KEULE SERVICES

Keule On-Prem: Technical Overview

Architecture, installation, updates, operations, and security

Technical companion document for administrators, IT managers, and technical decision-makers

Focused on on-prem operations, release artifacts, Docker Compose, native Ubuntu installation, preflight, diagnostics, health checks, and deliberate update processes.

Document status: June 24, 2026

Table of Contents

1. Executive Summary

Keule On-Prem is an integrated collaboration and organization platform for organizations that want to control data storage and operations within their own infrastructure. The application combines file storage, tasks, calendar, chat, notes, news/blog, notifications, areas, user management, and system monitoring in one web application.

For operators, the decisive shift is from developer checkout to release-based operation: customer installations run via prepared release artifacts or immutable container images. For finished releases, target systems do not require Composer installation or npm/Node builds.

> **Admin note**
>
> Docker Compose is the recommended standard path. Native Ubuntu installation remains a supported alternative when container operation is not desired for organizational or technical reasons.

Key message	Meaning for operators
Data sovereignty	MariaDB and the file system reside in the operator's own environment. Persistent data is not transported in images or release artifacts.
Controlled installation	CLI, preflight, diagnostics, and health checks make prerequisites and blockers traceable before services run in production.
Deliberate updates	Updates must be started actively via CLI. The web system monitor displays update information but does not start migrations.
Verified distribution	Signature, SHA-256, internal release manifest, and package contract are checked before activation.
Clear boundaries	No automatic rollback after migrations, no multi-node release, and WebDAV only for private user directories.

2. Target audience and use cases

This document is intended for administrators, IT managers, technical contacts, and decision-makers who want to evaluate, pilot, or operate Keule as an on-prem system. It is deliberately technical and focuses on architecture, operations, and responsibilities.

Scenario	Typical benefit	Special checkpoints
Association / federation	Shared file storage, tasks, appointments, news, and area structure without a large cloud suite.	Role model, self-registration, mail delivery, backup responsibility.
School / education context	Areas for classes, groups, or projects; central communication and organization.	Data protection, external reachability, sharing rules, administration process.
Municipal administration / organizational unit	Internal collaboration with controlled data storage and traceable operations.	TLS/reverse proxy, admin access, audit coverage, update and backup approvals.
Small to medium-sized company	Intranet-like collaboration with PWA, file storage, chat, calendar, and tasks.	Integration into existing infrastructure, SMTP, WebDAV, storage and restore tests.

3. Functional scope

Keule is designed as a modular web application. The following functions are relevant for on-prem evaluation without expanding every detail of the user interface.

Module	Functional scope	Technical notes
Hub / Dashboard	Start page with quick access, status notices, upcoming appointments, task status, and chat indicators; largely configurable.	PWA-capable navigation, mobile pane switchers, Command Palette.
Areas	Workspaces for teams, groups, or topics with join rules, memberships, and area file rules.	Visibility is created explicitly via area sharing; strict/legacy rules are configurable.
Files	Private and shared files, folders, upload, preview, share links, download codes, and soft-edit notices.	User files remain file-system data; shared/area rules run through ACL logic.
Tasks	Status, priority, due date, reminders, comments, checklists, attachments, assignees, and links.	Reminder state, file/event links, and main data are stored in MariaDB.
Calendar	Month, week, day, and agenda views, participants, reminders, and ICS/JSON import and export.	Calendar data and reminder state are stored in MariaDB.

Module	Functional scope	Technical notes
Notes	Text notes, checklists, colors, pins, archive, sharing, and task links.	Notes including shares and task links are DB-first.
Chat	Direct and group chats, attachments, history loading, read state, and task-from-chat.	Long polling, attachments with size/MIME limits, DB-first metadata.
News / Blog	System feed, area feeds, user feeds, posts, comments, favorites, and search.	Permission and moderation model in MariaDB.
Notifications / Push	In-app notices, status history, badge logic, and optional web push.	Push requires VAPID configuration and browser permission.
Administration / Monitoring	User management, system monitor, update view, diagnostics, certificate, and log views.	Update display remains read-only; operational updates remain CLI-only.

