offline:
    access_max_user_messages: max_user_offline_messages
  mod_ping:
    send_pings: true
  mod_privacy: {}
  mod_private: {}
  mod_pubsub:
    access_createnode: pubsub_createnode
    plugins:
      - flat
      - pep
    force_node_config:
      "eu.siacs.conversations.axolotl.*":
        access_model: open
      "storage:bookmarks":
        access_model: whitelist
  mod_push: {}
  mod_push_keepalive: {}
  mod_roster:
    versioning: true
  mod_s2s_dialback: {}
  mod_shared_roster: {}
  mod_stream_mgmt:
    resend_on_timeout: if_offline
  mod_stun_disco: {}
  mod_vcard:
    search: false
  mod_vcard_xupdate: {}
```

Lo importante de cada bloque

EN EL SERVIDOR

```
sudo mkdir -p /var/lib/ejabberd/upload
sudo chown ejabberd:ejabberd /var/lib/ejabberd/upload
sudo chmod 750 /var/lib/ejabberd/upload
```

Comprueba antes de seguir

La configuración está en su sitio y existe la carpeta de subidas, de `ejabberd`, con `drwxr-x---`.

Paso 7 Episodio 19

Arrancar y comprobar

Por qué: ejabberd comprueba la configuración al arrancar: si hay un error de sangría o un nombre mal escrito, lo dice en su registro.

EN EL SERVIDOR

```
sudo systemctl start ejabberd
sleep 10
sudo ejabberdctl status
sudo ss -tulnp | grep -E 'beam' | awk '{print $1, $5}' | sort -u
```

SALIDA ESPERADA

```
The node ejabberd@localhost is started with status: started
ejabberd 23.10 is running in that node

tcp *:5222
tcp *:5269
tcp *:5443
tcp 0.0.0.0:3478
tcp 127.0.0.1:5280
udp 0.0.0.0:3478
...
```

Si `status` no dice `started`, mira el registro: `sudo tail -n 30 /var/log/ejabberd/ejabberd.log`. Casi siempre es la sangría del YAML (espacios, nunca tabuladores).

Comprueba antes de seguir

`ejabberdctl status` dice `started` y escuchan el 5222, el 5269, el 5443 y el 3478 (UDP y TCP).

Paso 8 Episodio 19

Crear las cuentas

Por qué: con `ejabberdctl`.

```
EN EL SERVIDOR
read -rs -p "Contraseña para TU_USUARIO: " P; echo
sudo ejabberdctl register TU_USUARIO TU_DOMINIO "$P"
unset P
sudo ejabberdctl registered_users TU_DOMINIO
```

Repite el bloque para cada persona (con su nombre en lugar de `TU_USUARIO`). Una contraseña distinta y larga para cada una, generada con tu gestor.

Comprueba antes de seguir

`registered_users` muestra `admin` y tus cuentas.

Paso 9 Episodio 19

Conectar los móviles y el PC

Por qué: las apps XMPP modernas solo necesitan la cuenta y la contraseña: con los SRV del paso 2, encuentran el servidor solas.

Dispositivo	App recomendada	Cómo añadir la cuenta
Android	**Conversations** (gratis en F-Droid; de pago en Google Play)	«Ya tengo una cuenta» → `TU_USUARIO@TU_DOMINIO` y contraseña
iPhone	**Monal** (gratis)	Iniciar sesión → la cuenta y la contraseña
Windows	**Gajim** (gratis)	Añadir cuenta → iniciar sesión → la cuenta y la contraseña

1. Conecta dos cuentas, cada una en un dispositivo.
2. Desde una, añade a la otra como contacto (`otra@TU_DOMINIO`) y acepta la petición en la segunda.
3. Escribid unos mensajes. Fíjate en el **candado**: es OMEMO, el cifrado de punta a punta (en Conversations y Monal viene activado).

Verificad las huellas

Para estar seguros de que nadie se interpone, en la ficha del contacto hay una «huella» de su clave OMEMO. Compruébala en persona o por otra vía: si coincide, márcala como verificada.

Comprueba antes de seguir

Dos cuentas se escriben entre sí, con el candado de OMEMO.

PARTE C

Archivos, grupos y llamadas

Lo que hace que el chat sea útil de verdad: fotos y documentos, grupos de trabajo, y llamadas que conectan aunque estéis en redes distintas. Corresponde al **episodio 20**.

Paso 10 Episodio 20

Enviar fotos y archivos

Por qué: cuando envías una foto, la app la sube cifrada a tu servidor por el 5443 y envía al otro solo el enlace.

1. Desde una app, envía una foto al otro contacto.
2. Debe aparecer en la otra app al momento.

EN EL SERVIDOR

