KINTO Operations & QA

Linux/Ubuntu Deployment Guide

Production Deployment with Multi-Application Support

Last Updated: November 14, 2025

Conflict-Free Setup • 53 Tables • Production Ready

Table of Contents

1. Overview & Prerequisites

System Overview

This guide covers deploying KINTO Operations & QA on a Linux/Ubuntu server that may already be running other applications. We will configure the application to run on a different port to avoid conflicts.

Server Requirements

- Ubuntu 20.04 LTS or later (18.04+ supported)
- Node.js 18.x or 20.x
- PostgreSQL 13+ (14 or 15 recommended)
- Minimum 2GB RAM (4GB recommended)
- Minimum 10GB disk space
- Root or sudo access

Assumed Current Setup

- You have another application running on port 3000 or 5000
- PostgreSQL may or may not be installed
- Nginx may or may not be configured

& IMPORTANT: This guide assumes conflicts exist. We will use port 5001 for KINTO by default.

2. Server Preparation

Update System Packages

```
sudo apt update
sudo apt upgrade -y
```

Install Essential Build Tools

```
sudo apt install -y build-essential curl git
```

Install Node.js (if not installed)

```
# Using NodeSource repository for Node.js 20.x
curl -fsSL https://deb.nodesource.com/setup_20.x | sudo -E bash -
sudo apt install -y nodejs
# Verify installation
node --version  # Should show v20.x.x
npm --version   # Should show 10.x.x
```

Create Application User (Recommended)

```
# Create dedicated user for KINTO
sudo adduser --disabled-password --gecos "" kinto
sudo usermod -aG sudo kinto
```

3. PostgreSQL Installation & Setup

Check if PostgreSQL is Already Installed

```
psql --version
```

If PostgreSQL is already installed, skip to database creation.

Install PostgreSQL 15

```
# Add PostgreSQL APT repository
sudo sh -c 'echo "deb http://apt.postgresql.org/pub/repos/apt $(lsb_release -cs)-pgdg main" > /etc/apt/sources.list.d/pgdg.list'
wget --quiet -O - https://www.postgresql.org/media/keys/ACCC4CF8.asc | sudo apt-key add -
# Install PostgreSQL
sudo apt update
sudo apt install -y postgresql-15 postgresql-contrib-15
```

Start and Enable PostgreSQL

```
sudo systemctl start postgresql
sudo systemctl enable postgresql
sudo systemctl status postgresql
```

Configure PostgreSQL Authentication

```
# Edit pg_hba.conf to allow password authentication
sudo nano /etc/postgresql/15/main/pg_hba.conf
```

Change this line:

```
local   all             all                                     peer
```

To:

```
local   all             all                                     md5
```

```
# Restart PostgreSQL
sudo systemctl restart postgresql
```

4. Database Creation & Migration

Create Database and User

```
# Switch to postgres user
sudo -u postgres psql
-- Create database
CREATE DATABASE kinto_qa;
-- Create user with strong password
CREATE USER kinto_user WITH ENCRYPTED PASSWORD 'your_strong_password_here';
-- Grant privileges
GRANT ALL PRIVILEGES ON DATABASE kinto_qa TO kinto_user;
-- Exit psql
\q
```

Test Database Connection

```
psql -U kinto_user -d kinto_qa -h localhost
# Enter password when prompted
\q  # Exit after successful connection
```

Upload Application Files

```
# Option 1: Using Git (recommended)
cd /home/kinto
git clone https://your-repo-url.git kinto-app
cd kinto-app
# Option 2: Using SCP from local machine
# On your local machine:
scp -r /path/to/kinto-app user@server-ip:/home/kinto/
```

Execute Database Migration Scripts

```
cd /home/kinto/kinto-app
# Execute baseline schema (31 tables)
psql -U kinto_user -d kinto_qa -h localhost -f database_scripts/01_schema.sql
# Insert seed data
psql -U kinto_user -d kinto_qa -h localhost -f database_scripts/02_seed_data.sql
# Create performance indexes
psql -U kinto_user -d kinto_qa -h localhost -f database_scripts/03_indexes.sql
```

