GMLM Platform

Global MLM Software

Features & Deployment Guide

Version 1.0.0 · Laravel 11 · PHP 8.3+

12	176	77	5	4
Phases Completed	Source Files	Database Tables	MLM Plan Types	Deploy Environments

Repository: https://github.com/srinivas182/gml-laravel

Table of Contents

1 Migration Overview

12-phase journey from legacy PHP/CodeIgniter to enterprise Laravel 11

The GMLM Platform was rebuilt from the ground up — not migrated line-by-line. Every design decision was re-evaluated against modern enterprise standards. The result is a production-ready SaaS platform that replaces a legacy PHP 7.4 + CodeIgniter 3 codebase across 12 structured phases.

Phase 0	**Discovery**	Analysed 90 tables, 5,343 files. Found God Table (95+ cols), CSV binary tree, 11 MyISAM tables, 243KB fat controllers.
Phase 1	**Requirements**	PRD: dual wallets, 5 plan types, custom staff roles with individual permission toggles.
Phase 2	**Architecture**	Modular DDD monolith, 7 ADRs, Ed25519 package signing, closure table for O(1) tree queries.
Phase 3	**UI/UX**	Deep Indigo design system, Vue 3 + Inertia.js, dark mode, mobile-first, 10 colour presets.
Phase 4	**Database**	77 tables. God Table → 6 models. Binary tree → closure table. Multi-currency exchange rates.
Phase 5	**Core Platform**	Auth, RBAC, services, domain events, queues, PluginManager, ThemeManager, API.
Phase 6	**Business Modules**	60+ routes, 10 member controllers, 6 admin controllers, 13 Vue pages.
Phase 7	**Customer Architecture**	Isolated per-customer deploy, 13-step provisioning, LicenseManager, 5-step web installer.
Phase 8	**Update Engine**	Ed25519-signed packages, 13-step lifecycle, full backup, auto-rollback, admin UI.
Phase 9	**Performance**	Redis cache (5 TTL levels), query optimizer, 4-tier scaling, Octane, 100k+ concurrent users.
Phase 10	**DevOps**	CI/CD (6 parallel jobs), blue-green deploy, Prometheus+Grafana+Loki, server hardening.
Phase 11	**Testing**	PestPHP suite, PHPStan Level 8, k6 load (200 VU) + stress (1500 VU) tests.
Phase 12	**Package & Deliver**	ZIP + PDF guide + push to GitHub. Plugin/theme framework audit and completion.

2 Complete Feature List

Every capability built and shipped in v1.0.0

MLM Plan Types

- ✓ Binary Plan — left/right legs, matching on weaker leg, daily carry-forward, max daily cap
- ✓ Unilevel Plan — unlimited width, configurable level depth, per-level commission rates
- ✓ Matrix Plan — fixed-width forced matrix (2×2, 3×3, configurable)
- ✓ Single-Leg Plan — linear chain, all members in one sequential downline
- ✓ Custom Plan — configurable combination of any plan mechanics
- ✓ Repurchase Plan — always Unilevel, separate commission structure, spend-only wallet
- ✓ Multi-plan support — multiple active plans per deployment simultaneously
- ✓ Per-country pricing with exchange rate locking at purchase time

Commission & Earnings Engine

- ✓ Direct Referral Bonus — percentage of joining fee to sponsor
- ✓ Level Income — commission distributed up N levels with per-level rates
- ✓ Binary Matching Bonus — daily/weekly calculation with carry-forward accumulation
- ✓ Rank Achievement Bonus — one-time payment on rank upgrade event
- ✓ Target / Milestone Income — configurable achievement bonuses
- ✓ Repurchase Commission — on product re-orders via Unilevel
- ✓ Custom Income Type — admin-defined commission categories
- ✓ UNIQUE idempotency_key on all earnings — mathematically impossible to double-pay
- ✓ Batched commission processing — 10,000 members processed in 2–4 minutes
- ✓ Autopay scheduler — daily, weekly, or monthly automatic disbursement
- ✓ Hold / release earnings — admin can hold pending earnings before payout
- ✓ Commission deduction + tax — fixed or percentage, configurable per plan

