MP

MigrationPilot

Product Report & User Experience Walkthrough

Version 1.1.0

February 2026

Know exactly what your PostgreSQL migration will do to production — before you merge.

48 safety rules · Auto-fix · Risk scoring · GitHub Action
6 output formats · 14 framework detection · Watch mode

Author: Micke Samuel
Website: https://migrationpilot.dev
Repository: https://github.com/mickelsamuel/migrationpilot

Contents

Executive Summary

MigrationPilot is a PostgreSQL migration safety tool that analyzes DDL (Data Definition Language) statements for dangerous patterns **before** they reach production. It works as a CLI, GitHub Action, and Node.js library.

Key Statistics:

- **48 safety rules** (45 free, 3 Pro) — more than any competitor
- **6 auto-fixable rules** with `-fix` flag
- **6 output formats:** text, JSON, SARIF v2.1.0, markdown, quiet, verbose
- **14 migration frameworks** auto-detected
- **550+ tests** across 31 test files
- **8 CLI commands:** analyze, check, plan, init, detect, watch, hook, list-rules
- **3 config presets:** recommended, strict, ci
- **Risk scoring:** RED / YELLOW / GREEN (0–100)

Open-Core Business Model

Tier	What's Included	Price
Free	45 safety rules, CLI, GitHub Action, all output formats, auto-fix, PR comments, config, watch mode, hooks, framework detection	$0 forever
Pro	Everything in Free + production context queries (pg_stat_*, pg_class), 3 production rules (MP013/014/019), enhanced risk scoring	$29/mo
Enterprise	Everything in Pro + team license management, SSO/SAML, audit logs, dedicated support, custom rules	Custom

Competitive Position

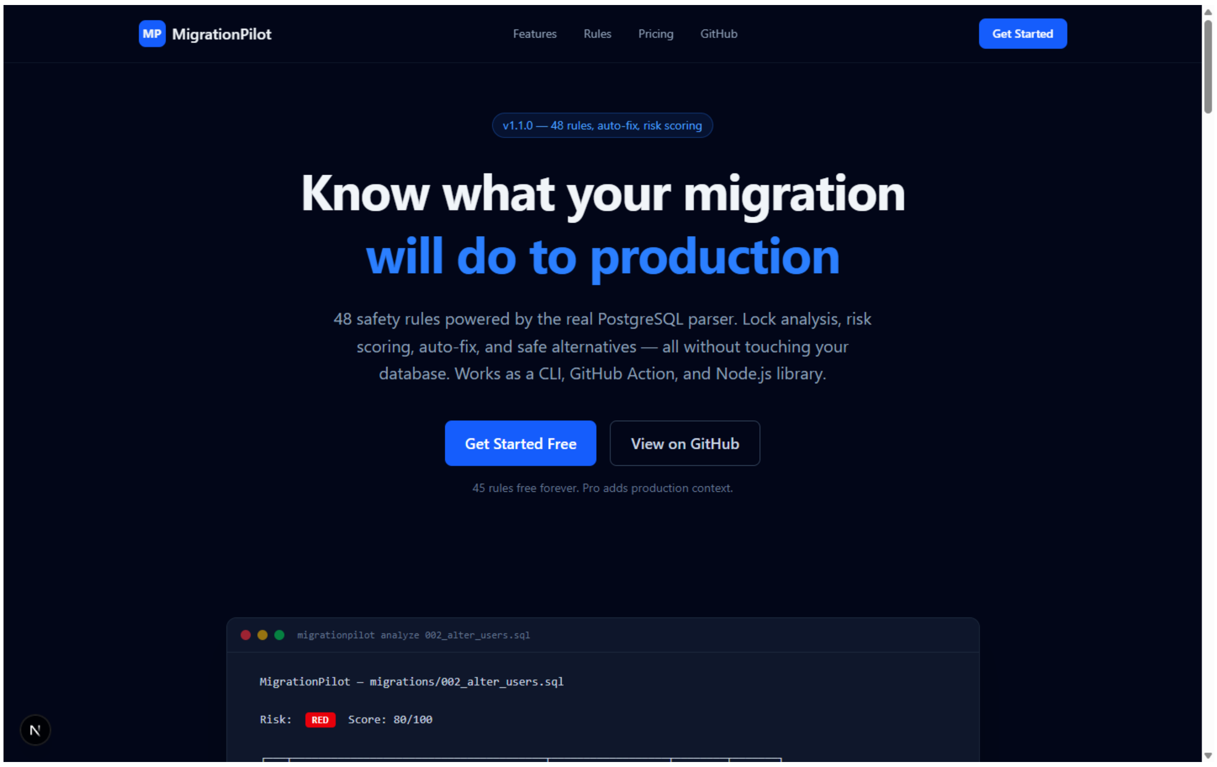
Metric	MigrationPilot	Squawk	Atlas
Total rules	**48**	31	∼15
Free rules	**45**	31	0 (paywalled)
Auto-fix	**6 rules**	0	0
Output formats	**6**	3	2
Framework detection	**14**	0	0
Watch mode	**Yes**	No	No
Config presets	**3**	0	0
Programmatic API	**Yes**	No	Yes (Go)

Website Walkthrough

The MigrationPilot landing page is a single-page Next.js 16 application with Tailwind CSS 4, featuring a dark-themed design optimized for developer audiences. It is deployed at `https://migrationpilot.dev` and statically generated for performance.

Hero Section

The hero section communicates the core value proposition immediately: "Know what your migration will do to production." The version badge (v1.1.0) and feature highlights (48 rules, auto-fix, risk scoring) establish credibility.



Know what your migration
will do to production

48 safety rules powered by the real PostgreSQL parser. Lock analysis, risk scoring, auto-fix, and safe alternatives — all without touching your database. Works as a CLI, GitHub Action, and Node.js library.

45 rules free forever. Pro adds production context.

```
migrationpilot analyze 002_alter_users.sql

MigrationPilot — migrations/002_alter_users.sql

Risk:  RED  Score: 80/100
```

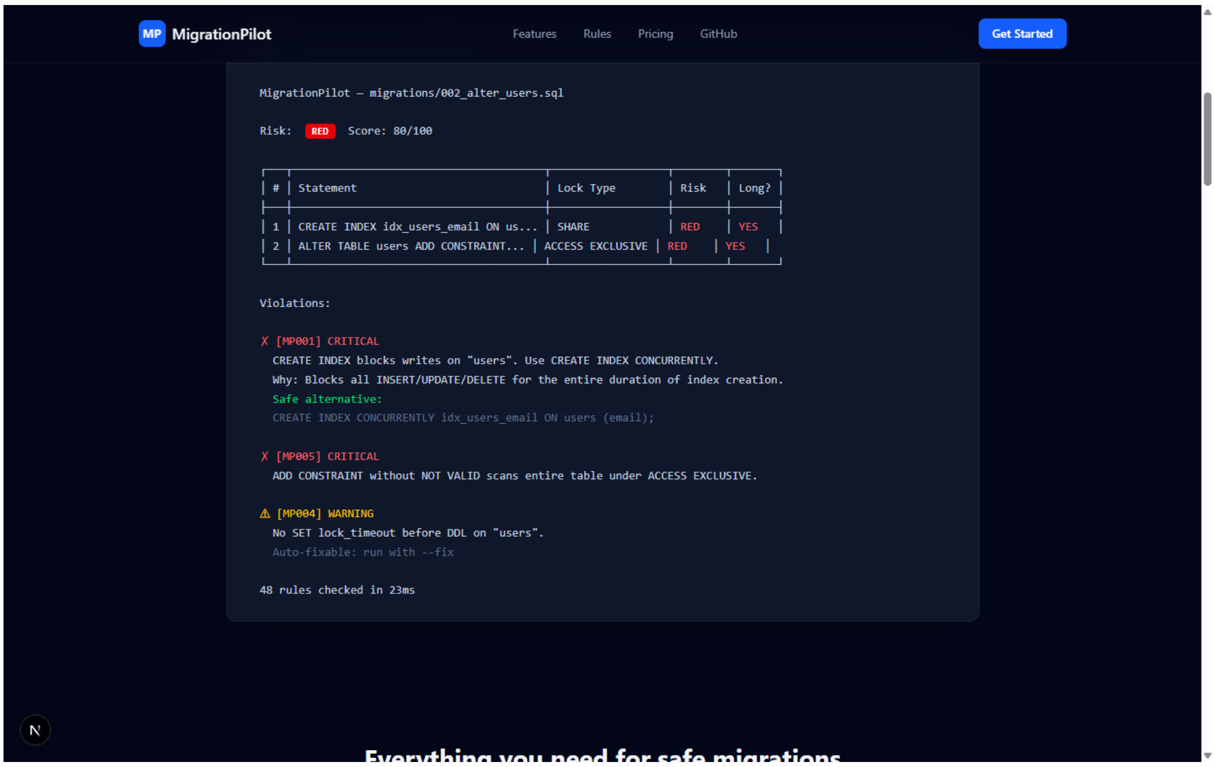
Figure 1: Landing page hero section with version badge, headline, value proposition, and CTAs.