Execute Incremental Migrations (22 Tables)

```
# Legacy migrations
psql -U kinto_user -d kinto_qa -h localhost -f
updated_dbscripts/20251106_163500_production_management.sql
psql -U kinto_user -d kinto_qa -h localhost -f
updated_dbscripts/20251107_020000_notification_config.sql
psql -U kinto_user -d kinto_qa -h localhost -f
updated_dbscripts/20251110_incremental_whatsapp_checklist.sql
psql -U kinto_user -d kinto_qa -h localhost -f
updated_dbscripts/20251111_add_photo_spare_parts_columns.sql
# Complete schema migrations
psql -U kinto_user -d kinto_qa -h localhost -f
updated_dbscripts/20251112_140000_financial_invoicing.sql
psql -U kinto_user -d kinto_qa -h localhost -f
updated_dbscripts/20251112_140001_sales_returns_credit_notes.sql
psql -U kinto_user -d kinto_qa -h localhost -f
updated_dbscripts/20251112_140002_production_management.sql
psql -U kinto_user -d kinto_qa -h localhost -f
updated_dbscripts/20251112_140003_configuration_assignments.sql
# Recent patches
psql -U kinto_user -d kinto_qa -h localhost -f
updated_dbscripts/20251112_150000_add_credit_notes_approved_by.sql
psql -U kinto_user -d kinto_qa -h localhost -f
updated_dbscripts/20251113_060000_product_category_type_display_order.sql
```

Verify Database Setup

```
    # Count tables (should be 53)
    psql -U kinto_user -d kinto_qa -h localhost -c "SELECT COUNT(*) FROM
information_schema.tables WHERE table_schema = 'public';"
    # Verify admin user
    psql -U kinto_user -d kinto_qa -h localhost -c "SELECT username, email FROM users WHERE
username = 'admin';"
```

Expected Results:

- ' Table count: 53
- ' Admin user exists: admin / admin@kinto.com

5. Application Deployment

Install Application Dependencies

```
cd /home/kinto/kinto-app
npm install --production
```

Build Application

```
# Build frontend
npm run build
```

& NOTE: If build fails with memory issues on small servers:

```
# Increase Node.js memory limit
export NODE_OPTIONS="--max-old-space-size=2048"
npm run build
```

6. Port Configuration (Avoiding Conflicts)

Understanding Port Usage

KINTO by default binds to port 5000. If you have another app on 5000, we will use 5001.

Check Which Ports Are In Use

```
# Check all listening ports
sudo netstat -tulpn | grep LISTEN
# Or using ss command
sudo ss -tulpn | grep LISTEN
```

Common Port Assignments

- Port 80: HTTP (Nginx)
- Port 443: HTTPS (Nginx)
- Port 3000: Common for other Node apps
- Port 5000: Your existing application (assumed)
- Port 5001: KINTO (recommended to avoid conflicts)
- Port 5432: PostgreSQL

Configure KINTO Port

Edit the application startup script or environment variable to use port 5001:

```
# In your .env file, set:
PORT=5001
```

7. Process Management with PM2

Install PM2 Globally

```
sudo npm install -g pm2
```

Create PM2 Ecosystem File

```
# Create ecosystem.config.js
nano ecosystem.config.js
```

Add the following content:

```
module.exports = {
   apps: [{
      name: "kinto-app",
      script: "server/index.js",  // Or your entry point
      cwd: "/home/kinto/kinto-app",
      instances: 1,
      exec_mode: "fork",
      env: {
         NODE_ENV: "production",
         PORT: 5001,
      },
      error_file: "/home/kinto/logs/kinto-error.log",
      out_file: "/home/kinto/logs/kinto-out.log",
      log_date_format: "YYYY-MM-DD HH:mm:ss Z",
      merge_logs: true,
      autorestart: true,
      watch: false,
      max_memory_restart: "1G"
   }]
};
```

Create Logs Directory

```
mkdir -p /home/kinto/logs
```

Start Application with PM2

```
cd /home/kinto/kinto-app
pm2 start ecosystem.config.js
# Check status
pm2 status
# View logs
pm2 logs kinto-app
```

Configure PM2 to Start on Boot

```
# Save PM2 process list
pm2 save
# Generate and configure startup script
pm2 startup systemd
# Follow the instructions shown (copy/paste the sudo command)
```