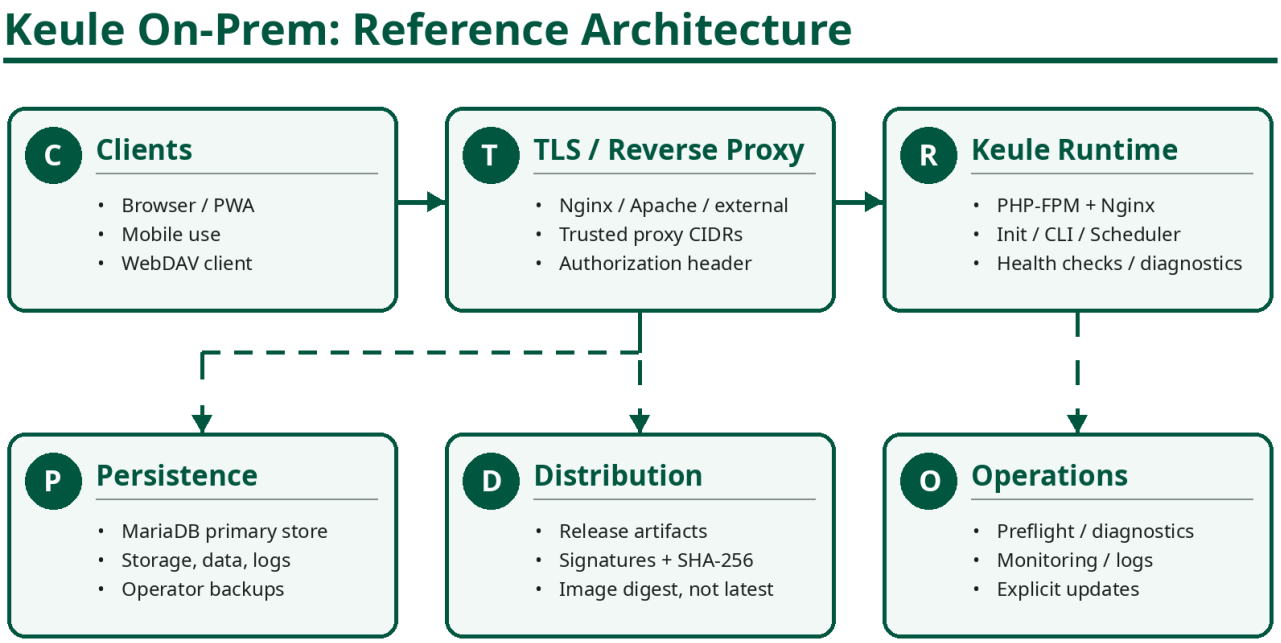
PWA limitation

The PWA is installable and supports service workers and push. It should currently not be understood as a full offline-first data application; modules are limited when the connection is unavailable.

4. Technical architecture

Keule is a server-side PHP web application with modular frontend bundling. The backend consists of PHP-FPM, shared app/DB logic, CLI commands, migrations, and background jobs. The frontend is delivered in the release as built assets. Productive domain storage is DB-first; binary data and logs remain in the file system.

Figure 1: Schematic Keule On-Prem reference architecture



Layer	Components	Operator relevance
Client	Browser, installed PWA, WebDAV client.	TLS trust, browser compatibility, and PWA origin are part of the operator environment.
Web/TLS	Native Nginx/Apache or Nginx in the Compose stack; optional external reverse proxy.	Document root is exclusively `public/`; forwarded headers are evaluated only for trusted CIDRs.
Application	PHP-FPM, app code, APIs, CLI, migrations, health checks.	Finished releases contain production dependencies and built assets.
Persistence	MariaDB, storage, data, logs, attachments.	Backups must capture database and file system consistently together.
Operations	Scheduler, preflight, diagnostics, update check, maintenance mode.	Exactly one scheduler; updates and migrations run under lock.

5. Data storage and storage locations

The central architectural decision is: core business/domain objects are stored in MariaDB; files, attachments, logs, and a few remaining domains remain file-system data. Consequently, a complete backup must always consider both the database and file paths together.

