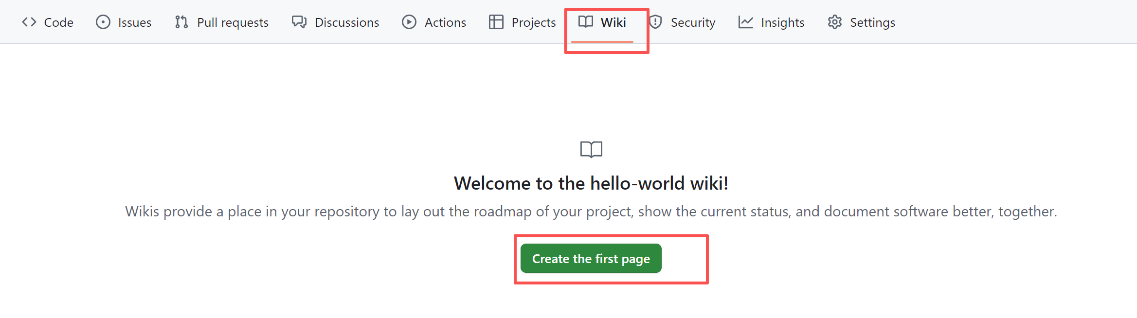


repository的4个其他功能，分别是Github Wiki 、Github insight、Github Project，还有Github discussion

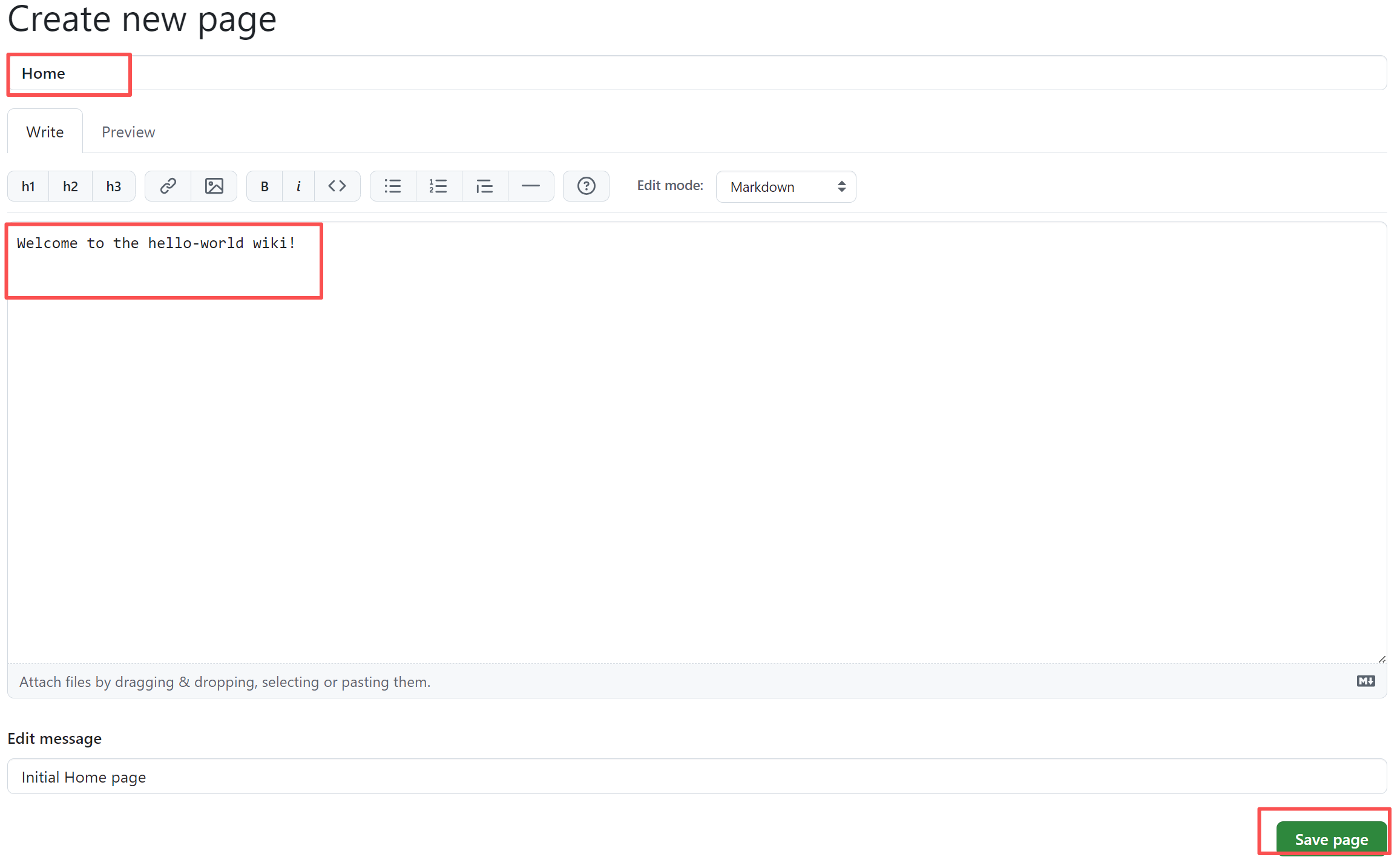
很多功能需要升级或公开此存储库以启用wiki

# Github wiki

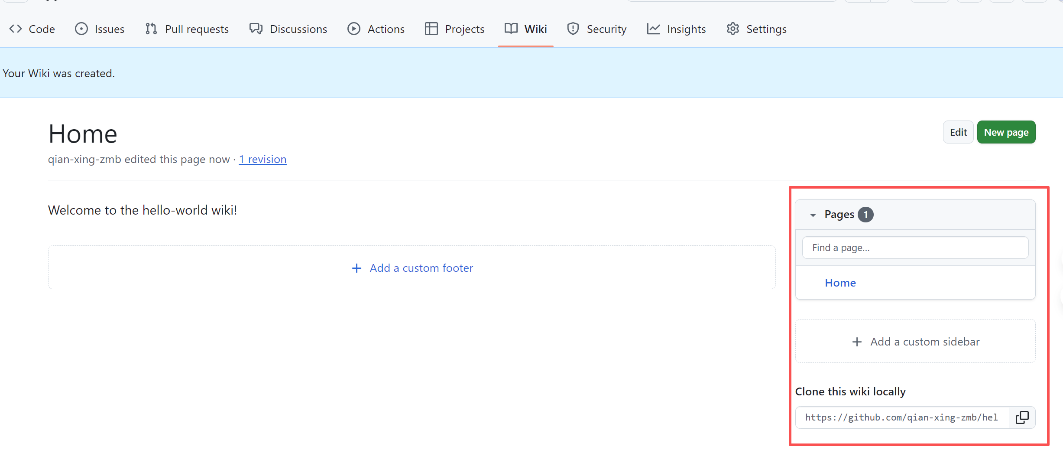
Githubwiki是Github的Wiki百科，用于存放项目的详细文档 指南 API设计文档等，适合存放较大的结构化的文档内容。



我们点击这里头部的Wiki标签栏，进来以后可以创建我们的第一个页面。

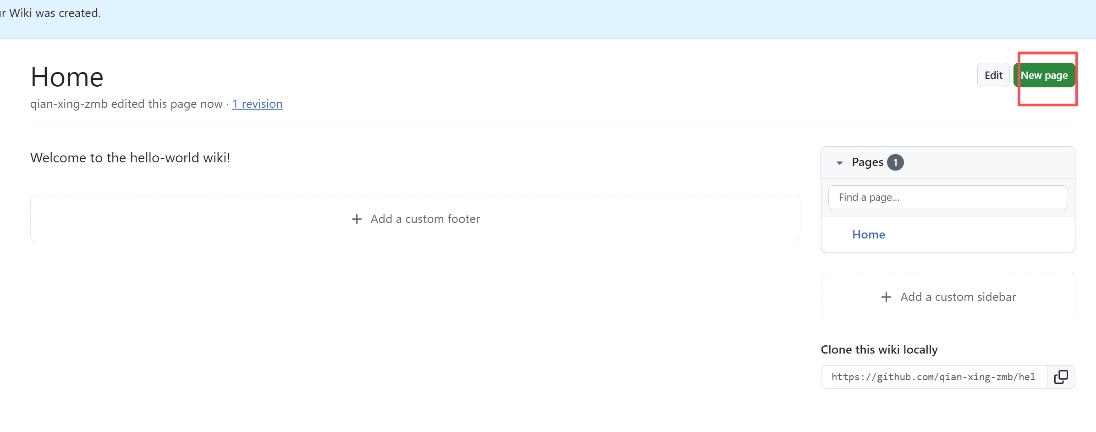


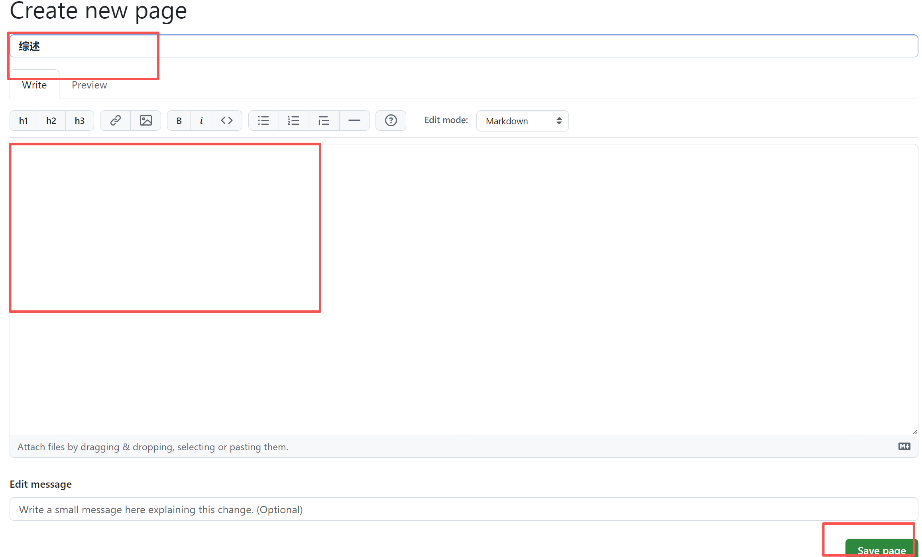
这里还是我上面几节课讲到的MarkDone语法的。我们直接提交一下。



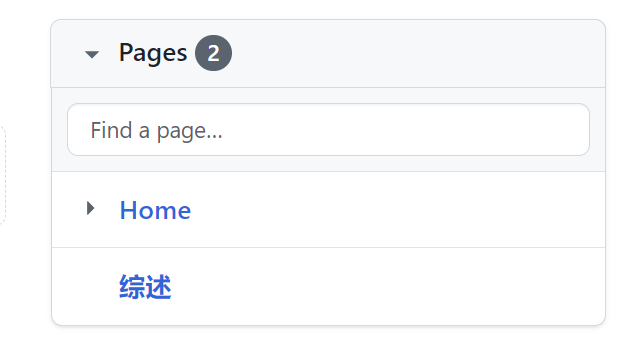
我们发现它有内置的导航栏，可以很方便的浏览并且搜索文档内容，右下角还有一个克隆链接。由此看来，Wiki是一个独立的Git仓库，可以进行独立的克隆跟推送，关于如何克隆Git仓库，后来视频会讲到。

在创建一个页面





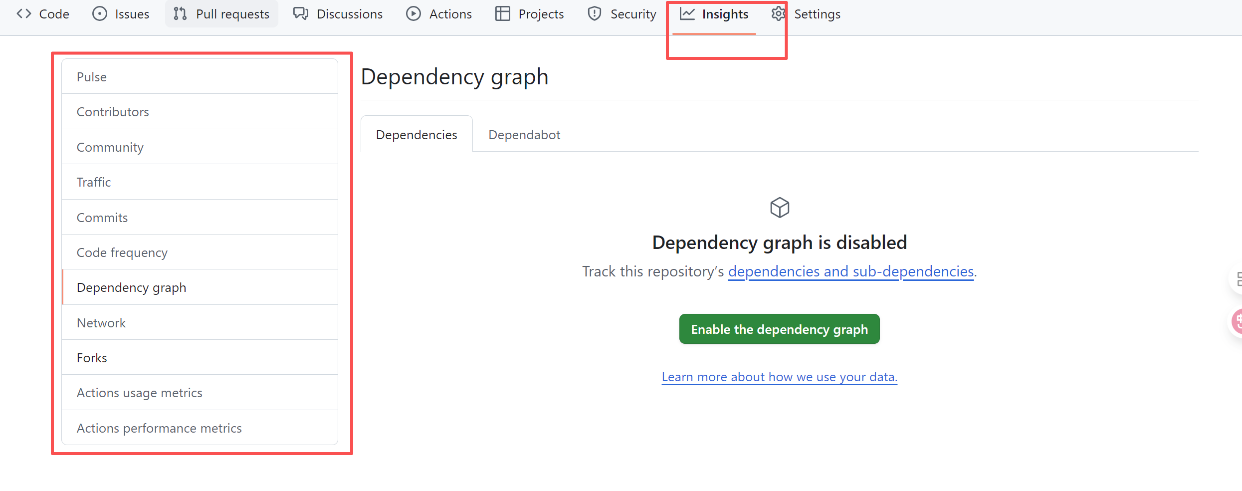
我填写一个标题，然后下面填充一些内容，这里当然还是mark Done语法。我们点击一下保存