Useful PM2 Commands

```
pm2 list                # List all processes
pm2 restart kinto-app   # Restart application
pm2 stop kinto-app      # Stop application
pm2 delete kinto-app    # Remove from PM2
pm2 logs kinto-app      # View logs
pm2 monit               # Monitor CPU/Memory
```

8. Nginx Reverse Proxy Configuration

Install Nginx (if not installed)

```
sudo apt install -y nginx
sudo systemctl start nginx
sudo systemctl enable nginx
```

Create Nginx Configuration for KINTO

```
sudo nano /etc/nginx/sites-available/kinto
```

Add the following configuration:

```
server {
   listen 80;
   server_name kinto.yourdomain.com;  # Change to your domain
   # Increase upload size for file uploads
   client_max_body_size 50M;
   location / {
      proxy_pass http://localhost:5001;
      proxy_http_version 1.1;
      proxy_set_header Upgrade $http_upgrade;
      proxy_set_header Connection 'upgrade';
      proxy_set_header Host $host;
      proxy_cache_bypass $http_upgrade;
      proxy_set_header X-Real-IP $remote_addr;
      proxy_set_header X-Forwarded-For $proxy_add_x_forwarded_for;
      proxy_set_header X-Forwarded-Proto $scheme;
   }
}
```

Enable the Site

```
# Create symbolic link
sudo ln -s /etc/nginx/sites-available/kinto /etc/nginx/sites-enabled/
# Test configuration
sudo nginx -t
# Reload Nginx
sudo systemctl reload nginx
```

Configure Subdirectory Access (Alternative)

If you want KINTO at yourdomain.com/kinto instead of a subdomain:

```
# Add to your existing Nginx config:
location /kinto/ {
   proxy_pass http://localhost:5001/;
   proxy_http_version 1.1;
   proxy_set_header Upgrade $http_upgrade;
   proxy_set_header Connection 'upgrade';
   proxy_set_header Host $host;
   proxy_cache_bypass $http_upgrade;
}
```

9. Environment Variables Configuration

Create .env File

```
cd /home/kinto/kinto-app
nano .env
```

Add the following environment variables:

```
# Server Configuration
NODE_ENV=production
PORT=5001
# Database Configuration
DATABASE_URL=postgresql://kinto_user:your_strong_password@localhost:5432/kinto_qa
PGHOST=localhost
PGPORT=5432
PGUSER=kinto_user
PGPASSWORD=your_strong_password
PGDATABASE=kinto_qa
# Session Configuration
SESSION_SECRET=your-super-secure-random-string-min-32-characters-long
# Optional: Email Configuration (SendGrid)
# SENDGRID_API_KEY=your_sendgrid_api_key
# Optional: WhatsApp Configuration (Meta Business API)
# META_PHONE_NUMBER_ID=your_phone_number_id
# META_ACCESS_TOKEN=your_access_token
```

Secure the .env File

```
chmod 600 .env
chown kinto:kinto .env
```

10. SSL/HTTPS Setup with Let's Encrypt

Install Certbot

```
sudo apt install -y certbot python3-certbot-nginx
```

Obtain SSL Certificate

```
# Make sure your domain points to this server's IP
sudo certbot --nginx -d kinto.yourdomain.com
```

Follow the interactive prompts. Certbot will automatically configure Nginx.

Auto-Renewal Setup

```
# Test auto-renewal
sudo certbot renew --dry-run
```

Certbot creates a systemd timer for auto-renewal. Verify:

```
sudo systemctl status certbot.timer
```

11. Running Multiple Applications on Same Server

Port Allocation Strategy

Assign unique ports to each application:

- Existing App 1: Port 3000
- Existing App 2: Port 5000
- KINTO: Port 5001
- Future App: Port 5002, 5003, etc.