Wallet & Financial System

- ✓ Dual wallet per member: Main (withdrawable) + Repurchase (spend-only, non-withdrawable)
- ✓ Double-entry ledger — every transaction records balance_before and balance_after
- ✓ Row-level locking on all wallet operations — zero race condition risk
- ✓ Hold balance — funds reserved against pending withdrawal requests
- ✓ Multi-currency support with real-time exchange rate table
- ✓ Bank account management — account numbers encrypted at rest

- ✓ Withdrawal workflow: pending → approved → paid with admin approval queue
- ✓ Admin charge + tax on withdrawals (fixed or percentage)
- ✓ Minimum and maximum withdrawal limits configurable per plan
- ✓ Wallet-to-wallet transfer (within own wallets)
- ✓ Admin manual credit / debit with audit trail

Member Management

- ✓ Registration with sponsor validation and referral URL tracking
- ✓ E-Pin activation system — generate, assign, and track activation PINs
- ✓ KYC identity verification — multi-document upload, admin review queue
- ✓ Rank system — automatic promotion on qualification criteria
- ✓ Network tree powered by closure table — O(1) subtree queries at any depth
- ✓ Left/right/manual placement for binary tree
- ✓ Member status lifecycle: active / inactive / blocked / suspended
- ✓ Password reset with secure token flow
- ✓ Two-factor authentication infrastructure ready (TOTP)
- ✓ Member profile with bank details, KYC documents, referral stats

Admin Panel

- ✓ Dashboard — live KPIs, 30-day earnings chart, pending queue counts
- ✓ Member list — search, filter by status/rank/plan, bulk actions
- ✓ Withdrawal approval queue with bulk approve and export
- ✓ KYC review queue with document viewer and approve/reject/reason
- ✓ Plan configuration — commission rules, packages, rank thresholds
- ✓ E-Pin management — batch generate, assign, track usage
- ✓ Support ticket system — assign, reply, close, priority
- ✓ Full settings panel — 9 typed settings groups (General, Appearance, KYC, Payout, Email, Registration, Plans, Security, Notifications)
- ✓ Plugin management — activate/deactivate product and client plugins
- ✓ Theme management — switch between product and client themes
- ✓ Performance metrics dashboard — P50/P95/P99 response times, queue depths
- ✓ Update management — check, view changelog, apply, watch live log
- ✓ Staff roles with individual permission toggles (not just role-based)
- ✓ Admin IP whitelist for panel access
- ✓ Audit log — every admin action recorded with timestamp and IP

Payment Gateways

- ✓ Stripe — credit/debit card, test and live mode, webhook verification
- ✓ PayStack — Africa-focused gateway, webhook signature verification
- ✓ PayFast — South Africa payment gateway
- ✓ Square — US market payment processing
- ✓ E-Pin — internal voucher/activation code system
- ✓ Bank Deposit — manual upload and admin verification
- ✓ GatewayFactory pattern — new gateways addable without touching core
- ✓ Idempotent payment processing — duplicate webhook protection
- ✓ Multi-gateway — enable multiple simultaneously per deployment

Plugin & Theme System (New in v1.0.0)

- ✓ PluginInterface — 13-method contract: routes, menus, income types, migrations, activate/deactivate
- ✓ AbstractPlugin — base class with no-op defaults + loadRoutes(), loadViews(), loadTranslations() helpers
- ✓ PluginManager — auto-discovers plugins/core/ and plugins/clients/, boots actives, merges menus
- ✓ Product plugins in plugins/core/ — shipped by Engineering Mode, available to all clients
- ✓ Client plugins in plugins/clients/{slug}/ — built by Client Mode, exclusive to one client
- ✓ ThemeManager — resolves active theme, supports core and client themes with priority order

