SHERKETYERP

System requirements

September 15, 2026

System requirements

Before deploying Sherkety ERP, check the container, database, network, and service requirements, then test the database connection.

System requirements

The image starts from Ubuntu 22.04 and the deployment uses Flectra 3.0 with an external PostgreSQL database. Check the environment before startup, then check connectivity after startup.

If you need to…	Go to
Check software, database, and network prerequisites	Before you start
Choose between deployment environments	Choose the deployment path
Configure values, mounts, ports, and startup	Configure and start the deployment
Select connection and security settings	Choose connection and security options
Test PostgreSQL access or diagnose startup	Test the connection and troubleshoot startup
Confirm deployment readiness	Confirm deployment readiness

Before you start

The deployment prerequisites are the software and access needed to run the container
and reach its external database.

Requirement	What to have ready
Container base	The deployment image uses Ubuntu 22.04.
Python	Use Python 3.11 for Flectra 3.0 compatibility.
Database	An Azure VM has PostgreSQL installed and running.

Requirement	What to have ready
PostgreSQL	Use PostgreSQL 14 or later as the recommended version.
Docker	Use Docker 20.10 or later for local development.
Docker Compose	Use Docker Compose 2.0 or later for local development.
Database access	Have access to the Azure VM PostgreSQL database.
Network access	Allow connections from the local development machine, Rancher/Kubernetes cluster IP ranges, and the Docker host network.

Choose the deployment path

Choose the path that matches where you are running the service. Local development uses
Docker Compose; the staging and production environments use an Azure VM PostgreSQL database.

If you need to…	Use this path
Run local development	Use the Docker Compose instructions and the `.env` file.
Run staging	Use the Azure VM database with the staging configuration.
Run production	Use the Azure VM database with the production configuration and subdomain routing.

The database remains outside the container in the Azure deployment. The container keeps Flectra data and sessions in its mounted data location.

Configure and start the deployment

Begin after Docker, Docker Compose, and database access are ready.

Prerequisites

- You have Docker 20.10 or later.
- You have Docker Compose 2.0 or later.
- PostgreSQL is running on the Azure VM.
- The database accepts connections from the machine or cluster that runs the service.

Steps

1. Open a shell in the Flectra directory.
2. Create the environment file from `.env.example`.

```
cp .env.example .env
```

3. Set the database host, database port, database user, database password, and deployment environment in `.env`.

Setting	Value or meaning
`DB_HOST`	Azure VM PostgreSQL host, or `localhost` for local development
`DB_PORT`	`5432`
`DB_USER`	Account used to connect
`DB_PASSWORD`	Password for the connection
`ENVIRONMENT`	`development`

4. Start the service with the environment file.

```
docker-compose --env-file .env up -d
```

5. Check the service logs.

```
docker-compose logs -f flectra
```

6. Confirm that the service health check requests `/web` on port `8069`.

Result: The container starts with the data, custom-addon, configuration, and log mounts. The image exposes HTTP on port `8069`, long polling on port `8072`, and XMLRPC/API on port `8073`; Compose maps the web and long-polling ports by default. Continue with the connection and security choices before exposing the service beyond local development.

Choose connection and security options

Choose the connection and security settings before exposing the service beyond local development.

If you need to…	Set in `.env`, PostgreSQL access rules, or network configuration
Connect to the Azure database	Set the database host in `.env` and allow the required source addresses in PostgreSQL access rules.
Permit local development	Allow the development machine's address to connect to PostgreSQL.
Permit Kubernetes or Rancher	Allow the cluster address range to connect to PostgreSQL.
Protect production traffic	Enable SSL/TLS and use a connection string that requires SSL.
Restrict database access	Use specific addresses in `pg_hba.conf` instead of broad ranges.
Expose the API	Confirm whether the XMLRPC/API service on port `8073` is required before opening it.

Use strong passwords. Restrict database access. Enable SSL/TLS for production. Schedule
automated backups. The container exposes port `8073`, while the Compose service maps the
web and long-polling ports only; confirm the API exposure decision for the deployment.

Result: The database connection path and the security settings match the intended deployment environment. Run the connection test next.

Test the connection and troubleshoot startup

Run the connection test after configuring the database host, user, and database.

Prerequisites

- You know the Azure VM host name or address.
- PostgreSQL is running.
- The database user has permission to connect.

Steps

1. Open a shell on the local machine.
2. Enter `psql -h <azure-vm-ip> -U <database-user> -d <database-name> -c "\l"` and press Enter.
3. Confirm that the command lists the databases.

4. If the container does not start, inspect the Flectra logs.

```
docker-compose logs -f flectra
```

5. If the connection fails, use the symptom in the table to choose the checks to perform.

If you see…	Check…
`could not connect to server: Connection refused`	PostgreSQL status, port `5432`, network security group rules, and `listen_addresses`
Authentication fails for the database user	The `.env` password, the `pg_hba.conf` authentication method, and the database user
`FATAL: no pg_hba.conf entry for host`	The client address in `pg_hba.conf`, then reload PostgreSQL
`FATAL: SSL connection is required`	The production SSL configuration and the connection string's SSL mode

Resolve the URL conflict before giving an access address to readers: one access URL is `http://localhost:7071`, while the image and Compose configuration use port `8069`.

Result: The PostgreSQL command lists the available databases, or the displayed failure identifies the database or network setting to correct. Complete the readiness checks before treating the environment as ready.

Confirm deployment readiness

Before treating the environment as ready, complete these deployment checks.

Readiness action	How to complete it
Confirm the host platform	Confirm that the host operating system and architecture support the container deployment.
Confirm capacity	Set CPU, memory, storage, and database capacity for the intended workload.
Confirm client access	Confirm the supported browsers and display requirements for the web interface.
Confirm network access	Confirm firewall, proxy, DNS, outbound, and PostgreSQL source-network rules.

Readiness action	How to complete it
Confirm the PostgreSQL version	Confirm whether PostgreSQL 14 is the required minimum for the deployment.
Confirm API exposure	Confirm whether port `8073` must be reachable, then apply the network rule.
Confirm recovery settings	Set backup retention and recovery targets for the deployment.
Confirm production security	Set the SSL/TLS enforcement and certificate policy.
Confirm distribution scope	Confirm whether the configuration is for internal operations before sharing it.

Result: The deployment owner has completed the readiness checks before the service
is treated as ready.