Kenya Farm IoT – Installation Manual

Version 1.0
Document type: Installation & setup guide
Last updated: February 2025

1. Introduction

This manual describes how to install and configure the **Kenya Farm IoT** system: a full-stack IoT farming dashboard for Kenyan farmers. The system includes:

- **Backend API** (Node.js, Express, TypeScript) for auth, sensors, readings, alerts, and SMS
- **Frontend** (React, Vite, Tailwind) – web dashboard and PWA
- **PostgreSQL + TimescaleDB** for time-series sensor data
- **MQTT** for real-time sensor ingestion
- **SMS** via Africa's Talking (or mock/email/WhatsApp modes)
- **Docker** support for local and production deployment

2. Prerequisites

Requirement	Version / notes
Node.js	18 or higher (LTS recommended)
npm	9+ (bundled with Node)
PostgreSQL	14+ (or use Docker)
(Optional) Docker	For database, MQTT, or full-stack
(Optional) MQTT broker	Mosquitto or similar, for sensor data

For production: A Linux server (e.g. Ubuntu 22.04), domain name for SSL, and (optional) Africa's Talking account for SMS.

3. Quick start (local development)

3.1 Clone the repository

```
git clone <repository-url> kenya-farm-iot
cd kenya-farm-iot
```

3.2 Database setup

Create a PostgreSQL database. Using Docker with TimescaleDB:

```
docker run -d --name timescale \
  -e POSTGRES_PASSWORD=postgres \
  -e POSTGRES_DB=kenya_farm_iot \
  -p 5432:5432 \
  timescale/timescaledb:latest-pg16
```

Or use an existing PostgreSQL 14+ instance and create a database:

```
createdb kenya_farm_iot
```

Set the connection URL:

```
export DATABASE_URL="postgresql://postgres:postgres@localhost:5432/kenya_farm_iot"
```

(On Windows PowerShell use `$env:DATABASE_URL = "postgresql://..."` .)

3.3 Backend setup

```
cd backend
cp .env.example .env
```

Edit `.env` and set at least:

- `DATABASE_URL` – PostgreSQL connection string
- `JWT_SECRET` – A long random string (e.g. `openssl rand -hex 32`)

Then:

```
npm install
npm run db:migrate
npm run dev
```

The API runs at **http://localhost:4000**. A `/health` endpoint is available for checks.

3.4 Frontend setup

In a new terminal:

```
cd frontend
npm install
npm run dev
```

The app runs at **http://localhost:5173** and proxies `/api` to the backend.

3.5 First use

1. Open **http://localhost:5173**
2. Click **Register** and enter a phone number (e.g. `0712345678`) and password
3. Log in and use **Dashboard**, **Sensors**, and **Alerts**

---

4. Environment variables

4.1 Backend (`backend/.env`)

Variable	Description	Required	Default
`PORT`	API port	No	`4000`
`FRONTEND_URL`	CORS allowed origin	No	`http://localhost:5173`
`DATABASE_URL`	PostgreSQL connection string	**Yes**	—
`JWT_SECRET`	Secret for JWT signing	**Yes**	—
`MQTT_URL`	MQTT broker URL	No	—
`MQTT_TOPIC_PREFIX`	Topic prefix for sensors	No	`kenya-farm`
`ALERT_THRESHOLD_LOW`	Low soil moisture % for alert	No	`20`
`ALERT_THRESHOLD_HIGH`	High soil moisture % for alert	No	`90`
`SMS_MODE`	`mock` | `email_sms` | `whatsapp` | `production`	No	`mock`
`AFRICAS_TALKING_USERNAME`	Africa's Talking username	For production SMS	—
`AFRICAS_TALKING_API_KEY`	Africa's Talking API key	For production	—

		SMS	
`SMTP_*`	SMTP settings for email_sms mode	For email_sms	—
`TWILIO_*`	Twilio for WhatsApp mode	For whatsapp	—

4.2 Frontend

Variable	Description	Default
`VITE_API_URL`	Backend API base URL	Unset in dev (uses Vite proxy)

For production build, set `VITE_API_URL` to your API URL (e.g. `https://api.yourdomain.com`).