Nginx Configuration for Multiple Apps

Example showing two applications:

```
# /etc/nginx/sites-available/apps
# Existing App
server {
    listen 80;
    server_name app1.yourdomain.com;
    location / {
        proxy_pass http://localhost:5000;
    }
}
# KINTO App
server {
    listen 80;
    server_name kinto.yourdomain.com;
    location / {
        proxy_pass http://localhost:5001;
    }
}
```

PM2 Process Management for Multiple Apps

```
# List all PM2 processes
pm2 list
```

You should see:

```
% % % % % % %,% % % % % % % % % % % % % %,% % % % % % % % % %,% % % % % % % % % %,% % % % % %
%  id  %  name          %  mode    %  status  %  port     %
% % % % % % %<% % % % % % % % % % % % % %<% % % % % % % % % %<% % % % % % % % % %<% % % % % %
%  0   %  existing-app %  fork    %  online  %  5000     %
%  1   %  kinto-app    %  fork    %  online  %  5001     %
% % % % % % %4% % % % % % % % % % % % % %4% % % % % % % % % %4% % % % % % % % % %4% % % % % %
```

12. Monitoring & Logs

Application Logs (PM2)

```
# View all logs
pm2 logs
# View specific app logs
pm2 logs kinto-app
# View error logs only
pm2 logs kinto-app --err
# Clear logs
pm2 flush
```

Nginx Logs

```
# Access logs
sudo tail -f /var/log/nginx/access.log
# Error logs
sudo tail -f /var/log/nginx/error.log
```

PostgreSQL Logs

```
# View PostgreSQL logs
sudo tail -f /var/log/postgresql/postgresql-15-main.log
```

System Resource Monitoring

```
# Real-time PM2 monitoring
pm2 monit
# System resources
htop        # Interactive process viewer
df -h       # Disk usage
free -h     # Memory usage
```

13. Security Best Practices

Firewall Configuration (UFW)

```
# Install UFW
sudo apt install -y ufw
# Allow SSH (IMPORTANT: Do this first!)
sudo ufw allow 22/tcp
# Allow HTTP and HTTPS
sudo ufw allow 80/tcp
sudo ufw allow 443/tcp
# Enable firewall
sudo ufw enable
# Check status
sudo ufw status
```

& DO NOT open ports 5001, 5432 to the internet. Only Nginx (80/443) should be public.

Database Security

' Use strong passwords for database users
' Restrict PostgreSQL to localhost only
' Regularly update PostgreSQL
' Enable SSL for database connections (production)

Application Security

' Keep Node.js and npm packages updated
' Use environment variables for secrets
' Never commit .env files to version control
' Enable HTTPS (SSL) in production
' Change default admin password immediately
' Delete test users before production

Delete Test Users (Production)

```
psql -U kinto_user -d kinto_qa -h localhost
DELETE FROM users WHERE username IN ('manager_test', 'operator_test', 'reviewer_test');
\q
```

14. Troubleshooting Common Issues

Issue: Port Already in Use

Error: EADDRINUSE: address already in use :::5001

```
# Find process using the port
sudo lsof -i :5001
# Kill the process (if needed)
sudo kill -9 <PID>
```

Or change KINTO to use a different port in .env file.

Issue: Database Connection Failed

```
# Check PostgreSQL is running
sudo systemctl status postgresql
# Check if you can connect manually
psql -U kinto_user -d kinto_qa -h localhost
# Check pg_hba.conf authentication
sudo nano /etc/postgresql/15/main/pg_hba.conf
```

Ensure md5 authentication is enabled for local connections.

Issue: PM2 App Keeps Restarting

```
# Check error logs
pm2 logs kinto-app --err
```

Common causes:

- Missing environment variables in .env
- Database connection issues
- Port conflicts
- Missing npm dependencies

Issue: Nginx 502 Bad Gateway

This means Nginx cannot reach the backend application.

```
# Check if PM2 app is running
pm2 status
# Check if app is listening on correct port
sudo netstat -tulpn | grep 5001
# Check Nginx error logs
sudo tail -f /var/log/nginx/error.log
```

Issue: Application Out of Memory

```
# Increase PM2 memory limit
# Edit ecosystem.config.js:
max_memory_restart: "2G"  // Instead of 1G
# Restart PM2
pm2 restart kinto-app
```

Issue: Cannot Upload Files

Increase Nginx upload limit:

```
# Edit Nginx config
sudo nano /etc/nginx/sites-available/kinto
# Add inside server block:
client_max_body_size 50M;
# Reload Nginx
sudo systemctl reload nginx
```

Issue: Session Expired Immediately

Check SESSION_SECRET is set in .env file and is at least 32 characters long.