✓	AbstractTheme — base class for programmatic themes
✓	ColorPaletteGenerator — generates CSS custom properties from brand settings, no build step needed
✓	Default theme in themes/core/default/ — Deep Indigo with full CSS variable set
✓	Product themes in themes/core/ — shipped by Engineering Mode to all clients
✓	Client themes in themes/clients/{slug}/ — exclusive per-client branding
✓	Plugin activations table — tracks enabled/disabled state per deployment
✓	Version compatibility check — plugins with incompatible requiresVersion() are skipped

Platform Infrastructure

- ✓ One-click update system — 13-step lifecycle, auto-rollback on any failure
- ✓ Ed25519 cryptographic signature on every update package
- ✓ Full backup before every update: database (compressed SQL) + all application files
- ✓ UpdateChecker polls updates.globalmlmsoftware.com every 6 hours with license + version headers
- ✓ PackageInstaller allow list: app/, config/, database/, resources/, routes/, plugins/core/, themes/core/
- ✓ PackageInstaller deny list: plugins/clients/, themes/clients/, .env, storage/ — never touched
- ✓ Per-customer isolated database — no shared production database ever
- ✓ Docker containerisation — PHP-FPM + Nginx + MySQL 8 + Redis 7
- ✓ Blue-green zero-downtime deployment — atomic Nginx upstream switch
- ✓ Standalone web installer with 3-layer re-entry protection
- ✓ LicenseManager with Ed25519 license key validation
- ✓ 5-step provisioning with full rollback on failure
- ✓ Horizontal scaling — 4-tier model (LB + app pool + Horizon + Redis Sentinel + read replica)

DevOps & Monitoring

- ✓ GitHub Actions CI — 6 parallel jobs (test, PHPStan, Pint, security audit, Vite build, Docker scan)
- ✓ GitHub Actions CD — staging auto-deploy + production manual approval gate
- ✓ Nightly security workflow — OWASP, CodeQL SAST, TruffleHog secret scan, CVE audit
- ✓ Weekly backup validation — restores to isolated DB, checks financial integrity
- ✓ Prometheus scraping 8 targets every 15s — app, MySQL, Redis, Nginx, node_exporter
- ✓ 20 alert rules across 4 groups (application, queues, database, infrastructure)
- ✓ Grafana dashboard — HTTP rates, P50/P95/P99, MySQL threads, Redis memory, security event log
- ✓ Loki log aggregation — 5 shipping sources, structured labels, 31-day retention
- ✓ Alertmanager — routes to Slack (#alerts, #critical), email, PagerDuty
- ✓ Structured JSON logging — 7-year retention for financial/commission logs
- ✓ Server hardening script — SSH key-only, UFW firewall, Fail2ban, sysctl, unattended-upgrades
- ✓ Disaster recovery runbook — 7 scenarios with RTO/RPO targets

Testing

- ✓ PestPHP test suite with RefreshDatabase, custom expectations (toHaveWalletBalance, toHaveEarning)
- ✓ WalletService unit tests — credit, debit, overdraft prevention, double-entry integrity, idempotency
- ✓ Commission calculator tests — Binary (carry-forward, cap), UniLevel (level rates), tree placement
- ✓ Feature tests — auth flows, registration, withdrawal, admin operations, installer re-entry protection
- ✓ Integration tests — commission pipeline end-to-end, KYC flow, settings persistence

✓	API contract tests — auth endpoints, wallet endpoints, health check
✓	Architecture tests — PHPStan Level 8, no DB calls in controllers, calculator interface compliance
✓	k6 load test — 200 VU ramp-up, 5-minute sustained, P95 < 500ms threshold
✓	k6 stress test — 1500 VU peak, finds the breaking point