Domain	Primary storage	Note
Users, profiles, settings	MariaDB	Identity and profile state is managed on the database side.
Remember, API, registration, and reset tokens	MariaDB	Tokens are not stored in clear text; observe hash and TTL logic.
Areas, memberships, requests	MariaDB	Foundation for the organization model and visibility.
Tasks, calendar, notes, chat, news/blog	MariaDB	Core modules including reminder state and links.
Notifications, push subscriptions, VAPID	MariaDB	Push additionally requires VAPID keys.
Private and shared files	File system	`storage/private`, `storage/shared`, or configured operator paths.
Chat/task attachments	File system	Under `DATA_PATH` or the attachment paths.

Domain	Primary storage	Note
Support tickets	JSON / file system	Active remaining domain; include it in backups.
Logs / mail dumps	File system	Operational diagnostics; retention according to operator policy.

Installation type	Code	Configuration / secrets	Persistence / logs
Docker Compose	Immutable image by digest; app, init, and scheduler use the same image.	`.env`, `docker/compose.env`, secret files under `secrets/` and in the container under `/run/secrets`.	Named volumes or bind mounts for MariaDB, storage, data, and logs.
Native Ubuntu	`/opt/keule/releases/<version>` and active symlink `/opt/keule/current`.	`/etc/keule/keule.env` with `0640 root:keule`; token and trust key under `/etc/keule`.	`/var/lib/keule/storage`, `/var/lib/keule/data`, `/var/log/keule`.
External DB / NAS	Unchanged per installation path.	DB credentials as env/secret; DB TLS is currently reserved.	DB schema and mounted storage must exist beforehand; Keule does not mount NFS/SMB itself.

6. Operating models

The standard path for new customer installations is Docker Compose. Native Ubuntu installation is useful when existing operating policies exclude containers or when direct system integration with Nginx/Apache, PHP-FPM, and systemd is preferred.

Operating model	Suitability	Prerequisites	Limitations
Docker Compose	Recommended standard for most on-prem pilots and small to medium installations.	Ubuntu 24.04/WSL2, Docker Engine 26+, Compose Plugin 2.24+, 2 vCPU, 4 GiB RAM, approx. 30 GiB free.	No multi-node operation; production TLS typically via external reverse proxy.
Native Ubuntu	Direct integration into existing Linux operating processes and web server landscapes.	Ubuntu 24.04/WSL2, PHP-FPM/CLI >= 8.2, MariaDB 10.11 or external schema, Nginx/Apache.	More OS responsibility for the operator; package, web server, and certificate state must be maintained.
External DB	Operation with existing DB infrastructure or a separate DB host.	Pre-created schema and unprivileged DB user.	DB TLS is reserved; until then secure via private network or tunnel.
Host/NAS bind mounts	Persistence on existing storage systems.	Mounted, available storage with UID/GID rights, locking, rename/delete.	Keule does not mount NAS itself; backup and mount availability are operator duties.

Recommended starting point

For a first PoC, use Docker Compose with local MariaDB, named volumes, and a reverse proxy in front of it. External DB and bind mounts should be added only after the basic smoke test is stable.

7. Configuration and preflight

Keule uses a versioned configuration schema. Process values take precedence over env files; native installations use `/etc/keule/keule.env` by default, while Docker Compose uses the same application configuration as `env_file` and transports sensitive values as secret files.

Group	Examples	Operator note
App	`APP_ENV`, `APP_NAME`, `APP_URL`, `APP_TIMEZONE`, `APP_UPDATE_CHANNEL`	`APP_URL` must be a root URL; URL subpaths are not supported.
Organization	`ORG_NAME`, `ORG_LEGAL_NAME`, `ORG_ADDRESS`, contact and privacy addresses	Relevant for legal notice, mail texts, and legal display.
Database	`DB_HOST`, `DB_PORT`, `DB_NAME`, `DB_USER`, `DB_PASS`	With Compose, `DB_PASS` may remain empty if a secret file is used.
Storage	`STORAGE_PATH`, `DATA_PATH`, `LOG_PATH`	Paths must be writable; preflight does not blindly create missing productive storage paths.
Auth / registration	`AUTH_REGISTRATION_MODE`, password and rate-limit values	Open registration requires mail transport. 2FA is visible in the schema but cannot currently be used in production.
Mail / push / WebDAV	`MAIL_*`, `PUSH_ENABLED`, `WEBDAV_ENABLED`	Optional services are checked only when enabled. WebDAV requires HTTPS.
Proxy / TLS	`KEULE_TRUSTED_PROXY_HEADERS`, `TRUSTED_PROXY_CIDRS`, certificate paths	Forwarded headers only from explicitly trusted source networks.