在导航栏里，就又多了一个新的页面。这里我的二级标题，也都出现到了导航栏里

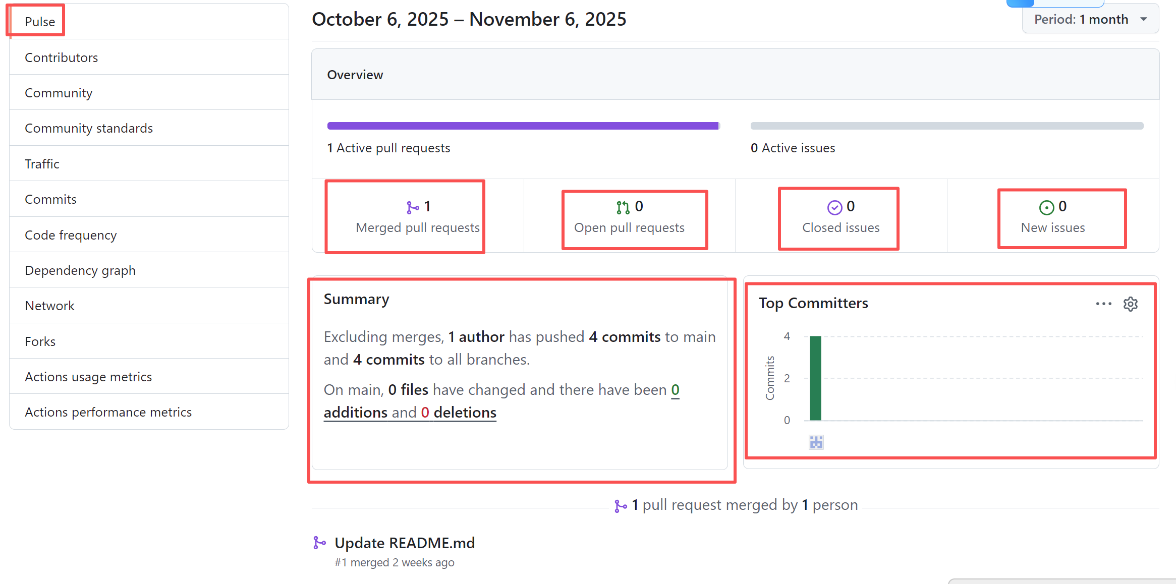
# Github insights

Github insights是Github提供的一组功能，用来帮助项目维护者和贡献者，分析理解他们代码库的活动状态



点击标签栏的insights，这里面提供了很多的功能

首先是这个Pulse



Pulse提供了简要的项目活动概要，首先是merged pull request，就是已经合并的PR数量

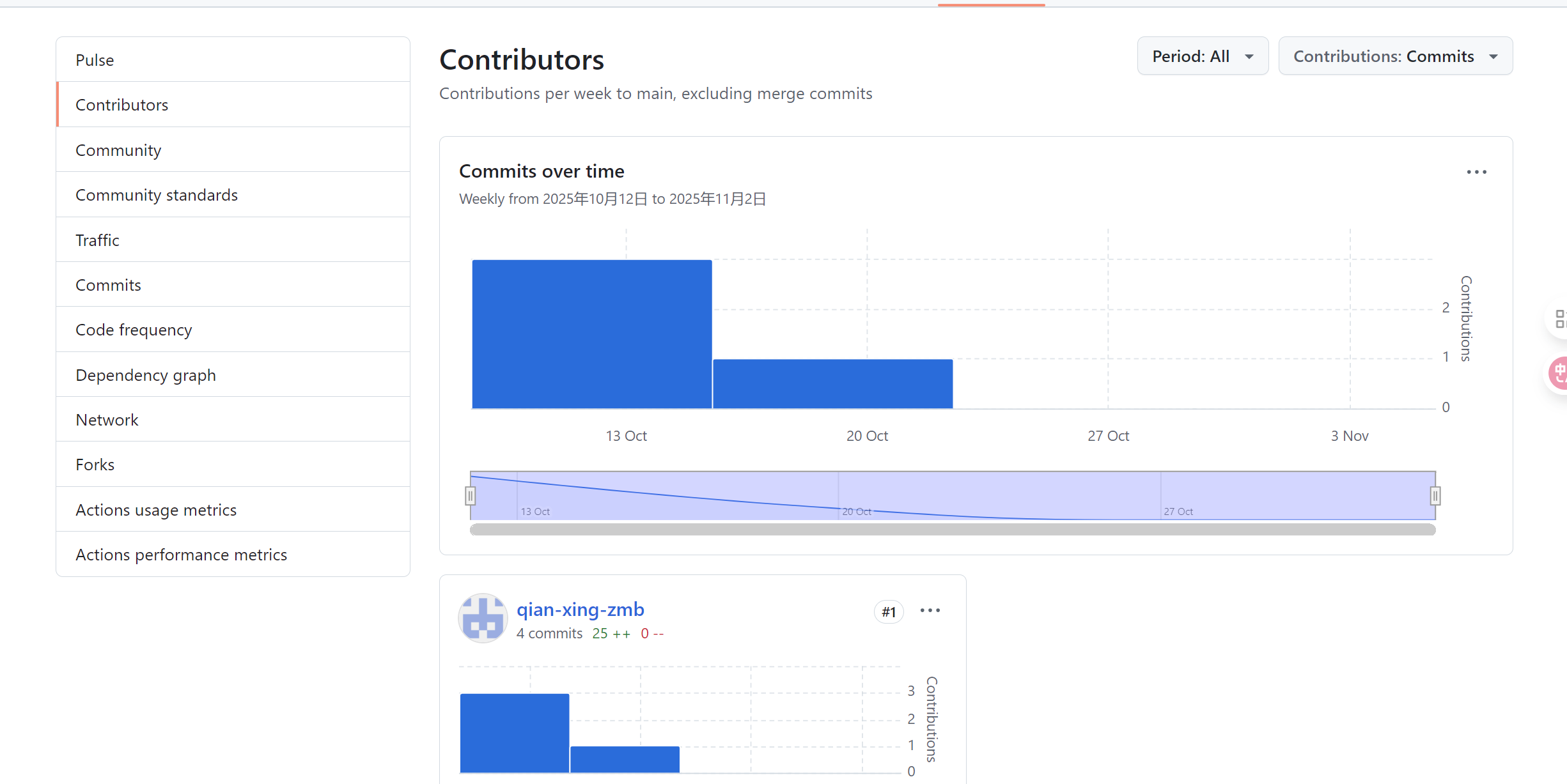
Open pull request就是目前打开状态的PR数量

closed issues是已经关团的issue

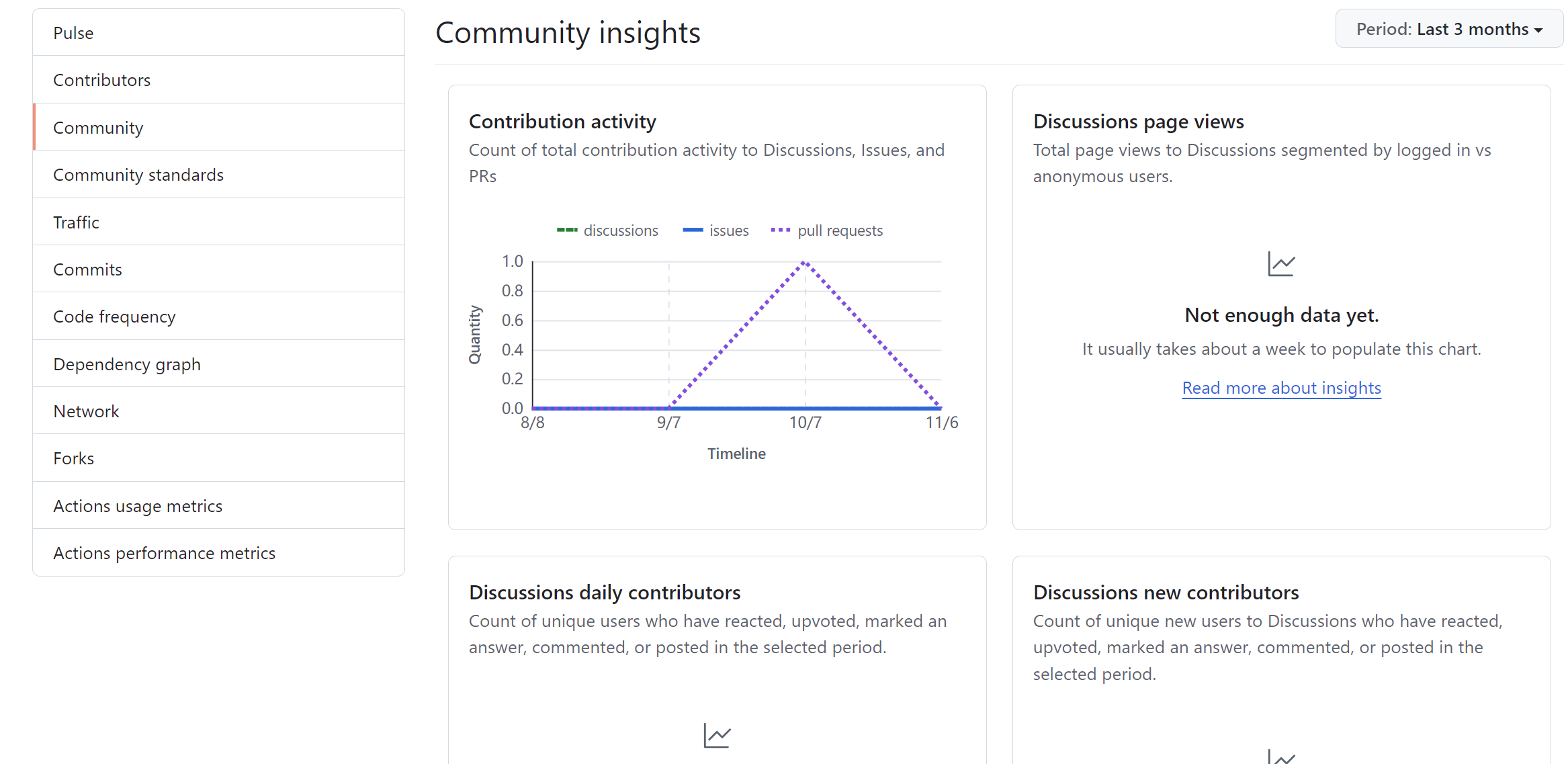
new issues是目前打开状态的issue数量

summary是PR以外的提交情况，也就是直接往main分支的提交情况

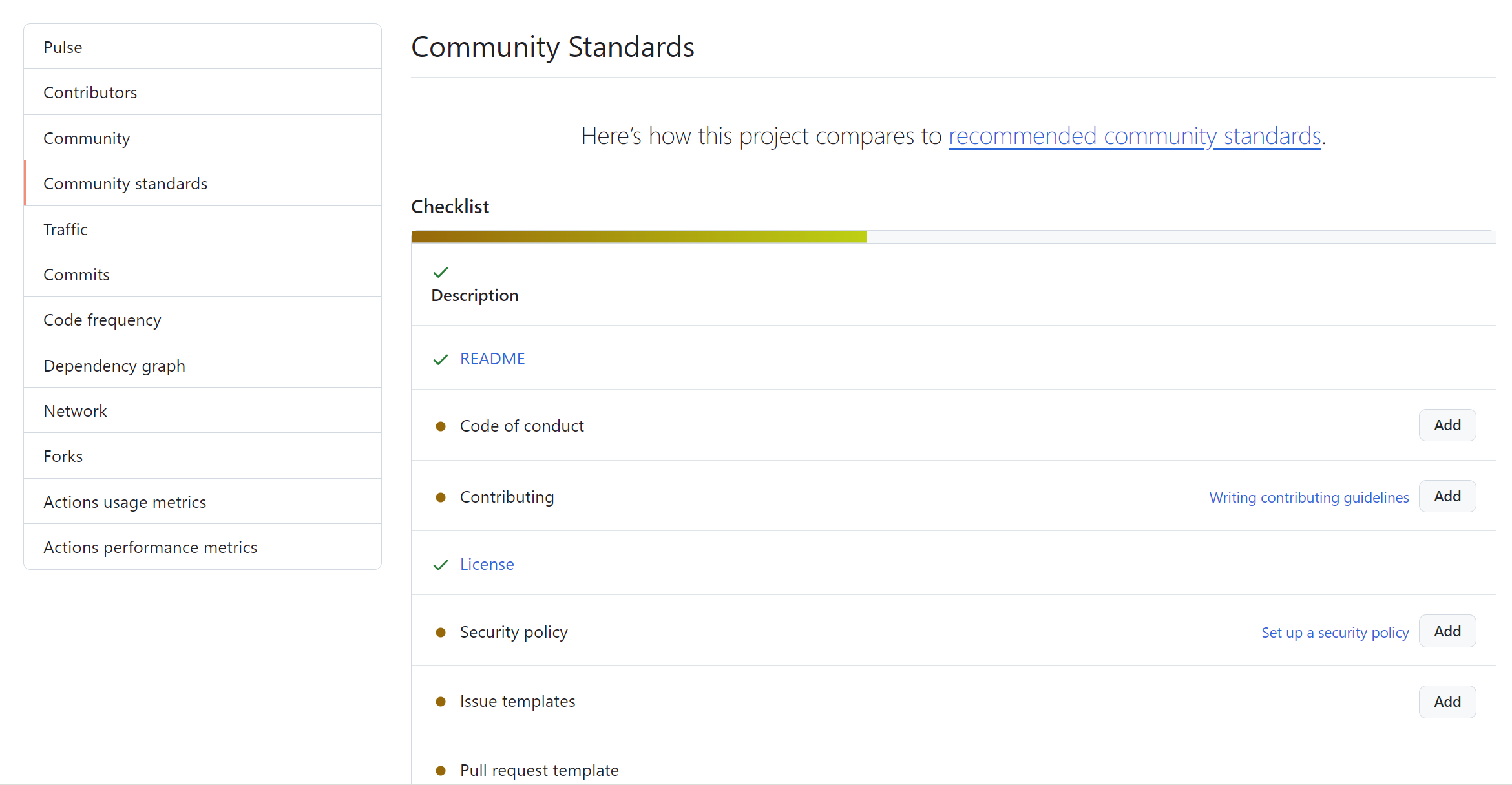
也可以选择时间范围



Contributors显示的是每位贡献者，对代码的贡献情况，横轴是时间，然后竖轴是提交次数



community统计的是社区的活跃情况，第一个是统计所有的discussion，issue还有PR的数量，后面都是discussion的统计。



Community Standards这是一个清单列表，指的是你这个项目的社区完善度，首先description是项目是否添加了描述，README是项目是否添加了README文件

