匠人学院
JR ACADEMY

vibe coding 实战版

不写代码，也能做出真实产品

Lovable 实战手册

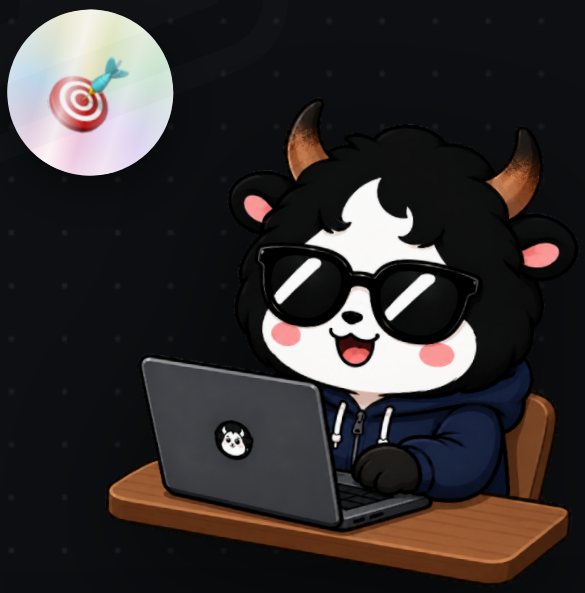
2026 年 8 月官方宣布 C 轮 $400M、估值 $13.3B，Lovable 已经是 vibe coding 赛道里绕不开的平台。从 5 分钟创建第一个应用，到 Supabase 全栈集成、Row Level Security、实时订阅，再到绑定自定义域名、接 GA4 埋点上线。7 章实战内容，含 Prompt 模板、SQL、TypeScript 代码，全程可复制。

⚡ 30 秒生成完整 React 全栈应用

🔐 Supabase Auth + RLS 数据安全

🌐 自定义域名 + GA4 埋点上线

一个下午，非技术产品经理用 Lovable 搭出了用户反馈系统



jiangren.com.au · 在线版随官网更新 /wiki/lovable-guide

AI 来匠人 ✦ Lovable 实战手册 ✦ JIANGREN.COM.AU ✦ vibe coding 时代的全栈应用

CONTENTS

目录（7 章）

📖 这本手册怎么读

这本书和官网 wiki《Lovable 实战手册》同源（jiangren.com.au/wiki/lovable-guide，免费、不用注册，官网那份持续更新）。完全没接触过 Lovable 的，第 1、2 章看清楚它能干什么、5 分钟跑通一个应用；想接 Supabase 做有真实用户账号和数据库的产品，第 4 章有完整的 Prompt 序列，第 6 章把 Auth / RLS / Storage / Realtime 四个模块拆开讲——SQL 和 TypeScript 代码直接用；想把应用推上自己的域名、接 GA4 追踪用户，第 7 章一步一步来。书里所有代码可直接复制粘贴到 Lovable 对话框。

💬 vibe coding 新人？扫码进群

群里每周分享真实的 Lovable 项目案例和 AI 应用搭建经验。

CHAPTER 01

🎯 Lovable 是什么：vibe coding 时代的应用构建革命

Lovable 是一个 AI 驱动的应用构建平台，你只需要用自然语言描述你想要什么，它就能生成完整的、可以直接部署的 Web 应用——包括前端界面、后端逻辑和数据库集成。

什么是 vibe coding

2026 年最热的技术趋势之一就是 **vibe coding**——不是传统意义上"敲代码"，而是用自然语言和 AI 对话，靠"感觉"（vibe）来构建软件。

《MIT Technology Review》将 vibe coding 列为 2026 年十大突破技术之首。这个概念的核心是：

- 你描述你想要什么（"做一个任务管理应用，支持拖拽排序"）
- AI 生成全部代码
- 你审查结果、提出调整（"把按钮改成蓝色，加上截止日期字段"）
- 反复迭代直到满意

Lovable 是 vibe coding 这个品类里增长很快的平台。官方博客在 2026 年 8 月写到：公司完成 $400M C 轮融资，估值 $13.3B；平台累计创建超过 1 亿个项目，累计用户超过 1000 万。数字会继续变，所以这本书不追融资八卦，重点讲你怎么把一个应用真的搭出来。

Lovable 的技术原理

Lovable 在底层做了三件事：

1. **代码生成**：调用顶级 AI 模型（包括 Claude）理解你的需求，生成 React + TypeScript + Tailwind CSS 代码
2. **实时预览**：在浏览器里直接运行生成的代码，你立刻看到结果
3. **持久化部署**：一键将应用部署到生产环境，生成公开可访问的 URL

生成的代码存储在 GitHub 仓库里（你的账户下），每次对话修改都会自动 commit。这意味着你完全拥有代码，随时可以 clone 下来本地开发。

Lovable 能做什么

典型用例：

- **内部工具**：HR 审批系统、销售仪表盘、库存管理
- **SaaS MVP**：在几小时内验证产品想法，而不是几个月
- **个人项目**：个人主页、博客、作品集
- **数据可视化**：连接数据库，生成图表和报告
- **表单和收集**：报名系统、调研问卷、反馈收集

一个真实案例：一位非技术背景的产品经理用 Lovable 在一个下午构建了完整的用户反馈收集系统，包括提交表单、管理后台和邮件通知，没有写过任何代码。

与王要竞品的对比

维度	Lovable	Bolt.new	v0.dev	Cursor
目标用户	非技术 + 开发者	开发者	开发者	开发者
生成内容	完整全栈应用	完整全栈应用	UI 组件	代码片段/功能
数据库集成	Supabase（原生）	需手动配置	无	无
部署	一键内置	需 Netlify/Vercel	无	无
代码所有权	GitHub 同步	下载/GitHub	复制粘贴	本地
迭代方式	对话	对话	对话	编辑器内
定价（免费额度）	5个项目/月	每日额度	有限组件数	免费版功能受限
适合场景	完整产品	原型到产品	组件库	已有代码库

核心差异：

- **vs Bolt.new**：Lovable 更强调 Supabase 原生集成，对非技术用户更友好；Bolt 在生成复杂后端逻辑上更灵活
- **vs v0.dev**：v0 专注 UI 组件生成（基于 shadcn/ui），不生成完整应用，定位不同
- **vs Cursor**：Cursor 是给已有代码库的开发者用的 AI IDE，不生成全新应用；两者可以组合使用——Lovable 生成初版，Cursor 做精细修改

谁适合用 Lovable

非常适合：

- 想验证产品想法的创业者
- 需要快速做出内部工具的运营/产品人员
- 学习编程的初学者（可以看 AI 生成的代码来学习）
- 开发者做快速原型验证（比自己从头写快 10 倍）

