

# LuckyPunt Development & Deployment Workflow (Updated)

## Overview for Human Readers:

- Casino App: luckypunt.net (Port 3333)
- Admin App: admin.luckypunt.net (Port 4444)
- Both deployed on VPS using Cursor SSH
- Current state: Clean baseline, live DB (luckypunt\_live), up-to-date GitHub branches
- Future dev workflow: Duplicate live DB, create dev branches, test & migrate

## VPS Deployment Summary:

- Apps deployed as luckypunt-casino (3333) and luckypunt-admin (4444)
- Redeploy via Cursor SSH:
  - Cursor connects directly to VPS
  - Pull latest from GitHub main branch
  - Restart processes (PM2/Docker/etc. as configured)

## Important Note:

Future development cycles will involve creating a dev DB from the live schema, updating .env for dev branches, testing migrations in staging, and applying to production.

# Supabase Dev Workflow Summary (Comprehensive)

## 1■■■ Project Overview

You are building:

- Casino App (App A): Customer-facing, deployed from GitHub main branch, live on VPS.
  - Admin App (App B): Separate admin interface, deployed from GitHub main branch, live on VPS.
- Both apps share a Supabase backend.

Environments:

- luckypunt\_live: Production DB
- luckypunt\_dev: Development DB (now redundant post-migration)
- luckypunt\_live\_staging: Temporary staging DB for testing migrations

## 2■■■ Goals

- Maintain an immutable, safe production DB (luckypunt\_live)
- Enable isolated development DBs for testing large changes
- Support repeatable, testable migrations
- Prevent schema drift and data loss
- Avoid cloning limitations in Supabase
- Provide context for future AI agents to reason about this system

## 3■■■ Challenges & Solutions

- Supabase blocks cloning a DB that's itself a clone → ■ Adopt schema-only migrations
- pg\_dump/pg\_restore cannot recreate Supabase system schemas (auth, storage) → ■ Preserve system schemas using pg\_dumpall
- CLI tools failed initially (IPv6-only) → ■ Enabled IPv4 for direct Postgres CLI access
- Full data clones risky post-live → ■ Use fake/anonymized data in dev DBs

## 4■■■ Current Migration-Based Workflow

### ■ Dev Phase

1. Export live schema from luckypunt\_live using pg\_dump (schema-only)
2. Create fresh Supabase project (new dev DB) and apply schema
3. Seed with fake/anonymized data for testing
4. Point dev branches of both apps to the dev DB

### ■ Commit Phase

1. Develop features, test in dev DB
2. Generate schema diff between dev DB and live DB using tools like diff or supabase db diff
3. Assemble a migration SQL file (dev\_to\_live\_migration.sql)

### ■ Test Phase

1. Restore luckypunt\_live to new project (luckypunt\_live\_staging)
2. Apply migration SQL to staging DB
3. Point dev apps to staging DB for full-stack testing
4. Verify DB changes, RLS policies, triggers, and Supabase functions work as expected

### ■ Deploy Phase

1. Apply tested migration to luckypunt\_live
2. Merge dev branches into main
3. Redeploy Casino and Admin apps on VPS pointing to luckypunt\_live
4. Clean up old dev DB

## 5■■■ Data Strategy

Environment	Data Strategy
Production (luckypunt_live)	Immutable schema, seeded with fake data, no direct writes
Development (luckypunt_dev)	Schema-only migrations, seeded with fake data
Staging (luckypunt_live_staging)	Full migration from dev, used for full-stack testing

----- -----	
Pre-Live	Optional full data copy
Post-Live	Seed dev DBs with anonymized data only

## 6■■■ Best Practices

- Always enable IPv4 on new Supabase projects for CLI tools (pg\_dump, psql)
- Never attempt to recreate Supabase system schemas manually
- Keep migration files clean and ordered (CREATE TABLE → INDEX → POLICIES → TRIGGERS)
- Test migrations in staging before applying to production
- Use feature branches for smaller changes, dev DBs for larger schema changes
- Document all DB changes for future AI agents

## 7■■■ Current State (July 2025)

- Migration tested successfully on staging DB (luckypunt\_live\_staging)
- Migration applied successfully to production DB (luckypunt\_live)
- Casino App verified working live
- Dev branches still point to luckypunt\_dev (to be cleaned up)
- Ready to merge dev → main, redeploy, and retire luckypunt\_dev

## 8■■■ Future Dev Cycle

1. Create new dev DB from live schema
2. Point dev branches to it
3. Develop & test
4. Generate and test migration
5. Apply to live DB and update apps
6. Clean up old dev DB

This workflow ensures a safe, repeatable, and AI-friendly development process for future cycles.