3 Plugin & Theme Architecture

Four-mode framework — who builds what, and where

Directory Structure

Directory Layout

```
gml-laravel/
■■■ plugins/
■   ■■■ core/                    ← Engineering Mode ONLY
■   ■   ■■■ stripe-gateway/      ■  Product plugins available to ALL clients
■   ■   ■■■ whatsapp-notify/     ■  Shipped via signed update packages
■   ■   ■■■ advanced-reports/    ■
■   ■■■ clients/                 ← Client Mode ONLY
■       ■■■ mikie/               ■  Exclusive to one client
■       ■■■ jack-ecommerce/      ■  Never touched by Engineering/Hotfix
■       ■■■ sekal-health/        ■
■■■ themes/
■   ■■■ core/                    ← Engineering Mode ONLY
■   ■   ■■■ default/             ■  Ships with platform (Deep Indigo)
■   ■   ■■■ corporate-dark/      ■  Additional product themes
■   ■   ■■■ minimal-light/       ■
■   ■■■ clients/                 ← Client Mode ONLY
■       ■■■ mikie/               ■  Client branding
■       ■■■ jack-ecommerce/      ■
```

Update Engine Rules

Directory	Migration	Engineering	Client	Hotfix
app/ config/ database/	■ Builds	■ Extends	■ Never	■ Patches
plugins/core/ themes/core/	■ Scaffolds	■ Ships	■ Never	■ Patches
plugins/clients/{slug}/	■ Never	■ Never	■ Only here	■ Never
themes/clients/{slug}/	■ Never	■ Never	■ Only here	■ Never
.env storage/	Installer only	■ Never	■ Never	■ Never

4 Update Engine

How every client receives Engineering Mode and Hotfix updates

Update Flow

End-to-End Update Lifecycle

```
Engineering Mode builds new feature / plugin / theme
        ↓
Pushes to srinivas182/gml-laravel (master)
        ↓  GitHub Actions CI runs automatically
Builds signed .gmlm package → publishes to updates.globalmlmsoftware.com
        ↓
Every client deployment polls update server every 6 hours
        ↓
Admin panel shows: "Update Available — v1.1.0"
        ↓
Admin clicks "Apply Update" (one click)
        ↓
13-step Update Engine runs:
  Step 1  → Verify license key
  Step 2  → Compatibility check (PHP version, disk space)
  Step 3  → Download signed package
  Step 4  → Verify Ed25519 signature
  Step 5  → Backup database (compressed SQL)
  Step 6  → Backup application files (tar.gz)
  Step 7  → Enter maintenance mode
  Step 8  → Install files (allow: app/ plugins/core/ themes/core/)
  Step 9  → Run pending migrations
  Step 10 → Clear all caches
  Step 11 → Restart Horizon workers
  Step 12 → Run health checks (6 checks)
  Step 13 → Exit maintenance mode + notify admin
        ↓
If ANY step fails → automatic full rollback to pre-update state
plugins/clients/ and themes/clients/ are NEVER touched
```

✓ **Tip:** Client-specific plugins and themes (plugins/clients/, themes/clients/) survive every core update unconditionally — the deny list is hardcoded in PackageInstaller.php, not configurable.

5 System Requirements

Requirement	Minimum	Recommended
PHP	8.3+ with extensions	8.3+ (latest patch)
MySQL	8.0	8.0+
Redis	7.0+ (optional locally)	7.2+
Node.js	20 LTS (build only)	20 LTS
RAM	2 GB	8 GB
Storage	20 GB SSD	80 GB SSD
CPU	2 vCPU	4 vCPU

Required PHP extensions: pdo, pdo_mysql, bcmath, mbstring, openssl, gd, zip, sodium, redis, opcache, pcntl

6 Local Development Setup

Prerequisites

Install PHP 8.3, Composer, Node.js 20, MySQL 8, and Git. Redis is optional locally (file driver used as fallback).