不太适合：

- 高度定制化的复杂企业级系统
- 需要精细性能优化的应用
- 对代码规范有严格要求的团队（AI 生成的代码质量参差不齐）

CHAPTER 02

🚀 Lovable 快速上手：5 分钟创建你的第一个 AI 应用

Lovable 的上手门槛极低——你只需要一个 Google 账号，不需要安装任何软件，打开浏览器就能开始构建应用。

第一步：注册账号

1. 打开 lovable.dev
2. 点击 **Sign up** → 选择 Google 登录（最快）
3. 完成后进入 Dashboard

免费计划包含：

- 每月 5 个项目
- 无限对话消息（在项目内）
- Lovable 托管的预览 URL
- GitHub 仓库同步（需连接 GitHub 账号）

第二步：创建第一个项目

点击 Dashboard 上的 **New Project** 按钮，进入一个空白的聊天界面。

这里有一个关键认知：**Lovable 的入口是一个对话框，不是代码编辑器**。你的第一条消息就是整个应用的"蓝图"。

写好第一条 Prompt 的技巧

差的 prompt：

```
做一个任务管理应用
```

好的 prompt：

```
创建一个任务管理应用，功能包括：
- 添加/删除/编辑任务
- 每个任务有标题、描述、截止日期、优先级（高/中/低）
- 按优先级或截止日期排序
- 标记任务为完成（带删除线效果）
- 深色模式支持
技术要求：用 React + TypeScript，界面参考 Linear 的风格，简洁现代
```

原则：越具体越好。列出你要的功能、UI 风格参考、技术偏好（如果你有的话）。

第三步：等待生成并预览

Lovable 一般在 30–60 秒内生成完整应用。生成完成后：

- **左侧**：AI 对话区域
- **右侧**：实时预览，可以直接点击交互
- **顶部**：代码视图切换按钮、部署按钮

你可以直接在预览里点击测试，也可以切换到 **Code** 视图查看生成的具体代码。

第四步：用对话迭代修改

这是 Lovable 最核心的体验。你发一条消息，它修改代码，右侧预览立刻更新。

几个实用的迭代 prompt 模式：

```
把主色调改成 #0066FF，按钮圆角改成 8px
```

```
任务列表太密集了，每个任务之间加 12px 间距，
加一个空状态图（当没有任务时显示"暂无任务"和一个图标）
```

```
添加一个搜索框，可以实时过滤任务标题
```

```
把现在的本地状态改成用 localStorage 持久化，
这样刷新页面后任务不会消失
```

每次对话都对应一个 Git commit，右上角可以看到版本历史，随时回滚。

第五步：分享预览链接

右上角 **Share** 按钮会生成一个公开的预览 URL，格式类似：

```
https://your-project-name.lovable.app
```

这个 URL 可以直接发给别人看，无需部署。如果你想绑定自己的域名，需要升级到付费计划。

连接 GitHub（推荐）

在项目设置里连接你的 GitHub 账号后，Lovable 会自动：

- 创建一个新的 GitHub 仓库
- 每次对话修改自动 commit + push
- 支持你在本地 clone 后用 Cursor/VS Code 继续开发

```
# 在 Lovable 项目设置里点击 Connect GitHub
# 然后 clone 到本地
git clone https://github.com/your-username/your-lovable-project.git
cd your-lovable-project

# 安装依赖并本地运行
npm install
npm run dev
```

一个完整的 5 分钟示例

以下是创建一个"链接收藏夹"应用的完整流程：

Prompt 1（创建应用）：

```
创建一个链接收藏夹应用：
- 可以添加链接（URL + 标题 + 描述 + 标签）
- 按标签筛选
- 搜索功能
- 网格卡片布局，每张卡片显示网站 favicon
- 数据暂时存 localStorage
```

Prompt 2（优化 UI）：

```
加一个深色/浅色模式切换按钮（右上角），
标签改成彩色徽章，不同标签不同颜色
```

Prompt 3（加功能）：

```
加一个"一键复制链接"按钮，点击后显示"已复制"提示（2秒后消失）
```

三条 prompt，大约 5 分钟，你就有了一个功能完整、界面美观的应用。

CHAPTER 03

⚙️ Lovable 核心功能详解：从对话到生产级应用

Lovable 不只是一个"帮你写代码"的工具，它构建了一套从想法到上线的完整工作流。理解它的核心功能，才能发挥它的全部潜力。

功能一：AI 代码生成引擎

Lovable 底层使用多个顶级 AI 模型（包括 Claude 系列），生成的代码技术栈固定为：

- **前端**：React 18 + TypeScript + Vite
- **样式**：Tailwind CSS + shadcn/ui 组件库
- **状态管理**：React Query（服务端状态）+ React hooks（本地状态）
- **后端**（可选）：Supabase（PostgreSQL 数据库 + Auth + Storage）

这个技术栈选择是有意为之：React + TypeScript + Tailwind 是 2025–2026 年最主流的 Web 开发组合，生成的代码质量高、可维护性强。

代码示例：当你说"添加一个带验证的登录表单"，Lovable 生成的代码大致是：

```tsx
// 使用 shadcn/ui 的 Form 组件 + react-hook-form + zod 验证
import { useForm } from "react-hook-form"
import { zodResolver } from "@hookform/resolvers/zod"
import * as z from "zod"
import { Button } from "@/components/ui/button"
import { Input } from "@/components/ui/input"

const loginSchema = z.object({
  email: z.string().email("请输入有效的邮箱地址"),
  password: z.string().min(8, "密码至少 8 位"),
})

export function LoginForm() {
  const form = useForm({
    resolver: zodResolver(loginSchema),
  })

  return (
    <Form {...form}>
      <form onSubmit={form.handleSubmit(onSubmit)} className="space-y-4">
        <FormField
          control={form.control}
          name="email"
          render={({ field }) => (
            <FormItem>
              <FormLabel>邮箱</FormLabel>
              <FormControl>
                <Input placeholder="you@example.com" {...field} />
              </FormControl>
              <FormMessage />
            </FormItem>
          )}
        />
        {/* password field... */}
        <Button type="submit" className="w-full">登录</Button>
      </form>
    </Form>
  )
}
```

这是地道的生产级 React 代码，不是玩具。

功能二：实时预览与可视化编辑

Lovable 的右侧预览不只是静态截图，它是一个真实运行的应用实例：

- 可以点击按钮、填写表单、触发交互
- 响应式预览：可以切换手机/平板/桌面视图
- **Select mode**：点击预览里的任何元素，AI 会自动定位到对应代码并告诉你可以怎么修改