```
sudo ls -la /var/lib/ejabberd/upload/
sudo du -sh /var/lib/ejabberd/upload/
```

Comprueba antes de seguir

La foto llega, y en `/var/lib/ejabberd/upload` aparece una carpeta nueva.

Paso 11 Episodio 20

Grupos

Por qué: los grupos (MUC, *multi-user chat*) viven en `conference.TU_DOMINIO`.

1. En Conversations: **+** → **Crear grupo** (o «Crear chat en grupo privado»).
2. Ponle nombre y elige a los participantes. La dirección será algo como `equipo@conference.TU_DOMINIO`.
3. Para que el grupo vaya cifrado con OMEMO, debe ser **privado** (solo miembros): así lo crea Conversations por defecto.

EN EL SERVIDOR

```
sudo ejabberdctl muc_online_rooms global
```

Comprueba antes de seguir

El grupo aparece en `muc_online_rooms` y sus miembros se escriben.

Paso 12 Episodio 20

Llamadas y videollamadas

Por qué: dos móviles en casas distintas están detrás de sus routers y no saben cómo llegar el uno al otro.

Situación	Cómo va la llamada
Los dos en la misma wifi	Directa, sin pasar por el servidor
Redes distintas, routers normales	Directa, gracias a STUN
Datos móviles, redes de empresa, hoteles	Por tu servidor (TURN), puertos 49152–65535

1. Con un móvil en la **wifi** y el otro con **datos móviles** (es la prueba difícil), haz una llamada de voz desde la ficha del contacto.
2. Prueba después una videollamada.

```
EN EL SERVIDOR
sudo grep -iE "turn|stun" /var/log/ejabberd/ejabberd.log | tail -n 5
```

Comprueba antes de seguir

Una llamada de voz y una de vídeo entre wifi y datos móviles conectan y se oyen y se ven en los dos sentidos.

PARTE D

Proteger y mantener

Cerramos lo que no debe estar abierto, frenamos a quien prueba contraseñas, y dejamos las copias y la limpieza automáticas. Corresponde al **episodio 20**.

Paso 13 Episodio 20

El puerto interno de ejabberd, solo en local

Por qué: ejabberd está hecho en Erlang, y Erlang abre dos puertos propios para que `ejabberdctl` hable con el servidor: **epmd** (4369), una especie de «guía telefónica» que dice dónde está ejabberd, y un puerto alto al azar.

13.1 · Preparar los dos cambios

EN EL SERVIDOR

```
sudo grep -n "INET_DIST_INTERFACE" /etc/ejabberd/ejabberdctl.cfg
sudo sed -i 's/^#\?INET_DIST_INTERFACE=.*/INET_DIST_INTERFACE=127.0.0.1/'
/etc/ejabberd/ejabberdctl.cfg
sudo grep -n "^INET_DIST_INTERFACE" /etc/ejabberd/ejabberdctl.cfg
```

EN EL SERVIDOR

```
F=/etc/systemd/system/epmd.socket.d/solo-local.conf
sudo mkdir -p /etc/systemd/system/epmd.socket.d
[ -e "$F" ] && echo "YA EXISTE, no lo toco" || sudo tee "$F" > /dev/null <<'EOF'
[Socket]
ListenStream=
ListenStream=127.0.0.1:4369
ListenStream=[::1]:4369
EOF
cat "$F"
```

13.2 · Aplicar con un reinicio ordenado

EN EL SERVIDOR

```
sudo systemctl stop ejabberd
sudo systemctl stop epmd.service epmd.socket
sudo systemctl daemon-reload
sudo systemctl start epmd.socket
sudo systemctl start ejabberd
sleep 15
sudo ejabberdctl status
sudo ss -tlnp | grep -E 'beam|epmd' | awk '{print $4, $6}'
```

Comprueba antes de seguir

`ejabberdctl status` sigue en `started`, epmd y el puerto alto de Erlang solo en local, y fuera de `127.0.0.1` solo los puertos del paso 3.

Paso 14 Episodio 20

fail2ban para el chat

Por qué: el 5222 recibe intentos de adivinar contraseñas, como el correo.

EN EL SERVIDOR