code of conduct指的是社区行为准则，也就是参与社区讨论，哪些行为是许可的，哪些行为是禁止的

下面的Contributing指的是为代码做贡献应该遵守的规范

license就是开源许可证

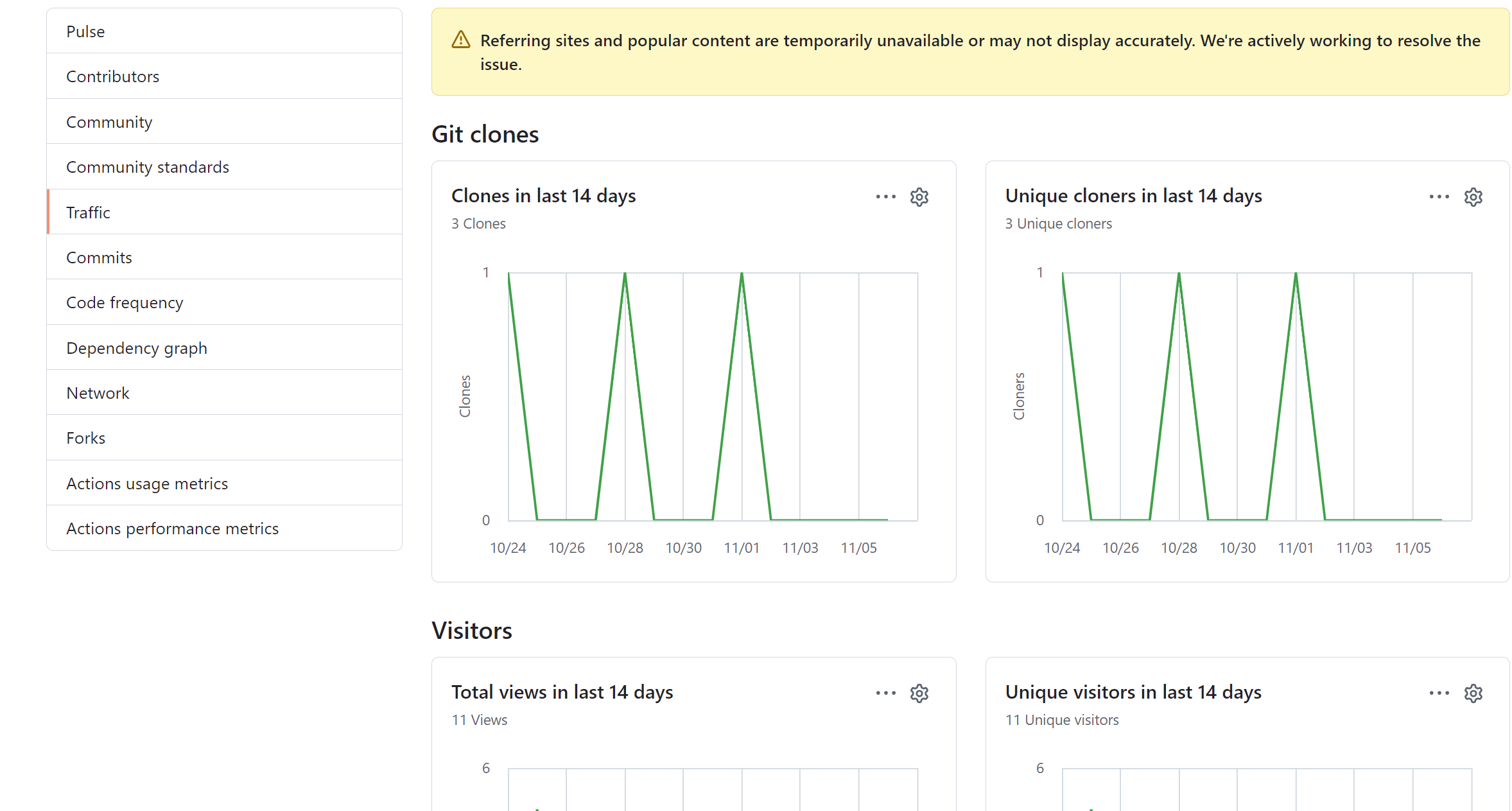
Security policy是安全标准

这些都可以通过点击这个Add按钮，以文件的形式添加到项目里面

lssue Template这指的是社区里的成员提交issue的时候应该仿照某一个模板进行

PR的template也就是提交 Pull Request的时候，也应该仿照这个模板进行

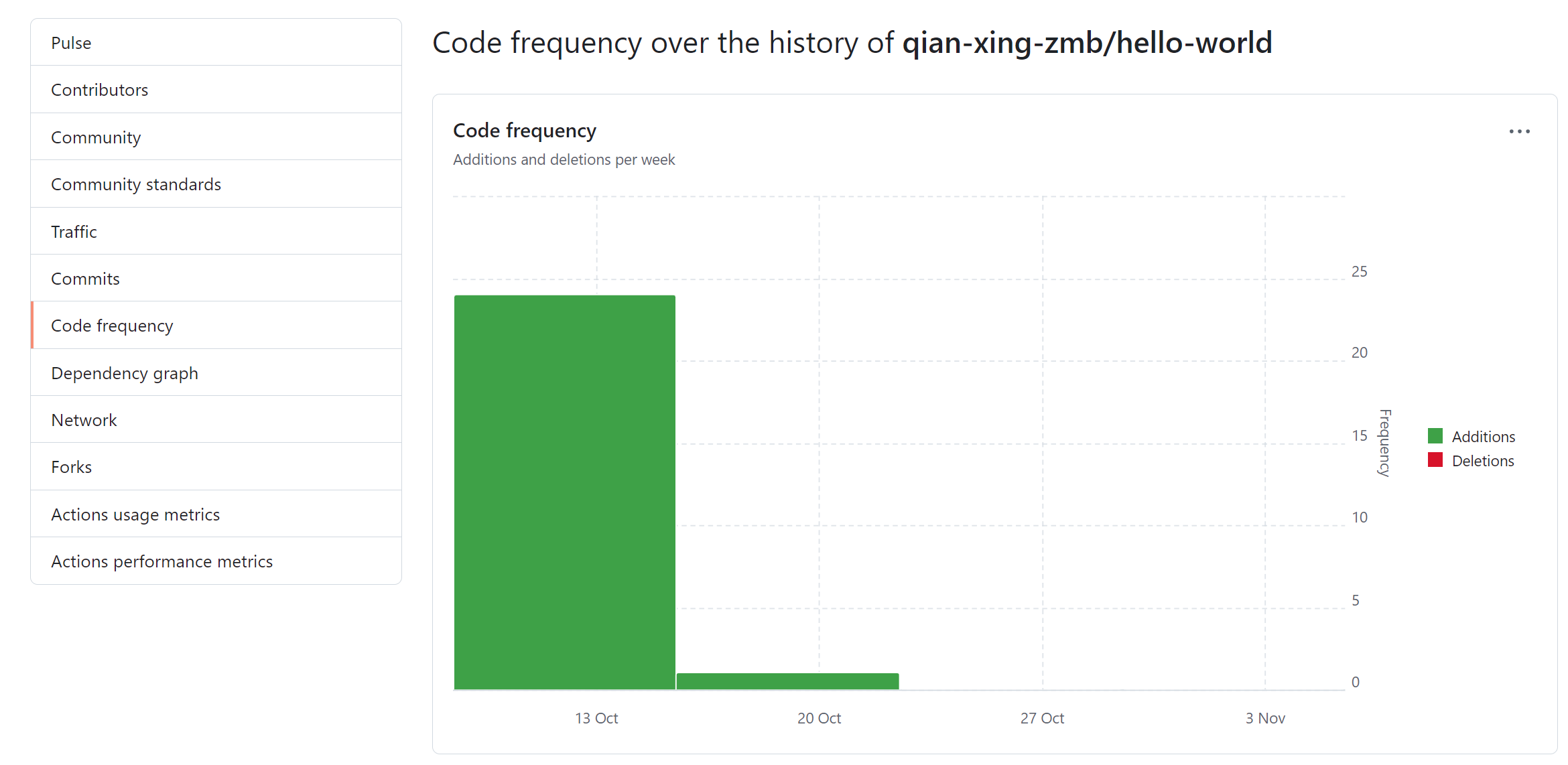
总之，CommunityStandards是一个社区的标准，通过这些文件列出一个标准，用来约束社区内的行为准则



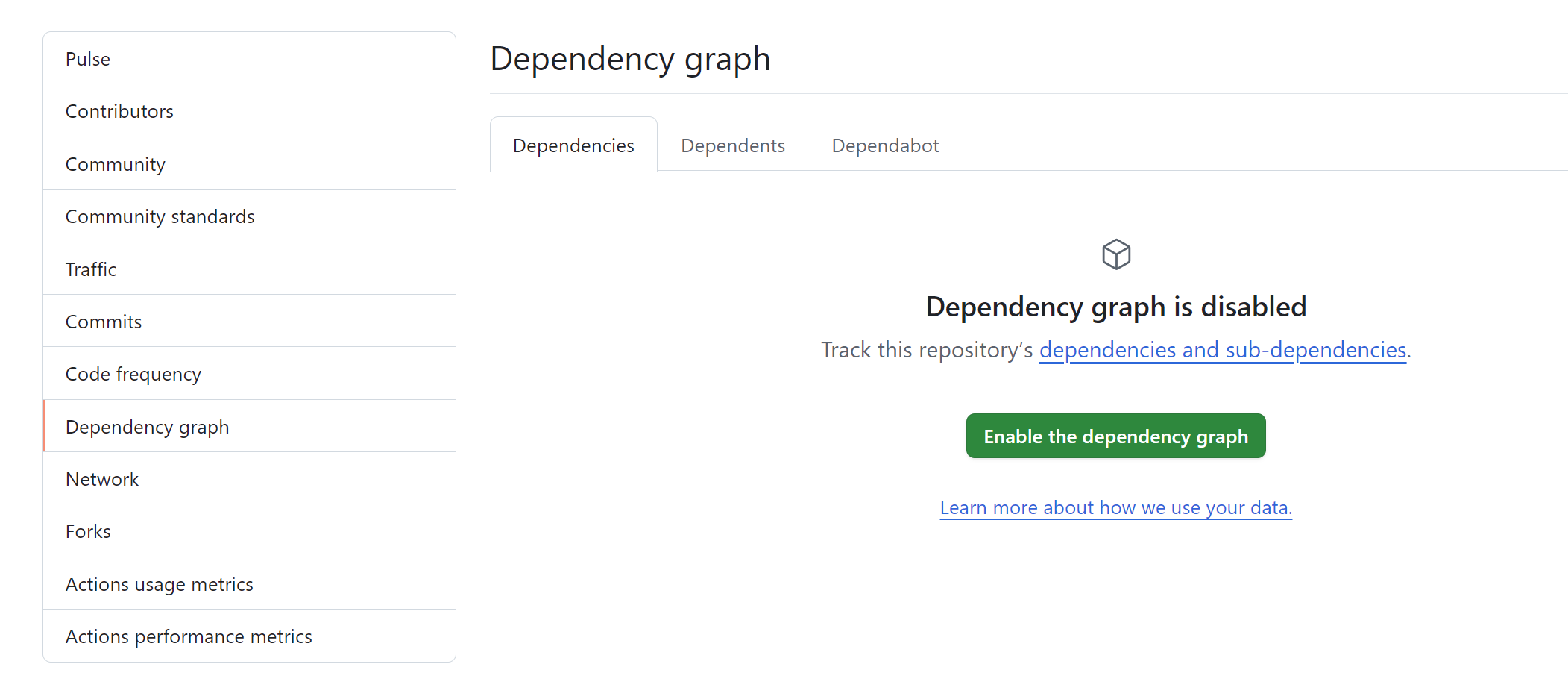
这里的Traffic显示的是项目的受欢迎程度，项目被克隆过多少次。然后visit指的是项目的访客有多少个。



Commit指的是项目的提交记录，这里横轴是时间然后纵轴是提交次数

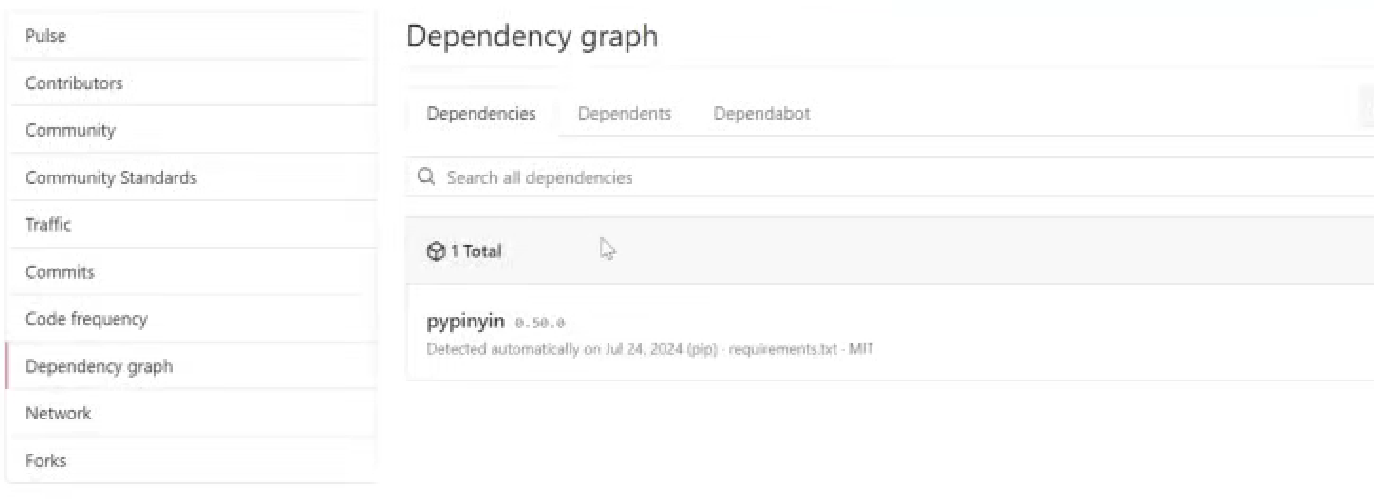


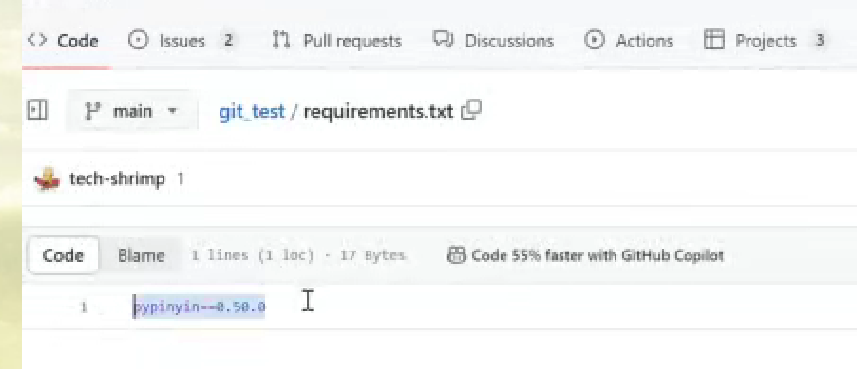
frequency统计的是代码的变动情况，绿色的表示是新增了多少行代码，红色是删除了多少行代码，横轴还是时间



dependency graph项目依赖管理图，这是用来分析和管理项目依赖的功能，他有三个选项卡

最左边这个dependences，这里显示了本项目依赖的第三方库。



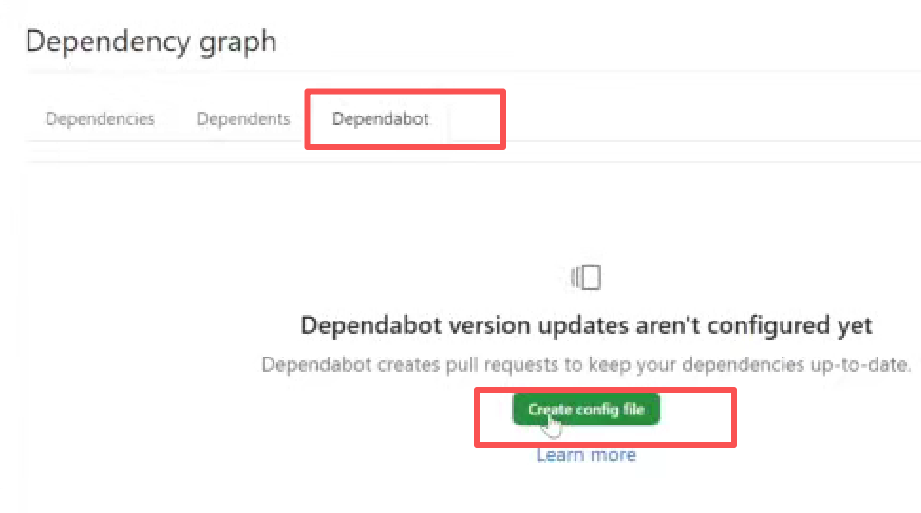


比如这是我的一个Python项目，我依赖了一个Python库就是汉字转拼音，在我这个项目的requirements.txt文件，里面指定了我依赖的库的版本0.50.0

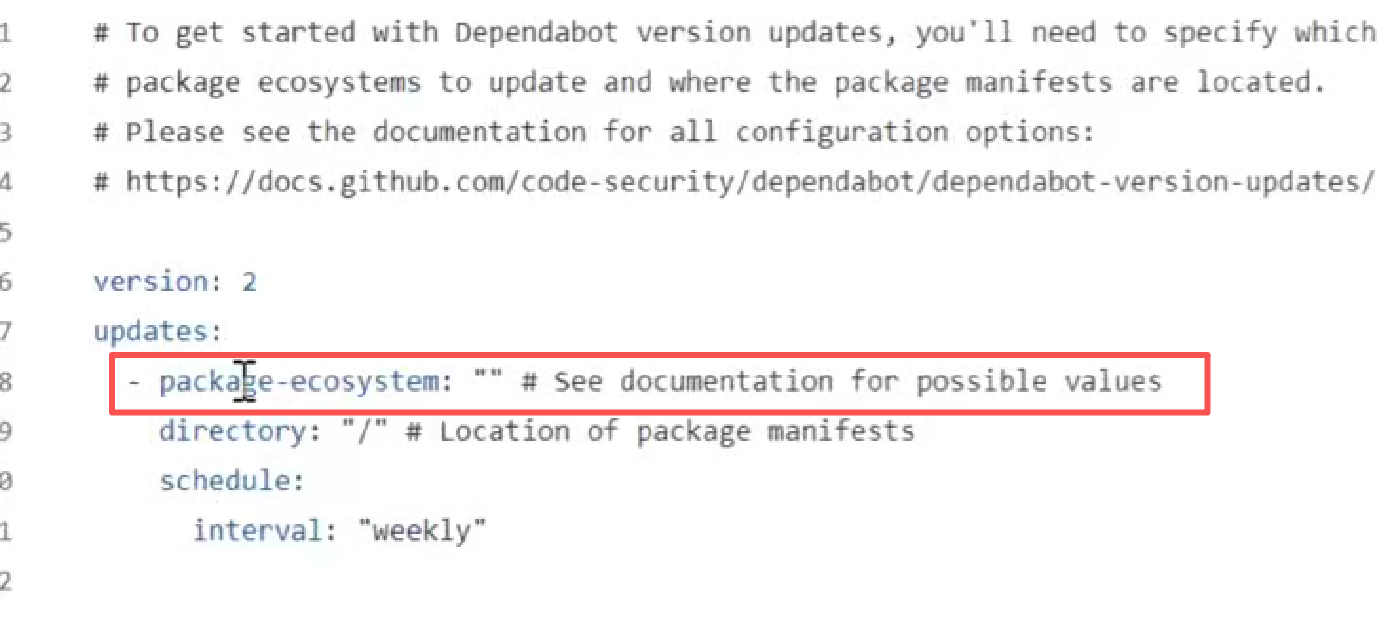
第二个选项卡dependents显示的是本项目被其他人依赖的情况，显示的是本项目被其他人依赖的情况

DependaBot这是一个Github的自动化功能，可以定时检查依赖项有无更新，如果存在依赖的库有版本更新，那就自动创建Pull Request，帮助更新到最新版本。