这个功能让非技术用户可以用"点击"代替"描述位置"：

```
# 与其说
把右上角第三个按钮的颜色改成红色

# 不如直接点击那个按钮，然后说
把这个按钮改成红色
```

功能三：版本历史与回滚

每次对话修改都是一个 Git commit，Lovable 提供可视化的版本历史：

- 右上角时钟图标 → 打开版本历史
- 每个版本显示：时间戳、对应的 prompt 摘要、代码变更文件列表
- 一键回滚到任意历史版本

实际使用场景：

```
你改了 UI，结果越改越乱 → 回滚到 3 个版本前
你误删了一个功能 → 查看历史找到删除前的版本 → 回滚
```

这本质上就是 Git，只是 Lovable 给它套了一个对话界面。

功能四：Supabase 原生集成

这是 Lovable 最强的差异化功能。Supabase 是一个开源的 Firebase 替代品，提供：

- **PostgreSQL 数据库**：真实的关系型数据库
- **Auth**：用户认证（邮箱/密码、Google OAuth、GitHub OAuth）
- **Storage**：文件存储（图片、PDF 等）
- **Edge Functions**：服务端逻辑

在 Lovable 里连接 Supabase 只需要三步：

Step 1：在 supabase.com 免费创建项目，获取 API URL 和 anon key

Step 2：在 Lovable 项目右上角点击 Supabase 图标，粘贴连接信息

Step 3：直接在对话里描述你的数据需求：

```
帮我创建一个 todos 表，字段：
- id (uuid, primary key)
- user_id (uuid, 关联 auth.users)
- title (text, not null)
- completed (boolean, default false)
- created_at (timestamp, default now())

然后更新应用，把本地的 todo 存储改成读写这个数据库
```

Lovable 会自动生成 Supabase 的 SQL migration 和前端的数据库操作代码。

功能五：一键部署

Lovable 自带托管服务，一键部署无需任何配置：

1. 点击右上角 **Publish** 按钮
2. 等待 30 秒左右
3. 得到一个 `https://xxx.lovable.app` 的 URL

生产级部署注意事项：

- 免费计划使用 Lovable 子域名（`xxx.lovable.app`）
- Pro 计划支持绑定自定义域名
- 应用运行在全球 CDN，访问速度快
- HTTPS 自动配置

如果你想部署到自己的服务器，也可以：

```
# clone 下来后自行部署
git clone https://github.com/your-username/your-project.git
cd your-project
npm install
npm run build

# dist/ 目录里就是静态文件，可以部署到任何静态托管
# 如 Vercel、Netlify、Cloudflare Pages
```

功能六：GitHub 双向同步

连接 GitHub 后，Lovable 和本地开发形成完美互补：

```
Lovable（快速迭代 UI）
        ↕ GitHub
本地编辑器（精细代码调整）
```

工作流示例：

```
# 1. 在 Lovable 里用对话生成初版 UI
# 2. clone 到本地
git clone https://github.com/your-username/project.git

# 3. 本地做精细调整（比如复杂的业务逻辑）
# 编辑文件...
git add . && git commit -m "add complex sorting logic"
git push

# 4. 回到 Lovable，继续用对话改 UI
# Lovable 会自动 pull 你的本地更改
```

注意：本地 push 的更改会同步到 Lovable，但 Lovable 的 AI 不会理解本地代码的语义——它只能基于当前代码状态继续生成，不会"记住"你的本地更改意图。

CHAPTER 04

🔥 Lovable 进阶技巧：构建真实可用的生产级应用

用了 Lovable 一段时间后，你会发现它真正的威力不在于"生成代码"，而在于如何通过精准的对话，把一个想法变成可以给真实用户使用的产品。

进阶技巧一：写出精准 Prompt 的框架

WHAT + WHERE + HOW 框架

```
WHAT（改什么）：添加一个用户头像上传功能
WHERE（在哪里）：用户设置页面，在"个人信息"表单里
HOW（怎么做）：
  - 点击头像区域触发文件选择
  - 支持 JPG/PNG，最大 2MB
  - 上传到 Supabase Storage 的 avatars bucket
  - 上传成功后立即更新页面显示
  - 上传失败时显示错误 toast
```

一次改一件事

错误做法（一次改太多）：

```
改一下 UI，加几个功能，修复那个 bug，还有调整数据库结构
```

正确做法（分步操作）：

```
第1条：修复登录后跳转到 /dashboard 的 bug
第2条：添加"记住登录状态"复选框
第3条：登录页面 UI 改成白底深色文字风格
```

分步操作有两个好处：出了问题更容易定位，回滚也更精准。

明确说"不要改什么"

```
只修改 UserProfile 组件里的头像部分，
不要动其他任何组件，不要修改数据库结构
```

这句话能避免 AI 在"顺手"帮你优化时引入意外变更。

进阶技巧二：完整的 Supabase 用户认证

实现真实的用户注册/登录系统，需要以下步骤：

Step 1：在 Supabase 开启 Auth

在 Supabase Dashboard → Authentication → Settings：

- 开启 Email 认证
- 可选开启 Google OAuth（需要配置 Google Cloud Console）

Step 2：在 Lovable 里描述认证需求

```
帮我实现完整的用户认证：
1. 注册页面（/signup）：邮箱 + 密码 + 确认密码，注册后发验证邮件
2. 登录页面（/login）：邮箱 + 密码 + 记住我
3. 忘记密码（/forgot-password）：输入邮箱发重置链接
4. 所有需要登录的页面加上路由守卫
5. 顶部导航显示用户头像和退出按钮

用 Supabase Auth，我已经连接了 Supabase
```

Lovable 会生成完整的认证流程代码，包括 Supabase Auth hooks：

```
// Lovable 生成的认证 hook 示例
import { supabase } from "@/integrations/supabase/client"

export const useAuth = () => {
  const [user, setUser] = useState(null)
  const [loading, setLoading] = useState(true)

  useEffect(() => {
    // 检查当前登录状态
    supabase.auth.getSession().then(({ data: { session } }) => {
      setUser(session?.user ?? null)
      setLoading(false)
    })

    // 监听认证状态变化
    const { data: { subscription } } = supabase.auth.onAuthStateChange(
      (_event, session) => {
        setUser(session?.user ?? null)
      }
    )

    return () => subscription.unsubscribe()
  }, [])

  return { user, loading }
}
```

进阶技巧三：Row Level Security 数据安全

连接 Supabase 后，**必须**开启 Row Level Security (RLS)，否则所有用户都能看到所有人的数据。

在对话里说：

```
帮我为 todos 表设置 Row Level Security：
- 用户只能看到自己的 todos（user_id = auth.uid()）
- 用户只能创建属于自己的 todos
- 用户只能更新和删除自己的 todos
```

