SELF-HOSTED · FOR IT AND SECURITY

Security memo

Every statement below was checked on 15 November 2026 on a clean Ubuntu 22.04 machine (2 CPU cores, 4 GB) with Layers 3.0.1 installed by the command from the licence email. Each one comes with a command so your team can check it on your server. Where we did not check something, we say so.

The administrator panel password

Since 15.11.2026 the installer sets a random password for the administrator panel (God Mode, /godmode) in /opt/layers/.env (ADMIN_PASSWORD, mode 600). Servers installed earlier accept admin / admin: run the install command from the email again, or the command below. Checked on a clean machine: afterwards admin / admin gets 401, also after a restart; running it again changes nothing.

```
cd /opt/layers && P=$(openssl rand -hex 16) && T=$(sudo docker compose exec -T layers wget -qO- --post-data "username=admin&password=admin&bearer=true" http://localhost:8080/data/auth/admin/login | sed -n 's/.*"token":"\([^"]*\)".*/\1/p') && [ -n "$T" ] && sudo docker compose exec -T layers wget -qO- --header "Authorization: Bearer $T" --header "Content-Type: application/json" --post-data "{\"current\":\"admin\",\"next\":\"$P\"}" http://localhost:8080/data/auth/admin/password >/dev/null && echo "ADMIN_PASSWORD=$P" | sudo tee -a .env >/dev/null && echo "Done: God Mode password is in /opt/layers/.env (ADMIN_PASSWORD)"
```

Check

```
cd /opt/layers && sudo docker compose exec -T layers wget -qO- --post-data "username=admin&password=admin" http://localhost:8080/data/auth/admin/login >/dev/null 2>&1 && echo "NOT CLOSED: admin/admin still works" || echo "Closed: admin/admin is refused"
```

Where the data lives

- Everything is in /opt/layers on your machine: the PostgreSQL database in data/db, people's files in data/layers/files, HTTPS certificates in data/caddy, the licence key in license/, server secrets in .env.
- .env is created with mode 600 (root only); the database folder is 700.
- People's files are written with mode 644. Since 15.11.2026 the installer closes their folder (750) so other local accounts cannot read them; on servers installed earlier run the command below — the server keeps working (checked, healthy after restart).

Check

```
sudo ls -la /opt/layers /opt/layers/data
sudo docker inspect layers-layers-1 -f '{{range .Mounts}}{{.Source}} → {{.Destination}}{{println}}{{end}}'
```

Restrict people's files to the server

```
sudo chmod 750 /opt/layers/data/layers && sudo docker compose -f /opt/layers/docker-compose.yml restart layer
s
```

Ports and where the server connects

- Listening: TCP 80 (redirect to HTTPS and certificate issue), TCP 443 (site and TURN over TLS for calls), UDP 443, TCP 7881, UDP 7882, UDP 3478 (calls). Nothing else.
- The document editors, a4 (.docx) and f1 (.xlsx), are separate containers without ports of their own: with them the list above is the same (checked on 15.11.2026), browsers reach them only through Caddy at /a4/ and /f1/. By their settings in docker-compose.yml they ask Layers who the person is and take and save the file through Layers, so the Drive decides who opens it; working copies of open documents are in data/editors.
- Since 15.11.2026 the administrator panel (/godmode and its API) answers only local networks and the machine itself; everyone else gets 403 (checked on a clean install). Other addresses are added with GODMODE_ALLOW in .env or --godmode-allow of the installer.
- While installing, the machine downloads from layers.md (install.sh, docker-compose.yml), Docker Hub (images), the Ubuntu archive (Docker, if missing) and asks api.ipify.org for its external address to warn about DNS.
- At work (10 minutes watched with outbound internet cut, after a full restart): the calls service asked public STUN servers (stun.l.google.com, global.stun.twilio.com) for its external address, and Layers tried api.layers.md — the hourly licence mark (next section). Nothing else; the browser makes no requests to outside addresses.
- Closed network: set NODE_IP (since 15.11.2026 the installer writes it itself) and switch the licence mark off. Checked: after a full restart without internet, 6 minutes, zero outbound attempts, calls service up, licence active.
- With the document editors (checked on 15.11.2026, outbound internet cut, full restart, 7 minutes watched): the server came up in 104 s, licence active, calls service up with NODE_IP, both editors healthy. The document editor (a4) tried support.sumoffice.com, the service of the editor’s maker, 7 times and kept working without an answer; the spreadsheet editor (f1) made no outbound attempts.
- Without NODE_IP and without internet the calls service does not start (it keeps restarting: “could not resolve external IP”) — the rest of Layers works.