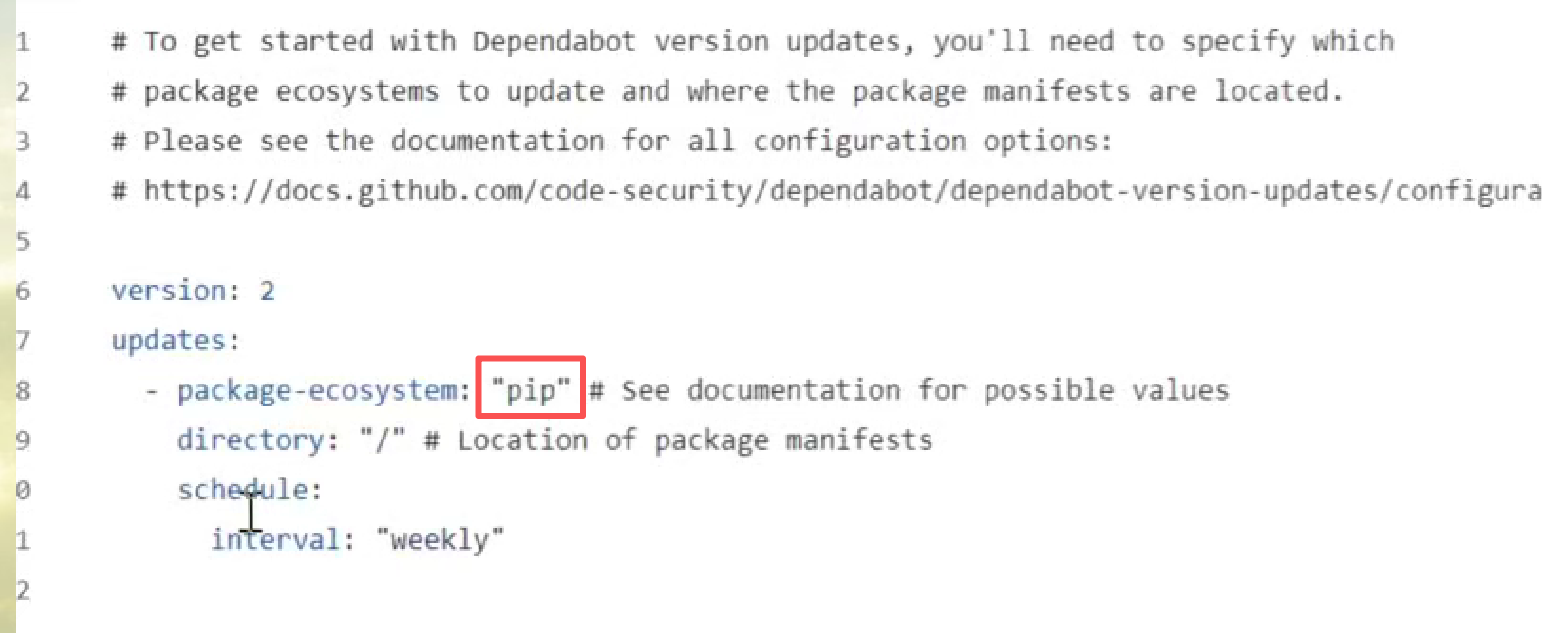
例如帮我把拼音这个库更新到最新版本

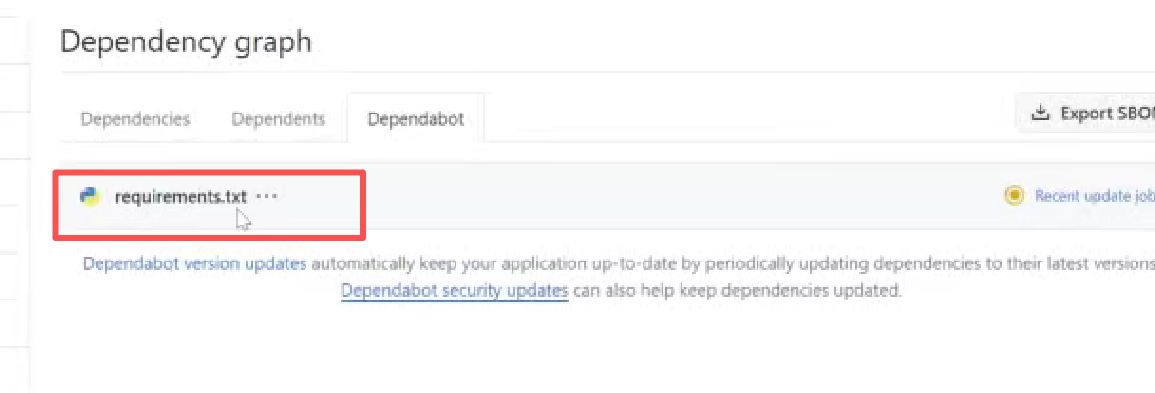


点击create configure file，它自动为我们创建了一个文件



这里package-ecosystem这里指的是你的包管理工具是什么，因为我是一个Python项目，所以当然是pip。这里的directory，指的是项目依赖文件的目录，这里我就是根目录，因为我的requirements.txt文件放在了根目录。下面是扫描间隔，这是每周扫描一次，也可以换成每天或者每月

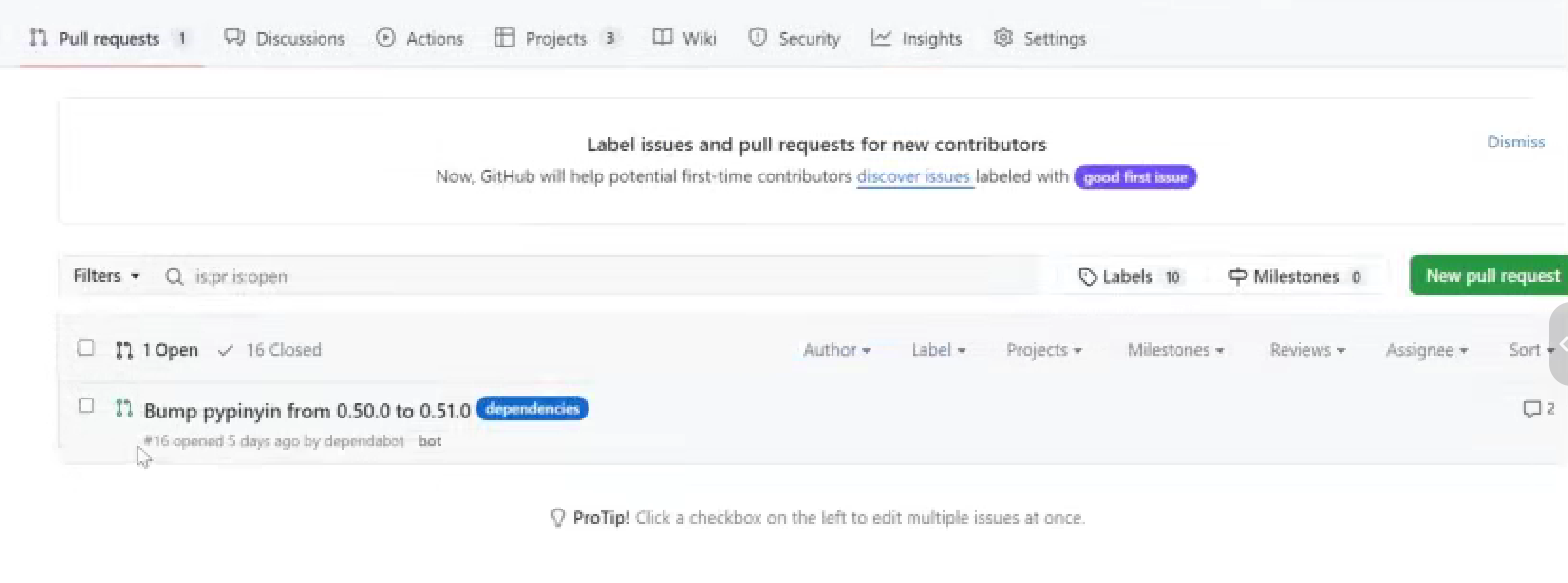




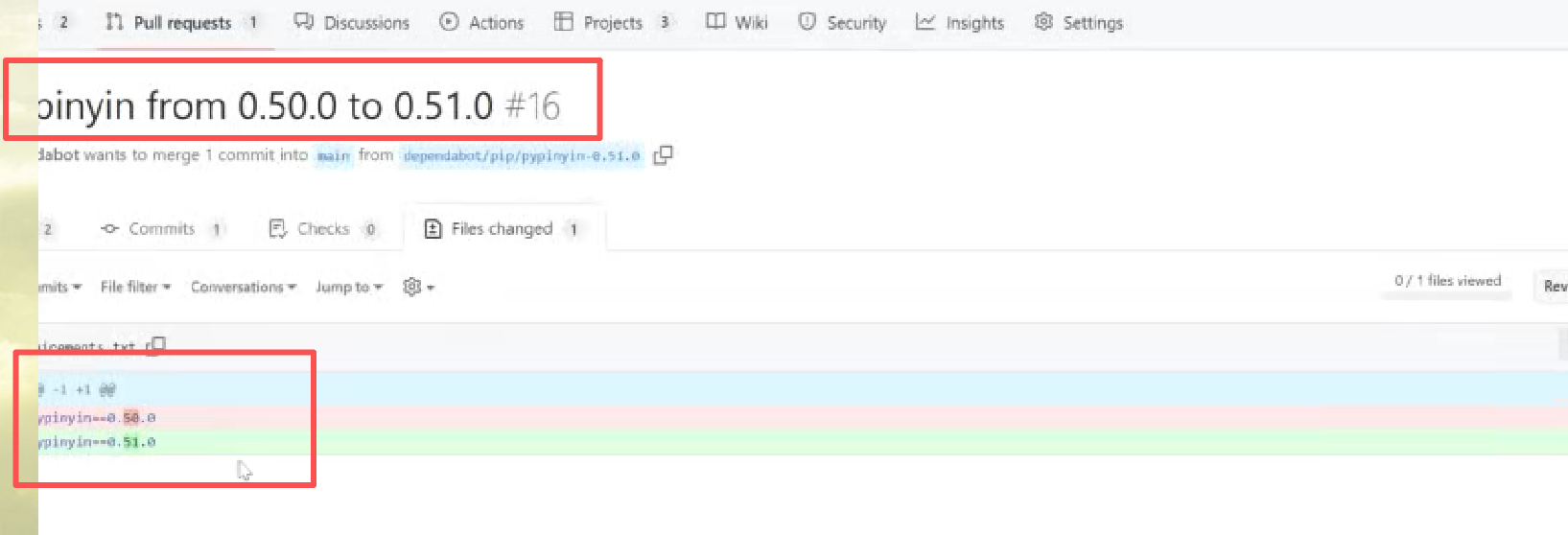
把这个文件提交一下，还是切换到dependabot，它已经识别出了我的项目依赖文件，requirements.txt，右边有一个黄色的按钮在转，就是代表他正在对我的项目依赖进行扫描。如果发现有新的项目依赖，他就会自动创建一个PullRequest帮我更新。

