数独乐乐软件配置与运维文档

1. 配置管理

1.1 环境配置

环境	用途	配置示例
开发环境	本地开发与调试	npm run dev
测试环境	自动化测试	Jest 测试套件
生产环境	用户访问	Netlify 全局部署

1.2 关键配置文件

文件	用途
babel.config.js	Babel 转译配置。
rollup.config.js	Rollup 打包配置。
tailwind.config.js	Tailwind CSS 定制配置。
jest.config.js	Jest 单元测试配置。

1.3 依赖管理

依赖文件

文件	用途
package.json	项目元数据、脚本命令、依赖声明（dependencies/devDependencies）。
package-lock.json	锁定依赖版本树以确保生产环境一致性。
jsconfig.json	JavaScript 项目配置（路径别名、模块解析规则等，替代 tsconfig.json）。

工具规范

- 包管理工具：npm + package-lock.json（确保依赖树精确锁定）。
- 禁止直接修改 package-lock.json（需通过 npm 命令操作）。

关键命令

命令	用途说明	适用环境
npm install	根据 package-lock.json 安装精确版本的依赖	生产环境
npm update	更新依赖到 package.json 允许的最新版本	开发环境
npm ci	清空 node_modules 后重新安装	生产环境/CI

最佳实践

1. 开发环境:
    - 使用 npm update 定期更新依赖，但需测试兼容性后再提交 package-lock.json。
2. 生产环境:
    - 始终通过 npm install 安装依赖，避免意外升级导致问题。
3. 版本控制:
    - 必须提交 package-lock.json 到 Git 仓库, 确保团队和 CI 环境依赖一致。

---

2. 版本控制

2.1 分支策略（Git）

分支	描述	保护规则
main	生产稳定版本	Require PR + CI Pass
develop	集成测试分支	Require 2 Reviews
feature	功能开发分支	合并后删除

2.2 提交规范

我们团队在开发过程中未使用命令提交版本，而是使用了 vscode 中的 git 提交版本，在提交版本过程中也遵守了相关规范。

3. 持续集成（CI）

3.1. CI 核心目标

- 自动化验证：代码提交后自动运行测试、构建和代码检查。
- 快速反馈：10 分钟内反馈构建状态，阻断问题代码进入主分支。
- 生产就绪：通过 CI 的代码方可部署到生产环境。

3.2 质量门禁

文件/目录　用途

---

coverage/　测试覆盖率报告（由 Jest 生成）。

```
============================== Coverage summary ==============================
Statements   : 61.19% ( 615/1005 )
Branches     : 53.46% ( 216/404 )
Functions    : 61.34% ( 119/194 )
Lines        : 61.3% ( 534/871 )
==============================================================================
```

- 测试覆盖率如上
- ESLint 规则: Airbnb JavaScript 规范
- 构建时间 ⩽ 3 分钟

---

4. 持续部署（CD）

4.1. CD 核心原则

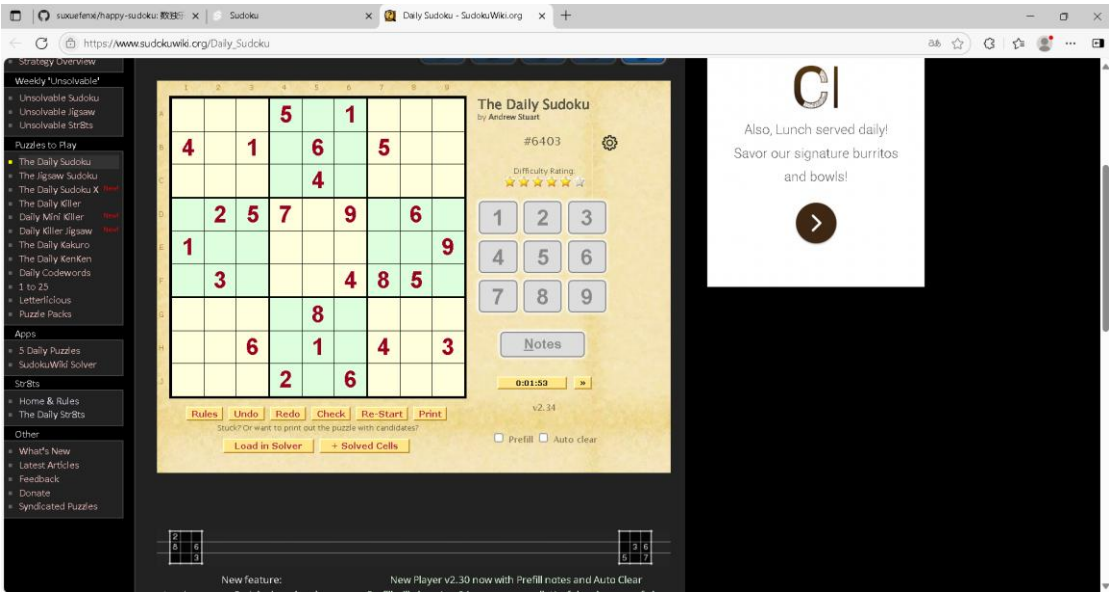
- 自动化：通过 CI 后自动部署到对应环境（测试/生产）。
- 渐进式发布：先灰度发布，验证无误后全量部署。
- 回滚机制：10 分钟内可回退到上一稳定版本。