Key elements:

- Navigation bar: Features, Rules, Pricing, GitHub, Get Started CTA
- Version badge: "v1.1.0 — 48 rules, auto-fix, risk scoring"
- Two CTAs: "Get Started Free" (links to pricing) and "View on GitHub"
- Clarifying note: "45 rules free forever. Pro adds production context."

Interactive Demo Terminal

Below the hero, an interactive terminal demo shows real CLI output — a RED-risk migration with violations, safe alternatives, and timing.

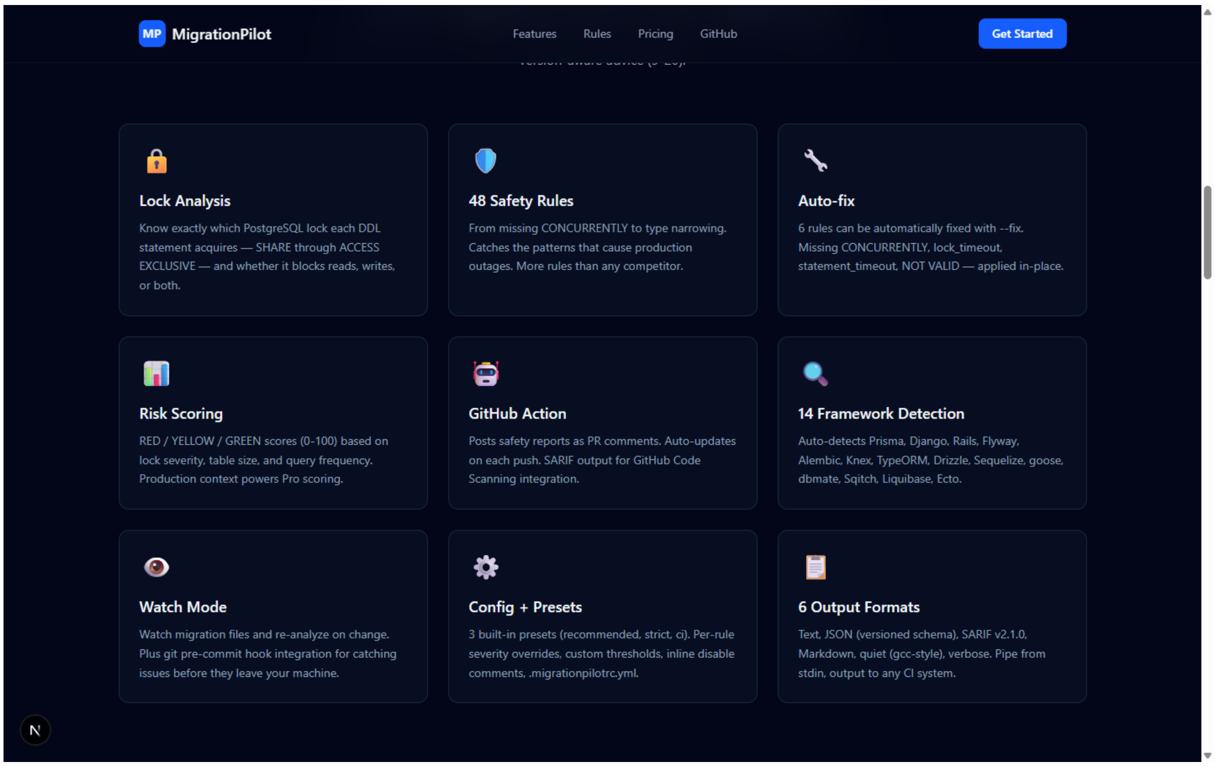


```
MigrationPilot — migrations/002_alter_users.sql

Risk:  RED  Score: 80/100

┌───┬──────────────────────────────────────┬──────────────────┬──────┬───────┐
│ # │ Statement                            │ Lock Type        │ Risk │ Long? │
├───┼──────────────────────────────────────┼──────────────────┼──────┼───────┤
│ 1 │ CREATE INDEX idx_users_email ON us... │ SHARE            │ RED  │ YES   │
│ 2 │ ALTER TABLE users ADD CONSTRAINT... │ ACCESS EXCLUSIVE │ RED  │ YES   │
└───┴──────────────────────────────────────┴──────────────────┴──────┴───────┘

Violations:

✗ [MP001] CRITICAL
  CREATE INDEX blocks writes on "users". Use CREATE INDEX CONCURRENTLY.
  Why: Blocks all INSERT/UPDATE/DELETE for the entire duration of index creation.
  Safe alternative:
  CREATE INDEX CONCURRENTLY idx_users_email ON users (email);

✗ [MP005] CRITICAL
  ADD CONSTRAINT without NOT VALID scans entire table under ACCESS EXCLUSIVE.

⚠ [MP004] WARNING
  No SET lock_timeout before DDL on "users".
  Auto-fixable: run with --fix

48 rules checked in 23ms
```

Figure 2: Simulated terminal showing analysis output with risk score, lock types, and violations.

Features Grid

Nine feature cards organized in a 3×3 grid, each with an emoji icon, title, and description:



Lock Analysis
Know exactly which PostgreSQL lock each DDL statement acquires — SHARE through ACCESS EXCLUSIVE — and whether it blocks reads, writes, or both.

48 Safety Rules
From missing CONCURRENTLY to type narrowing. Catches the patterns that cause production outages. More rules than any competitor.

Auto-fix
6 rules can be automatically fixed with --fix. Missing CONCURRENTLY, lock_timeout, statement_timeout, NOT VALID — applied in-place.

Risk Scoring
RED / YELLOW / GREEN scores (0-100) based on lock severity, table size, and query frequency. Production context powers Pro scoring.

GitHub Action
Posts safety reports as PR comments. Auto-updates on each push. SARIF output for GitHub Code Scanning integration.

14 Framework Detection
Auto-detects Prisma, Django, Rails, Flyway, Alembic, Knex, TypeORM, Drizzle, Sequelize, goose, dbmate, Sqitch, Liquibase, Ecto.

Watch Mode
Watch migration files and re-analyze on change. Plus git pre-commit hook integration for catching issues before they leave your machine.