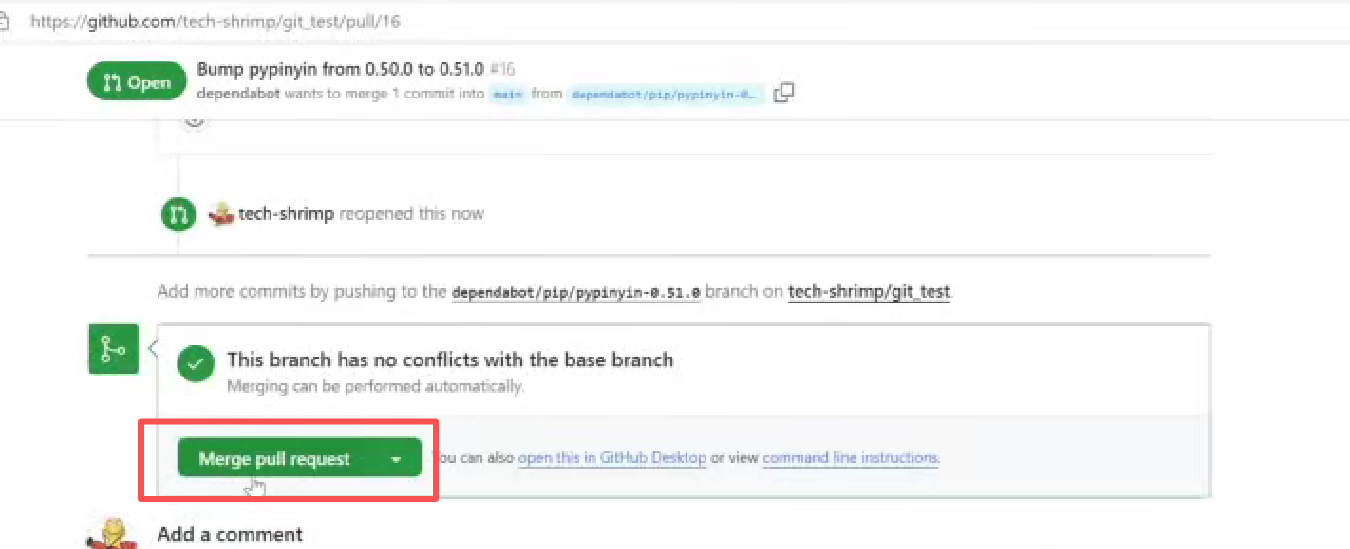


扫描结束后，我们可以看到这里多了一个PullRequest

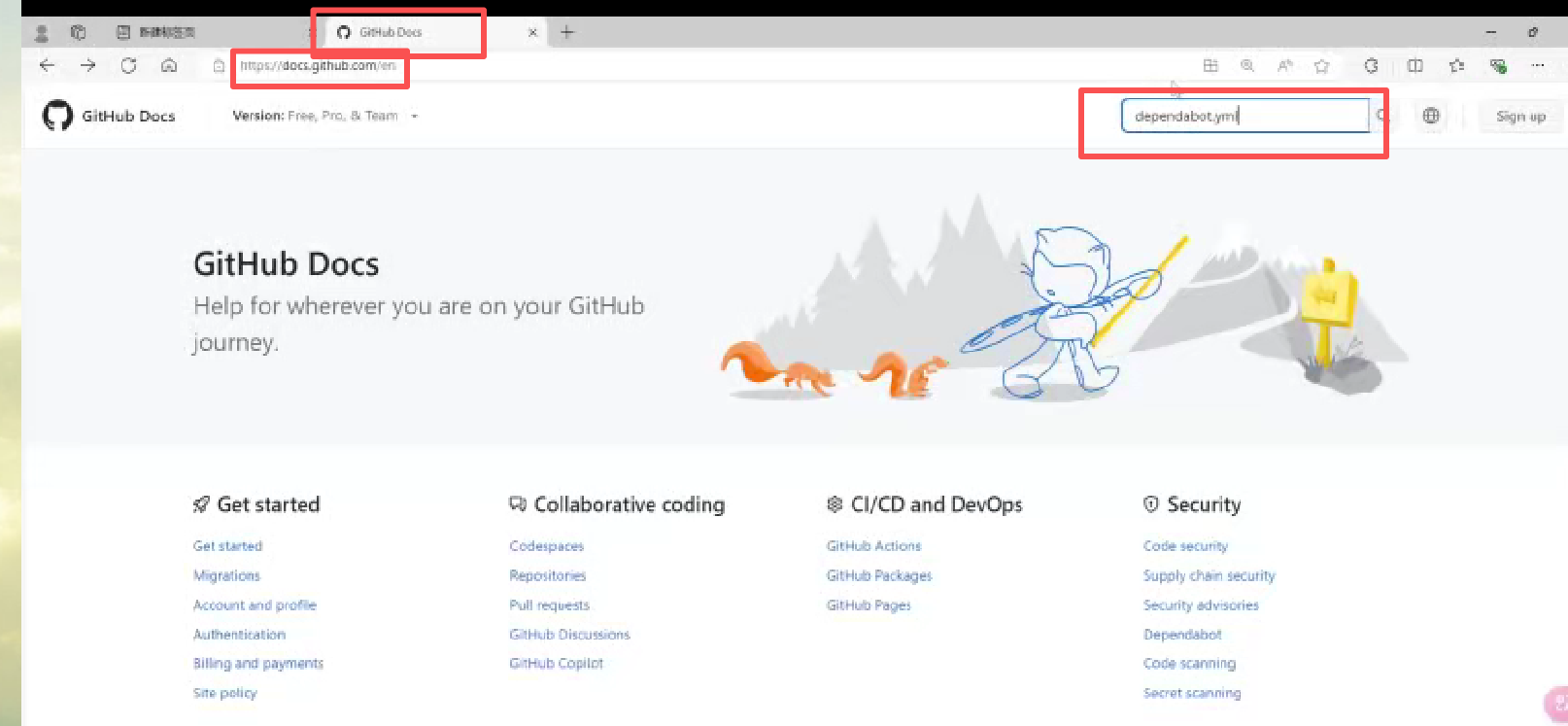


点进去，这个就是机器人为我们创建的Pull Request，标题就是把这个库的版本从0.50.0更新到0.51.0

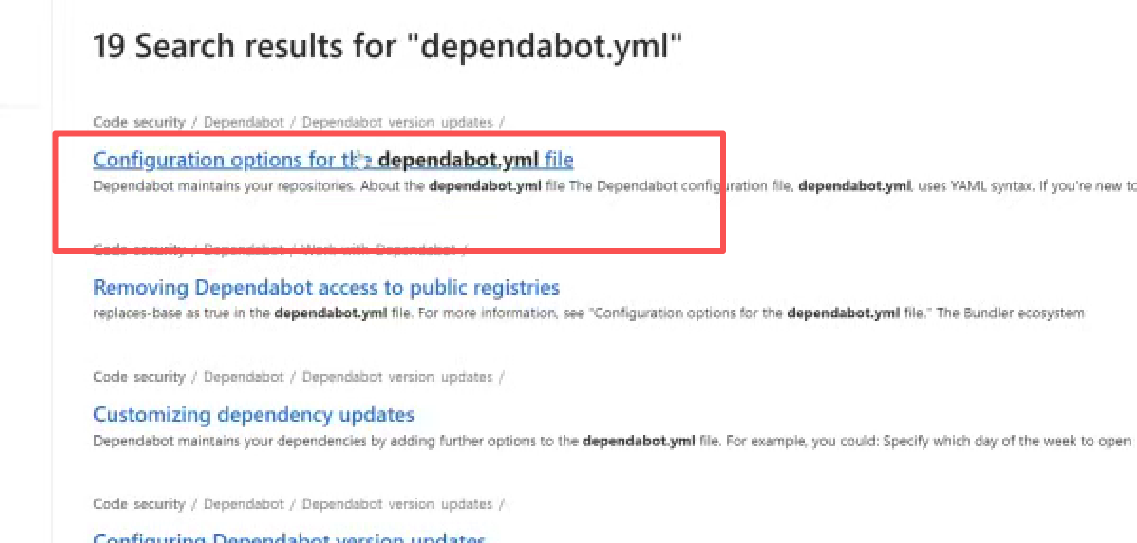




可以看到他进行了文件的修改，帮我这么更新一下，如果您觉得这个更新有用，就直接在下面点击合并。这个扫描跟PR都是自动进行的。



如果想了解DependaBot的更多功能，我们进入这个网址docs.github.com，进来以后我们搜索dependabot.yml



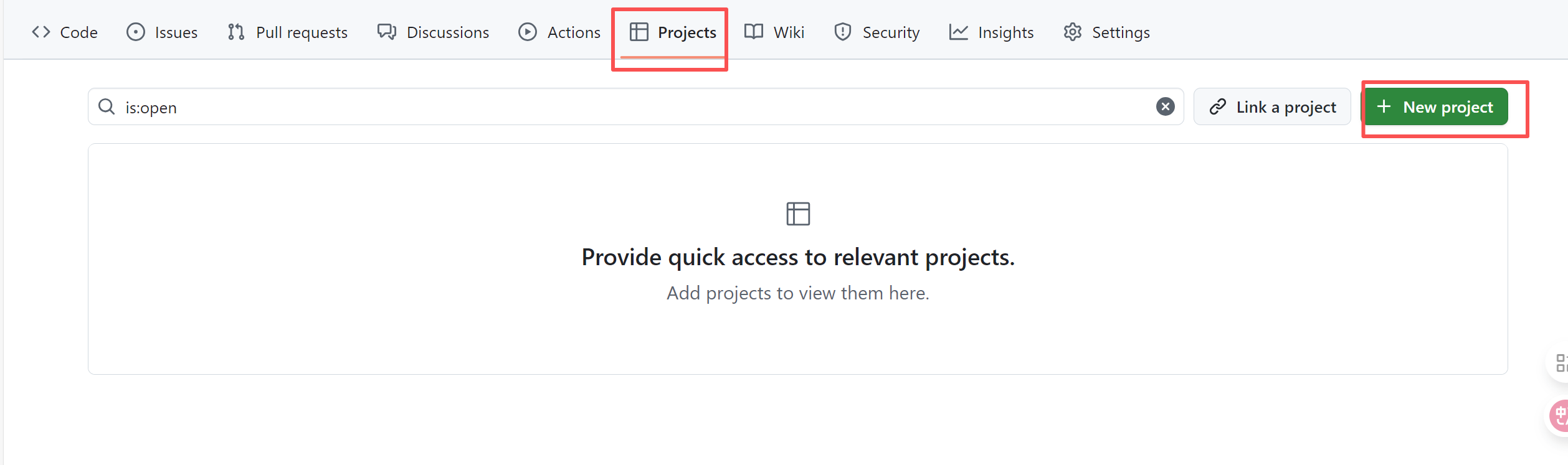
这里有一个文档教你如何配置DependaBot.yml。文档的内容也非常多，主要的几个第一个是package-ecosystem，就是你的项目包管理系统用的哪个，这里是所有支持的项目包管理系统，比如Python就是pip，Java可以用Maven或者Gradle，前端的话npm。第二个，第三个上面也讲了。

额外讲的是这个group, 对更新的规则进行分组

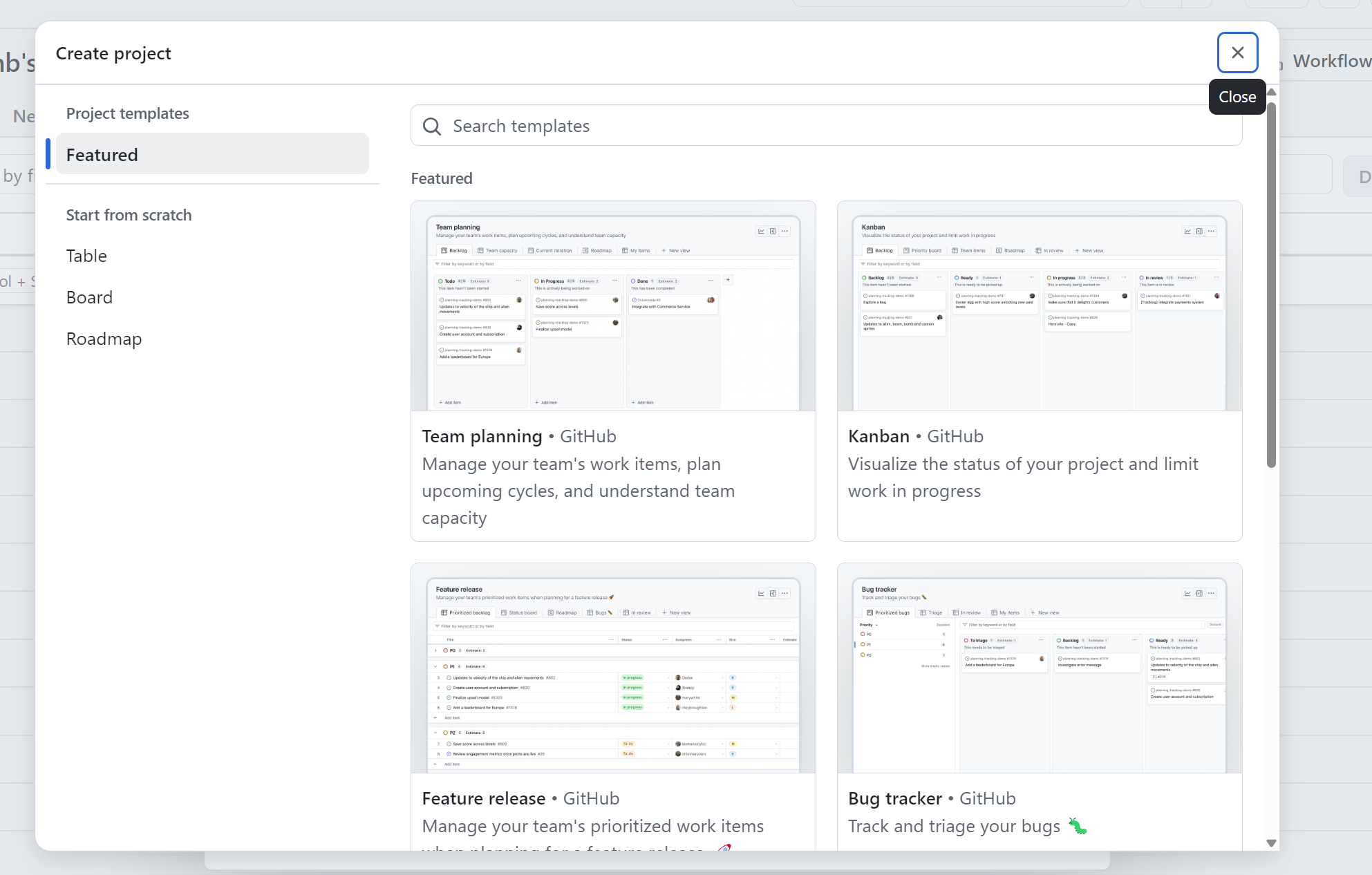
再一个我想讲的配置是这个ignore，指定哪些版本更新是被忽略的

# Github project

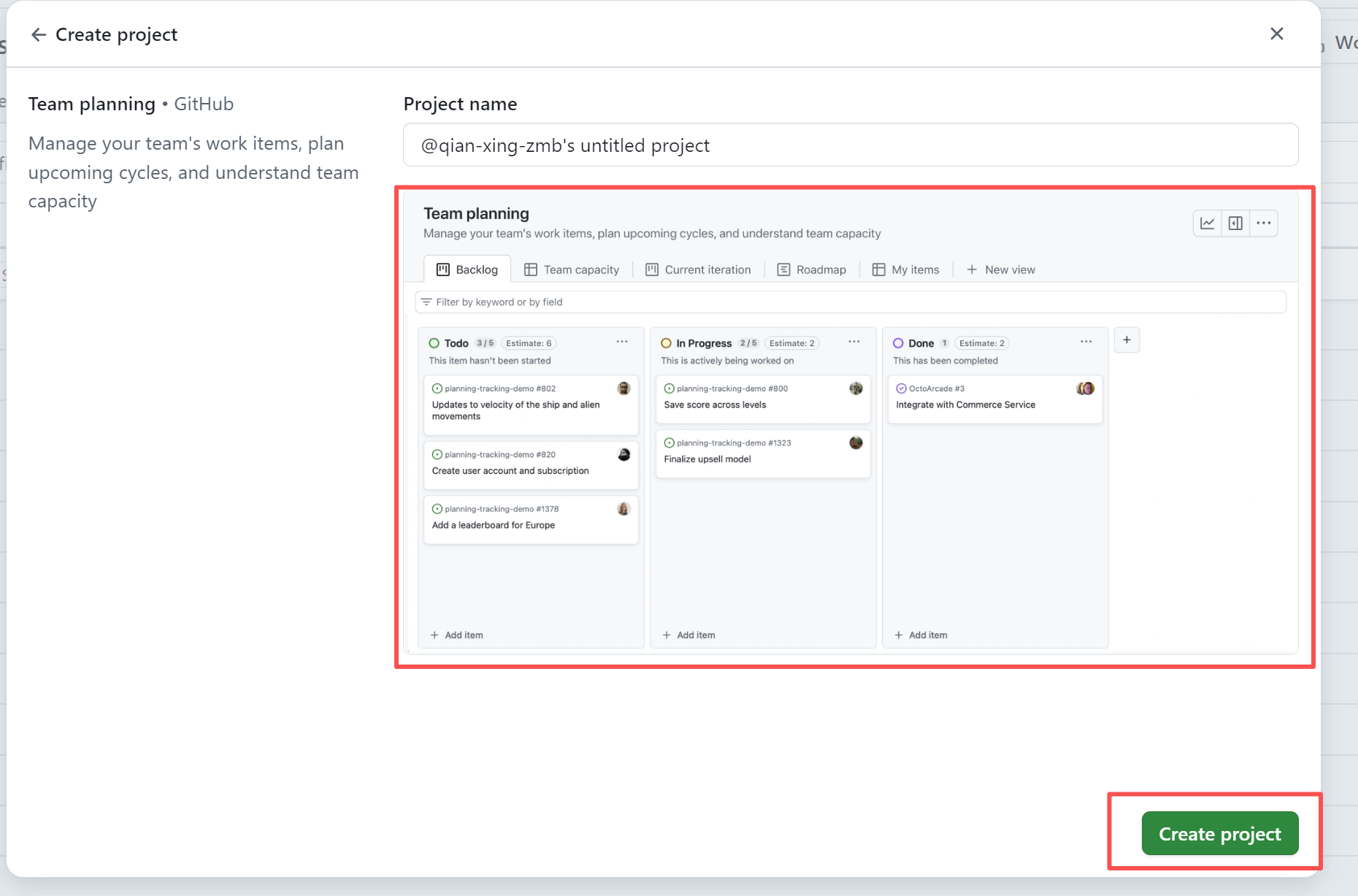
Github project就是项目看板，是一种可视化管理工具，用于组织和跟踪项目的进度任务。



