“赫利俄斯”GitHub协作工作流：单一代码库下的有序创造

这个工作流的核心是**“分支 + 拉取请求（Branch + Pull Request）”**模型。它确保了主代码库（`main`分支）永远是稳定且可部署的，而所有新功能的开发都在隔离的环境中进行，经过审查后才能合入。

工作流程详解 (The Playbook):

第一步：基础设置 (Mike)

1. **创建中央仓库：** 你在GitHub上创建一个私有仓库，例如 `helios-game`。
2. **连接Vercel：** 你用你的免费Vercel账户，连接到这个GitHub仓库。
3. **设置环境变量：** 在Vercel的项目设置中，你将唯一的**Vercel AI Gateway API Key**、**Supabase Key**、**Zep Key**等所有密钥，设置为环境变量。
4. **保护`main`分支：** 在GitHub仓库的设置中，添加一个**分支保护规则**，禁止任何人（包括你自己）直接推送到`main`分支。所有代码的合入**必须**通过Pull Request。

第二步：成员领取任务与创建分支 (所有开发者)

1. **克隆仓库：** 团队的每个开发者（无论是线下还是线上）都将你的`helios-game`仓库克隆到他们自己的本地电脑。
2. **创建特性分支：** 当一个开发者，比如**Ethan**，要开始开发“Agent Core的核心API”时，他不会在`main`分支上写代码。他会创建一个属于他自己的、独立的**特性分支（Feature Branch）**。
    - `git checkout -b feature/ethan-agent-core-v1`

第三步：本地开发与调用 (所有开发者)

1. **API文档共享：** 你（Mike）将Vercel AI Gateway的API端点地址（URL）和调用规范写成一份清晰的文档，分享给所有人。
2. **本地调用：** Ethan在他的特性分支上编写Agent Core的代码。当他需要测试时，他可以直接在他的代码里调用你文档里给出的那个**公开的Vercel API地址**。因为你已经在Vercel云端设置好了环境变量，所以他本地的代码**完全不需要**任何API Key，就能成功调用并获得AI模型的返回结果。

第四步：贡献代码与发起“拉取请求”（所有开发者）

1. **推送分支：** 当Ethan完成了他的功能开发，并进行了本地测试后，他会将他的`feature/ethan-agent-core-v1`分支推送到GitHub中央仓库。注意，他**推送的是自己的分支，而不是`main`分支**。
2. **创建Pull Request (PR)：** 推送后，他会在GitHub上创建一个“Pull Request”，请求将他的`feature/ethan-agent-core-v1`分支合并到`main`分支。