- With a real domain Caddy obtains a Let’s Encrypt certificate over the internet; we did not check this on the test machine (it used layers.localhost and its own certificate).

Open ports

```
sudo ss -ltnup | grep docker-proxy
```

How we counted outbound attempts

```
SUB=$(sudo docker network inspect layers_default -f '{{(index .IPAM.Config 0).Subnet}}')
sudo iptables -I DOCKER-USER 1 -s $SUB -j DROP
sudo iptables -I DOCKER-USER 1 -s $SUB -j LOG --log-prefix "LAYERS-OUT "
for n in 10.0.0.0/8 172.16.0.0/12 192.168.0.0/16; do sudo iptables -I DOCKER-USER 1 -s $SUB -d $n -j RETURN;
done
sudo dmesg | grep LAYERS-OUT          # what the server tried to send out
```

Closed network

```
cd /opt/layers
echo "NODE_IP=10.0.0.10" | sudo tee -a .env >/dev/null       # the server's address in your network
sudo tee docker-compose.override.yml >/dev/null <<'EOF'
services:
  layers:
    environment:
      LAYERS_LICENSE_SEEN_ENABLED: "false"
EOF
sudo docker compose up -d
```

What leaves the server; is the licence check offline?

- The key is verified on the server itself. Checked: outbound internet cut, full restart of all services — licence LICENSED, mode full, nobody locked out.
- If the server has internet, once an hour it sends api.layers.md the key id, the installation id, the instance id and the version — no people, no files, no content. The answer may carry a renewed key for this installation, which the server checks (signature, installation) and installs itself. Without an answer nothing changes.
- Switch the mark off with LAYERS_LICENSE_SEEN_ENABLED=false (see “Closed network” above); renewed keys are then installed by copying the file into /opt/layers/license.

Check

```
curl -s https://layers.example.com/layers/license/status | grep -o '"mode":"[A-Z]*"\|"regime":"[a-z-]*"'
# "mode":"LICENSED"
# "regime":"full"
```

Backups and restore

- All state is the database, people’s files, .env and the licence folder. Two commands make a copy without stopping the server.
- Restore checked: database dropped and files deleted, then restored from the copy — same 5 people, 8 pages, the same file with the same checksum, licence active, the owner sees their files. The server was back in about 1 min 40 s.

Copy

```
cd /opt/layers
sudo docker compose exec -T db pg_dump -U layers -d layersdb -Fc -Z6 > ~/layersdb.dump
sudo tar czf ~/layers-files.tar.gz -C /opt/layers data/layers .env license
```

Restore

```
cd /opt/layers
sudo docker compose stop layers
sudo docker compose exec -T db psql -U layers -d postgres -c "drop database layersdb with (force)"
sudo docker compose exec -T db psql -U layers -d postgres -c "create database layersdb owner layers"
sudo docker compose exec -T db pg_restore -U layers -d layersdb --no-owner < ~/layersdb.dump
sudo tar xzf ~/layers-files.tar.gz -C /opt/layers data/layers
sudo docker compose up -d
```

Updates and rollback

- The version is one line in .env (LAYERS_IMAGE). Update: copy the database, change the line, docker compose up -d; database changes apply by themselves. The break was about 80 seconds.
- Rollback checked: 3.0.1 → 3.0.0 on the same database started and kept the licence (the database tool warns that the schema is newer), then back to 3.0.1. The reliable rollback is the old version plus the copy made before the update (restore above).

Update

```
cd /opt/layers
sudo docker compose exec -T db pg_dump -U layers -d layersdb -Fc -Z6 > ~/before-update.dump
sudo sed -i '/^LAYERS_IMAGE=/d' .env && echo "LAYERS_IMAGE=hissih/layers:3.0.1" | sudo tee -a .env >/dev/null
sudo docker compose up -d
sudo docker inspect layers-layers-1 -f '{{.Config.Image}} {{.State.Health.Status}}'
```

Roll back

```
cd /opt/layers
sudo sed -i '/^LAYERS_IMAGE=/d' .env && echo "LAYERS_IMAGE=hissih/layers:3.0.0" | sudo tee -a .env >/dev/null
sudo docker compose up -d
```

Logs and audit

- Every request is a line in the server log: method, path, status, time, user id and browser.
- The administrator panel keeps a log of sign-in attempts (time, address, success); it holds recent attempts since the last restart.
- People’s sign-ins are recorded in the database (auth_journal: time, method, result, address, browser).
- Container logs rotate: docker-compose.yml keeps at most 5 files of 20 MB per container (json-file driver, max-size and max-file); change the limits there or ship the logs to your system. A node installed earlier gets rotation after the new docker-compose.yml and docker compose up -d.