Lovable 会生成这样的 SQL：

```sql
-- 开启 RLS
ALTER TABLE todos ENABLE ROW LEVEL SECURITY;

-- 查询策略：只能看自己的
CREATE POLICY "Users can view own todos"
  ON todos FOR SELECT
  USING (auth.uid() = user_id);

-- 插入策略：只能插入属于自己的
CREATE POLICY "Users can insert own todos"
  ON todos FOR INSERT
  WITH CHECK (auth.uid() = user_id);

-- 更新策略
CREATE POLICY "Users can update own todos"
  ON todos FOR UPDATE
  USING (auth.uid() = user_id);

-- 删除策略
CREATE POLICY "Users can delete own todos"
  ON todos FOR DELETE
  USING (auth.uid() = user_id);
```

进阶技巧四：文件上传功能

```
实现图片上传功能：
- 用户可以上传图片到帖子
- 图片上传到 Supabase Storage 的 post-images bucket
- 展示上传进度条
- 上传成功后返回图片 URL，存到 posts 表的 image_url 字段
- 图片大小限制 5MB，只接受 image/* 类型
```

Lovable 生成的核心上传逻辑：

```
const uploadImage = async (file: File): Promise<string> => {
  const fileExt = file.name.split('.').pop()
  const fileName = `${Date.now()}.${fileExt}`
  const filePath = `${user.id}/${fileName}`

  const { error: uploadError } = await supabase.storage
    .from('post-images')
    .upload(filePath, file, {
      onUploadProgress: (progress) => {
        setUploadProgress(Math.round((progress.loaded / progress.total) * 100))
      }
    })

  if (uploadError) throw uploadError

  const { data } = supabase.storage
    .from('post-images')
    .getPublicUrl(filePath)

  return data.publicUrl
}
```

实战案例：构建一个简单的 SaaS 产品

以下是用 Lovable 构建一个"团队任务管理 SaaS"的完整 prompt 序列：

Prompt 1 – 初始框架：

```
构建一个团队任务管理 SaaS，技术栈用 React + Supabase：
- 工作空间概念（一个账号可以有多个工作空间）
- 项目和任务的层级结构
- 团队成员邀请
- 任务分配给特定成员
设计风格参考 Linear，深色主题
```

Prompt 2 – 数据库设计：

```
帮我设计并创建以下 Supabase 数据表：
- workspaces（工作空间）
- workspace_members（成员关系表，含权限角色）
- projects（项目）
- tasks（任务，含优先级、状态、截止日期、指派人）
每个表加上 RLS 策略，确保用户只能访问自己所在工作空间的数据
```

Prompt 3 – 邀请功能：

```
实现团队邀请功能：
- 工作空间 owner 可以生成邀请链接（有效期 7 天）
- 通过链接注册的新用户自动加入工作空间
- 邀请记录存 workspace_invitations 表
```

Prompt 4 – 看板视图：

```
添加看板视图（Kanban Board）：
- 状态列：待办、进行中、审核中、已完成
- 任务卡片可以拖拽到不同状态列
- 使用 @dnd-kit/core 库实现拖拽
```

四步之后，你有了一个功能完整的 SaaS 雏形，核心功能全部可用。

CHAPTER 05

🤔 Lovable 常见问题 FAQ：踩坑避坑指南

使用 Lovable 的过程中会遇到各种问题，这里汇总了最常见的疑问和解决方案。

关于生成质量

Q：AI 生成的代码不是我想要的，怎么办?

A：最常见的原因是 prompt 不够具体。尝试以下策略：

1. **更具体地描述**：不说"美化一下界面"，改成"把背景色改成 #F8F9FA，标题字体改成 24px 加粗，卡片加 1px #E5E7EB 边框和 8px 圆角"
2. **分解需求**：一次只改一个地方
3. **提供参考**：说"参考 Stripe Dashboard 的风格"或粘贴一个 UI 截图
4. **使用 Select 模式**：直接点击预览里的元素，让 AI 定位到具体组件

Q：AI 修改了我不想改的地方，怎么恢复?

A：立刻使用版本回滚：

1. 右上角点击时钟图标（History）
2. 找到修改前的版本
3. 点击 Restore

预防方法：在 prompt 里明确说 "**只修改 [具体组件名]，不要改其他任何文件**"

Q：生成的代码有 TypeScript 错误，但 AI 说修好了还是报错?

A：这是 Lovable 的已知问题。解决步骤：

1. 把完整的错误信息（包括文件名和行号）粘贴到对话框，让 AI 重新修复
2. 如果多次失败，切换到 Code 视图，把出错的文件内容复制出来，加上错误信息，重新描述让 AI 修
3. 实在不行，clone 到本地用 VS Code/Cursor 手动修

Q：我想用 Lovable 生成后端 API，可以吗?

A：Lovable 主要生成前端代码（React），"后端"逻辑通过 Supabase 实现：

- 数据库操作：Supabase 的 JavaScript SDK
- 服务端逻辑：Supabase Edge Functions（基于 Deno）
- 文件存储：Supabase Storage

如果你需要独立的后端 API（Node.js/Python/Go），Lovable 不是最佳选择，建议用 Cursor 或直接手写。

关于 Supabase 集成

Q：连接 Supabase 后，数据库操作报 403 权限错误?

A：99% 是 Row Level Security (RLS) 配置问题：

1. 检查表是否开启了 RLS：Supabase Dashboard → Table Editor → 对应表 → 查看 RLS 状态
2. 如果开启了 RLS，确保有对应的 Policy
3. 快速调试方法：临时在 Supabase SQL Editor 运行：

```
-- 查看某个表的 Policy
SELECT * FROM pg_policies WHERE tablename = 'your_table_name';
```

Q：Supabase 免费计划有什么限制?

A：Supabase 免费计划（2026 年）：

限制	额度
数据库容量	500 MB
文件存储	1 GB
月活跃用户	50,000
Edge Functions 调用	500,000 次/月
暂停条件	连续 7 天无活动自动暂停

对于早期验证项目完全够用，正式上线后建议升级到 Pro（$25/月）。

Q：怎么在 Lovable 里调用外部 API（比如天气 API、支付 API）？

A：直接描述：

```
集成 OpenWeatherMap API：
- API key：[你的key，注意：不要把真实 key 写在对话里，
  让 AI 用环境变量，然后你在 Supabase 或部署设置里配置]
- 添加一个天气卡片组件，根据用户地理位置显示当前天气
```

涉及敏感 API key，正确做法是：

1. 让 AI 生成使用 `import.meta.env.VITE_API_KEY` 读取环境变量的代码
2. 把 key 配置在 Lovable 项目设置的 Environment Variables 里

关于定价和限制

Q：Lovable 免费计划够用吗?