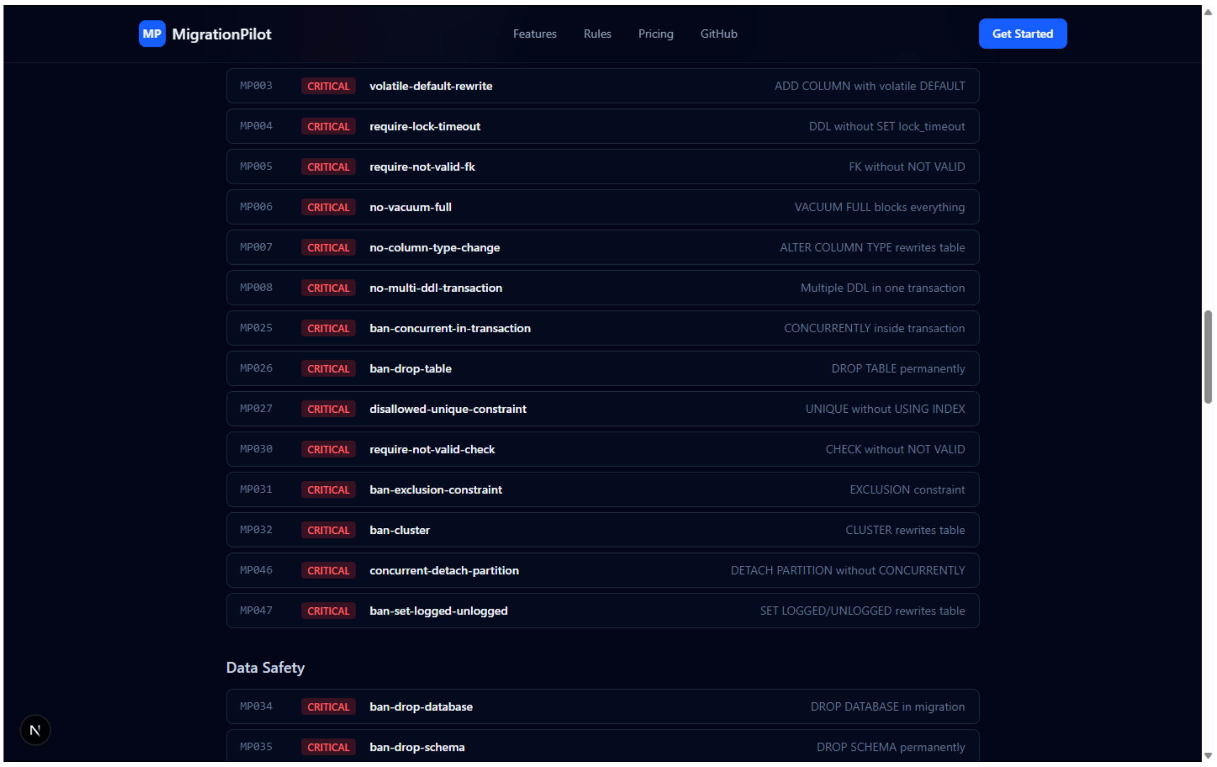
Config + Presets
3 built-in presets (recommended, strict, ci). Per-rule severity overrides, custom thresholds, inline disable comments, .migrationpilotrc.yml.

6 Output Formats
Text, JSON (versioned schema), SARIF v2.1.0, Markdown, quiet (gcc-style), verbose. Pipe from stdin, output to any CI system.

Figure 3: Feature grid: Lock Analysis, 48 Safety Rules, Auto-fix, Risk Scoring, GitHub Action, 14 Framework Detection, Watch Mode, Config + Presets, 6 Output Formats.

Complete Rules Catalog

All 48 rules displayed in four categories, each showing the rule ID, severity badge, name, and one-line description:



MP003	CRITICAL	volatile-default-rewrite	ADD COLUMN with volatile DEFAULT
MP004	CRITICAL	require-lock-timeout	DDL without SET lock_timeout
MP005	CRITICAL	require-not-valid-fk	FK without NOT VALID
MP006	CRITICAL	no-vacuum-full	VACUUM FULL blocks everything
MP007	CRITICAL	no-column-type-change	ALTER COLUMN TYPE rewrites table
MP008	CRITICAL	no-multi-ddl-transaction	Multiple DDL in one transaction
MP025	CRITICAL	ban-concurrent-in-transaction	CONCURRENTLY inside transaction
MP026	CRITICAL	ban-drop-table	DROP TABLE permanently
MP027	CRITICAL	disallowed-unique-constraint	UNIQUE without USING INDEX
MP030	CRITICAL	require-not-valid-check	CHECK without NOT VALID
MP031	CRITICAL	ban-exclusion-constraint	EXCLUSION constraint
MP032	CRITICAL	ban-cluster	CLUSTER rewrites table
MP046	CRITICAL	concurrent-detach-partition	DETACH PARTITION without CONCURRENTLY
MP047	CRITICAL	ban-set-logged-unlogged	SET LOGGED/UNLOGGED rewrites table

Data Safety

MP034	CRITICAL	ban-drop-database	DROP DATABASE in migration
MP035	CRITICAL	ban-drop-schema	DROP SCHEMA permanently

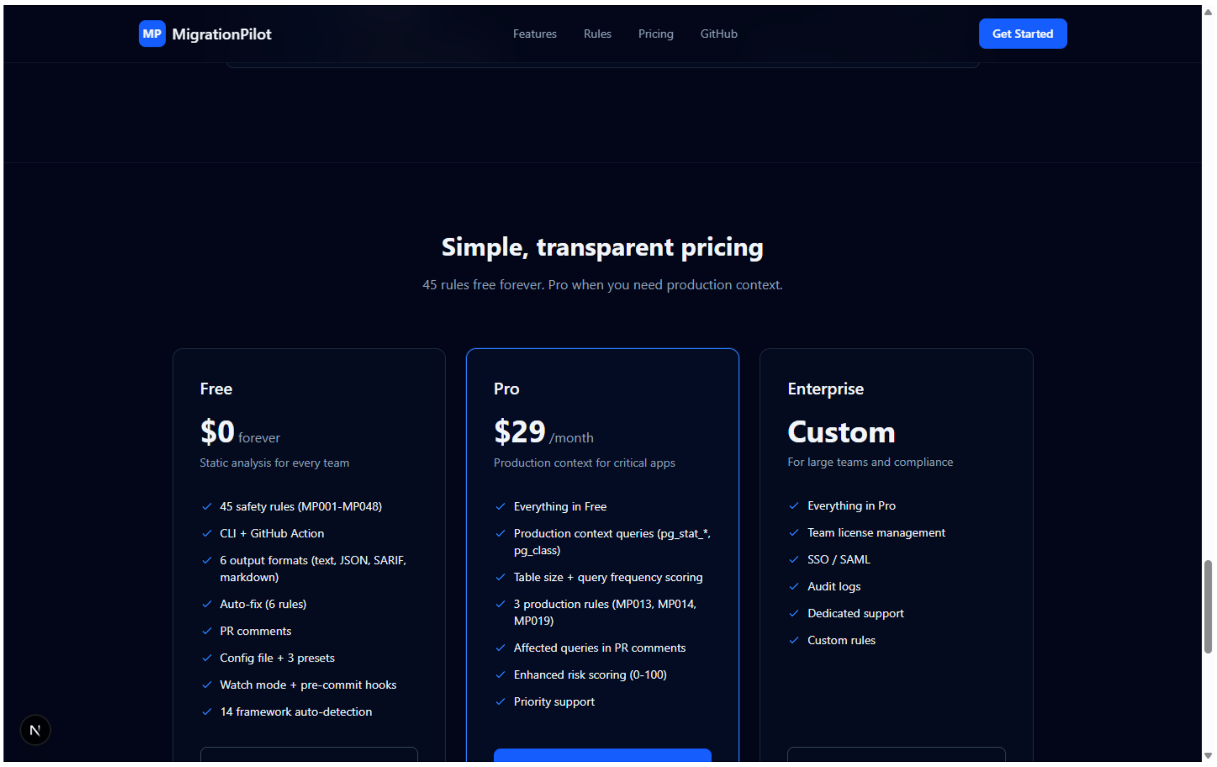
Figure 4: Rules section showing Lock Safety rules (critical severity) and Data Safety rules. All 48 rules are listed.

Rule categories:

- **Lock Safety** (16 rules) — Critical patterns that acquire dangerous locks
- **Data Safety** (3 rules) — Irreversible data destruction
- **Best Practices** (26 rules) — Schema design and migration hygiene
- **Production Context** (3 rules, Pro) — Traffic and size-aware checks

Pricing Section

Three-tier pricing with transparent feature lists:



Simple, transparent pricing

45 rules free forever. Pro when you need production context.