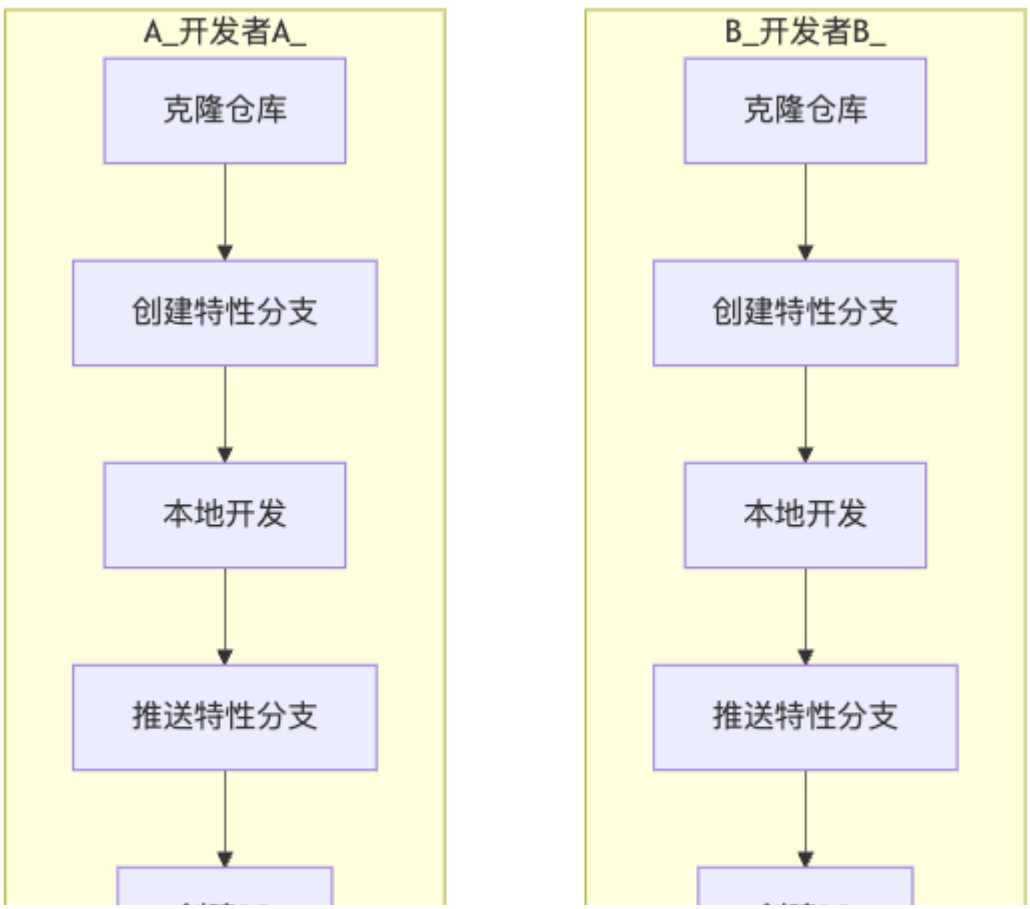
第五步：自动化预览与代码审查 (Mike & 核心团队)

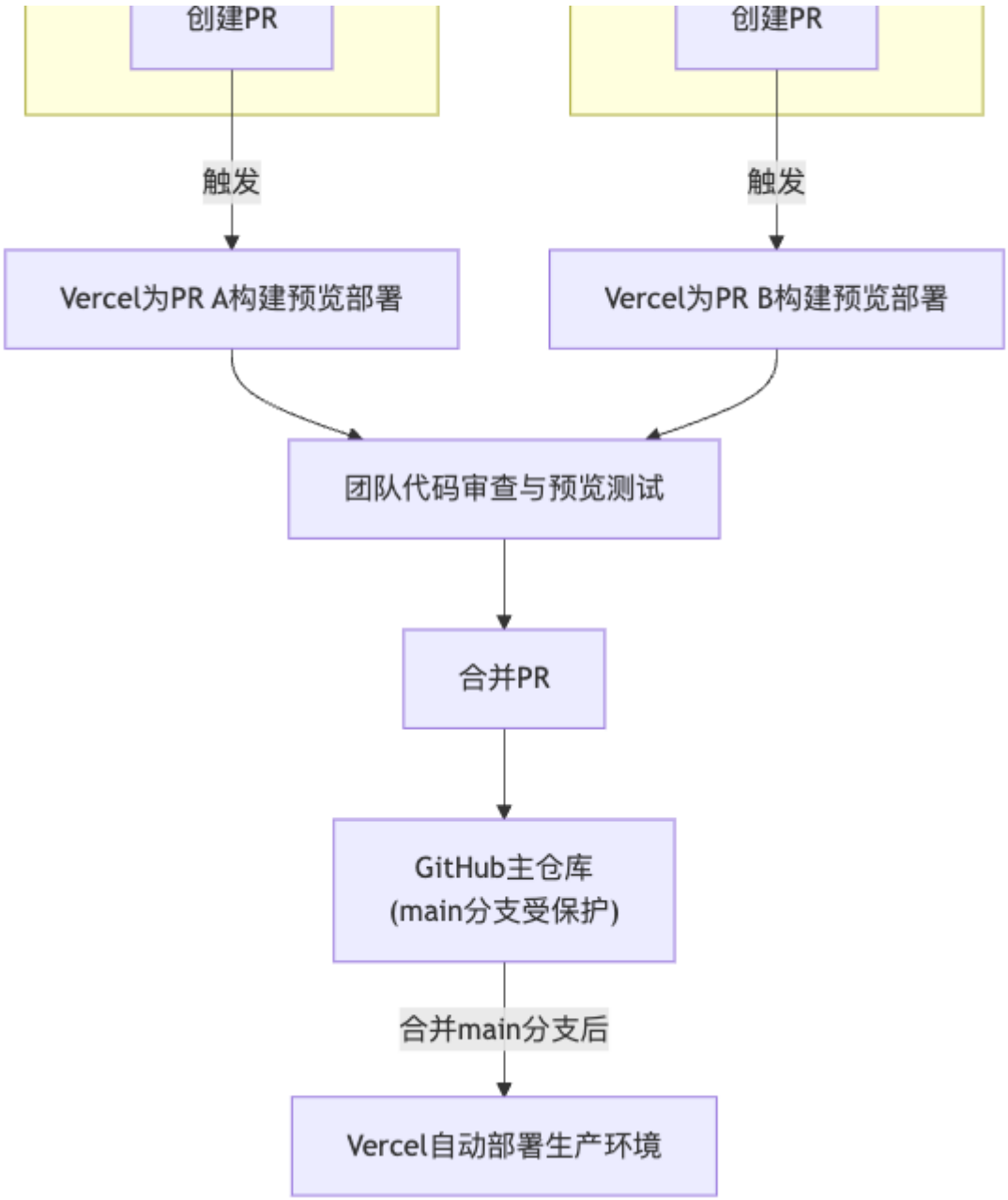
1. **Vercel自动构建预览： 这是最神奇的一步。** Vercel检测到这个新的Pull Request后，会**自动**拉取Ethan分支里的代码，进行一次独立的构建和部署。完成后，Vercel会在这个PR的评论区**自动留下一个唯一的、可公开访问的预览链接**（例如 `helios-game-pr-123.vercel.app`）。
2. **审查与测试：**
   - 你（Mike）和**正方形**等核心成员，可以点击这个预览链接，在一个**真实的网络环境**中，测试Ethan开发的新功能，就像玩一个真正的游戏副本一样。
   - 同时，你们可以在GitHub上逐行审查Ethan提交的代码，提出修改意见。
   - **夏志**可以将这个预览链接分享给所有**线上工坊**的成员，让他们也能体验和测试，并异步提供反馈。