A：免费计划适合：学习、个人项目、验证想法。限制是每月 5 个项目（注意：是项目数量，不是消息数量）。

付费计划（Pro $25/月）增加：

- 无限项目
- 自定义域名
- 更高的 AI 生成速度
- 优先客服

Q：Lovable 生成的代码，我拿去商用有版权问题吗?

A：根据 Lovable 的服务条款，用户完全拥有生成代码的所有权，可以商用、分发、修改。Lovable 不对生成内容主张任何版权。

关于工作流最佳实践

Q：Lovable 和 Cursor 怎么配合使用?

A：推荐工作流：

```
1. 在 Lovable 快速生成初版（1–2小时）
2. 连接 GitHub，clone 到本地
3. 用 Cursor 做精细修改：
   – 复杂业务逻辑
   – 性能优化
   – 集成测试
4. push 后回 Lovable 继续迭代 UI
```

两个工具各有优势：Lovable 快（新功能对话即可），Cursor 精（精细代码控制）。

Q：Lovable 适合团队协作吗?

A：目前（2026年）Lovable 的协作功能相对基础：

- 可以把项目分享给团队成员查看
- 多人同时编辑同一项目可能导致冲突
- 推荐做法：一个 Lovable 项目对应一个 GitHub repo，团队通过 GitHub 协作，Lovable 作为"快速修改 UI"的工具

Q：我的项目越来越大，AI 开始出现奇怪的修改，怎么办?

A：这是 Lovable（以及所有 AI 编辑器）的通病：上下文窗口限制导致大项目理解能力下降。解决方案：

1. **保持项目规模合理**：一个 Lovable 项目不要超过 50 个文件
2. **使用精确指令**：指定具体文件名和组件名，不让 AI 猜
3. **定期整理代码**：在本地用 Cursor 重构，保持代码清洁后再回 Lovable
4. **拆分项目**：把复杂系统拆成多个独立的小项目

CHAPTER 06

🗄️ Lovable + Supabase 完整集成：auth / database / storage / realtime

Lovable 自带前端生成，但真实产品离不开后端：用户账号、持久化数据、文件存储、多端同步。Supabase 是目前和 Lovable 配合最顺的后端选择——两者都走"告诉 AI 你想要什么"的思路，官方也做了原生连接器。

这一章把四个核心模块拆开讲，每个模块给出真实可用的代码和配置，不绕弯子。

第一步：连接 Supabase 项目

在 Supabase 创建项目，拿到凭证

登录 supabase.com，创建一个新项目（Organization → New project）。创建完成后进入：

Settings → API

你需要两个值：

- **Project URL**：格式是 `https://xxxxxxxxxxx.supabase.co`
- **anon key**：公开匿名密钥，以 `eyJ...` 开头，很长

> 只复制 `anon`（public）key，绝对不要把 `service_role` key 粘到 Lovable。`service_role` 可以绕过所有 RLS 权限检查，泄露等于数据库裸奔。

在 Lovable 项目里接入

打开你的 Lovable 项目，找到右上角设置图标 → **Settings → Connectors → Supabase**，粘贴 Project URL 和 anon key，点 Save。出现绿色 "Connected" 状态即为成功。

连接后，Lovable 的 AI 会读取你的 Supabase 表结构和 RLS 规则——之后你说"帮我加一个 todos 表，每个用户只能看自己的数据"，AI 会直接生成 SQL migration 和前端查询代码，不用你手动来回切换。

模块一：用户认证（Auth）

开启邮箱登录

Supabase 默认开启了邮箱/密码认证。开发期间建议先关掉邮件确认，不然每次测试都要去邮箱点链接：

Supabase Dashboard → Authentication → Providers → Email → 关闭 "Confirm email"

然后告诉 Lovable：

```
添加用户登录/注册功能：
- 邮箱 + 密码登录
- 注册后自动跳转 /dashboard
- 登录状态持久化（刷新页面不掉登录）
- 右上角加退出按钮
```

Lovable 会生成完整的认证逻辑，包括 `supabase.auth.signUp()`、`signInWithPassword()`、`signOut()` 和 session 监听。

加 OAuth 登录（Google / GitHub）

在 Supabase Dashboard 里先启用 OAuth Provider（Authentication → Providers → 选对应平台），填入该平台的 Client ID 和 Secret。

然后在 Lovable 里：

```
登录页加 "使用 Google 登录" 按钮，
调用 supabase.auth.signInWithOAuth({ provider: 'google' }),
回调地址配置到 /auth/callback 路由
```

读取当前用户

Supabase SDK 的 session 通过 `supabase.auth.getSession()` 获取，或者监听变化：

```
// 在 React 应用顶层监听 auth 状态
useEffect(() => {
  const { data: { subscription } } = supabase.auth.onAuthStateChange(
    (event, session) => {
      setUser(session?.user ?? null)
    }
  )
  return () => subscription.unsubscribe()
}, [])
```

模块二：数据库 + Row Level Security

建表和基本查询

Lovable 可以直接帮你生成 SQL migration。告诉它你想要的表结构，它会在 Supabase SQL Editor 里执行：

```
-- Lovable 生成的 todos 表示例
CREATE TABLE todos (
  id UUID DEFAULT gen_random_uuid() PRIMARY KEY,
  user_id UUID REFERENCES auth.users(id) ON DELETE CASCADE,
  title TEXT NOT NULL,
  completed BOOLEAN DEFAULT FALSE,
  created_at TIMESTAMPTZ DEFAULT NOW()
);
```

前端查询同样由 Lovable 生成，但了解基本写法很有必要：

```
// 查询当前用户的所有 todos（RLS 会自动过滤）
const { data, error } = await supabase
  .from('todos')
  .select('*')
  .order('created_at', { ascending: false })

// 插入新 todo
const { error } = await supabase
  .from('todos')
  .insert({ title: '买咖啡', user_id: user.id })

// 更新
const { error } = await supabase
  .from('todos')
  .update({ completed: true })
  .eq('id', todoId)
```

Row Level Security（RLS）是什么，为什么必须开

默认情况下，Supabase 数据库对所有请求都开放——只要有 anon key，任何人都能读写你的表。这在开发时方便，但上线前必须关掉这扇门。

RLS（Row Level Security）是 PostgreSQL 的原生特性，作用是给每张表加一道"行级过滤器"：每条 SQL 执行时，Postgres 都会检查当前用户是否满足 Policy 定义的条件，不满足的行直接不可见，不是返回空而是根本不存在。