Free

$0 forever

Static analysis for every team

- ✓ 45 safety rules (MP001-MP048)
- ✓ CLI + GitHub Action
- ✓ 6 output formats (text, JSON, SARIF, markdown)
- ✓ Auto-fix (6 rules)
- ✓ PR comments
- ✓ Config file + 3 presets
- ✓ Watch mode + pre-commit hooks
- ✓ 14 framework auto-detection

Pro

$29 /month

Production context for critical apps

- ✓ Everything in Free
- ✓ Production context queries (pg_stat_*, pg_class)
- ✓ Table size + query frequency scoring
- ✓ 3 production rules (MP013, MP014, MP019)
- ✓ Affected queries in PR comments
- ✓ Enhanced risk scoring (0-100)
- ✓ Priority support

Enterprise

Custom

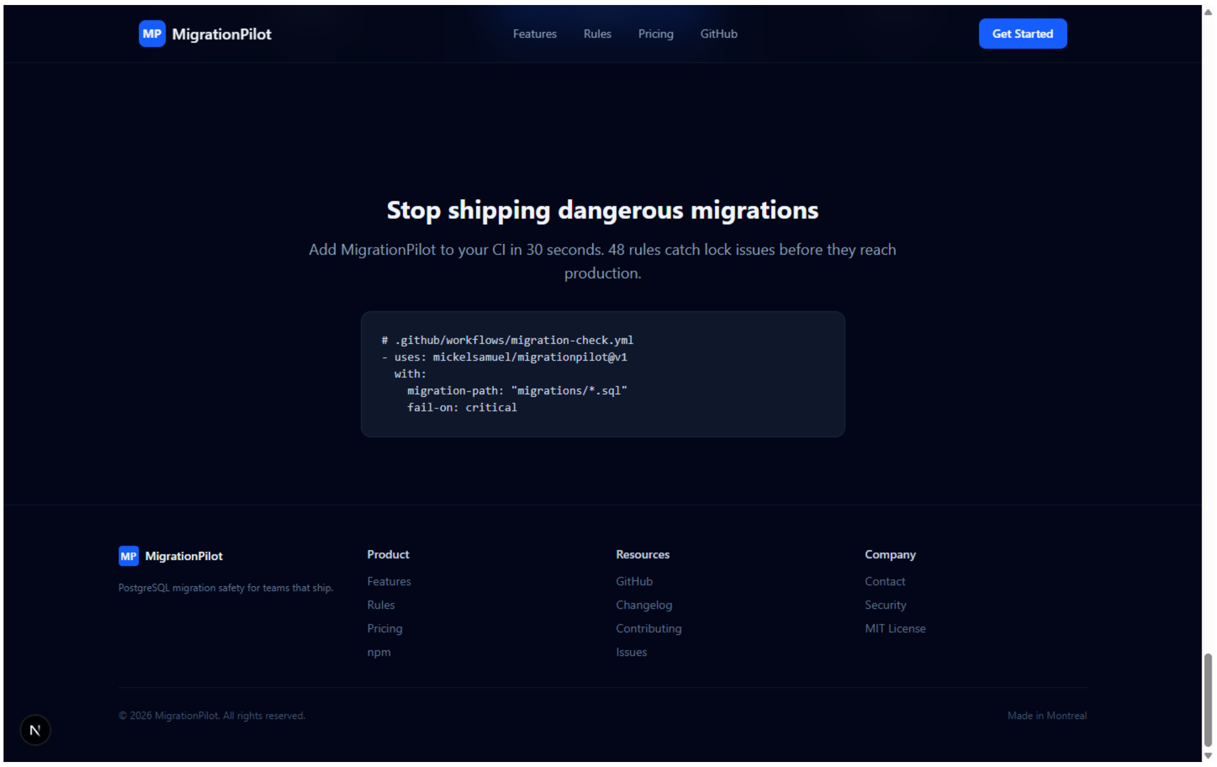
For large teams and compliance

- ✓ Everything in Pro
- ✓ Team license management
- ✓ SSO / SAML
- ✓ Audit logs
- ✓ Dedicated support
- ✓ Custom rules

Figure 5: Pricing section: Free ($0 forever), Pro ($29/month), Enterprise (custom).

CTA Section & Footer

The bottom CTA shows a GitHub Action YAML snippet for instant adoption. The footer has four columns (Product, Resources, Company) with comprehensive links.



Stop shipping dangerous migrations

Add MigrationPilot to your CI in 30 seconds. 48 rules catch lock issues before they reach production.

```
# .github/workflows/migration-check.yml
- uses: mickelsamuel/migrationpilot@v1
  with:
    migration-path: "migrations/*.sql"
    fail-on: critical
```

MigrationPilot

PostgreSQL migration safety for teams that ship.

Product	Resources	Company
Features	GitHub	Contact
Rules	Changelog	Security
Pricing	Contributing	MIT License
npm	Issues	

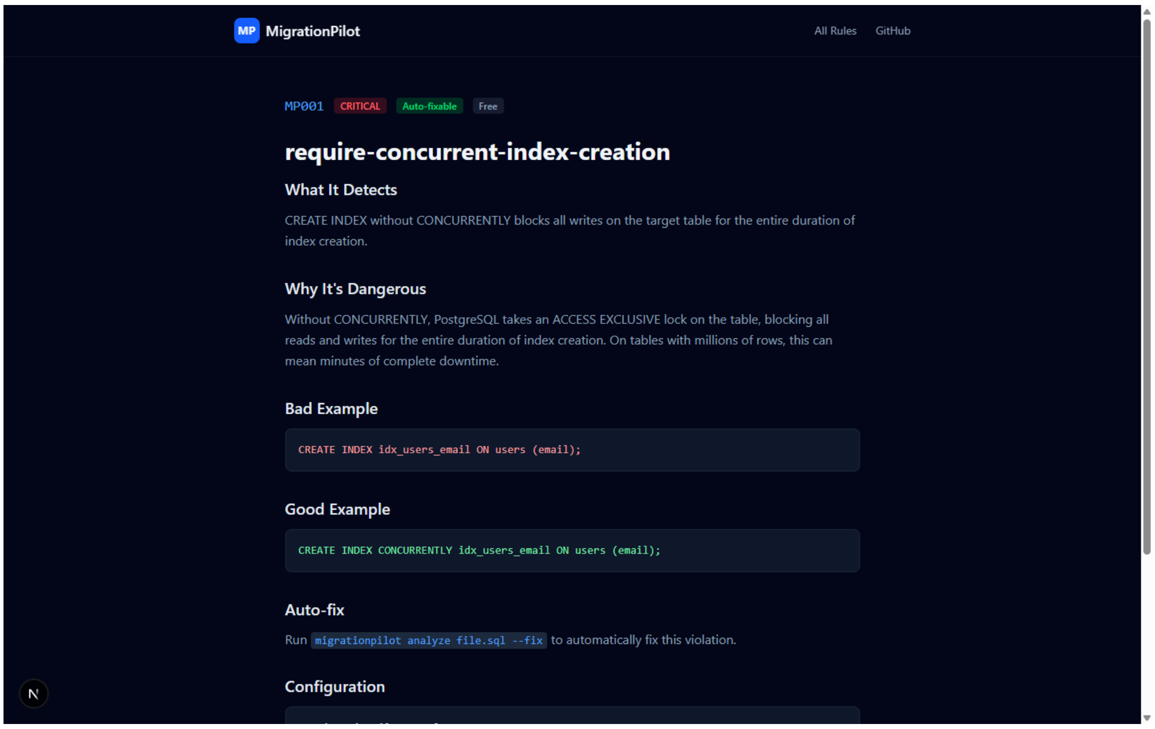
© 2026 MigrationPilot. All rights reserved.