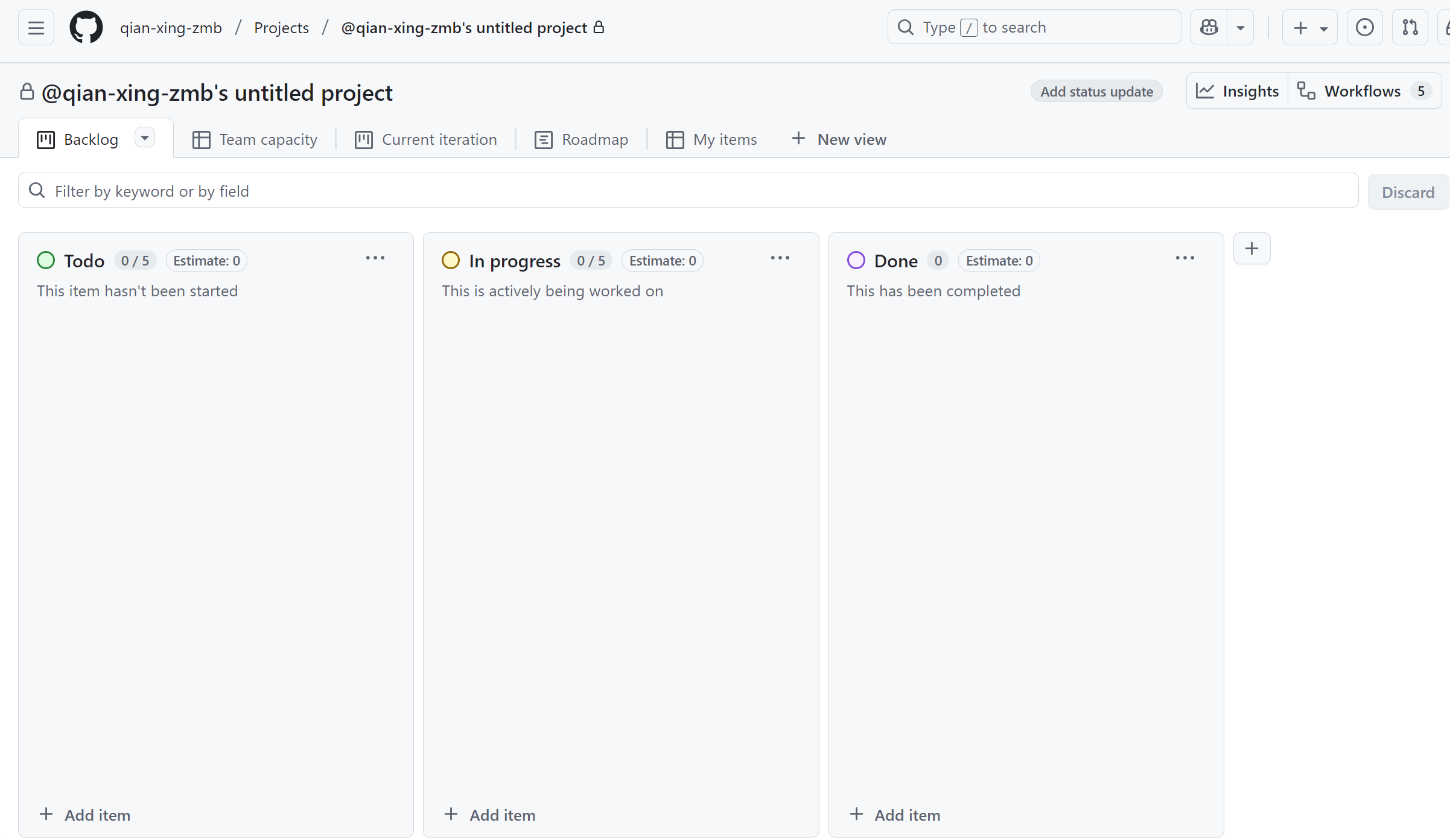
我们可以创建一个新的project，我们这里点击new project



这里可以选择一种模板



点击create project

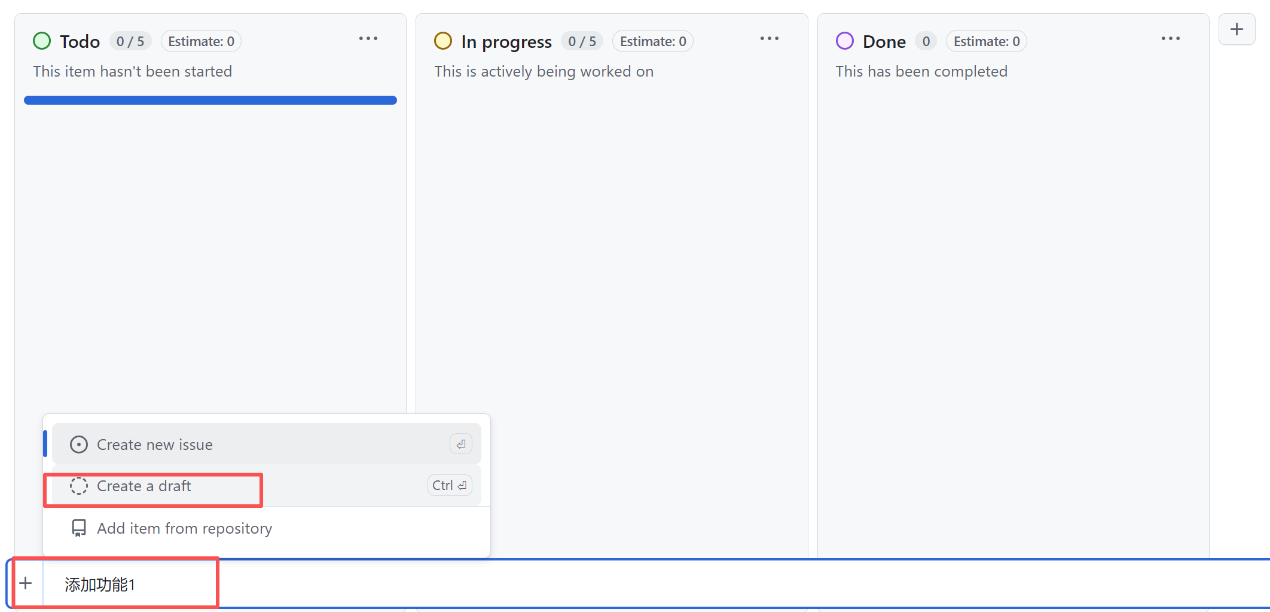


看板就创建出来，这里的任务通常以卡片的形式，放置在了不同的列中，比如Todo是代办事项，In Progress是进行中，Done就是完成。

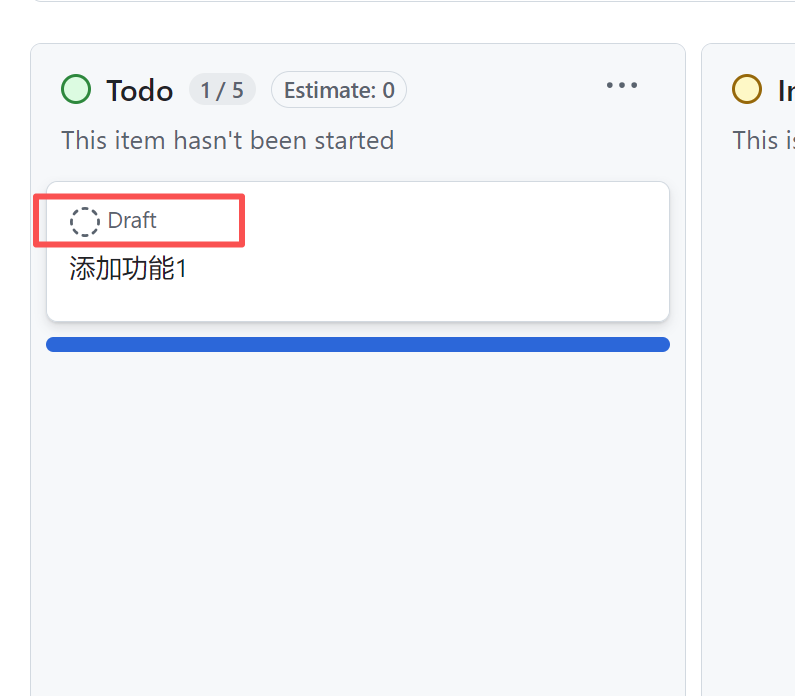
这种视觉化的方式能让团队更清楚地看到每个任务的状态和进展。



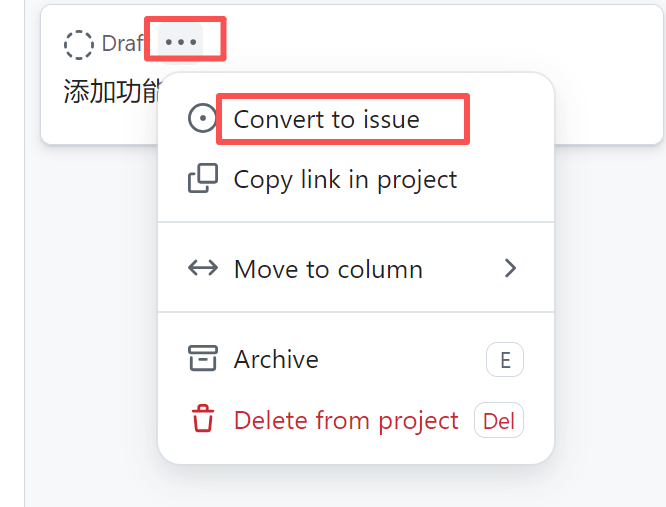
我们试着创建一个，在左下角点击Add ltem，



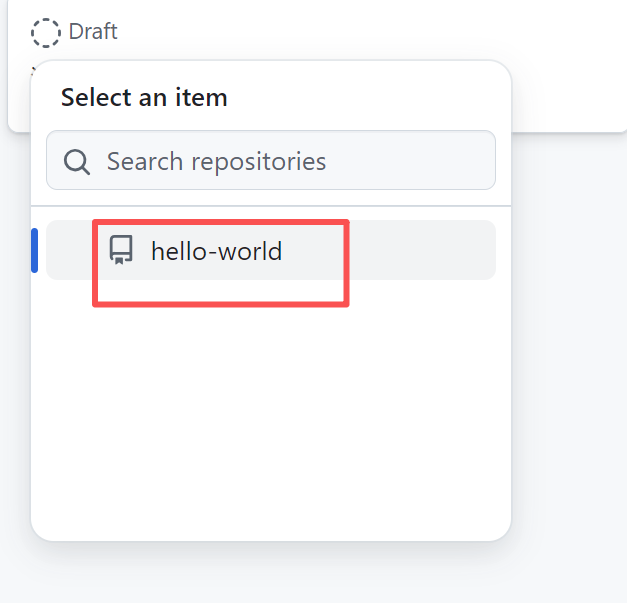
然后输入一个名字，选择Create a draft



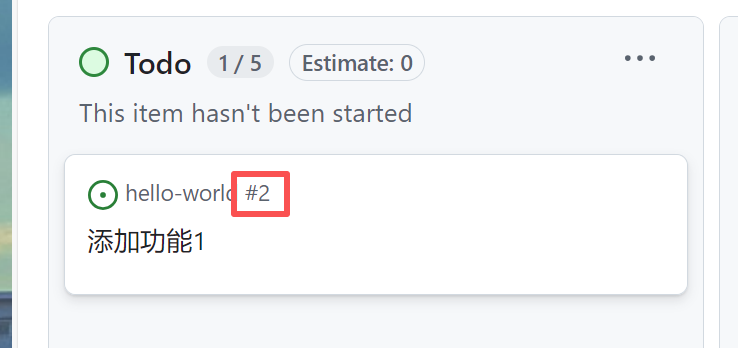
这里显示一个draft的状态，就是还在编辑中，它的意思是还没跟一个lssue关联起来