Group	Examples	Operator note
Updates	`UPDATE_SERVER_URL`, `UPDATE_TOKEN_FILE`, `UPDATE_TRUST_KEY_FILE`	Update check is anonymous; downloads use customer-specific tokens.

```
bin/keule config:assistant --output=/etc/keule/keule.env
bin/keule config:validate --env-file=/etc/keule/keule.env
bin/keule preflight --env-file=/etc/keule/keule.env
bin/keule diagnose --env-file=/etc/keule/keule.env --json
```

Preflight behavior

`preflight` and `diagnose` are read-only. They report blockers, warnings, and OK states, but do not output passwords, tokens, or private keys.

8. Installation

8.1 Docker Compose

Docker Compose starts MariaDB, the init service, the PHP-FPM app, Nginx, and exactly one scheduler. Nginx is the only service with a host port; MariaDB and PHP-FPM remain internal. The init service validates configuration, waits for MariaDB, runs migrations, and creates the initial administrator account idempotently.

```
cp .env.example .env
cp docker/compose.env.example docker/compose.env
install -d -m 0700 secrets
openssl rand -base64 36 > secrets/db_password
openssl rand -base64 48 > secrets/db_root_password
openssl rand -base64 36 > secrets/admin_password
chmod 0600 .env docker/compose.env
sudo chgrp 10001 secrets/*
chmod 0640 secrets/*

docker/bin/keule-compose config --quiet
docker/bin/keule-compose pull
docker/bin/keule-compose up -d
docker/bin/keule-compose ps
curl --fail http://127.0.0.1:8080/healthz
```

Compose file	Purpose
compose.yaml	Standard stack with app, init, scheduler, Nginx, and local MariaDB.
compose.bind-mounts.yaml	Supplement for host or NAS paths instead of named volumes.
compose.external-db.yaml	Variant for externally provided DB without a local DB container.
compose.env.example	Compose-adjacent values such as image digest, bind address, port, and secret directory.
.env.example	Application configuration for app, init, and scheduler.

8.2 Native Ubuntu installation

The native installation separates immutable release code from persistent data. The first installer run is a read-only plan. Changes are applied only with `--apply`. Missing packages are confirmed interactively or installed unattended only with `--install-packages`.

```
bin/keule config:assistant --output="$HOME/keule-native.env"

sudo ./bin/keule install \
  --env-file="$HOME/keule-native.env" \
  --webserver=nginx \
  --database=local \
  --scheduler=systemd

sudo ./bin/keule install \
  --apply \
  --env-file="$HOME/keule-native.env" \
  --webserver=nginx \
  --database=local \
  --scheduler=systemd
```

Native component	Path / behavior

Releases	`/opt/keule/releases/<version>`; active via `/opt/keule/current`.
Configuration	`/etc/keule/keule.env`, owner `root:keule`, mode `0640`.
Storage / data	`/var/lib/keule/storage`, `/var/lib/keule/data`, sessions, and temporary uploads.
Logs	`/var/log/keule`.
Web server	Nginx or Apache, document root `public/`, configuration is tested before reload.
Scheduler	systemd timer as standard or cron fallback; parallel managed schedulers are blockers.

8.3 TLS, domains, and reverse proxy

Mode	Purpose	Checkpoints
letsencrypt	Public domain with HTTP-01.	`APP_URL=https://DOMAIN`, DNS and port 80 reachable from outside.
existing	Own certificate or internal CA.	Full chain/key readable, hostname/SAN, expiration date, and key match.
self-signed	Internal names or IPs.	Clients must trust the certificate manually.
reverse-proxy	TLS terminates in front of Keule.	`KEULE_TRUSTED_PROXY_HEADERS=true` and narrow `TRUSTED_PROXY_CIDRS`.
http	Deliberately confirmed internal exception.	Only with `--allow-http`; no silent fallback.

```
proxy_set_header Host $host;
proxy_set_header X-Forwarded-For $proxy_add_x_forwarded_for;
proxy_set_header X-Forwarded-Proto $scheme;
proxy_set_header Authorization $http_authorization;
```

9. Updates and release verification

Security as a principle: publicly retrievable metadata, protected artifacts, local verification, and deliberate activation. `update:check` remains anonymous. `update` uses customer-specific download tokens, checks signatures, and enables maintenance mode before productive changes are made. Automated via the updater.