Server log and log driver

```
cd /opt/layers
sudo docker compose logs --since 1h layers | grep "\[access\]"
sudo docker inspect layers-layers-1 -f '{{.HostConfig.LogConfig.Type}} {{json .HostConfig.LogConfig.Config}}'
# json-file {"max-file":"5","max-size":"20m"}
```

Administrator panel sign-ins

```
J=$(mktemp)
curl -s -c $J -o /dev/null https://layers.example.com/data/auth/admin/login --data "username=admin&password=
$(sudo sed -n 's/^ADMIN_PASSWORD=//p' /opt/layers/.env)"
curl -s -b $J https://layers.example.com/data/auth/admin/logins; rm -f $J
# [{"at":"2026-11-15T00:31:13Z","ip":"…","ok":true}, …]
```

Who has access

- root on the server has everything: .env, database, files. Keep the machine to your administrators.
- The administrator panel (God Mode): one service account, admin, with a random password in /opt/layers/.env (ADMIN_PASSWORD), reachable only from local networks unless you list other addresses in GODMODE_ALLOW. On servers installed before 15.11.2026 the password is admin and the panel is open from anywhere — change the password (command at the top) and take the install file of 15.11.
- Inside Layers, the workspace owner and the administrators they appoint manage people and access.
- The Layers team has no access to your server: nothing is installed for remote access, and the only outbound call from Layers is the licence mark, which you can switch off.

Change the God Mode password

```
cd /opt/layers && P=$(openssl rand -hex 16) && T=$(sudo docker compose exec -T layers wget -qO- --post-data
"username=admin&password=admin&bearer=true" http://localhost:8080/data/auth/admin/login | sed -n 's/.*"toke
n":"\([^"]*\)".*/\1/p') && [ -n "$T" ] && sudo docker compose exec -T layers wget -qO- --header "Authorizatio
n: Bearer $T" --header "Content-Type: application/json" --post-data "{\"current\":\"admin\",\"next\":\"$P\"}"
http://localhost:8080/data/auth/admin/password >/dev/null && echo "ADMIN_PASSWORD=$P" | sudo tee -a .env >/de
v/null && echo "Done: God Mode password is in /opt/layers/.env (ADMIN_PASSWORD)"
```

Check

```
cd /opt/layers && sudo docker compose exec -T layers wget -qO- --post-data "username=admin&password=admin" ht
tp://localhost:8080/data/auth/admin/login >/dev/null 2>&1 && echo "NOT CLOSED: admin/admin still works" || ec
ho "Closed: admin/admin is refused"
```

Who may open God Mode

```
curl -s -o /dev/null -w "%{http_code}\n" https://layers.example.com/godmode      # from a machine not in GODM
ODE_ALLOW
# 403
cd /opt/layers && sudo sed -i '/^GODMODE_ALLOW=/d' .env && echo "GODMODE_ALLOW=private_ranges 203.0.113.7" |
sudo tee -a .env >/dev/null && sudo docker compose up -d
```

Encryption

- In transit: HTTPS with TLS 1.3 (Caddy); port 80 answers 308 to HTTPS. Calls in networks where only 443 is open go through TURN over TLS on 443.
- At rest: the database and people's files are stored as they are — a .docx on disk is the same file. If your policy requires encryption at rest, put /opt/layers on an encrypted disk (for example LUKS). We did not check this setup.
- People's passwords are stored as bcrypt hashes; the God Mode password, after the change, as a bcrypt hash; database and Redis passwords and server secrets are random and live in .env (600).

In transit

```
echo | openssl s_client -connect layers.example.com:443 -servername layers.example.com 2>/dev/null | grep -E
"Protocol|Cipher"
curl -s -o /dev/null -w "%{http_code} %{redirect_url}\n" http://layers.example.com/
# 308 https://layers.example.com/
```

At rest

```
sudo file /opt/layers/data/layers/files/* | head
sudo docker compose -f /opt/layers/docker-compose.yml exec -T db psql -U layers -d layersdb -Atc "select left
(password,4) from users limit 3"
# $2a$  (bcrypt)
```

Checked on 15.11.2026, Layers 3.0.1

This page was prepared with the help of AI from our own test logs.