Rashenal Dev Environment Runbook (v1)

Purpose: Make the dev environment boring■reliable. Capture what we set up, why, how to use it, and how to recover when things go sideways. Single source of truth for local dev, Git, Supabase, CI, and release hygiene.

System Prerequisites

- OS: Windows 10/11 (macOS/Linux fine). Optional: WSL2 on Windows.
- Core: Git ≥ 2.40, Node.js 20.x, npm/pnpm, Docker Desktop, Supabase CLI.
- Editors: VS Code + ESLint, Prettier, GitHub PRs, SQLTools.
- Accounts: GitHub, Supabase project (project ref + DB password), Supabase Personal Access Token.

Repository Conventions

- Default branch: main (protected/gated).
- Working branches: wip/-.
- Commit style: feat/fix/chore/docs/wip(scope): message.
- Line endings: LF via .gitattributes; Windows .bat/.cmd allowed CRLF.
- Editor rules: .editorconfig (LF, UTF■8, trim whitespace, 2■space indent).

One■Time Local Setup (per machine)

- git clone https://github.com/rashenal/rashenal.git && cd rashenal
- git config core.autocrlf true | git config core.eol lf
- supabase login (paste Personal Access Token)
- supabase link --project-ref (stores DB creds locally; do not commit supabase/config.toml)
- Optional: git config core.safecrlf warn

Branch Protection (Gate main)

- GitHub → Repo → Settings → Branches → New rule → pattern: main.
- Require PR, require status checks, require up■to■date branch, require conversation resolution.
- After first PR runs CI, mark the check “build-and-test” as required.

Continuous Integration (CI)

- .github/workflows/ci.yml runs on PRs and pushes to main.
- Steps: Checkout → Setup Node 20 → Setup Supabase CLI → supabase start → supabase db reset → npm ci → typecheck → lint → test → build.
- See docs/ci-workflow-guide-v1.pdf for YAML ↔ plain English.

Local Dev Quickstart

- supabase start (first time pulls images)
- supabase db reset (apply all migrations from scratch)
- npm ci && npm run dev (start the app)

Migrations – Golden Workflow

- Rule: schema only changes via migrations (no ad■hoc remote edits).
- Create: supabase migration new → edit SQL under supabase/migrations/_.sql
- Idempotency: use CREATE IF NOT EXISTS; CREATE INDEX IF NOT EXISTS; INSERT ... ON CONFLICT (id) DO NOTHING; wrap in BEGIN/COMMIT.
- Test locally: supabase db reset (must be green).
- Push to remote: supabase db push.
- Remote changed first? supabase db diff --linked --schema public --file supabase/migrations/_remote.sql
- Histories disagree? supabase migration repair --status applied

Seeds: Dev vs Production

- Schema seeds (small essentials) live in migrations and are idempotent.
- Dev/demo seeds live in supabase/seed.sql and run locally only.

Env Vars & Secrets

- Local: .env.local (never commit).
- CI: GitHub → Settings → Secrets (SUPABASE_ACCESS_TOKEN, other API keys).
- Supabase: supabase link stores local DB creds; do not commit supabase/config.toml.

Docker (Optional for App)

- Keep app compose minimal; Supabase stack is managed by supabase start.
- Example service: command npm run dev, bind■mount repo, expose dev port.

Git Safety Net

- Daily: git checkout -b wip/- → small commits → push often (Draft PR).
- Crash/panic: rescue/ snapshot or stash branch.
- Restore: git stash pop or git stash branch wip/recover- stash@{0}.
- One■liner checkpoint: add script "wip:save" to package.json (git add/commit/push).

Troubleshooting

- Migrations fail: supabase db reset locally; fix newest migration.
- History mismatch: supabase migration repair --status applied .
- Password prompts: supabase login + supabase link once per machine.
- LF/CRLF: ensure .gitattributes exists; git add --renormalize .
- Ports busy: supabase stop; restart Docker; supabase start.
- CI red on migrations: reproduce locally and fix.

Maintenance

- Weekly: git pull on all machines; supabase db reset locally to sanity■check.
- Monthly: review CI runtime; rotate tokens and secrets.

Supabase CLI Automation (for Claude■Dev)

- Expose safe npm scripts so automations call npm, not raw CLIs:
- package.json scripts:
- db:new → supabase migration new
- db:reset → supabase db reset
- db:push → supabase db push
- db:diff:remote → supabase db diff --linked --schema public
- db:repair:hint → echo "Use: supabase migration repair --status applied "
- wip:save → git add -A && git commit -m "wip: checkpoint" && git push
- Then instruct Claude■Dev to use `npm run db:push` etc., avoiding secrets and keeping output standardized.