4.3 Production (`.env.prod`)

Copy `.env.prod.example` to `.env.prod` and set `POSTGRES_PASSWORD`, `JWT_SECRET`, `DOMAIN`, and any SMS/API keys. See `docs/DEPLOYMENT.md` for details.

---

5. Database migrations

Migrations create and update tables (farmers, farms, sensors, sensor_readings, alerts, irrigation_logs, sms_messages) and the TimescaleDB hypertable for readings.

Run migrations:

```
cd backend
npm run db:migrate
```

For production Docker:

```
docker compose -f docker-compose.prod.yml --env-file .env.prod exec backend node dist/
```

(Use the actual entry point if your app exposes migrations differently.)

---

6. Docker (full stack local)

From the project root:

```
cp .env.example .env
# Edit .env with POSTGRES_PASSWORD, JWT_SECRET, etc.
docker-compose up -d
```

This starts:

- **Backend** – http://localhost:4000
- **Frontend** – http://localhost:5173
- **PostgreSQL (TimescaleDB)** – port 5432

To include the MQTT broker:

```
docker-compose --profile full up -d
```

Set `MQTT_URL=mqtt://localhost:1883` (or `mqtt://mqtt:1883` from inside the backend container) in `.env` if you use MQTT.

Run migrations after first start (see Section 5).

---

7. MQTT sensor ingestion

Sensors can send data via MQTT. Configure `MQTT_URL` and optionally `MQTT_TOPIC_PREFIX`.

Option A – Topic with sensor ID in path

- Topic: `kenya-farm/{farmer_id}/sensors/{sensor_id}`
- Payload (JSON): `{"value": 42, "unit": "%", "recorded_at": "2025-02-07T12:00:00Z"}`

Option B – Single topic, sensor_id in payload

- Topic: `kenya-farm/readings`
- Payload: `{"sensor_id": "uuid-of-sensor", "value": 42, "unit": "%"}`

The backend subscribes, stores readings, and can create alerts and send SMS when thresholds are breached.

---

8. SMS configuration

8.1 Modes

- **mock** – Logs to console and database; no real SMS. Good for development.
- **email_sms** – Sends via SMTP (e.g. email-to-SMS gateway).

- **whatsapp** – Uses Twilio WhatsApp.
- **production** – Africa's Talking API (real SMS in Kenya).

Set `SMS_MODE` in `backend/.env`.

8.2 Africa's Talking (production)

1. Sign up at Africa's Talking
2. Get **username** and **API key** from the dashboard
3. Set `AFRICAS_TALKING_USERNAME` and `AFRICAS_TALKING_API_KEY` in `backend/.env`

When an alert is created (e.g. low soil moisture), the system can send an SMS to the farmer's registered phone number.

---

9. Production deployment (overview)

For a full production guide see **docs/DEPLOYMENT.md**. Summary:

1. Use `docker-compose.prod.yml` with `.env.prod`
2. Configure Nginx as reverse proxy and enable SSL (e.g. Let's Encrypt)
3. Run migrations after first deploy
4. Schedule database backups (e.g. `scripts/backup-db.sh`) and optional S3/GCS upload
5. Use `scripts/deploy.sh` or `scripts/deploy.ps1` for updates

---

10. Troubleshooting

Issue	Suggestion
Database connection failed	Check `DATABASE_URL`, ensure PostgreSQL is running and the database exists
Migrations fail	Ensure TimescaleDB extension is available if using hypertables; check DB user permissions
CORS errors	Set `FRONTEND_URL` to the exact origin of the frontend (e.g. `https://app.yourdomain.com`)
MQTT not receiving	Verify `MQTT_URL` and that the broker is reachable; check topic and payload format
SMS not sent	For production, check Africa's Talking credentials and `SMS_MODE`; for mock, check logs and `sms_messages` table
Frontend can't reach API	In production set `VITE_API_URL` and rebuild; in dev ensure backend is on port 4000 and proxy is used

---

11. Security checklist

- Use a **strong random JWT_SECRET** in production (e.g. 32+ bytes hex).
- Use **HTTPS** and restrict **FRONTEND_URL** / CORS to your frontend origin.
- Restrict **POST /api/readings** (e.g. API key or auth) if exposed to the internet.
- Store database URLs and API keys in secrets (e.g. environment or secrets manager), never in code.
- Keep dependencies updated (`npm audit`, `npm update`).

End of Installation Manual