Made in Montreal

Figure 6: Bottom CTA with GitHub Action snippet and 4-column footer with “Made in Montreal” tag.

Individual Rule Pages

Each of the 48 rules has a dedicated page at `/rules/mp{id}` with full documentation:



MP001 CRITICAL Auto-fixable Free

require-concurrent-index-creation

What It Detects

CREATE INDEX without CONCURRENTLY blocks all writes on the target table for the entire duration of index creation.

Why It's Dangerous

Without CONCURRENTLY, PostgreSQL takes an ACCESS EXCLUSIVE lock on the table, blocking all reads and writes for the entire duration of index creation. On tables with millions of rows, this can mean minutes of complete downtime.

Bad Example

```
CREATE INDEX idx_users_email ON users (email);
```

Good Example

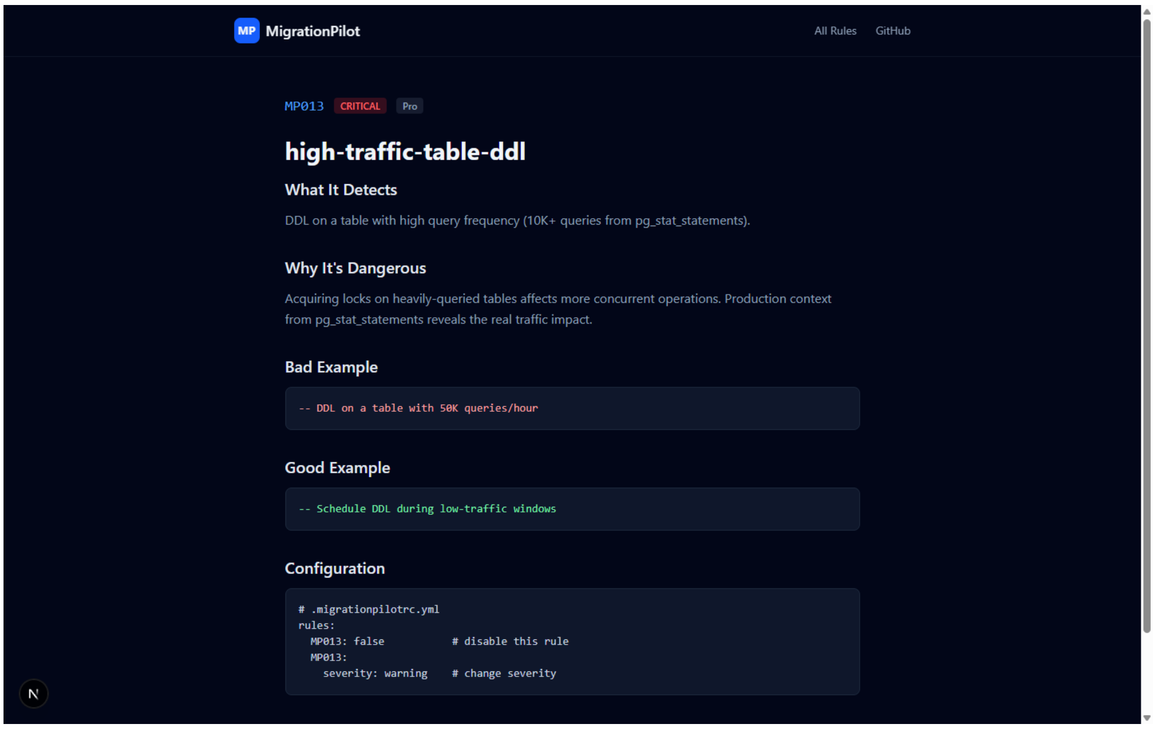
```
CREATE INDEX CONCURRENTLY idx_users_email ON users (email);
```

Auto-fix

Run `migrationpilot analyze file.sql --fix` to automatically fix this violation.

Configuration

Figure 7: Rule detail page for MP001 (require-concurrent-index-creation) showing severity badges, detection description, bad/good examples, auto-fix instructions, and configuration.



MP013 CRITICAL Pro

high-traffic-table-ddl

What It Detects

DDL on a table with high query frequency (10K+ queries from pg_stat_statements).

Why It's Dangerous

Acquiring locks on heavily-queried tables affects more concurrent operations. Production context from pg_stat_statements reveals the real traffic impact.

Bad Example

```
-- DDL on a table with 50K queries/hour
```

Good Example

```
-- Schedule DDL during low-traffic windows
```

Configuration

```
# .migrationpilotrc.yml
rules:
  MP013: false          # disable this rule
  MP013:
    severity: warning   # change severity
```

Figure 8: Pro rule page for MP013 (high-traffic-table-ddl) showing the “Pro” tier badge.

Each rule page includes:

- Severity badge (CRITICAL/WARNING), auto-fixable badge, tier badge (Free/Pro)
- “What It Detects” description
- “Why It’s Dangerous” explanation
- Bad Example (red-tinted SQL code block)
- Good Example (green-tinted SQL code block)
- Auto-fix instructions (if applicable)
- Configuration YAML snippet
- Links: back to all rules, view on GitHub

CLI Walkthrough

Installation & Version

```
$ npm install -g migrationpilot

$ migrationpilot --version
1.1.0
node v24.13.0
platform win32-x64
rules: 48 (45 free, 3 pro)
```

The enriched version output shows the Node.js version, platform, and rule breakdown.

Help & Commands

```
$ migrationpilot --help
Usage: migrationpilot [options] [command]

Know exactly what your PostgreSQL migration will do to
production -- before you merge.

Options:
  -V, --version             output the version number
  --no-color                Disable colored output
  -h, --help                display help for command

Commands:
  init                      Generate a .migrationpilotrc.yml config
  list-rules [options]      List all available safety rules
  detect [dir]              Auto-detect migration framework
  watch [options] <dir>     Watch migration files and re-analyze
  hook <action>             Install/uninstall git pre-commit hook
  plan [options] <file>     Show a visual execution plan
  analyze [options] [file]  Analyze a SQL migration file for safety
  check [options] <dir>     Check all migration files in a directory
  help [command]            display help for command
```

Core Analysis — Unsafe Migration

Given an unsafe migration file:

```sql
-- demo-unsafe.sql
CREATE INDEX idx_users_email ON users (email);
ALTER TABLE orders ADD COLUMN total numeric DEFAULT 0;
ALTER TABLE users ALTER COLUMN name TYPE varchar(50);
VACUUM FULL users;
DROP TABLE legacy_data;
```

Running `migrationpilot analyze demo-unsafe.sql` produces:

CRITICAL

Risk: YELLOW — Score: 40/100
5 statements, 9 critical violations, 4 warnings.
The analysis identifies:

- **MP001** — CREATE INDEX without CONCURRENTLY (blocks all writes)
- **MP004** — Missing lock_timeout on 5 DDL statements
- **MP006** — VACUUM FULL blocks all reads and writes
- **MP007** — ALTER COLUMN TYPE rewrites entire table
- **MP020** — Missing statement_timeout on 3 statements
- **MP023** — CREATE INDEX without IF NOT EXISTS
- **MP026** — DROP TABLE is irreversible

Each violation includes a **safe alternative** with corrected SQL, a **why** explanation, and a **docs link**.

Core Analysis — Safe Migration

A properly written migration:

```
-- demo-safe.sql
SET lock_timeout = '5s';
SET statement_timeout = '30s';
CREATE INDEX CONCURRENTLY IF NOT EXISTS
  idx_users_email ON users (email);
RESET lock_timeout;
RESET statement_timeout;
```

SAFE

Risk: GREEN — Score: 10/100
5 statements, 0 violations. “No violations found — migration is safe.”
Lock: SHARE UPDATE EXCLUSIVE (allows reads), checked in 8ms.