开启方式：

```sql
-- 开启 RLS（必须）
ALTER TABLE todos ENABLE ROW LEVEL SECURITY;

-- 允许用户查看自己的 todos
CREATE POLICY "用户只能查看自己的 todos"
ON todos FOR SELECT
TO authenticated
USING (auth.uid() = user_id);

-- 允许用户插入自己的数据（WITH CHECK 确保 user_id 只能是自己）
CREATE POLICY "用户只能插入自己的 todos"
ON todos FOR INSERT
TO authenticated
WITH CHECK (auth.uid() = user_id);

-- 允许用户修改自己的 todos
CREATE POLICY "用户只能更新自己的 todos"
ON todos FOR UPDATE
TO authenticated
USING (auth.uid() = user_id)
WITH CHECK (auth.uid() = user_id);

-- 允许用户删除自己的 todos
CREATE POLICY "用户只能删除自己的 todos"
ON todos FOR DELETE
TO authenticated
USING (auth.uid() = user_id);
```

`auth.uid()` 是 Supabase 提供的函数，返回当前请求的用户 ID。未登录时返回 `NULL`，`NULL = user_id` 永远是 false，所以匿名请求拿不到任何数据。

Policy 类型速查

操作	USING	WITH CHECK
SELECT	✅ 必须	❌ 不用
INSERT	❌ 不用	✅ 必须
UPDATE	✅ 建议	✅ 建议
DELETE	✅ 必须	❌ 不用

`USING` 是"过滤已有行的条件"，`WITH CHECK` 是"验证写入新数据的条件"。

常见 403 报错排查

连接 Supabase 后出现 `403 Forbidden` 或 `new row violates row-level security policy`，基本是这几个原因：

1. **表开了 RLS 但没有 Policy** → 开启 RLS 等于给表加了一把没有钥匙的锁，必须显式创建 Policy 才能让用户进来
2. **Policy 里的字段名写错** → 比如 `author_id` 写成了 `user_id`，检查列名
3. **前端没传 auth token** → 确保 Supabase client 初始化时设置了 `auth.persistSession: true`，以及请求时 session 还有效

在 Supabase SQL Editor 里快速诊断：

```
-- 查看某张表的所有 Policy
SELECT policyname, cmd, qual, with_check
FROM pg_policies
WHERE tablename = 'todos';
```

模块三：文件存储（Storage）

创建 Bucket

在 Supabase Dashboard → Storage → Create a new bucket。

两种类型：

- **Public bucket**：文件可直接通过 URL 访问，适合公开图片（产品图、封面图）
- **Private bucket**：需要生成签名 URL 才能访问，适合用户私有文件（合同、隐私照片）

头像上传一般用 public bucket（`avatars`），用户上传的文档用 private bucket。

上传文件

```
// 上传用户头像
async function uploadAvatar(file: File, userId: string) {
  const fileExt = file.name.split('.').pop()
  const filePath = `${userId}/avatar.${fileExt}`

  const { data, error } = await supabase.storage
    .from('avatars')
    .upload(filePath, file, {
      upsert: true,          // 覆盖已有文件
      contentType: file.type
    })

  if (error) throw error

  // 获取公开 URL
  const { data: { publicUrl } } = supabase.storage
    .from('avatars')
    .getPublicUrl(filePath)

  return publicUrl
}
```

私有文件的临时访问

Private bucket 里的文件需要生成限时签名 URL：

```
// 生成 60 秒有效的临时访问链接
const { data, error } = await supabase.storage
  .from('private-docs')
  .createSignedUrl('contracts/agreement.pdf', 60)

if (data) {
  window.open(data.signedUrl)
}
```

Storage 的 RLS

Storage 同样支持 RLS Policy，通过 SQL 配置。例如，只允许用户访问自己的文件夹：

```
-- 允许已登录用户上传到自己的文件夹
CREATE POLICY "用户可以上传到自己的文件夹"
ON storage.objects FOR INSERT
TO authenticated
WITH CHECK (bucket_id = 'avatars' AND (storage.foldername(name))[1] = auth.uid()::text);

-- 允许公开读取 avatars bucket
CREATE POLICY "avatars 公开可读"
ON storage.objects FOR SELECT
USING (bucket_id = 'avatars');
```

告诉 Lovable "头像上传到 Supabase Storage 的 avatars bucket，文件路径用 userId/avatar.png，上传后更新 profiles 表的 avatar_url 字段"，它会生成完整的组件代码，包括文件选择、上传进度、错误处理。

模块四：实时订阅（Realtime）

Supabase Realtime 基于 WebSocket，监听数据库表的变更并实时推送到前端。典型场景：聊天室、协作文档、实时仪表盘、通知系统。

监听表变更

```
// 监听 messages 表的新增消息
const channel = supabase
  .channel('messages-channel')
  .on(
    'postgres_changes',
    {
      event: 'INSERT',
      schema: 'public',
      table: 'messages'
    },
    (payload) => {
      console.log('新消息:', payload.new)
      setMessages(prev => [...prev, payload.new])
    }
  )
  .subscribe((status) => {
    if (status === 'SUBSCRIBED') {
      console.log('实时监听已就绪')
    }
  })

// 组件卸载时清理
return () => {
  supabase.removeChannel(channel)
}
```

支持三种事件类型：

- `INSERT` — 新增行
- `UPDATE` — 行被修改
- `DELETE` — 行被删除
- `*` — 以上三种全监听

也可以加过滤条件，只监听特定行的变更：

```
// 只监听当前用户的通知
.on('postgres_changes', {
  event: 'INSERT',
  schema: 'public',
  table: 'notifications',
  filter: `user_id=eq.${userId}`
}, callback)
```

Realtime 的前置条件

1. **Supabase Dashboard → Replication → 开启对应表的 Realtime**（默认关闭）
2. 表必须开启 RLS，且有对应的 SELECT Policy——Realtime 推送也受 RLS 过滤

用 Lovable 接入 Realtime

```
在 messages 页面添加实时消息功能：
- 监听 messages 表的 INSERT 事件
- 有新消息时立即追加到列表底部，不刷新整页
- 组件卸载时取消订阅
```

Lovable 会生成对应的 `useEffect` 钩子和 channel 清理逻辑。

完整流程：从零搭一个 Todo 应用

把以上四个模块组合起来，按顺序跑通一个完整应用：

```
第1步：
  在 Supabase 创建 todos 表，开启 RLS，加四条 CRUD Policy

第2步：
  在 Lovable 连接 Supabase，告诉 AI 表结构

第3步：
  添加邮箱登录/注册页面，登录后跳转 /todos

第4步：
  /todos 页面：查询当前用户的 todos，支持新增/完成/删除

第5步：
  监听 todos 表 INSERT 事件，多个标签页同时打开时实时同步

第6步：
  加一个头像上传，传到 Supabase Storage avatars bucket
```