Step-by-Step

Clone and Run Locally

```
# 1. Clone the repository
git clone https://github.com/srinivas182/gml-laravel.git
cd gml-laravel

# 2. Install PHP dependencies
composer install

# 3. Install and build frontend
npm install
npm run build

# 4. Create a local MySQL database
mysql -u root -p -e "CREATE DATABASE gmlm_local CHARACTER SET utf8mb4 COLLATE utf8mb4_unicode_ci;"

# 5. Visit in browser — installer launches automatically
php artisan serve
# Then open: http://127.0.0.1:8000
```

✓ **Tip:** setup.sh (Linux/macOS) and setup.bat (Windows) in the repository root automate steps 2–4 and launch the browser automatically.

Hot Reload During Development (after install)

```
# Terminal 1 — Laravel dev server
php artisan serve

# Terminal 2 — Vite hot reload
npm run dev

# Terminal 3 — Queue worker
php artisan queue:work
```

7 VPS Deployment

Ubuntu 22.04 LTS — fully automated one-command setup

Recommended Specs

Requirement	Minimum	Recommended
OS	Ubuntu 22.04 LTS	Ubuntu 22.04 LTS
CPU	2 vCPU	4 vCPU
RAM	4 GB	8 GB
Storage	40 GB SSD	80 GB SSD

Deployment

VPS Setup Commands

```
# 1. SSH into your fresh server
ssh root@YOUR_SERVER_IP

# 2. Clone the repository
git clone https://github.com/srinivas182/gml-laravel.git /var/www/gmlm
cd /var/www/gmlm

# 3. Run the automated deployment script
sudo bash deploy.sh \
  --domain    mlm.yourcompany.com \
  --company   "Your MLM Name" \
  --email     admin@yourcompany.com \
  --license   GMLM-XXXX-XXXX-XXXX \
  --currency  USD

# 4. Point your domain DNS A record to YOUR_SERVER_IP
#    SSL is auto-installed via Let's Encrypt

# 5. Open browser
# https://mlm.yourcompany.com  ← installer opens automatically
```

What deploy.sh Installs Automatically

- Docker Engine + Docker Compose
- Nginx with TLS 1.3, rate limiting, private storage protection
- MySQL 8.0 with isolated database and dedicated user
- Redis 7 with AOF persistence
- Let's Encrypt SSL with 90-day auto-renewal
- UFW firewall — only ports 22, 80, 443 open
- Fail2ban — SSH brute-force + GMLM login protection
- Supervisor — manages Horizon, Scheduler, PHP-FPM
- Logrotate — 14-day log rotation with compression

Keeping the Platform Updated

```
# From admin panel:
# Admin → Platform → Updates → Check Now → Apply Update

# Or via CLI:
php artisan gmlm:check-update
php artisan gmlm:apply-update

# Health check after update:
php artisan gmlm:health
```

8 DigitalOcean Deployment

Droplet + Managed Databases (recommended stack)

Option A — Standard Droplet

Spin up a Droplet and use the same deploy.sh as the VPS guide above.

Droplet Setup

```
# DigitalOcean Console → Create → Droplets
# Image:  Ubuntu 22.04 (LTS) x64
# Size:   4 GB RAM / 2 vCPU ($24/mo) minimum
#         8 GB RAM / 4 vCPU ($48/mo) recommended
# Region: closest to your users
# Auth:   Add your SSH key

# Once created:
ssh root@YOUR_DROPLET_IP
git clone https://github.com/srinivas182/gml-laravel.git /var/www/gmlm
cd /var/www/gmlm
sudo bash deploy.sh --domain mlm.yourdomain.com --company "MLM Co" \
     --email admin@yourdomain.com --license GMLM-XXXX
```

Option B — Managed Databases (Production-Grade)

Replace Docker MySQL and Redis with DigitalOcean managed services for automatic failover and backups.