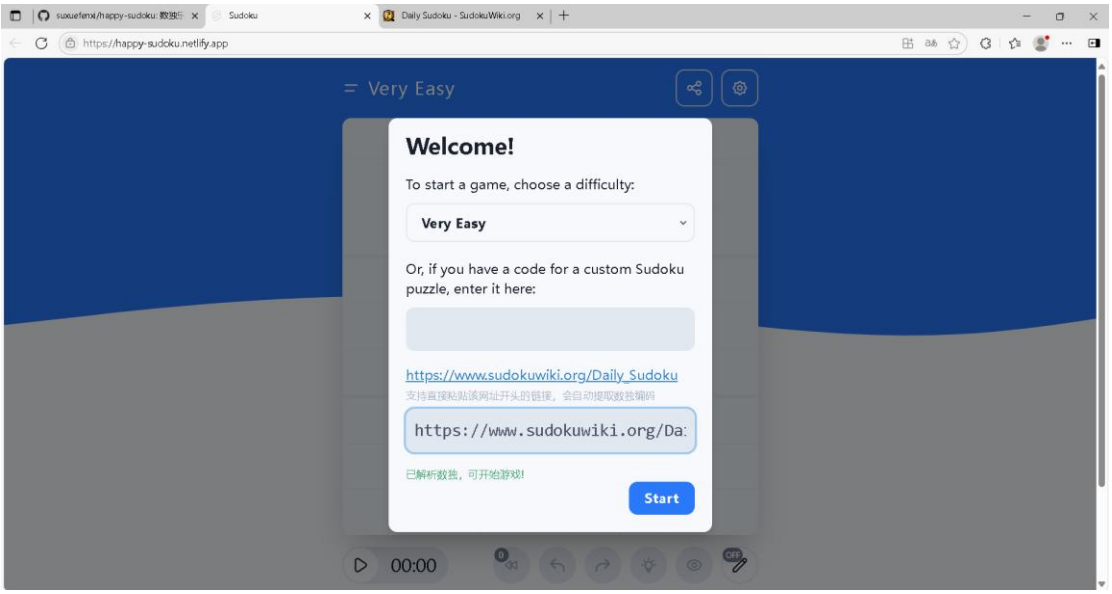
4.2 自动化部署（Netlify）

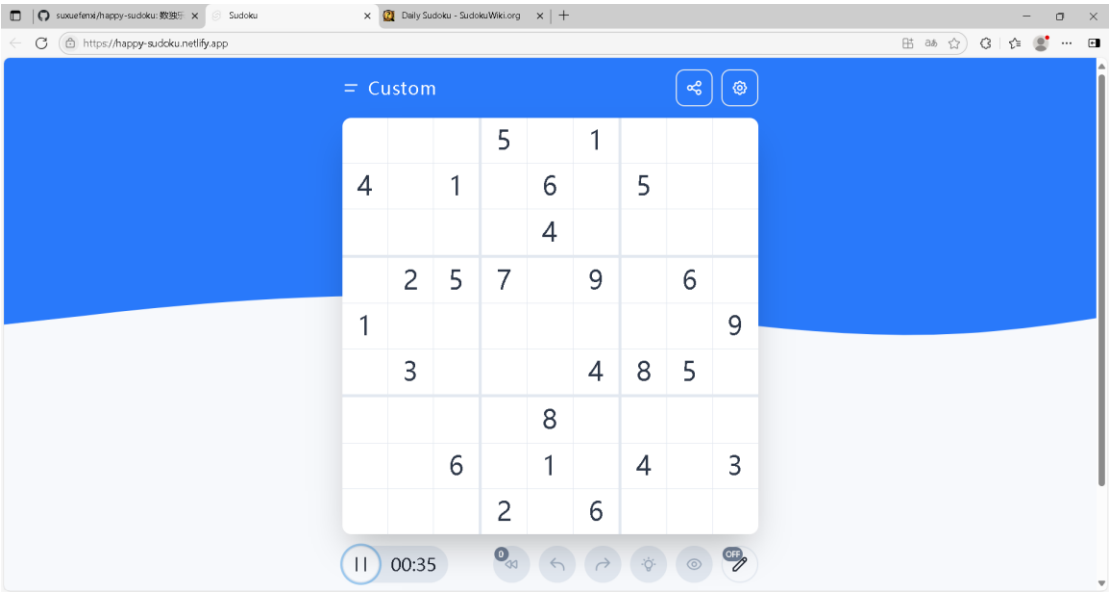
- 触发条件: main 分支合并后自动部署
- 配置要点:

- 构建命令: npm run build
- 发布目录: dist
- 环境变量: 通过 Netlify 控制台注入 SUDOKU_API_KEY

注：由于团队资金有限且该项目后端部分只是一个简单的爬虫功能，而且免费平台在部署时出现问题，因此我们并未部署该项目中的后端部分，用 netlify 部署得到的网址已经可以实现除此以外的所有功能，下图展示后端实现的复制 wiki 题目的功能：







4.3 回滚机制

1. 快速回滚: Netlify 后台一键回滚至任一历史版本
2. 紧急修复: 从 main 分支打标签触发重部署

---

5. 运维计划

5.1 运维目标

- 稳定性：保障生产环境可用性。
- 安全性：发现漏洞尽快修复。
- 可追溯：所有变更需记录工单。

5.2 监控与日志

工具	用途	配置示例
Sentry	前端错误监控	Svelte 错误边界集成

工具	用途	配置示例
Google Analytics	用户行为分析	跟踪游戏完成率/提示使用次数
Netlify Logs	访问日志/性能指标	过滤 5xx 错误请求

5.3 备份策略

- 频率: 每次变更时备份
- 执行: GitHub 自动备份
- 内容:
  - 源代码
  - 依赖清单
- 工具: 团队使用 git 进行版本控制以及变更备份

5.4 运维责任矩阵

角色	职责	工具权限
开发团队	提交变更、验证测试环境	GitHub、VS Code
运维工程师	生产发布、监控响应	Netlify、Sentry
技术负责人	审批生产变更、灾备决策	全部

注：由于团队只有 3 人，故没有特别去分配角色，而是一起负责这些任务