每一步都是一条 Lovable prompt，独立可验证，出了问题精准回滚。

上线前必查清单

检查项	说明
RLS 已开启	每张表都要 `ALTER TABLE xxx ENABLE ROW LEVEL SECURITY`
所有表都有 Policy	开了 RLS 但没有 Policy = 无法访问任何数据
anon key 没有泄露为 service_role	看 Lovable 项目里配置的 key 以 `eyJ...` 开头且比 service_role 短
Storage bucket 权限配对	public bucket 不要放敏感文件；private bucket 记得建 Policy
Realtime 只在需要时开	不需要实时更新的表不要开 Replication，减少资源消耗
Email Confirm 重新开启	开发时关掉的 "Confirm email" 上线前要打开

📦 **用 Supabase 搭完了数据层，这个项目能进简历吗？Amelia 帮你评估：**

匠人学院™
JR ACADEMY

Amelia

IT Career Consultant

在澳洲 IT 业务咨询领域拥有10年丰富经验和专业知识。对 IT 行业的发展趋势和技术需求有着深入的了解，擅长提供转码就业问题解决方案，能够准确把握最新的技术趋势和就业需求，为您提供最具前瞻性的建议。

探索IT领域，点亮你的科技未来

提供问题解决方案　工作内推机会

IT就业群提供　IT技术提升服务

300+
已帮助300+人成功就业

1000+
帮助1000+学员答疑解惑

200+
已帮助数百余人成功转码

CHAPTER 07

🌐 从 Lovable 到生产：自定义域名、SEO meta 和 Google Analytics

Lovable 默认给每个项目分配一个 `yourapp.lovable.app` 子域名，发开发原型够用，但上线产品不行——用户不信任 `lovable.app` 结尾的域名，SEO 权重也归不到你自己。这一章把三件事拆开讲：把域名换成你的、让搜索引擎读到正确的元数据、接 GA4 追踪用户行为。

1. 先发布，再接域名

自定义域名必须在项目发布后才能生效。

打开 Lovable 项目，右上角 → **Share** 按钮 → **Publish**。首次发布会生成 `yourapp.lovable.app` 的公开访问地址，后续每次发布只更新内容，域名不变。

> **Plan 限制**：自定义域名是付费功能，Free 计划只有 `lovable.app` 子域名。Starter 及以上方可绑定自己的域名。

2. 绑定自定义域名

2.1 在 Lovable 添加域名

项目页面 → **Settings** → **Domains** → **Add custom domain**，输入你的域名（如 `app.yourdomain.com` 或根域名 `yourdomain.com`），确认后 Lovable 会展示两条 DNS 记录：

类型	名称	值
A	`@` 或指定主机名	Lovable 提供的 IP 地址
TXT	验证用子域名	Lovable 提供的验证字符串

具体的 IP 和验证字符串会在你的 Lovable dashboard 里实时显示，每个项目不同，以 dashboard 为准。

2.2 在域名注册商添加 DNS 记录

登录你的域名注册商（Cloudflare / Namecheap / GoDaddy / 阿里云解析等），找到对应域名的 DNS 管理，照着 Lovable dashboard 给出的值添加：

A 记录（根域名）

```
Type:  A
Name:  @
Value: <Lovable 提供的 IP>
TTL:   Auto 或 3600
```

子域名（如 app.yourdomain.com）

```
Type:  CNAME
Name:  app
Value: <Lovable 给出的 CNAME 目标>
TTL:   Auto
```

TXT 验证记录

```
Type:  TXT
Name:  <Lovable 指定的主机名>
Value: <Lovable 提供的验证字符串>
TTL:   Auto
```

> **Cloudflare 用户注意**：A 记录的 Proxy 状态（橙色云）会干扰 Lovable 的 SSL 证书签发，先设为 **DNS only**（灰色云），等证书签发完再按需开启。

2.3 等待 DNS 传播和 SSL 颁发

保存 DNS 记录后回到 Lovable dashboard，域名状态会显示验证进度。SSL 证书由 Let's Encrypt 自动签发，全程不需要手动操作。

- 通常：几分钟到 1 小时
- 最长：DNS TTL 较长时可能 72 小时

可以用 dnschecker.org 查全球 DNS 传播情况，看 A 记录是否已经指向 Lovable 的 IP。

2.4 验证生效

浏览器访问你的域名，看到项目内容 + 地址栏显示 🔒 HTTPS，则绑定成功。

3. SEO 元数据配置

3.1 Lovable 的内置 SEO 工具

2026 年 5 月 13 日，Lovable 发布了 **Discoverability** 功能，集成了 SEO 审查、SSR/预渲染、Semrush 对接。对新项目来说：

- **2026-05-13 之后创建的项目**：默认使用 TanStack Start + SSR，每次请求返回完整 HTML，对搜索引擎和社交预览 bot 完全可见。
- **此前的 React + Vite 项目**：对 Google、Bing、社交预览 bot（Twitter、Facebook）、AI 引擎（ChatGPT、Perplexity）等验证过的爬虫启用按需预渲染。

两种情况下，SEO 元数据的配置方式一致。

3.2 基础元数据：标题、描述、Favicon

项目页面 → **Settings** → **SEO** （或 **Share** 面板 → **SEO**）：

字段	对应 HTML	作用
Page title	`<title>`	浏览器标签 + Google 搜索结果标题
Meta description	`<meta name="description">`	搜索结果摘要（建议 120–160 字符）
Favicon	`<link rel="icon">`	浏览器标签图标，默认是 Lovable Logo
Share image / OG image	`<meta property="og:image">`	社交分享时的预览图

修改后立即保存，Lovable 自动更新 `index.html` 里的对应 `<meta>` 标签。如果用 Lovable 的 AI 聊天改，直接说：

```
把 meta description 改成"…你的描述…"，并把 OG image 换成 https://…/og.png
```

Lovable 会定位到 `index.html` 或 `vite.config.ts` 的对应位置修改。

3.3 Open Graph 完整写法（手动编辑 index.html）

如果需要精确控制，打开 Lovable 的 **Dev Mode**，找到项目根目录下的 `index.html`，在 `<head>` 里加入：

```
<!-- 基础 SEO -->
<title>你的应用名 | 品牌名</title>
<meta name="description" content="120 字以内，说清楚这个应用是干什么的" />

<!-- Open Graph（Facebook、LinkedIn、微信） -->
<meta property="og:type" content="website" />
<meta property="og:title" content="你的应用名 | 品牌名" />
<meta property="og:description" content="120 字以内的描述" />
<meta property="og:image" content="https://yourdomain.com/og-image.png" />
<meta property="og:url" content="https://yourdomain.com" />

<!-- Twitter Card -->
<meta name="twitter:card" content="summary_large_image" />
<meta name="twitter:title" content="你的应用名 | 品牌名" />
<meta name="twitter:description" content="120 字以内的描述" />
<meta name="twitter:image" content="https://yourdomain.com/og-image.png" />
```