15. Backup & Maintenance

Database Backup Strategy

Manual Backup:

```
# Full database backup
pg_dump -U kinto_user -d kinto_qa -h localhost -F c -f /home/kinto/backups/kinto_$(date +%Y%m%d_%H%M%S).dump
```

Automated Daily Backup:

```
# Create backup script
sudo nano /home/kinto/backup-db.sh
```

```
#!/bin/bash
BACKUP_DIR="/home/kinto/backups"
DATE=$(date +%Y%m%d_%H%M%S)
BACKUP_FILE="$BACKUP_DIR/kinto_$DATE.dump"
# Create backup directory if it doesn't exist
mkdir -p $BACKUP_DIR
# Perform backup
pg_dump -U kinto_user -d kinto_qa -h localhost -F c -f $BACKUP_FILE
# Delete backups older than 30 days
find $BACKUP_DIR -name "kinto_*.dump" -mtime +30 -delete
```

```
# Make executable
chmod +x /home/kinto/backup-db.sh
```

```
# Add to crontab (daily at 2 AM)
crontab -e
# Add this line:
0 2 * * * /home/kinto/backup-db.sh
```

Database Restore

```
# Stop the application
pm2 stop kinto-app

# Restore from backup
pg_restore -U kinto_user -d kinto_qa -h localhost -c /home/kinto/backups/kinto_20251114_020000.dump

# Start the application
pm2 start kinto-app
```

Application Updates

```
# Pull latest code (if using Git)
cd /home/kinto/kinto-app
git pull origin main

# Install new dependencies
npm install --production

# Rebuild application
npm run build

# Restart with PM2
pm2 restart kinto-app
```

System Maintenance

```
# Update system packages (monthly)
sudo apt update && sudo apt upgrade -y

# Update Node.js packages
cd /home/kinto/kinto-app
npm outdated  # Check for updates
npm update    # Update packages

# Clean up old logs
pm2 flush
sudo find /var/log/nginx -name "*.log" -mtime +30 -delete
```

Production Deployment Checklist

- [ ] Ubuntu server set up and accessible via SSH
- [ ] Node.js 18+ installed
- [ ] PostgreSQL 13+ installed and running
- [ ] Database kinto_qa created
- [ ] Database user created with strong password
- [ ] All 53 tables created (baseline + incremental)
- [ ] Application files uploaded to server
- [ ] npm dependencies installed
- [ ] Application built successfully
- [ ] .env file created with all required variables
- [ ] Port 5001 configured (or other available port)
- [ ] PM2 installed and ecosystem.config.js created
- [ ] Application started with PM2
- [ ] PM2 startup configured for auto-start
- [ ] Nginx installed and configured
- [ ] Domain pointed to server IP
- [ ] SSL certificate obtained (Let's Encrypt)
- [ ] Firewall (UFW) configured
- [ ] Admin password changed from default
- [ ] Test users deleted (production only)
- [ ] Database backup configured
- [ ] Application tested and accessible
- [ ] Monitoring and logs verified
- [ ] Documentation updated with server details

Quick Reference Commands

PM2 Management

```
pm2 list                    # List all apps
pm2 logs kinto-app          # View logs
pm2 restart kinto-app       # Restart app
pm2 stop kinto-app          # Stop app
pm2 start kinto-app         # Start app
pm2 monit                   # Monitor resources
```

Nginx Management

```
sudo nginx -t               # Test config
sudo systemctl reload nginx # Reload config
sudo systemctl restart nginx# Restart Nginx
sudo systemctl status nginx # Check status
```

PostgreSQL Management

```
sudo systemctl status postgresql  # Check status
psql -U kinto_user -d kinto_qa   # Connect to DB
pg_dump -U kinto_user -d kinto_qa -F c -f backup.dump  # Backup
```

Log Files

```
PM2 Logs:      /home/kinto/logs/
Nginx Access:  /var/log/nginx/access.log
Nginx Error:   /var/log/nginx/error.log
PostgreSQL:    /var/log/postgresql/
```

For support, refer to system administrator or deployment documentation.

Document Version: 1.0 | Last Updated: November 14, 2025
KINTO Operations & QA Management System
Production Deployment Guide - Linux/Ubuntu

' Production Ready