# Railway Deep Dive 2026

The solo developer's backend. Push code, get a URL. The cheat sheet for the platform we run on.

| $5-30 | 3 | ~5min | Zero |
|---|---|---|---|
| TYPICAL SOLO /MO | CORE CONCEPTS | FIRST DEPLOY | DOWNTIME DEPLOYS |

## Mental model — projects, services, environments

CONTAINER

### Project

Top-level. RDR2 Companion = one project. Groups all related services + env config.

### Service

A running piece of software. Typical: backend + maybe worker + maybe cron.

### Environment

Copy of services with different config. **production** + **staging** standard.

## Managed databases (one-click)

DEFAULT

### PostgreSQL

Relational. Use this unless you have a specific reason otherwise.

### Redis

Key-value cache. Rate limiting state. Sessions.

### MySQL

Legacy compat.

### MongoDB

Document store.

**SERVICE REFERENCES**

Cross-service variables: `${{Postgres.DATABASE_URL}}` resolves automatically. Swap the DB; your backend code doesn't change.

**WHEN TO GROW OUT**

Bill consistently > $500/mo after optimization · need niche service Railway doesn't offer · regulated compliance requirements · global multi-region · then AWS / GCP.

**TIPS YOU ONLY LEARN LATER**

1. **Pin Node version**
   In package.json engines field
2. **Use service references**
   Internal calls go over fast private network
3. **Implement /health**
   Prevents broken deploys getting traffic
4. **Set NODE_ENV=production**
   Libraries change behavior on it
5. **Match staging vars to prod**
   Biggest "works in staging" cause

**CLI ESSENTIALS**

1. **railway link**
   Link local repo to project
2. **railway run npm start**
   Local with prod env vars loaded
3. **railway up**
   Deploy directly w/o GitHub
4. **railway logs**
   Tail from terminal
5. **railway open**
   Open dashboard for project

Full guide: **djenterprises.ai/blog/railway-deep-dive**