LOGZILLA DOCUMENTATION

Docker Containers

Reference for the Docker containers that make up a LogZilla deployment, including syslog, parser, storage, query, API, AI, and auxiliary services

Administration · Generated April 27, 2026 · logzilla.ai/docs/administration/docker-containers

Docker Containers Used by LogZilla

LogZilla operates by means of multiple docker containers handling various facets of its operation. The following are the containers used:

Container Name	Description
lz_celerybeat	Schedules periodic background tasks for the platform.
lz_gunicorn	API server for the LogZilla backend.
lz_celeryworker	Executes background jobs and module orchestration tasks.
lz_parsermodule	Parses, normalizes, and enriches incoming events using rules/apps.
lz_querymodule	Runs searches/analytics, evaluates triggers, and publishes live results.
lz_storagemodule-1	Event storage and indexing service (dedup, archiving, aggregates).
lz_ai	AI Copilot backend; powers AI features when enabled.
lz_watcher	Orchestrates, monitors, and restarts LogZilla containers.
lz_snmptrapd	Receives SNMP traps (UDP 162) and injects them as events.
lz_httpreceiver	HTTP/HTTPS event ingest API (token-protected) into LogZilla.
lz_sec	Simple Event Correlator for stateful, multi-event correlation.
lz_scriptserver	Runs user-defined scripts for trigger actions/correlation.
lz_front	Web frontend serving UI and proxying API/ingest endpoints.
lz_syslog	Syslog-ng receiver for BSD/RFC5424, JSON, RAW, and TLS inputs.
lz_logcollector	Collects and consolidates internal container logs for diagnostics.
lz_mailer	Outbound SMTP service for alerts and test emails.
lz_telegraf	Collects internal metrics and ships them to time-series storage.
lz_qdrant	Vector database backing AI features (embeddings/search).

Container Name	Description
lz_influxdb	Time-series database for metrics and aggregates.
lz_postgres	Primary relational store for users, dashboards, triggers, settings.
lz_redis	In-memory caching and queues for ephemeral data.

Note: AI containers appear only when the AI feature is enabled and configured in the web interface under Settings → System → AI or via the CLI with the `logzilla settings update` command. (`AI_ENABLED` true with `API_KEY` and `MODEL_NAME`). When enabled, the web interface proxies AI under the `/ai` path.

Overview and Architecture Placement

LogZilla uses a set of Docker containers that cooperate to receive, normalize, store, query, and present log data. These containers are grouped by function so the platform can scale pieces independently and restart modules without impacting the entire system.

At a high level:

- Ingestion enters through `lz_syslog` and optional SNMP and HTTP receivers.
- Processing and parsing occur in `lz_parsermodule`, with supporting modules such as `lz_dictionarymodule` and `lz_forwardermodule` for enrichment and forwarding.
- Durable application state is stored in `lz_postgres`; time-series metrics in `lz_influxdb`.
- Queries and correlation are executed by `lz_querymodule` and `lz_sec`, with storage managed by `lz_storagemodule-1`.
- Results are returned through the API (`lz_gunicorn`) and UI proxy (`lz_front`).
- Background workers and the watcher manage orchestration and periodic tasks.

Container Groups and Roles

Frontend and API

- `lz_front`
  - Web frontend serving UI and proxying API/ingest endpoints.
  - Terminates HTTP/HTTPS and proxies traffic to internal services.
- `lz_gunicorn`
  - API server for the LogZilla backend.

- REST API service used by the UI and CLI.

Workers and Orchestration

- `lz_celeryworker`
  - Executes background jobs and module orchestration tasks.
- `lz_celerybeat`
  - Schedules periodic background tasks for the platform.
- `lz_watcher`
  - Orchestrates, monitors, and restarts LogZilla containers.

Ingestion and Parsing

- `lz_syslog`
  - Syslog-ng receiver for BSD/RFC5424, JSON, RAW, and TLS inputs.
  - Port defaults are documented in Network Communications. Settings live under `syslogng` in Server Settings.
- `lz_snmptrapd`
  - Receives SNMP traps (UDP 162) and injects them as events.
- `lz_httpreceiver`
  - HTTP/HTTPS event ingest API (token-protected) into LogZilla.
- `lz_parsermodule`
  - Parses, normalizes, and enriches incoming events using rules/apps.

Query, Storage, and Correlation

- `lz_storagemodule-1`
  - Event storage and indexing service (dedup, archiving, aggregates).
- `lz_querymodule`
  - Runs searches/analytics, evaluates triggers, and publishes live results.
- `lz_sec`
  - Simple Event Correlator for stateful, multi-event correlation.
- `lz_scriptserver`
  - Runs user-defined scripts for trigger actions/correlation.

Datastores and Metrics

- `lz_postgres`
  - Primary relational store for users, dashboards, triggers, settings.
- `lz_influxdb`
  - Time-series database for metrics and aggregates.
- `lz_redis`
  - Ephemeral cache and queues (e.g., query results and task coordination).
- `lz_telegraf`
  - Collects internal metrics and ships them to time-series storage.
- `lz_mailer`
  - Outbound SMTP service for alerts and test emails.
- `lz_logcollector`
  - Collects and consolidates internal container logs for diagnostics.

AI (Conditional)

- `lz_ai`
  - AI Copilot backend; powers AI features when enabled.
  - Available only when AI is enabled and configured.
- `lz_qdrant`
  - Vector database backing AI features (embeddings/search).
  - Runs only when AI is enabled. Persistent volume name: `qdrant`.

Enablement: set `AI_ENABLED=true` and configure `API_KEY` and `MODEL_NAME` in the `ai` settings group. When enabled, `lz_front` proxies the AI backend under `/ai`.

Lifecycle and Control

Container lifecycle is managed through the LogZilla CLI (see System Commands for full details). For Docker-specific troubleshooting, check container status:

```
docker ps --format 'table {{.Names}}\t{{.Status}}\t{{.Image}}'
```

Logs and Troubleshooting

Primary platform log:

```
sudo tail -f /var/log/logzilla/logzilla.log
```

Per-container logs (examples):

```
docker logs -n 100 lz_front
docker logs -n 100 lz_syslog
docker logs -n 100 lz_gunicorn
```

Quick checks:

- Frontend is reachable on configured HTTP/HTTPS ports.
- Syslog receivers are listening on the configured ports (see Network Communications for defaults and options).
- Postgres and InfluxDB are running and healthy.

If a module configuration has been updated and does not appear to take effect, use a targeted restart for that component.

Data Persistence and Volumes

The platform persists data across container restarts via Docker volumes.

- Postgres stores durable application data.
- Archive data is managed by the storage pipeline (see Data Archiving and Retention for lifecycle and relocation guidance).
- InfluxDB stores time-series metrics.
- Qdrant stores AI vectors when AI is enabled (volume `qdrant`).

Before upgrades or large configuration changes, snapshots and backups are recommended. See Offline Installs and Upgrades.

Networking Model

`lz_front` terminates HTTP/HTTPS and proxies to internal services by container name. Syslog receiver ports are configurable; defaults are documented in Network Communications. Name resolution for containers uses the Docker network and service names. When custom host mappings are needed, use `/etc/logzilla/hosts.in` (see Custom DNS).

Security and Best Practices

- Use HTTPS in production for the web UI and API. See Using HTTPS.
- Restrict inbound syslog ports to authorized sources using firewalls and network policy.
- Prefer the `logzilla` CLI for lifecycle and configuration changes.
- Take snapshots/backups before upgrades or significant configuration changes.
- Monitor disk utilization for Postgres, archives, and InfluxDB volumes.