Visual Execution Plan

The `plan` command shows a step-by-step breakdown:

```
$ migrationpilot plan demo-unsafe.sql

  +==========================================================+
  |  MigrationPilot -- Execution Plan                        |
  +==========================================================+

  Statements: 5  Violations: 13  Risk: YELLOW

  Step 1: CREATE INDEX idx_users_email ON users (email)
    Lock: SHARE
    Impact: blocks writes
    Duration: ? unknown
    Tables: users
    X [MP001] CREATE INDEX without CONCURRENTLY
    X [MP004] Missing lock_timeout
    ! [MP020] Missing statement_timeout
    ! [MP023] Missing IF NOT EXISTS

  Step 2: ALTER TABLE orders ADD COLUMN total numeric DEFAULT 0
```

```
    Lock: ACCESS EXCLUSIVE
    Impact: blocks reads, blocks writes
    Duration: instant
    X [MP004] Missing lock_timeout

  Step 3: ALTER TABLE users ALTER COLUMN name TYPE varchar(50)
    Lock: ACCESS EXCLUSIVE
    Impact: blocks reads, blocks writes
    Duration: ? unknown
    X [MP004] Missing lock_timeout
    X [MP007] Table rewrite under ACCESS EXCLUSIVE

  Step 4: VACUUM FULL users
    Lock: ACCESS EXCLUSIVE
    Impact: blocks reads, blocks writes
    Duration: hours
    X [MP004] Missing lock_timeout
    X [MP006] VACUUM FULL blocks everything

  Step 5: DROP TABLE legacy_data
    Lock: ACCESS EXCLUSIVE
    Impact: blocks reads, blocks writes
    Duration: instant
    X [MP004] Missing lock_timeout
    X [MP026] Irreversible DROP TABLE

  9 critical, 4 warning -- migration is NOT safe to deploy
```

Auto-Fix (Dry Run)

```
$ migrationpilot analyze demo-unsafe.sql --fix --dry-run

Dry run: 9 fix(es) would be applied:

--- original
+++ fixed
- -- Unsafe migration example
+ SET lock_timeout = '5s';
- CREATE INDEX idx_users_email ON users (email);
+ SET statement_timeout = '30s';
+ -- Unsafe migration example
+ CREATE INDEX idx_users_email ON users (email);
  ...

4 violation(s) require manual fixes:
  - MP023: Missing IF NOT EXISTS
  - MP007: Table rewrite (use expand-contract pattern)
  - MP006: VACUUM FULL (use pg_repack instead)
  - MP026: DROP TABLE (rename first, drop later)
```

The auto-fixer handles 6 rules: MP001 (CONCURRENTLY), MP004 (lock_timeout), MP009 (DROP INDEX CONCURRENTLY), MP020 (statement_timeout), MP030 (NOT VALID CHECK), MP033 (REFRESH CONCURRENTLY). Violations that require architectural changes are flagged for manual attention.

Output Formats

MigrationPilot supports 6 output formats for integration with any CI/CD system:

Quiet Mode (gcc-style)

One violation per line, ideal for editor integration:

```
$ migrationpilot analyze demo-unsafe.sql --quiet
demo-unsafe.sql:1: [MP001] CRITICAL: CREATE INDEX without
  CONCURRENTLY will lock all writes on "users".
demo-unsafe.sql:1: [MP004] CRITICAL: DDL statement acquires
  SHARE lock without lock_timeout.
demo-unsafe.sql:3: [MP007] CRITICAL: ALTER COLUMN TYPE on
  "users"."name" rewrites the entire table.
demo-unsafe.sql:4: [MP006] CRITICAL: VACUUM FULL blocks
  ALL reads and writes.
demo-unsafe.sql:5: [MP026] CRITICAL: DROP TABLE "legacy_data"
  permanently removes the table.
  ...13 total violations
```

JSON Output

Machine-readable with versioned schema:

```
$ migrationpilot analyze demo-unsafe.sql --format json
{
  "$schema": "https://migrationpilot.dev/schemas/report-v1.json",
  "version": "1.1.0",
  "riskLevel": "YELLOW",
  "riskScore": 40,
  "statements": [
    {
      "sql": "CREATE INDEX idx_users_email ON users (email)",
      "lockType": "SHARE",
      "blocksReads": false,
      "blocksWrites": true,
      "riskLevel": "YELLOW",
      "riskScore": 30
    },
    ...
  ],
  "violations": [
    {
      "ruleId": "MP001",
      "ruleName": "require-concurrent-index-creation",
      "severity": "critical",
      "line": 1,
      "message": "CREATE INDEX without CONCURRENTLY...",
      "safeAlternative": "CREATE INDEX CONCURRENTLY ...",
      "docsUrl": "https://migrationpilot.dev/rules/mp001"
    },
    ...
  ]
}
```

SARIF v2.1.0

For GitHub Code Scanning integration:

```
$ migrationpilot analyze demo-unsafe.sql --format sarif
{
  "version": "2.1.0",
  "$schema": "https://raw.githubusercontent.com/oasis-tcs/
    sarif-spec/.../sarif-schema-2.1.0.json",
  "runs": [{
    "tool": {
      "driver": {
        "name": "MigrationPilot",
        "version": "1.1.0",
        "rules": [ ... 48 rule definitions ... ]
      }
    },
    "results": [ ... violations as SARIF results ... ]
  }]
}
```

Markdown Output

For PR comments and documentation:

```
$ migrationpilot analyze demo-unsafe.sql --format markdown
# Migration Safety Report
**Risk Level**: YELLOW (score: 40/100)
**Statements**: 5 | **Critical**: 9 | **Warnings**: 4

## DDL Operations
| # | Statement | Lock Type | Blocks | Risk |
|---|-----------|-----------|--------|------|
| 1 | CREATE INDEX ... | SHARE | Writes | YELLOW |
| 2 | ALTER TABLE ... | ACCESS EXCLUSIVE | R+W | YELLOW |
...

## Violations
### CRITICAL: MP001 (line 1)
CREATE INDEX without CONCURRENTLY will lock all writes...
> **Why:** Without CONCURRENTLY, PostgreSQL takes an
> ACCESS EXCLUSIVE lock on the table...
```

Framework Detection

```
$ migrationpilot detect .
No migration framework detected.
Supported frameworks: Flyway, Liquibase, Alembic, Django,
Knex, Prisma, TypeORM, Drizzle, Sequelize, goose, dbmate,
Sqitch, Rails, Ecto
```

Auto-detects 14 frameworks by examining project files (e.g., `prisma/migrations/`, `alembic.ini`, `db/migrate/`) and suggests appropriate configuration.

List Rules

```
$ migrationpilot list-rules
MigrationPilot -- 48 safety rules (45 free, 3 pro)
```

```
MP001 require-concurrent-index-creation [FREE] critical [auto-fix]
  CREATE INDEX without CONCURRENTLY blocks all writes.
MP002 require-check-not-null-pattern [FREE] critical
  SET NOT NULL requires a full table scan to validate.
...
MP048 ban-alter-default-volatile-existing [FREE] warning
  Setting a volatile default on an existing column
  has no effect on existing rows.
```

Also supports `-json` for machine consumption.

GitHub Action Integration

MigrationPilot ships as a GitHub Action that runs automatically on pull requests, posting safety analysis as PR comments.

Workflow Configuration

```
# .github/workflows/migration-check.yml
name: Migration Safety Check
on:
  pull_request:
    paths: ['migrations/**/*.sql']

jobs:
  check:
    runs-on: ubuntu-latest
    steps:
      - uses: actions/checkout@v4
      - uses: mickelsamuel/migrationpilot@v1
        with:
          migration-path: "migrations/*.sql"
          fail-on: critical
```

Action Inputs

Input	Default	Description
`migration-path`	(required)	Glob pattern for SQL migration files
`fail-on`	`critical`	Minimum severity to fail the check (`critical` or `warning`)
`database-url`	—	PostgreSQL connection string for production context (Pro)
`license-key`	—	MigrationPilot Pro license key
`pg-version`	`16`	Target PostgreSQL version for version-aware rules
`config`	—	Path to `.migrationpilotrc.yml` config file

PR Comment Output

The action posts a formatted PR comment with:

- Risk score badge (RED/YELLOW/GREEN)
- DDL operations table (statement, lock type, risk level)
- All violations with safe alternatives
- Risk factor breakdown
- Timing information

The comment is automatically updated on each push to the PR branch.