```
sudo tee /etc/fail2ban/jail.d/chat.local > /dev/null <<'EOF'
[ejabberd-auth]
enabled  = true
port     = 5222,5269,5443
logpath  = /var/log/ejabberd/ejabberd.log
maxretry = 5
findtime = 10m
bantime  = 1h
EOF
sudo fail2ban-client -t && sudo systemctl reload fail2ban
sudo fail2ban-client status ejabberd-auth
```

Comprueba antes de seguir

`fail2ban-client status ejabberd-auth` muestra la jaula activa vigilando `/var/log/ejabberd/ejabberd.log`.

Paso 15 Episodio 20

Copias y limpieza automática

Por qué: tres cosas que guardar (la base de datos de ejabberd, los archivos subidos y la configuración) y una que limpiar: el archivo de mensajes, que crece sin parar.

EN EL SERVIDOR

```
sudo tee /etc/cron.d/ejabberd-mantenimiento > /dev/null <<'EOF'
# Ebenezer IT · ejabberd
# Cada noche a las 02:30: copia de la base de datos de ejabberd
30 2 * * * ejabberd /usr/sbin/ejabberdctl backup /var/lib/ejabberd/copia-ejabberd.backup >
/dev/null
# Los domingos a las 04:00: borrar del servidor los mensajes de más de un año
0 4 * * 0 ejabberd /usr/sbin/ejabberdctl delete_old_mam_messages all 365 > /dev/null
EOF
sudo -u ejabberd /usr/sbin/ejabberdctl backup /var/lib/ejabberd/copia-ejabberd.backup
sudo ls -lh /var/lib/ejabberd/copia-ejabberd.backup
```

Qué incluir en tus copias de seguridad

- `/var/lib/ejabberd/copia-ejabberd.backup` (la copia de la base de datos de esta noche).
- `/var/lib/ejabberd/upload` (los archivos subidos).
- `/etc/ejabberd` (la configuración; el certificado se regenera con Certbot).

Comprueba antes de seguir

Existe `copia-ejabberd.backup` y el archivo `/etc/cron.d/ejabberd-mantenimiento`.

Paso 16 Episodio 20

Comprobación final e instantánea

Por qué: un repaso, el probador público de servidores XMPP, un reinicio y la instantánea.

EN EL SERVIDOR