OG image 规格建议：1200×630px，PNG 或 JPG，文件大小 < 1MB。可以用 Figma / Canva 做，导出后放到 Lovable 项目的 `public/` 目录下（Dev Mode 里拖进去），URL 就是 `https://yourdomain.com/og-image.png`。

3.4 接入 Google Search Console

让 Google 知道你的域名存在，加速首次索引：

1. 打开 search.google.com/search-console，添加 **Domain property**（填裸域名，不加 `https://`）
2. Google 会要求在 DNS 添加 TXT 验证记录，格式：

```
Type:  TXT
Name:  @
Value: google-site-verification=xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
```

1. 在域名注册商添加后点 **Verify**，通常几分钟内验证通过
2. 验证成功后，到 **URL Inspection** 里输入你的首页 URL，点 **Request Indexing**——告诉 Google 可以爬了

Search Console 也能看到哪些关键词带来了流量，以及哪些页面有抓取错误，接上去之后定期看一下。

4. Google Analytics GA4 埋点

4.1 获取 GA4 Measurement ID

1. 登录 analytics.google.com，创建账号 → 属性（Property）
2. 数据流（Data Streams）→ 添加网站 → 填你的域名
3. 创建完成后，进入该数据流，看到 **Measurement ID**，格式是 `G-XXXXXXXXXX`

4.2 在 index.html 里加载 GA4

在 Dev Mode 打开 `index.html`，把下面两段脚本插到 `<head>` 结尾处，`<title>` 之后都行：

```
<!-- Google Analytics GA4 -->
<script async src="https://www.googletagmanager.com/gtag/js?id=G-XXXXXXXXXX"></script>
<script>
  window.dataLayer = window.dataLayer || [];
  function gtag(){dataLayer.push(arguments);}
  gtag('js', new Date());
  gtag('config', 'G-XXXXXXXXXX');
</script>
```

把 `G-XXXXXXXXXX` 替换成你在 GA4 拿到的 Measurement ID。

> **为什么放 `<head>` 而不是 `<body>`？** GA4 脚本需要在 React 渲染之前完成加载，否则初始页面访问会被漏掉。

4.3 SPA 路由追踪（关键！）

Lovable 生成的应用是 React SPA，默认 GA4 只记录第一次页面加载，用户在应用内跳转不会触发新的 `page_view`。要追踪路由变化，在你的主组件（通常是 `src/App.tsx`）里加一个 hook：

```
import { useEffect } from 'react';
import { useLocation } from 'react-router-dom';

function usePageTracking() {
  const location = useLocation();

  useEffect(() => {
    if (typeof window.gtag !== 'function') return;
    window.gtag('event', 'page_view', {
      page_path: location.pathname + location.search,
      page_location: window.location.href,
    });
  }, [location]);
}
```

然后在 `App` 组件的函数体里调用：

```
function App() {
  usePageTracking(); // 每次路由变化触发 page_view

  return (
    <Routes>
      {/* 你的路由 */}
    </Routes>
  );
}
```

如果用 Lovable 的 AI 来加，直接说：

```
在 App.tsx 里加一个 usePageTracking hook，在每次路由变化时调用 window.gtag('event', 'page_view', ...)，追踪 GA4 SPA 路由。
```

Lovable 会自动处理 TypeScript 类型声明（`window.gtag`）和导入。

4.4 验证埋点是否生效

1. 打开 Google Analytics → **Reports** → **Realtime** → **Users in last 30 minutes**
2. 在另一个标签页打开你的应用，来回点几个页面
3. Realtime 报表里出现用户活动，并且每次路由跳转都有新的 `page_view` 事件，说明埋点正常

也可以在浏览器 DevTools → Network 里过滤 `google-analytics` 或 `gtag`，看到有 `collect` 请求发出去即为正常。

5. 上线前检查清单

项目	检查方法
域名 HTTPS 正常	浏览器地址栏显示 🔒，访问 HTTP 自动跳转 HTTPS
自定义域名内容正确	打开域名，内容与 Lovable 项目一致
页面 `<title>` 和 description	浏览器查看源代码，搜索 `<title>` 和 `description`
OG 图片预览	用 opengraph.xyz 粘贴你的 URL 看预览效果
Google Search Console 验证	状态显示 "Domain verified"
GA4 实时数据	Realtime 报表看到自己的访问记录
SPA 路由追踪	手动点 3–4 个页面，Realtime 出现对应 `page_view` 事件

6. 常见问题

域名连接后显示空白页或 ERR_SSL_PROTOCOL_ERROR

SSL 证书还没签发完，等 10–30 分钟。如果用了 Cloudflare 且 Proxy 是开的，先关掉 Proxy（灰色云）再等证书，签发后再决定是否开 Proxy。

Google 搜索不到我的网站

新域名被 Google 索引通常需要几天到几周。先到 Search Console 手动 Request Indexing，再检查 `robots.txt` 里没有 `Disallow: /` 把自己屏蔽掉。

GA4 实时报表看不到数据

首先排查是否装了 uBlock 或 AdGuard——广告拦截器会屏蔽 `googletagmanager.com` 请求。用隐身窗口（不带扩展）测试。其次确认 `G-XXXXXXXXXX` 里的 ID 和 GA4 属性里的 Measurement ID 一致，字母 O 和数字 0 很容易搞混。

OG 图片在微信里不显示

微信的图片爬虫对 CORS 和 Content-Type 有要求，图片必须放在自己的域名（不能是跨域的图床），且 Content-Type 必须是 `image/png` 或 `image/jpeg`。把 OG 图放到 Lovable 项目 `public/` 目录，用你的自定义域名 URL 引用。

🌐 **应用上线了，用户数据怎么追踪、SEO 怎么做，下一步聊什么？Rain 帮你想：**

匠人学院
JR ACADEMY

用 Lovable 搭出来了，接下来怎么写进简历？

这本手册的在线版在 jiangren.com.au/wiki/lovable-guide，随官网更新。匠人学院还有 20+ 本同款 AI 工具指南（Claude Code、Cursor、Windsurf、CrewAI、Dify、n8n…）和 AI Engineer 方向的项目制课程——想知道 Lovable + Supabase 项目怎么落到简历里、做成找工作的作品，来聊聊。

扫码进社群，Lovable 实操答疑

把这本 PDF 转给还在从零手写 React 样板代码的同事。

jiangren.com.au