然后我们点击这三个点，可以把它转换成一个lssue



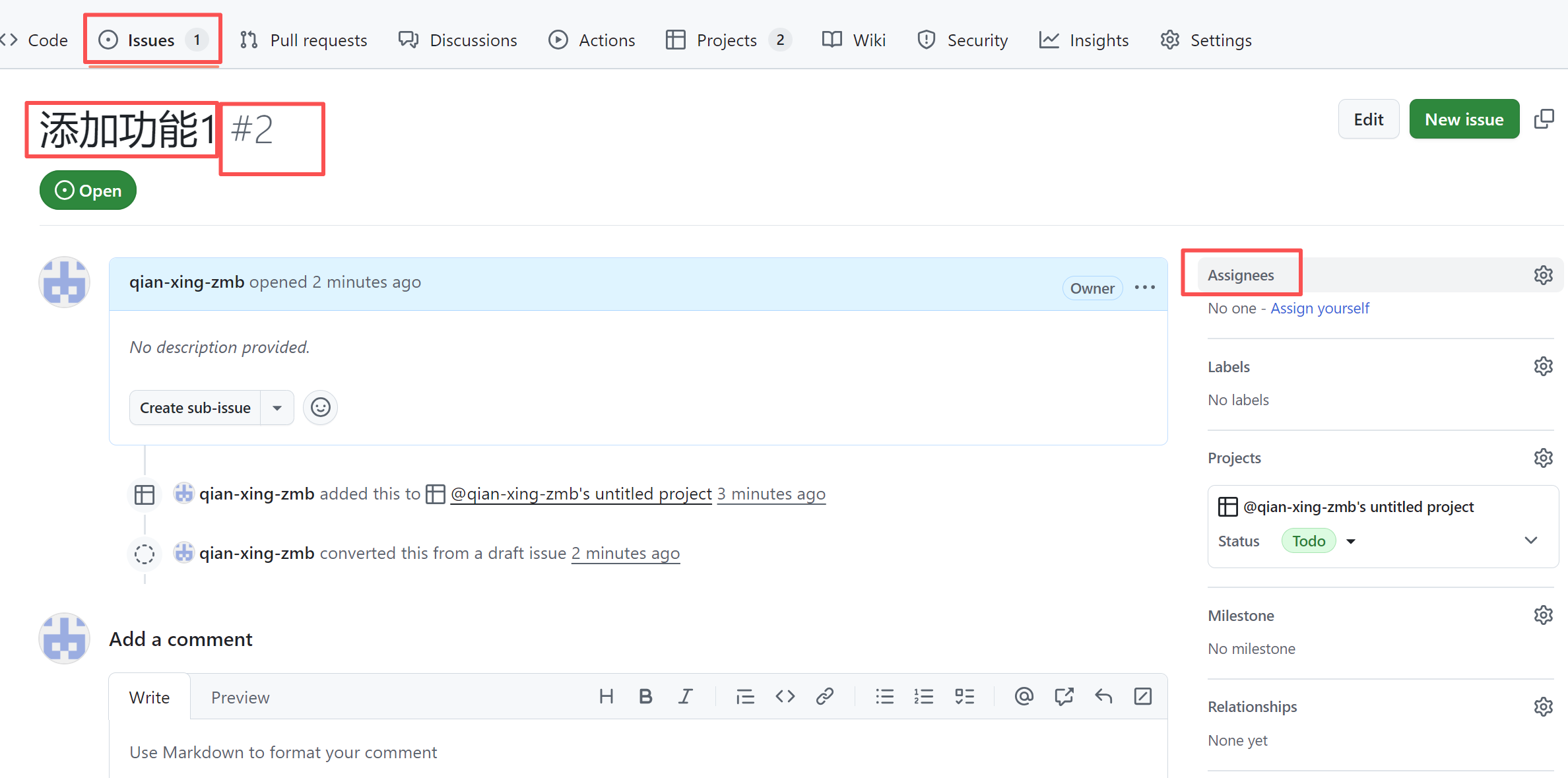
这里我还是选择一个项目



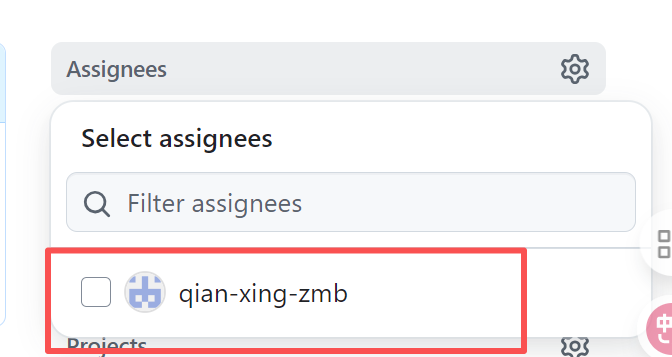
转换成功之后，这里会有一个lssue的号，如#2。可以点进去看一下



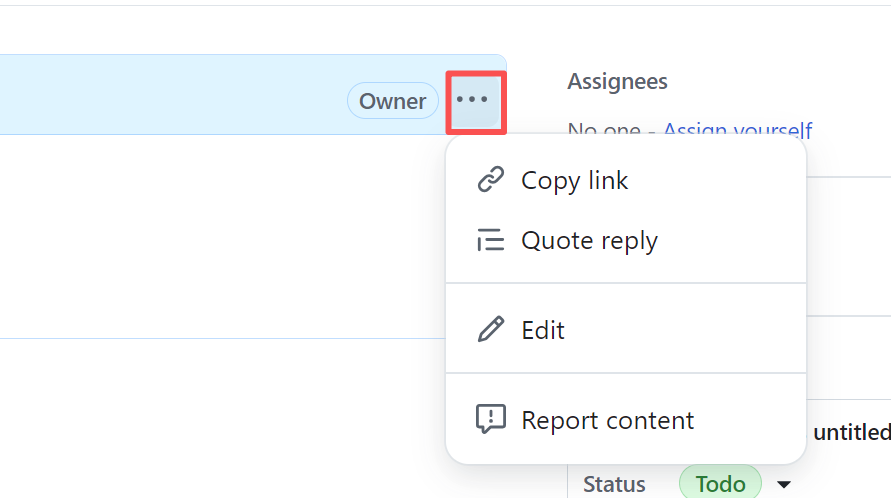
这里点击open in new tab



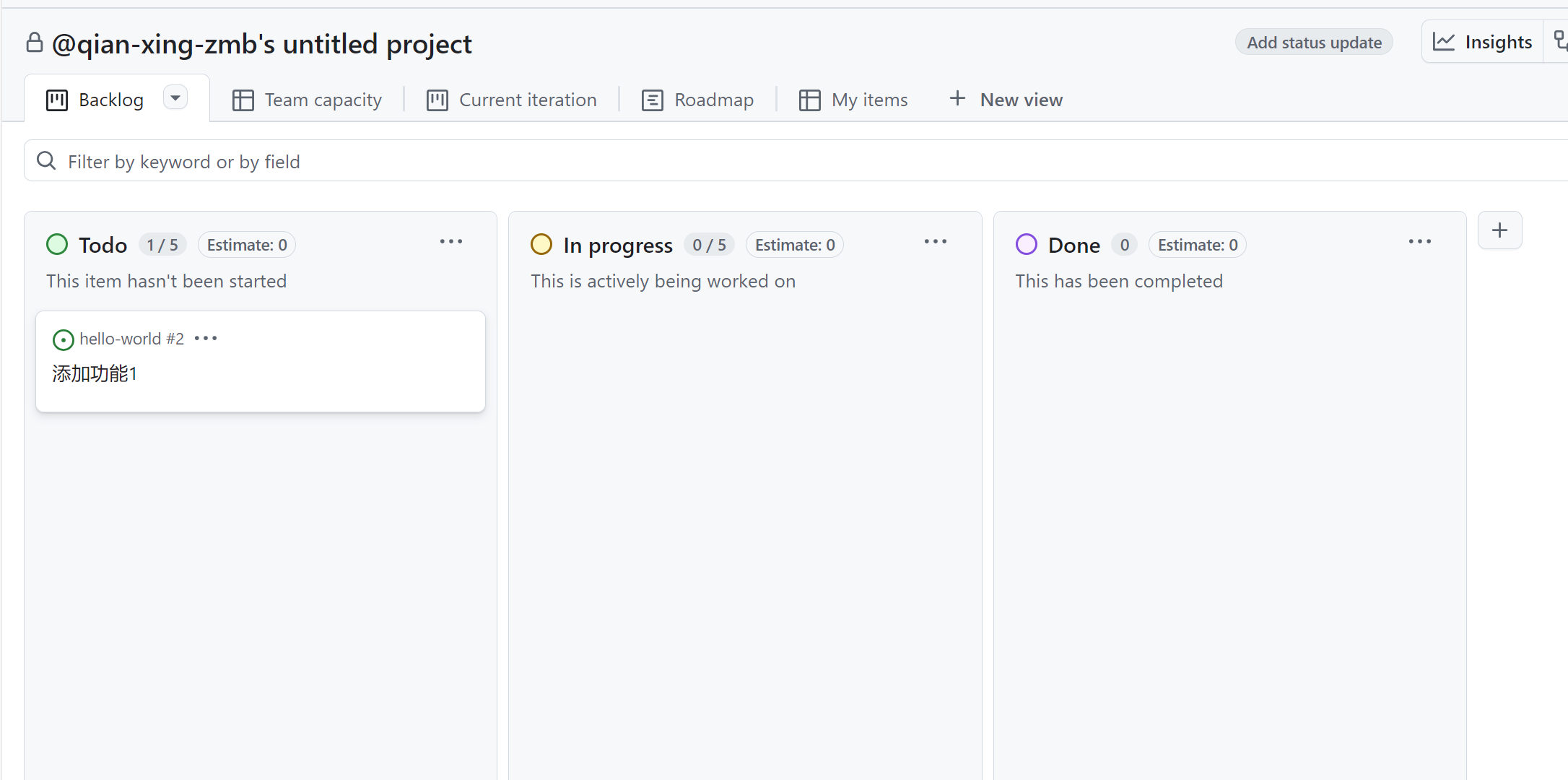
这样我们的issue就创建出来了，这里可以把它分配给某个成员进行开发。



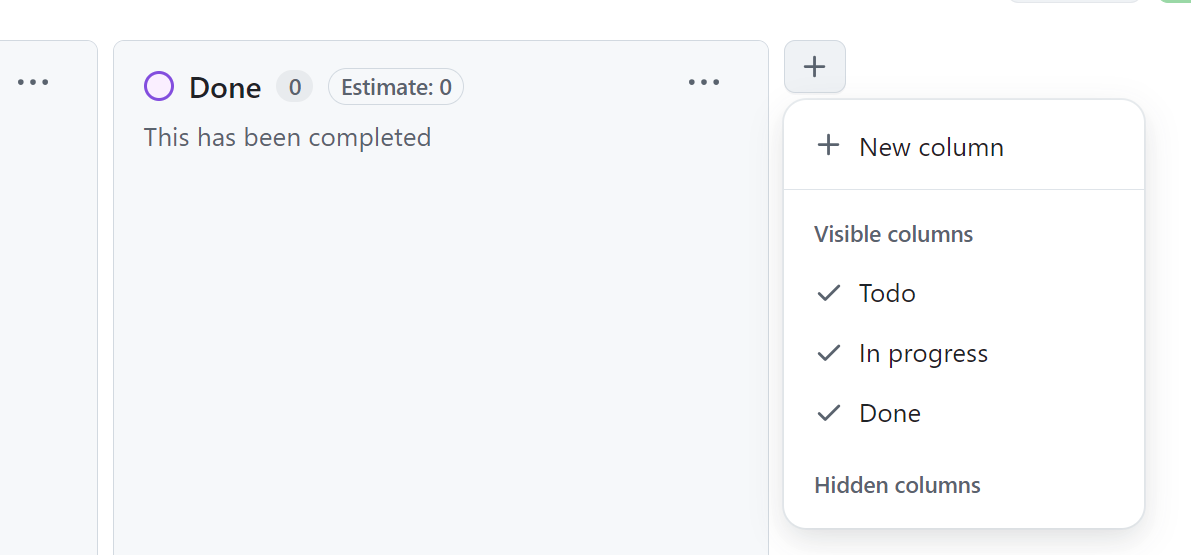
我们右侧找到这个Assignee，选择一个人。



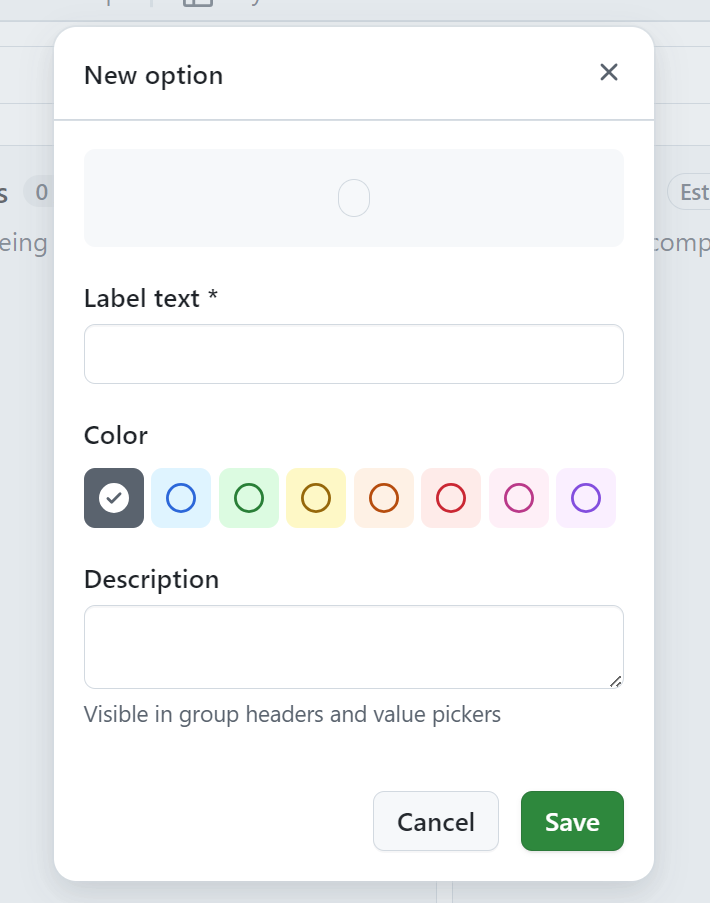
领导还有团队成员可以编辑这个issue的详细描述，添加一些图片，添加附件，并且发表评论等等。这里提供了一个跟团队成员交流的平台。



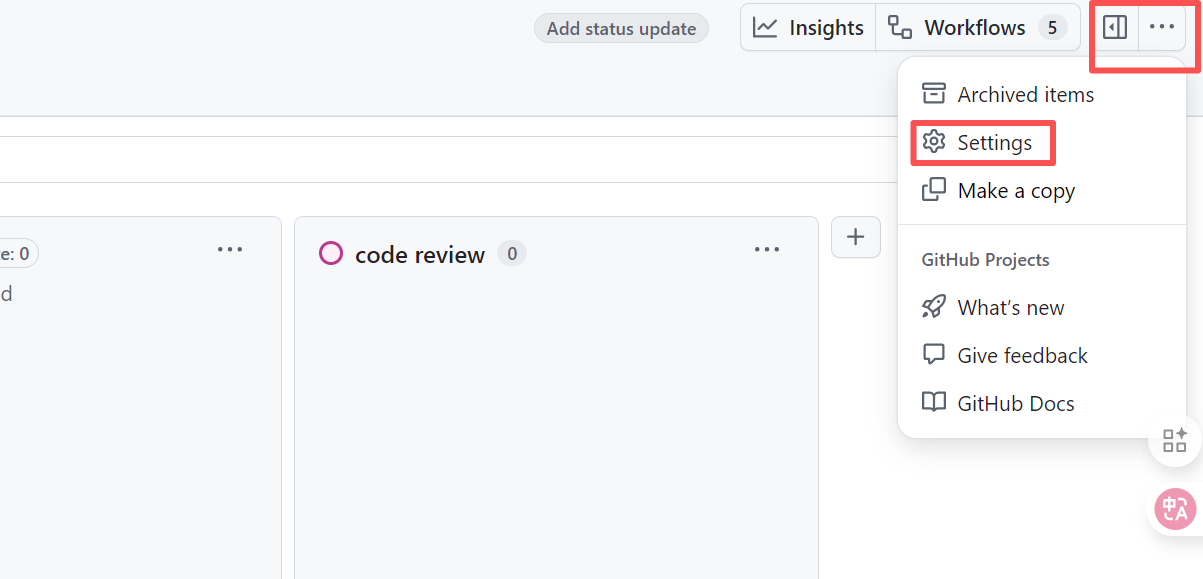
当这个功能开始开发的时候，我们可以把它从Todo这一列，拖拽到in progress里面，这样代表这个功能已经正在开发了，如果完成以后，我们把它拖拽到Done里面。