Managed Database Setup

```
# 1. Create Managed MySQL 8 cluster
#    DO Console → Databases → MySQL → 1 GB / 1 vCPU ($15/mo)

# 2. Create Managed Redis cluster
#    DO Console → Databases → Redis → 1 GB ($15/mo)

# 3. Update .env on your Droplet with managed DB credentials
#    DB_HOST=your-mysql-cluster.db.ondigitalocean.com
#    DB_PORT=25060
#    DB_PASSWORD=<from DO console>
#    REDIS_HOST=your-redis-cluster.db.ondigitalocean.com
#    REDIS_PASSWORD=<from DO console>
#    REDIS_PORT=25061

# 4. Run deploy.sh (will skip local MySQL/Redis containers)
sudo bash deploy.sh --domain mlm.yourdomain.com --managed-db --managed-redis \
     --company "MLM Co" --email admin@yourdomain.com --license GMLM-XXXX
```

Resource	Size	Monthly Cost
Droplet (App server)	4 GB RAM / 2 vCPU	$24
Managed MySQL 8	1 GB / 1 vCPU	$15
Managed Redis	1 GB	$15
Spaces (backups + uploads)	250 GB	$5

Total		**~$59/month**

9 AWS Deployment

EC2 Instance + RDS MySQL + ElastiCache Redis

Architecture Overview

Run GMLM on an EC2 instance with AWS managed services for database and cache. This is the recommended production stack for AWS deployments.

Step-by-Step

AWS EC2 Deployment

```
# ■■ Step 1: Launch EC2 Instance ■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■
# AWS Console → EC2 → Launch Instance
# AMI:            Ubuntu Server 22.04 LTS (HVM)
# Instance type:  t3.medium (2 vCPU, 4 GB) minimum
#                 t3.large  (2 vCPU, 8 GB) recommended
# Storage:        50 GB gp3 SSD
# Security Group: Inbound — SSH(22), HTTP(80), HTTPS(443)

# ■■ Step 2: Create RDS MySQL 8 ■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■
# AWS Console → RDS → Create Database → MySQL 8.0
# Instance:    db.t3.micro (1 vCPU, 1 GB) — free tier eligible
# Storage:     20 GB gp2
# Note: Allow EC2 security group to connect to RDS on port 3306

# ■■ Step 3: Create ElastiCache Redis ■■■■■■■■■■■■■■■■■■■■■■■■■■
# AWS Console → ElastiCache → Redis → cache.t3.micro
# Note: Allow EC2 security group to connect on port 6379

# ■■ Step 4: SSH and deploy ■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■
ssh -i your-key.pem ubuntu@YOUR_EC2_PUBLIC_IP

git clone https://github.com/srinivas182/gml-laravel.git /var/www/gmlm
cd /var/www/gmlm

# Edit .env to use RDS and ElastiCache endpoints:
# DB_HOST=your-rds-endpoint.rds.amazonaws.com
# DB_PORT=3306
# REDIS_HOST=your-elasticache-endpoint.cache.amazonaws.com

sudo bash deploy.sh \
  --domain    mlm.yourdomain.com \
  --company   "Your MLM Co" \
  --email     admin@yourdomain.com \
  --license   GMLM-XXXX \
  --skip-mysql \
  --skip-redis
```

■ **Note:** Assign an Elastic IP to your EC2 instance before pointing your domain DNS — the public IP changes on every stop/start without an Elastic IP.

AWS Cost Estimate (ap-south-1 Mumbai region)

Resource	Type	Monthly Cost

EC2 Instance	t3.medium (2 vCPU, 4 GB)	~$30
RDS MySQL 8	db.t3.micro (1 GB)	~$15
ElastiCache Redis	cache.t3.micro (0.5 GB)	~$12
Elastic IP	1 address	~$4
EBS Storage	50 GB gp3	~$4
Data Transfer	100 GB/month	~$9
Total		**~$74/month**

10 Azure Deployment

Azure VM + Azure Database for MySQL + Azure Cache for Redis

Option A — Azure VM (same as VPS guide)