Step	Check / action	Goal
1	Retrieve public channel metadata from `UPDATE_SERVER_URL`.	Anonymous check for available version.
2	Backup notice and deliberate confirmation.	Operator confirms that DB and file backups exist.
3	Download artifact, `.sha256`, and `.sha256.sig` with customer token.	Protected download without operator credentials.
4	Verify the signature of `.sha256`.	The update server alone is not sufficient as a trust anchor.
5	Verify archive SHA-256 and internal `RELEASE-MANIFEST.json`.	Detect manipulation before activation.
6	Check package contract via internal verifier.	Only permitted release contents are installed.
7	Enable maintenance mode, switch release or Docker digest, run migrations under lock.	Consistent, traceable update path.
8	Check version, health, DB, storage, and scheduler.	Release only after a successful smoke test.

```
bin/keule update:check --env-file=/etc/keule/keule.env

sudo bin/keule update \
  --env-file=/etc/keule/keule.env \
  --confirm-backup \
  --yes \
  --allow-production \
  --confirm-database=EXACT_SCHEMA
```

9.1 Docker update

Docker updates accept only metadata with an immutable, signed image digest. `latest` is not a valid update contract. Without `--apply-compose`, the verified digest can be prepared; pull and recreate remain deliberate operator steps.

```
bin/keule update:check
bin/keule update --mode=docker --confirm-backup --yes

docker/bin/keule-compose pull
docker/bin/keule-compose up -d --force-recreate
docker/bin/keule-compose ps
curl --fail http://127.0.0.1:8080/healthz
```

Backup requirement before updates

Before every real update, operators are responsible for database and file backups that match the target version.

10. Security and access control

The security model combines classic web protection mechanisms, role-based administration, area-related visibility, file ACLs, rate limits, and strict path checks. Operators are additionally responsible for TLS, reverse proxy, firewall, backup protection, and operating system hardening.

Area	Mechanism	Operator relevance
Authentication	Password hashes, session cookies, remember-me rotation, password reset tokens.	Enforce HTTPS and secure cookie contexts in production.
Authorization	Roles `Administrator`, `Moderation`, `User`, area rights, and ACL checks.	Discuss and test the permission model before the pilot.
CSRF	Session token and header/form checks for mutating actions.	Reverse proxy must not distort headers.
Rate limits	Login, registration, and support limits.	Align thresholds with real usage.
File security	Path containment, upload validation, ACL, soft locks.	Control storage rights, NAS behavior, and backup access.
WebDAV	Private directories only, Basic Auth with API token, HTTPS required, write audit.	Document API token process and client trust.
Audit	Structured app and error logs, audit views.	Audit coverage is not the same as a complete OS/web server journal.

Visibility mode

In strict operation, conflict cases are handled fail-closed. Legacy compatibility should be understood as a transition path and should not run unnoticed in parallel with strict expectations.

11. Operations, monitoring, and diagnostics

Operations rely on health checks, CLI diagnostics, scheduler heartbeat, logs, the system monitor, and optional certificate/storage monitoring. The decisive point is that diagnostic tools show blockers and warnings without outputting secrets or changing production paths without being asked.