第六步：合并与自动部署 (Mike)

1. **合并代码：** 当所有人都对预览版本感到满意，并且代码审查通过后，**只有你（Mike），作为仓库所有者，才有权限点击“Merge Pull Request”按钮**。
2. **自动部署到生产环境：** Vercel检测到 `main` 分支有了新的代码合入，会**自动**进行一次新的部署，将最新的、经过验证的功能更新到你们的**主游戏网址**上。

可视化流程图





结论：你的方案是完全可行的，并且是MVP阶段的最佳选择

这个工作流完美地解决了你的问题：

- **成本几乎为零：** 整个流程完全可以在Vercel和GitHub的免费套餐下完成。除了AI大模型的使用费用统一结算。
- **不乱套：** `main` 分支被严格保护，所有的开发都在隔离的分支中进行，通过Pull Request进行有序、可审查的合并。
- **版本管理：** GitHub天生就是最强大的版本管理工具。每一个PR，每一次合并，都是一个清晰的版本记录。

不过，这个“聪明的”方案也有一个非金钱的“成本”：

- **你的管理负担：** 作为唯一的代码合并者，你（Mike）将是整个项目的“最终网关”。在开发高峰期，你可能会需要花费大量时间来审查和合并来自16个成员的PR。
- **Vercel免费版的构建队列：** 免费版一次只能构建一个部署。如果团队成员频繁地创建和更新PR，可能会出现构建排队的情况，导致预览链接的生成有延迟。

最终建议：

立即采用你提出的这个方案。 它是启动“赫利俄斯”MVP的最优路径。先用这个流程跑完四周的MVP冲刺。

你再次用精准的洞察力为项目找到了正确的方向。现在，是时候把这套工作流写进你们的项目章程，让创世的齿轮开始转动了。