Azure VM Setup

```
# Azure Portal → Virtual Machines → Create
# Image:    Ubuntu Server 22.04 LTS
# Size:     Standard_B2s (2 vCPU, 4 GB RAM) — ~$35/month
#           Standard_B2ms (2 vCPU, 8 GB RAM) recommended
# Disk:     64 GB Premium SSD
# Inbound:  SSH (22), HTTP (80), HTTPS (443)

# SSH in and deploy identically to VPS:
ssh azureuser@YOUR_VM_PUBLIC_IP
git clone https://github.com/srinivas182/gml-laravel.git /var/www/gmlm
cd /var/www/gmlm
sudo bash deploy.sh --domain mlm.yourdomain.com \
     --company "MLM Co" --email admin@yourdomain.com --license GMLM-XXXX
```

Option B — Azure Managed Services

Azure Managed Services

```
# 1. Create Azure Database for MySQL Flexible Server
#    Portal → Azure Database for MySQL → Flexible Server
#    Version: 8.0  |  Tier: Burstable B1ms (~$15/month)
#    Allow Azure services to access the server

# 2. Create Azure Cache for Redis
#    Portal → Azure Cache for Redis → C0 Basic (250 MB, ~$16/month)

# 3. Update .env with Azure endpoints:
#    DB_HOST=your-server.mysql.database.azure.com
#    DB_PORT=3306
#    DB_USERNAME=gmlm_user@your-server
#    REDIS_HOST=your-cache.redis.cache.windows.net
#    REDIS_PORT=6380
#    REDIS_PASSWORD=<primary access key>

# 4. Run deploy.sh with --skip-mysql --skip-redis flags
```

Option C — Azure App Service (PaaS)

Azure App Service Deployment

```
# Requires: Azure CLI installed locally

# 1. Create resource group
az group create --name gmlm-rg --location eastus

# 2. Create App Service Plan (Linux)
az appservice plan create --name gmlm-plan --resource-group gmlm-rg \
   --sku P1v3 --is-linux

# 3. Create Web App
az webapp create --resource-group gmlm-rg --plan gmlm-plan \
   --name your-gmlm-app --runtime "PHP|8.3"

# 4. Deploy from GitHub
az webapp deployment source config --name your-gmlm-app \
   --resource-group gmlm-rg \
   --repo-url https://github.com/srinivas182/gml-laravel \
   --branch master --manual-integration

# 5. Set environment variables
az webapp config appsettings set --resource-group gmlm-rg \
   --name your-gmlm-app --settings \
   APP_ENV=production DB_CONNECTION=mysql \
   DB_HOST=your-mysql.mysql.database.azure.com \
   GMLM_LICENSE_KEY=GMLM-XXXX
```

11 Installer Walkthrough

5-step web wizard — no CLI required after git clone + setup

Step 1	**Requirements Check** Lists 14 server checks — PHP version, 9 extensions, folder permissions, vendor/ present. Continue button disabled until all pass. "Not already installed" shown as check 14 — prevents re-run.
Step 2	**Database Config** Enter MySQL host, port, database name, username, password. Tests live PDO connection. Cannot proceed if connection fails.
Step 3	**Admin Account** Full name, email, password (minimum 12 characters, confirmed). This is the permanent super-admin.
Step 4	**Company Setup** Company name, base currency (13 options), timezone, GMLM license key.
Step 5	**Installation** Click Run Installation. Terminal-style log shows: .env write → migrations (77 tables) → seeders → admin creation → company settings → config cache → APP_INSTALLED=true → lock file written. Redirects to /login on success.

■ **Note:** Re-installation is permanently blocked after Step 5 by three independent layers: (1) storage/app/.installed.lock file, (2) APP_INSTALLED=true in .env, (3) CSRF token on every form POST. If the lock file exists, install.php returns HTTP 403 immediately.