Tool / view	Checks / shows	Typical operator action
`/healthz`	DB reachability, migrations, admin account, runtime paths.	Check after start, update, and restore.
`bin/keule preflight`	Configuration, PHP, DB, storage, TLS/proxy, scheduler, and optional services.	Before installation, after domain changes, before go-live.
`bin/keule diagnose --json`	Detailed diagnostics with OK/WARNING/BLOCKER.	Export for operating documentation and support cases.
Scheduler heartbeat	Updated after successful periodic runs.	If heartbeat is missing, check timer/cron/container logs.
System monitor	Resources, storage, update view, certificates, and logs depending on configuration.	Control view for admins; not a substitute for infrastructure monitoring.
App/error logs	Structured operational and error events.	Log rotation, access, and retention according to operator policy.

```
docker/bin/keule-compose ps
docker/bin/keule-compose logs init
docker/bin/keule-compose logs app nginx scheduler
curl --fail http://127.0.0.1:8080/healthz

systemctl status keule-scheduler.timer
```

Tool / view	Checks / shows	Typical operator action

```
systemctl list-timers keule-scheduler.timer
journalctl -u keule-scheduler.service
```

12. Backup and recovery

A resilient backup always includes DB and file-system data. A restore must be tested at least once in an isolated environment before production use.

Backup component	Why required	Note
MariaDB / DB	Core objects, users, areas, tasks, calendar, chat, notes, news, tokens, and metadata.	Create a consistent dump; test restore against the matching Keule version.
Private/shared files	User data and shared storage.	Preserve permissions, ownership, and path structure.
Chat/task attachments	Binary attachments for modules.	Back up together with the DB so references remain consistent.
Support tickets	Remaining domain not yet fully migrated to DB.	Do not forget it, even though core modules are DB-first.
Logs / mail dumps	Diagnostics and traceability.	Back up or rotate depending on compliance and storage policy.
Configuration / secrets	Env file, update token, trust key, certificates.	Back up separately and with special protection; never copy into release artifacts.

```
# Native example: DB dump
mysqldump -u"$DB_USER" -p"$DB_PASS" "$DB_NAME" > keule.sql

# File-system example: back up relevant paths
rsync -aHAX /var/lib/keule/storage/ backup/keule-storage/
rsync -aHAX /var/lib/keule/data/    backup/keule-data/
rsync -aHAX /var/log/keule/         backup/keule-logs/
```

Restore rule

After a failed migration, do not simply start an older image against an already migrated schema. Restore means: a DB state matching the version plus matching file data.

13. Interfaces and integrations

Interface	Status / function	Operational requirements
Web / PWA	Browser access, installable PWA, service worker, and push handling.	Stable origin, TLS, possibly service-worker reload after domain change.
WebDAV	Private user directories via `/dav` or `public/dav.php`.	HTTPS, API token, `Authorization` header in proxy, observe PUT limit.
E-mail	SMTP, file mail, or disabled mail transport.	Registration/password reset require active mail transport. Test with `mail:test`.
Push	Web Push via VAPID and browser subscription.	Generate VAPID keys, browser permission, observe diagnostic blockers.
Update server	Anonymous `update:check`, protected downloads with customer token.	Maintain trust key, token file, and update channel.
Reverse proxy	External TLS termination possible.	Trust forwarded headers only from configured CIDR networks.

13.1 WebDAV scope

WebDAV is deliberately limited to private directories. `shared` and area directories are not part of the current WebDAV scope. Supported methods are `OPTIONS`, `PROPFIND`, `GET`, `HEAD`, `PUT`, `MKCOL`, `DELETE`, and `MOVE`; write operations are audited.

WebDAV note

A WebDAV client works with username and API token. The normal user password should not be used as an integration credential.

14. Quality assurance and acceptance

For operators, not only the initial installation is relevant, but reproducible acceptance: basic smoke, update smoke, backup/restore test, optional services, and negative tests should be documented before production use.

Acceptance area	Minimum check
Installation	Release/image reference verified, configuration validated, preflight without blockers.
Application	Login with `KeuleAdmin`, session, start page, central modules reachable.
Persistence	DB connection, migrations, storage read/write/rename/delete.
Scheduler	Exactly one active scheduler; reminder/heartbeat run successful.
TLS / Proxy	APP_URL, certificate, secure context, forwarded-header trust, and WebDAV HTTPS gate.
Updates	`update:check`, signature/hash verification, maintenance-mode behavior, health check after update.
Backup / Restore	Recovery in isolated environment with matching version successful.
Optional services	Mail test, Push/VAPID, WebDAV client test, registration according to chosen mode.
Negative test	**Expected behavior**
Manipulated signature or checksum	Update aborts before installation.
Manipulated archive	SHA-256 or internal verifier blocks.
Untrusted forwarded headers	HTTPS context is not accepted; WebDAV remains protected.
Parallel schedulers	Preflight blocks or second instance aborts via lock.
Source tree as update target	Updater refuses without explicitly isolated test target.
Migration error	Maintenance mode remains active; no automatic rollback.