Complete Rule Catalog

Lock Safety Rules (16 critical)

ID	Sev.	Name & Description	Fix
MP001	CRIT	**require-concurrent-index** — CREATE INDEX without CONCURRENTLY	Auto
MP002	CRIT	**require-check-not-null** — SET NOT NULL without CHECK pattern	—
MP003	CRIT	**volatile-default-rewrite** — ADD COLUMN with volatile DEFAULT	—
MP004	CRIT	**require-lock-timeout** — DDL without SET lock_timeout	Auto
MP005	CRIT	**require-not-valid-fk** — FK without NOT VALID	—
MP006	CRIT	**no-vacuum-full** — VACUUM FULL blocks everything	—
MP007	CRIT	**no-column-type-change** — ALTER COLUMN TYPE rewrites table	—
MP008	CRIT	**no-multi-ddl-transaction** — Multiple DDL in one transaction	—
MP025	CRIT	**ban-concurrent-in-transaction** — CONCURRENTLY inside transaction	—
MP026	CRIT	**ban-drop-table** — DROP TABLE permanently	—
MP027	CRIT	**disallowed-unique-constraint** — UNIQUE without USING INDEX	—
MP030	CRIT	**require-not-valid-check** — CHECK without NOT VALID	Auto
MP031	CRIT	**ban-exclusion-constraint** — EXCLUSION constraint	—
MP032	CRIT	**ban-cluster** — CLUSTER rewrites table	—
MP046	CRIT	**concurrent-detach-partition** — DETACH without CONCURRENTLY	—
MP047	CRIT	**ban-set-logged-unlogged** — SET LOGGED/UNLOGGED rewrites table	—

Data Safety Rules (3 critical)

ID	Severity	Name & Description
MP034	CRITICAL	**ban-drop-database** — DROP DATABASE in migration file
MP035	CRITICAL	**ban-drop-schema** — DROP SCHEMA permanently
MP036	CRITICAL	**ban-truncate-cascade** — TRUNCATE CASCADE across tables

Best Practice Rules (26 warnings)

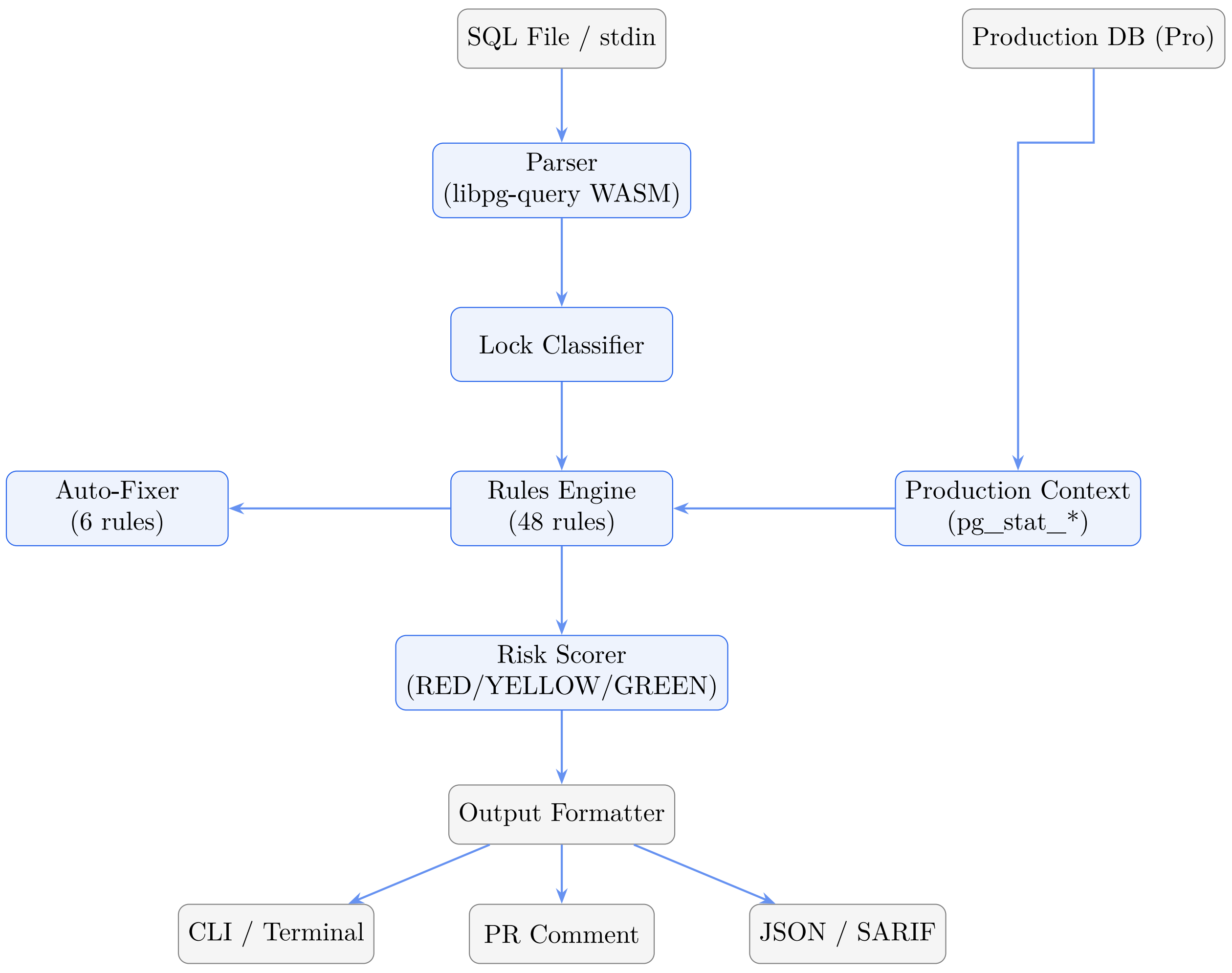
ID	Fix	Name & Description
MP009	Auto	**require-drop-index-concurrently** — DROP INDEX without CONCURRENTLY
MP010	—	**no-rename-column** — RENAME COLUMN breaks queries
MP011	—	**unbatched-backfill** — UPDATE without WHERE
MP012	—	**no-enum-add-in-transaction** — ADD VALUE inside transaction
MP015	—	**no-add-column-serial** — SERIAL, use IDENTITY
MP016	—	**require-fk-index** — FK without index
MP017	—	**no-drop-column** — DROP COLUMN risks
MP018	—	**no-force-set-not-null** — SET NOT NULL scan
MP020	Auto	**require-statement-timeout** — DDL without timeout
MP021	—	**require-concurrent-reindex** — REINDEX without CONCURRENTLY
MP022	—	**no-drop-cascade** — CASCADE drops dependents
MP023	—	**require-if-not-exists** — Non-idempotent CREATE
MP024	—	**no-enum-value-removal** — DROP TYPE destroys enum
MP028	—	**no-rename-table** — RENAME TABLE breaks refs
MP029	—	**ban-drop-not-null** — DROP NOT NULL risks
MP033	Auto	**concurrent-refresh-matview** — REFRESH without CONCURRENTLY
MP037	—	**prefer-text-over-varchar** — VARCHAR has no benefit, use TEXT
MP038	—	**prefer-bigint-over-int** — INT PK can overflow, use BIGINT
MP039	—	**prefer-identity-over-serial** — SERIAL quirks, use IDENTITY (PG 10+)
MP040	—	**prefer-timestamptz** — TIMESTAMP loses timezone info
MP041	—	**ban-char-field** — CHAR(n) wastes space
MP042	—	**require-index-name** — Unnamed index hard to reference
MP043	—	**ban-domain-constraint** — Domain constraint validates all columns
MP044	—	**no-data-loss-type-narrowing** — Narrowing column type risks data loss
MP045	—	**require-primary-key** — Table without PK hurts replication
MP048	—	**ban-alter-default-volatile** — Volatile SET DEFAULT no effect on existing rows