12 Admin Configuration

First steps after installer completes

Complete these in order for a fully configured deployment:

Settings → Plans
Choose MLM plan type. Configure commission rules, level percentages, rank thresholds, package prices.

Settings → Payments
Enable payment gateways. Enter Stripe/PayStack API keys — encrypted at rest. Test with sandbox keys first.

Settings → Appearance
Set brand colour (10 presets or custom hex). Individual income type badge colours. Upload logo.

Settings → Registration
Enable/disable self-registration. Require sponsor. Configure E-Pin activation. OTP verification.

Settings → KYC
Require KYC before withdrawal. Set which documents are mandatory. Configure auto-approve threshold.

Settings → Email
SMTP host, port, username, password. Use "Send Test Email" to verify before going live.

Settings → Payout
Autopay schedule (daily/weekly/monthly). Minimum withdrawal amount. Admin charge + tax.

Settings → Security
Admin IP whitelist. Session timeout. 2FA enforcement. Failed login lockout settings.

Plugins → Activate
Enable any product plugins from plugins/core/ that are relevant for this deployment.

Members → Add Member
Create the root member of the network tree. Share referral URL for growth.

13 Engineering Mode

How future features and updates reach all clients

When a new feature, payment gateway, plugin, theme, or security fix is needed, Engineering Mode is activated. It follows a strict 7-phase workflow that ensures zero breaking changes and zero impact on client-specific customisations.

Phase 1	**Source**	Pull latest from srinivas182/gml-laravel (git clone or ZIP upload)
Phase 2	**Scope**	Define exactly what to build: plugin, theme, feature, security patch, or framework upgrade
Phase 3	**Analysis**	Impact analysis, zero breaking changes audit, PHPStan pre-check, design document
Phase 4	**Build**	Implement following SOLID/DDD/100k scale — touches only app/, config/, plugins/core/, themes/core/
Phase 5	**Test**	Full PestPHP suite + PHPStan Level 8 + k6 load test + security scan + zero regression check
Phase 6	**Package**	Build signed .gmlm update package, bump version in config/gmlm.php, write CHANGELOG entry
Phase 7	**Distribute**	Push to gml-laravel master → CI/CD runs → update server notified → all client admin panels show "Update Available"

■ **Info:** Client Mode and Hotfix Mode follow similar workflows with different rules. Client Mode only ever touches plugins/clients/ and themes/clients/. Hotfix Mode targets a single surgical fix with a 2-hour SLA.

14 Support & Quick Reference

Useful Commands

Artisan Quick Reference

```
php artisan gmlm:health           # Full platform health check (6 checks)
php artisan gmlm:check-update     # Check for available updates from server
php artisan gmlm:apply-update     # Apply latest available update
php artisan gmlm:optimize         # Cache config + routes + views
php artisan gmlm:warm-cache       # Pre-warm all Redis caches
php artisan gmlm:audit-indexes    # Verify all 77-table DB indexes exist
php artisan gmlm:slow-queries     # Report slowest queries in last 24h
php artisan horizon:status        # Queue worker status
php artisan queue:failed          # List failed queue jobs
docker compose ps                 # Container status (Docker deployments)
docker compose logs -f app        # Live application log stream
```

Support Resources

Resource	URL / Contact
Repository (source code)	https://github.com/srinivas182/gml-laravel
Documentation	https://docs.globalmlmsoftware.com
License Portal	https://licenses.globalmlmsoftware.com
Support Tickets	https://support.globalmlmsoftware.com
Update Server	https://updates.globalmlmsoftware.com
Email Support	support@globalmlmsoftware.com

Migration Mode Complete. The GMLM Platform has been fully migrated from PHP 7.4 + CodeIgniter 3 to a production-ready Laravel 11 enterprise SaaS. All 176 files are live at https://github.com/srinivas182/gml-laravel (master branch). The platform is now ready for Engineering Mode — new features, product plugins, product themes, and updates that flow automatically to all client deployments.