15. Known limitations and items to verify

The following points are not showstoppers for a pilot operation, but they should be evaluated deliberately in the operator concept.

Limitation / checkpoint	Classification
No URL subpaths	`APP_URL` must be a root URL; operation under e.g. `/keule` is not intended.
No multi-node operation	Scheduler and storage locks are designed for exactly one active instance.
WebDAV only private directories	Shared/area folders remain outside the current DAV scope.
DB TLS reserved	Secure external DB connections via private network or tunnel until full implementation is complete.
2FA not yet production-ready	In development. Runtime should remain fail-closed; use `disabled`. Coming soon!
PWA not offline-first	Offline notices and push yes; complete offline data editing no.
No automatic DNS/firewall/router configuration	Especially WSL2, Let's Encrypt HTTP-01, and public DAV domains require operator verification.
No automatic DB rollback	Backups must exist and restore paths must be tested before updates and migrations.
Audit coverage partial	App/error log analysis does not replace a complete SIEM or OS journal.
Docker digest instead of latest	Deliberately stable; requires clean digest/signature verification in the update process.

16. Recommended pilot / PoC workflow

1. Define target and scope: user groups, modules, internal/external reachability, registration, mail, and WebDAV.
2. Choose operating model: initially prefer Docker Compose with local MariaDB and named volumes.
3. Prepare DNS/TLS/reverse proxy and finalize `APP_URL`.

4. Create env file with `config:assistant`, store secrets separately, and run `config:validate`.
5. Reach preflight without blockers; document warnings and consciously accept or fix them.
6. Start installation, check health and central modules.
7. Map roles, areas, ACLs, and sample users to the real organization structure.
8. Enable mail, push, and WebDAV only if they are really tested in the PoC.
9. Run backup and restore in an isolated environment.
10. Run update smoke with a disposable or test instance.
11. Create acceptance protocol: version, configuration, blockers/warnings, smoke results, remaining points.
12. Decide go/no-go based on operational capability, security requirements, and restore capability.

PoC recommendation

A PoC should not merely show that the interface works. The decisive points are restore, update path, TLS/proxy trust, permission model, and a realistically tested scheduler.

17. Appendix: important commands and terms

17.1 Commands

Purpose	Command
Show version	`bin/keule version`
Configuration assistant	`bin/keule config:assistant --output=/etc/keule/keule.env`
Validate configuration	`bin/keule config:validate --env-file=/etc/keule/keule.env`
Preflight	`bin/keule preflight --env-file=/etc/keule/keule.env`
Diagnostics as JSON	`bin/keule diagnose --env-file=/etc/keule/keule.env --json`
Migration	`bin/keule migrate --allow-production --confirm-database=EXACT_SCHEMA`
Create admin	`bin/keule admin:create --password-stdin --allow-production --confirm-database=EXACT_SCHEMA`
Mail test	`bin/keule mail:test --to=admin@example.invalid`
Generate VAPID	`bin/keule vapid:generate --allow-production --confirm-database=EXACT_SCHEMA`
Check update	`bin/keule update:check`
Start update	`bin/keule update --confirm-backup --yes`
Maintenance mode	`bin/keule maintenance:on` / `bin/keule maintenance:off`

17.2 Terms

Term	Meaning
Release artifact	Finished `keule-<version>.tar.gz` with PHP runtime files, migrations, assets, production dependencies, and manifest.
RELEASE-MANIFEST.json	Internal file contract with version, commit, configuration schema, and SHA-256 per package file.
Preflight	Read-only check of installation and operating prerequisites.
Health check	Read-only runtime check via `/healthz`.
Digest	Immutable container image reference `image@sha256:<digest>` instead of tag `latest`.
Update token	Customer-specific download token for protected release downloads; not a GitHub token.
Trust key	Public key for verifying release signatures.

Term	Meaning
Scheduler lock	Lock preventing parallel background runs.
Strict visibility	Operating mode that handles visibility conflicts fail-closed.