```
sudo ejabberdctl status
sudo ss -tulnp | grep beam | awk '{print $1, $5}' | sort -u
sudo ufw status | grep -E '5222|5269|5443|3478|49152|5280|5223|1883'
echo | openssl s_client -connect 127.0.0.1:5222 -starttls xmpp -xmpphost TU_DOMINIO 2>/dev/null
| openssl x509 -noout -enddate
sudo fail2ban-client status | grep "Jail list"
```

Comprobación	Resultado correcto
ejabberd	`started`
Puertos de ejabberd	5222, 5269, 5443 y 3478 (UDP y TCP) para todos; 5280 y el interno de Erlang solo en `127.0.0.1`
UFW	Las cinco reglas del paso 3; nada para 5280, 5223 ni 1883
Certificado del 5222	La misma fecha que `sudo certbot certificates` da para `chat.TU_DOMINIO`
fail2ban	Incluye `ejabberd-auth`

El probador de cumplimiento

`compliance.conversations.im` comprueba si tu servidor tiene todo lo que necesitan las apps modernas (archivo de mensajes, subidas, avisos, llamadas...). Pide una cuenta para conectarse: crea una de prueba con el paso 8, pásale su dirección y contraseña, y bórrala al terminar con `sudo ejabberdctl unregister prueba TU_DOMINIO`. Lo normal con esta configuración es un 100 % o muy cerca.

Reinicia el servidor y, al volver, escribe desde el móvil y haz una llamada corta. Después, en IONOS, la instantánea `10-chat`.

Comprueba antes de seguir

Tu equipo tiene su propio WhatsApp: cifrado de punta a punta, con grupos, fotos, archivos y llamadas, y con los mensajes en tu servidor. Con esto termina la temporada del chat.

ANEXOS

Para cuando algo no sale

El registro de ejabberd cuenta casi todo: `sudo tail -f /var/log/ejabberd/ejabberd.log` mientras repites la prueba (Ctrl+C para salir).

Anexo A · Problemas frecuentes

Lo que ves	Qué pasa	Qué hacer
ejabberd no arranca tras cambiar la configuración	Error en el YAML (casi siempre la sangría)	`sudo tail -n 30 /var/log/ejabberd/ejabberd.log`: dice la línea. Solo espacios, nunca tabuladores
La app: *certificate not trusted*	ejabberd no tiene el certificado bueno	`sudo ls -l /etc/ejabberd/certs/`; si falta, ejecuta el hook a mano: `sudo RENEWED_LINEAGE=/etc/letsencrypt/live/chat.TU_DOMINIO /etc/letsencrypt/renewal-hooks/deploy/ejabberd-certificado.sh`
La app no encuentra el servidor	Faltan los SRV o el 5222 está cerrado	Paso 2 y paso 3; prueba en la app con el servidor `TU_DOMINIO` y el puerto 5222 a mano
Not authorized al conectar	Usuario o contraseña incorrectos	La cuenta va completa: `usuario@TU_DOMINIO`. Cámbiala: `ejabberdctl change_password` (Anexo C)
Las fotos no se envían	El 5443 cerrado o la carpeta de subidas sin permisos	`sudo ufw status | grep 5443` y `ls -ld /var/lib/ejabberd/upload` (dueño `ejabberd`)
Las llamadas conectan en la misma wifi pero no entre wifi y datos	TURN no funciona	Revisa UFW e IONOS (3478 y 49152–65535 UDP) y que `turn_ipv4_address` es tu IP pública exacta. Con `sudo tail -f /var/log/ejabberd/ejabberd.log | grep -iE "stun|turn"` durante la llamada deben salir *Creating TURN allocation*; al cerrarse, *Relayed N KiB* dice si el audio y el vídeo pasaron por tu servidor (MiB) o fueron directos (pocos KiB)
La videollamada conecta pero la imagen no es nítida	Las apps ajustan la calidad a la red	TURN reenvía los paquetes tal cual, sin límite: ejabberd no afecta a la calidad. Compara con otra app en las mismas redes
El candado de OMEMO no aparece o «no hay claves»	El contacto aún no ha publicado sus claves	Que el otro abra la app y vuelva a intentarlo; revisa el bloque `mod_pubsub`
No llegan avisos con la app cerrada	Ahorro de batería del móvil	En Android: Ajustes → Batería → la app → «Sin restricciones»

Anexo B · Puertos al terminar esta guía

Puerto	Servicio	Accesible desde	Qué lo protege
5222/tcp	Apps de chat	Todo Internet	Cifrado obligatorio, contraseñas SCRAM, fail2ban
5269/tcp	Otros servidores XMPP	Todo Internet (opcional)	Cifrado obligatorio
5443/tcp	Subida y descarga de archivos	Todo Internet	HTTPS; solo se sube con sesión iniciada
3478/udp+tcp	STUN/TURN	Todo Internet	Credenciales temporales de `mod_stun_disco`
49152–65535/udp	Audio y vídeo por TURN	Todo Internet	Solo durante una llamada
127.0.0.1:5280	Panel web de administración	Solo el servidor (túnel SSH)	No escucha fuera
127.0.0.1:(alto)	Puerto interno de Erlang	Solo el servidor	`INET_DIST_INTERFACE`

Anexo C · Chuleta de ejabberdctl

Para...	Cómo
Estado	`sudo ejabberdctl status`
Crear una cuenta	Paso 8 (con `read -rs`)
Cambiar una contraseña	`read -rs P; sudo ejabberdctl change_password usuario TU_DOMINIO "$P"; unset P`
Borrar una cuenta	`sudo ejabberdctl unregister usuario TU_DOMINIO`
Cuentas y conectados	`sudo ejabberdctl registered_users TU_DOMINIO` · `sudo ejabberdctl connected_users`
Grupos activos	`sudo ejabberdctl muc_online_rooms global`
Avisar a todos	Desde una app con la cuenta `admin`: mensaje a `TU_DOMINIO/announce/online`
Recargar la configuración	`sudo systemctl reload ejabberd`
Panel web (túnel)	En tu PC: `ssh -L 5280:127.0.0.1:5280 TU_SERVIDOR` y abre `http://localhost:5280/admin` con la cuenta `admin@TU_DOMINIO`

Si te atascas

Pega las últimas 30 líneas de `/var/log/ejabberd/ejabberd.log` justo después de repetir la prueba, y la salida de `sudo ss -tulnp | grep beam`.