Production Context Rules (3 Pro)

ID	Severity	Name & Description
MP013	WARNING	**high-traffic-table-ddl** — DDL on table with 10K+ queries/hour
MP014	WARNING	**large-table-ddl** — Lock on 1M+ row table
MP019	WARNING	**exclusive-lock-connections** — ACCESS EXCLUSIVE + many connections

Architecture

System Overview



Module Breakdown

Module	Responsibility
`src/parser/`	DDL parsing with libpg-query WASM (real PostgreSQL parser compiled to WebAssembly)
`src/locks/`	Lock type classification — pure lookup table mapping AST nodes to PG lock levels
`src/rules/`	48 safety rules, engine, registry, shared helpers, inline disable comments
`src/production/`	Production context queries (Pro tier): pg_stat_*, pg_class, pg_stat_statements
`src/scoring/`	Risk scoring engine: 0–100 scale based on lock severity, table size, query frequency
`src/fixer/`	Auto-fix engine for 6 rules (MP001, MP004, MP009, MP020, MP030, MP033)
`src/output/`	6 formatters: text, JSON, SARIF v2.1.0, markdown, quiet, verbose, PR comment
`src/frameworks/`	Migration framework auto-detection (14 frameworks)
`src/config/`	Config file system + 3 built-in presets (recommended, strict, ci)
`src/analysis/`	Shared analyzeSQL pipeline, transaction boundaries, migration ordering
`src/generator/`	Safe migration SQL generation with lock_timeout retry wrappers
`src/watch/`	File watcher with debounce for `watch` command
`src/hooks/`	Git pre-commit hook installer/uninstaller
`src/license/`	License key validation (HMAC-SHA256, client-side, no telemetry)
`src/billing/`	Stripe checkout + webhook + Resend email integration
`src/cli.ts`	CLI entry point (8 commands, commander.js)
`src/action/`	GitHub Action entry point
`src/index.ts`	Programmatic API (17 exports)

Build Outputs

Bundle	Size	Format	Purpose
CLI	835 KB	CJS	`npx migrationpilot` / global install
Action	1.2 MB	CJS	GitHub Action (single-file, no node_modules)
API	219 KB	ESM	Programmatic `import {analyzeSQL} from 'migrationpilot'`

Test Coverage

- **550+ tests** across **31 test files**
- Framework: Vitest + PGlite (in-process PostgreSQL for integration tests)
- Coverage areas: all 48 rules, CLI commands, output formats, config loading, auto-fix, framework detection, risk scoring, transaction analysis, migration ordering
- All tests pass with `pnpm test`, typecheck clean, lint clean

Configuration

Config File

Generated with `migrationpilot init`:

```
# .migrationpilotrc.yml
extends: migrationpilot:recommended

failOn: critical
pgVersion: 16

rules:
  MP037: false              # disable prefer-text-over-varchar
  MP004:
    severity: warning       # downgrade lock_timeout to warning
  MP011:
    severity: critical      # upgrade unbatched-backfill

exclude:
  - "migrations/legacy/**"
```

Built-in Presets

Preset	Description
`migrationpilot:recommended`	Default severities, all rules enabled, fail on critical
`migrationpilot:strict`	All rules at critical severity, fail on warning
`migrationpilot:ci`	Optimized for CI: fail on critical, concise output

Inline Disable Comments

```sql
-- migrationpilot-disable-next-line MP001
CREATE INDEX idx_users_email ON users (email);

-- migrationpilot-disable MP004, MP020
ALTER TABLE users ADD COLUMN bio TEXT;
-- migrationpilot-enable MP004, MP020
```

PostgreSQL Lock Reference

MigrationPilot classifies every DDL statement by the PostgreSQL lock level it acquires. This table shows the lock hierarchy from least to most restrictive:

Lock Level	**Risk**	**Reads**	**Writes**	**Common Operations**
ACCESS SHARE	GREEN	OK	OK	SELECT, ANALYZE
ROW SHARE	GREEN	OK	OK	SELECT FOR UPDATE
ROW EXCLUSIVE	GREEN	OK	OK	INSERT, UPDATE, DELETE
SHARE UPD EXCL	GREEN	OK	OK	CREATE INDEX CONCURRENTLY
SHARE	YELLOW	OK	Blocked	CREATE INDEX (non-concurrent)
SHARE ROW EXCL	YELLOW	OK	Blocked	CREATE TRIGGER
EXCLUSIVE	RED	OK	Blocked	REFRESH MATVIEW CONCURRENTLY
ACCESS EXCLUSIVE	RED	Blocked	Blocked	ALTER TABLE, DROP, VACUUM FULL

ACCESS EXCLUSIVE is the most dangerous — it blocks **all** concurrent operations, including SELECT queries. MigrationPilot flags every statement that acquires this lock.

Community & Project Files

MigrationPilot includes a complete set of community and project metadata files:

File	Purpose
`README.md`	Comprehensive documentation with rules table, features, comparison, pricing
`LICENSE`	MIT License (Copyright 2026 Micke Samuel)
`CHANGELOG.md`	Version history with all changes
`CONTRIBUTING.md`	Contributor guide (setup, testing, rule development, PR process)
`CODE_OF_CONDUCT.md`	Contributor Covenant v2.1
`SECURITY.md`	Security policy and vulnerability reporting
`.editorconfig`	Editor settings (2-space indent, LF, UTF-8)
`.env.example`	Environment variable template
`.github/FUNDING.yml`	GitHub Sponsors configuration
`.github/ISSUE_TEMPLATE/`	Bug report and feature request templates
`.github/pull_request_template.md`	PR template with checklist
`.github/workflows/ci.yml`	CI pipeline (test, lint, typecheck, build)
`.github/workflows/publish.yml`	npm publish pipeline
`action.yml`	GitHub Action metadata
`docs/rules/*.md`	48 per-rule documentation files (MP001–MP048)
`site/public/robots.txt`	SEO robots configuration
`site/public/sitemap.xml`	49-URL sitemap (homepage + 48 rule pages)

SEO & Metadata

Open Graph / Social Sharing

The site generates a dynamic Open Graph image at build time using Next.js edge runtime:

- **Size:** 1200×630 pixels (standard OG)
- **Content:** MP logo, “MigrationPilot” title, tagline, 4 feature pills, domain
- **Twitter card:** summary_large_image

HTML Metadata

```
<title>MigrationPilot -- PostgreSQL Migration Safety</title>
<meta name="description" content="Know exactly what your
  PostgreSQL migration will do to production -- before you
  merge. 48 safety rules, auto-fix, risk scoring..." />
<meta name="keywords" content="postgresql, migration,
  database, safety, DDL, locks, github action, CLI,
  linter, static analysis, zero downtime" />
<link rel="canonical" href="https://migrationpilot.dev" />
<meta property="og:image"
  content="https://migrationpilot.dev/opengraph-image" />
```

npm Package Metadata

```
{
  "name": "migrationpilot",
  "version": "1.1.0",
  "description": "PostgreSQL migration safety CLI...",
  "keywords": ["postgresql", "migration", "database",
    "safety", "linter", "github-action", "ddl", "sql",
    "static-analysis", "zero-downtime", "lock", "pg"],
  "bugs": {
    "url": "https://github.com/mickelsamuel/migrationpilot/issues"
  },
  "funding": {
    "type": "github",
    "url": "https://github.com/sponsors/mickelsamuel"
  }
}
```

MP

MigrationPilot v1.1.0

Know exactly what your PostgreSQL migration will do to production.

https://migrationpilot.dev · https://github.com/mickelsamuel/migrationpilot · `npm install migrationpilot`