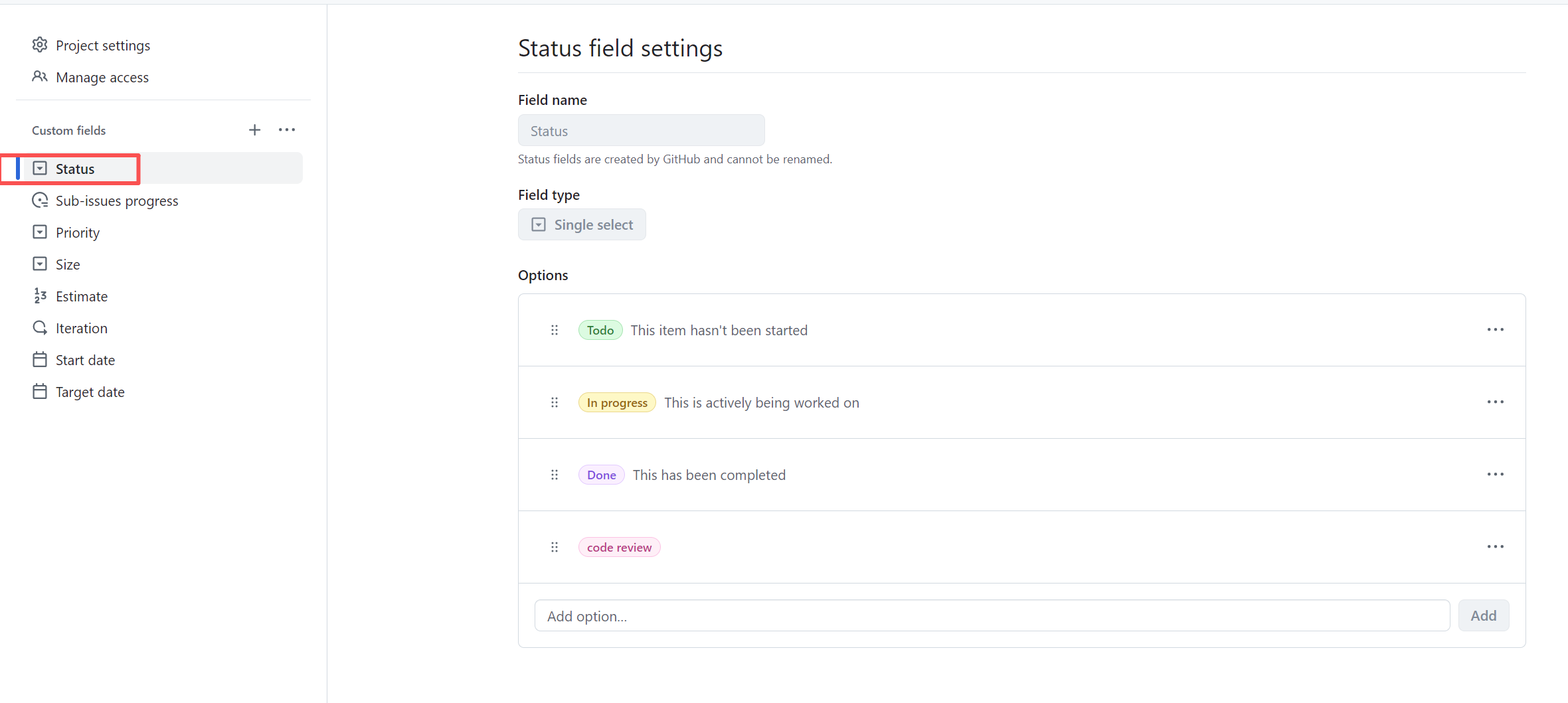
这里还可以添加更多的状态，新加一列



可以编辑

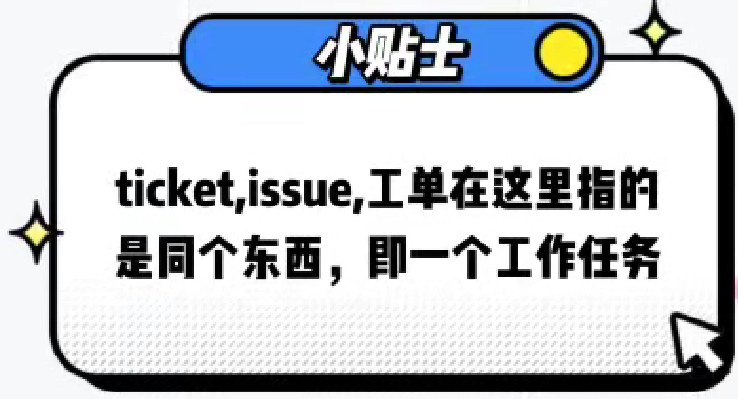


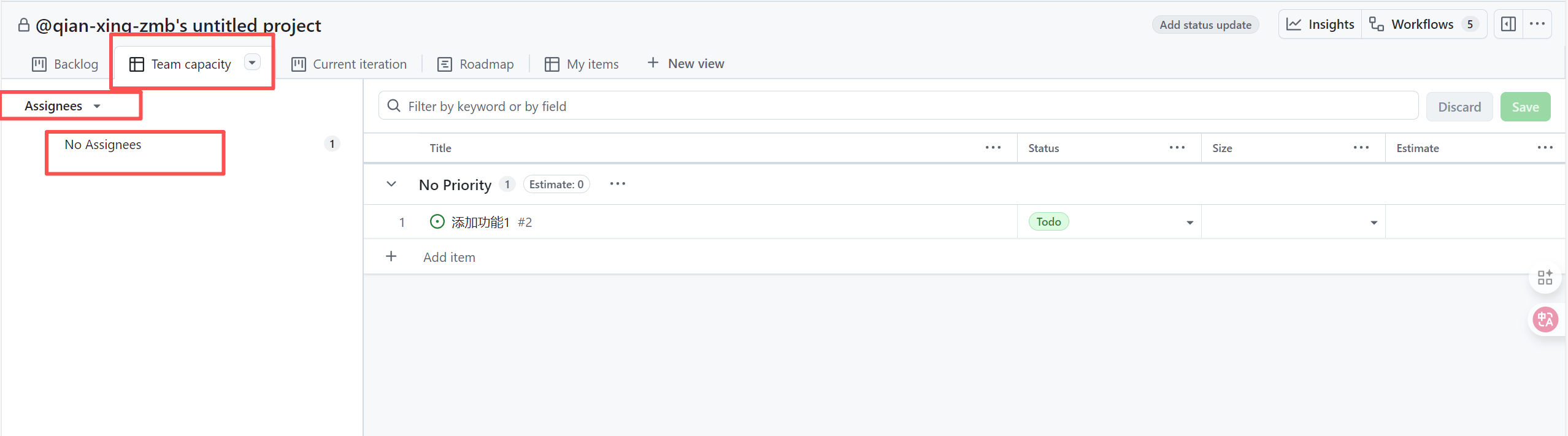
每种状态还可以调整不同的顺序，然后点击settings。



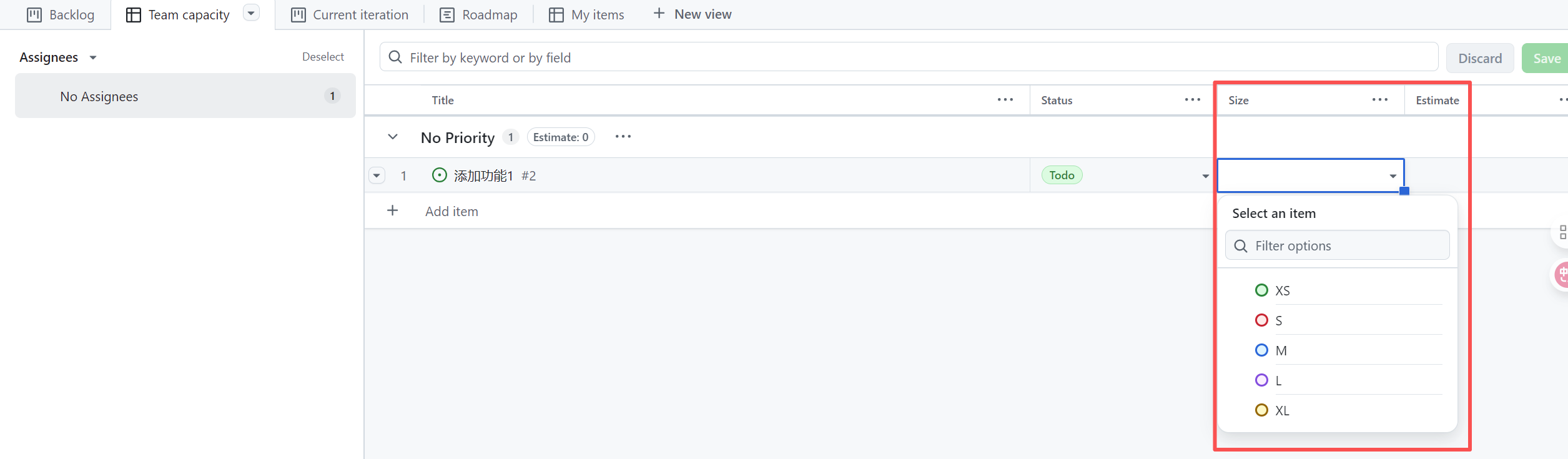
选择这里的status，这里就是我们刚才创建的几个status，我们调一下顺序，颜色。

每个ticket的进行到什么状态，我们直接把它拖拽过来就可以了啊

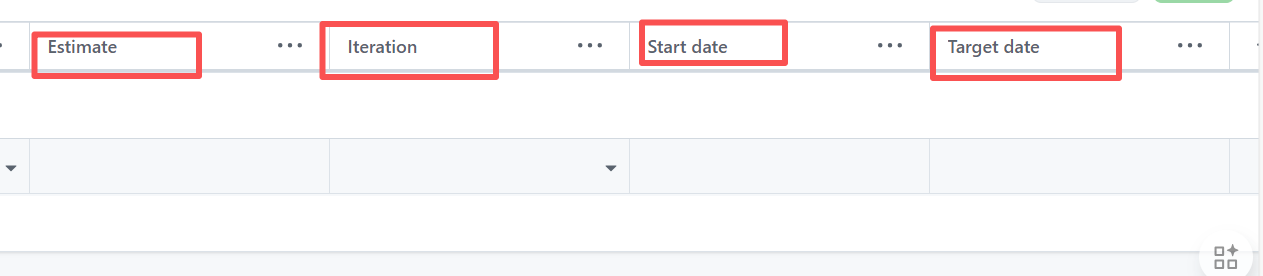




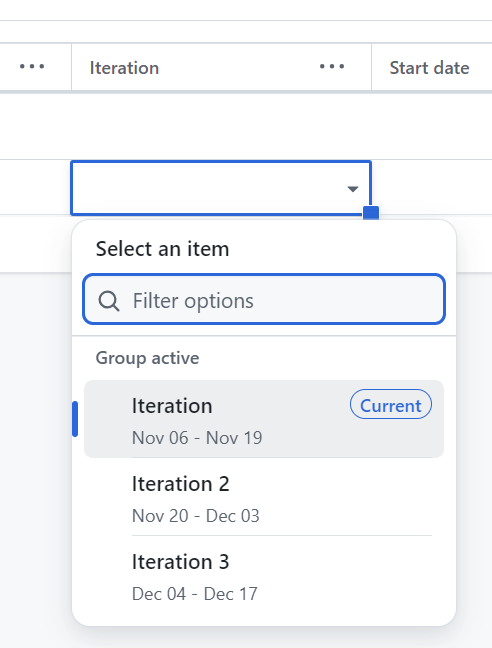
第二个选项卡叫做team capacity，这里的意思是团队的工作量，好我们先看左边的开发人员。可以看到有一个issue，处于todo状态



我们可以定义这个issue的大小，每种状态代表了不同的工作量，比如最小的可能需要半天，然后1天 3天5天10天



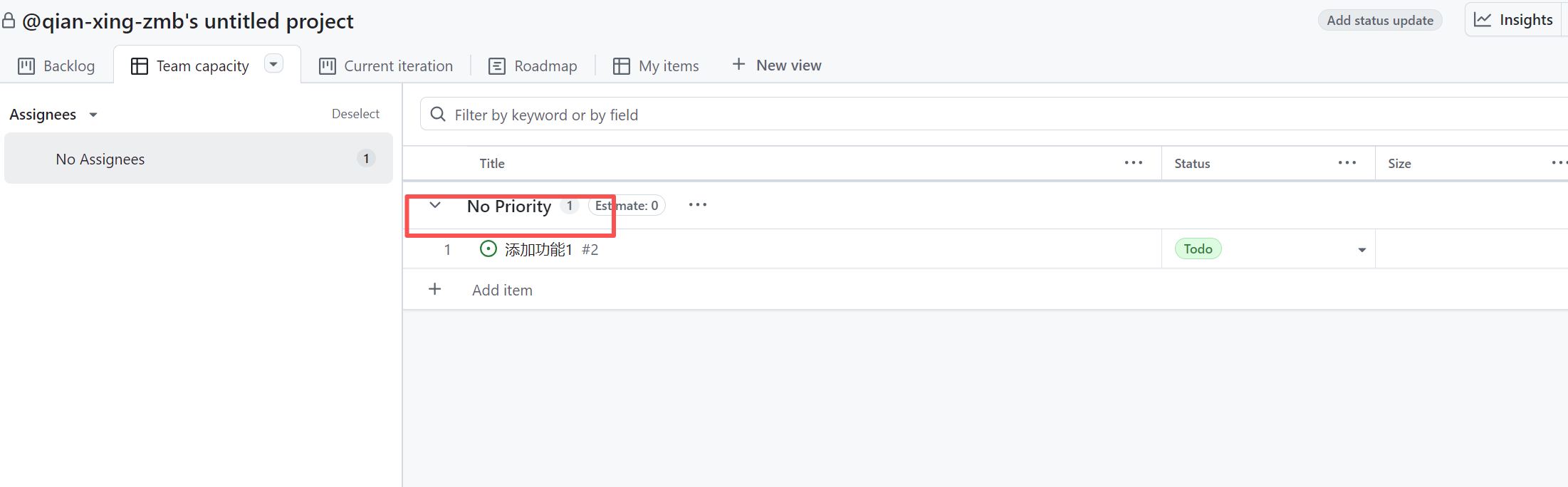
estimate是大约需要多少天才能完成，



下面是iteration是敏捷开发里的选代的概念，或者叫做Sprint，可以选择这个功能应该哪个时间点做完，就是给这个功能选择一个选代

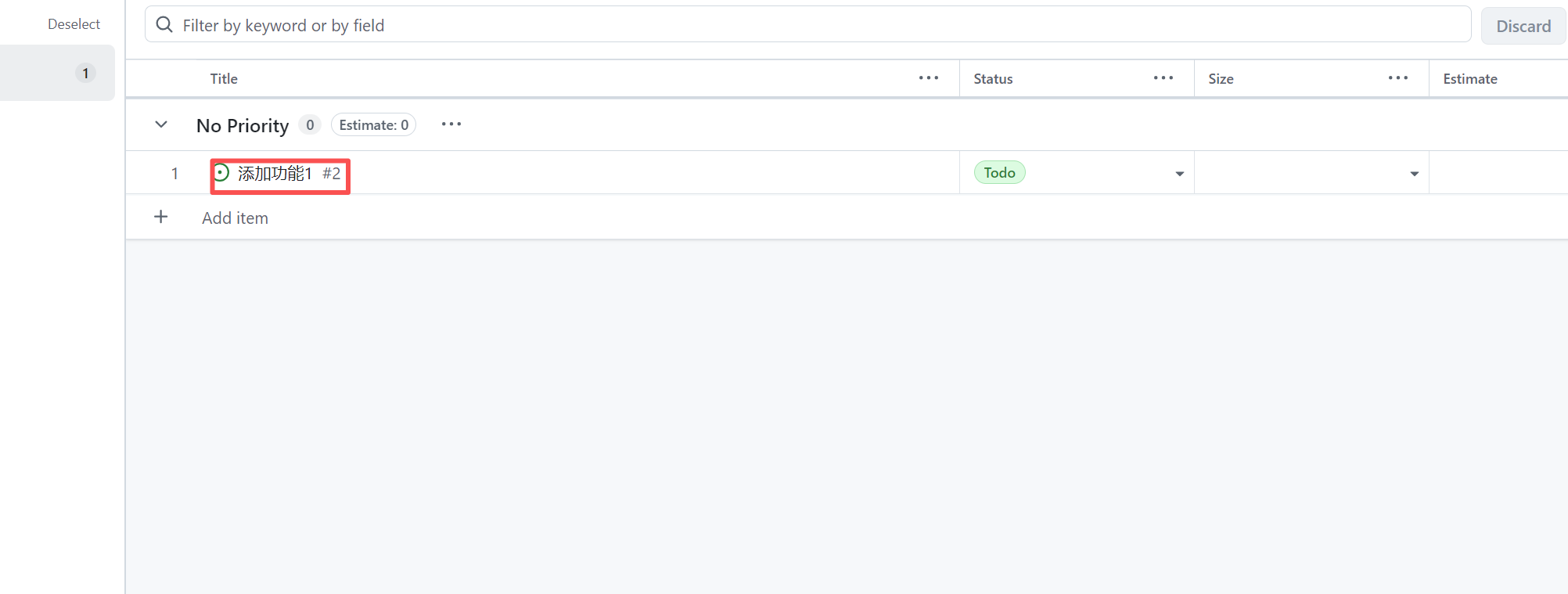
Start date是开发者可以在这里填一个开始时间跟结束时间

这些也可以在settings中添加每一项的其他选项。

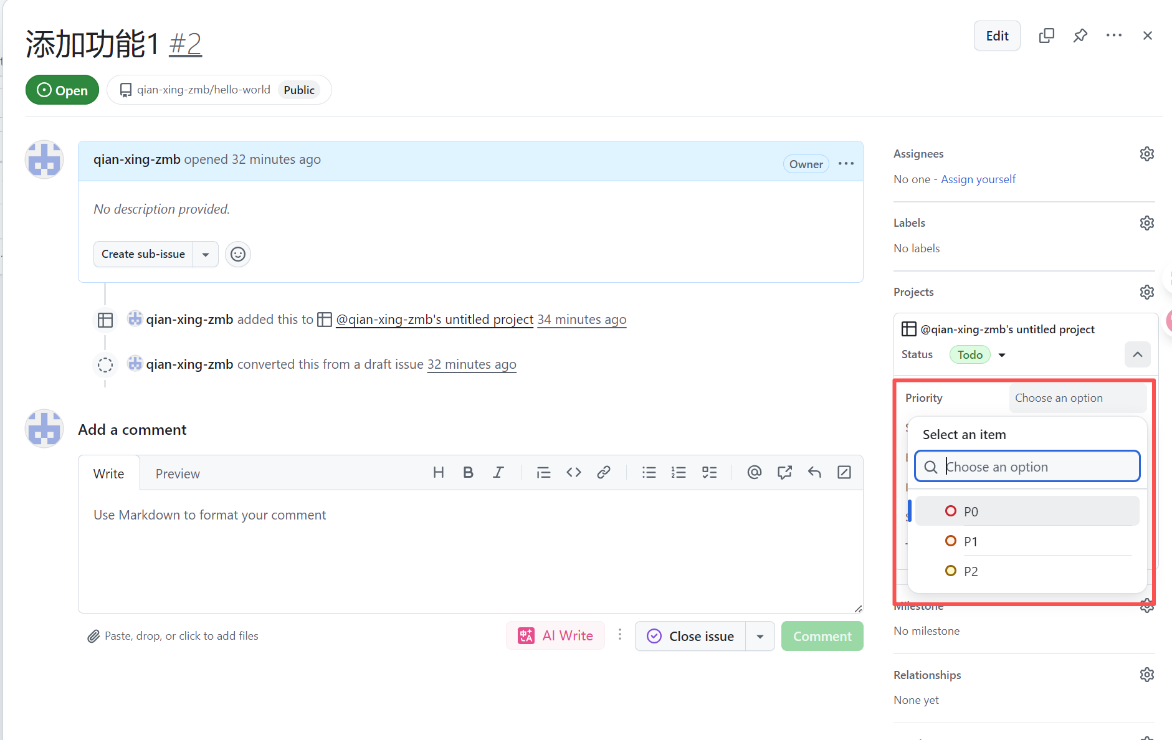


左侧根据团队成员进行切换

同时还有一个细节，给他们分配的工单都是no priority的，就是没有优先级。我们可以修改这些工单，给他们设上一个优先级

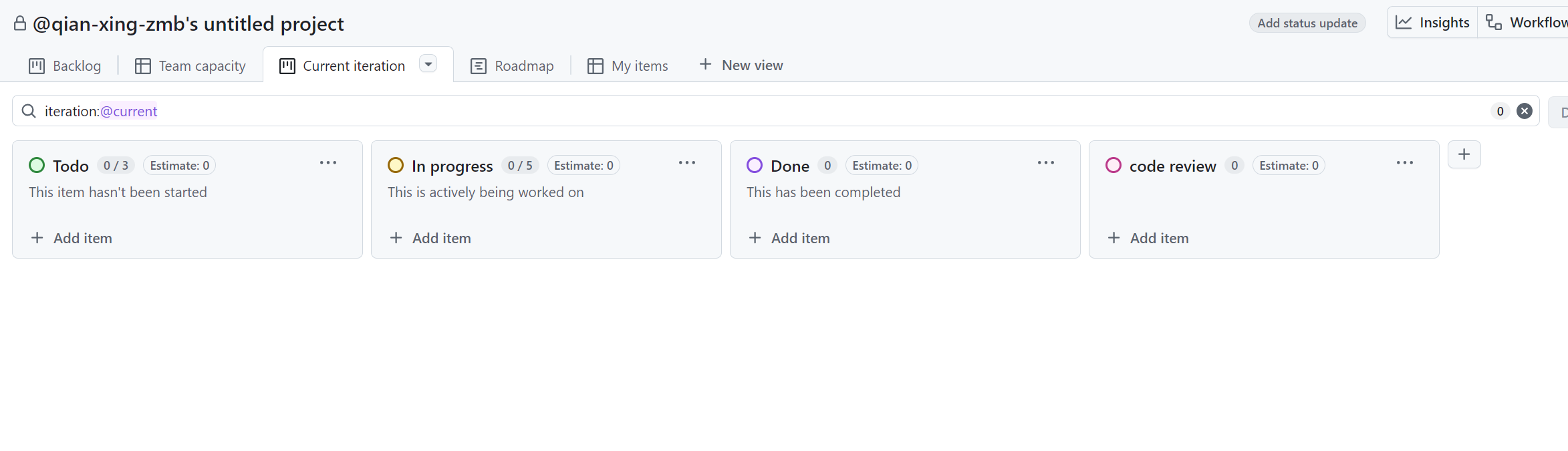


点击项目

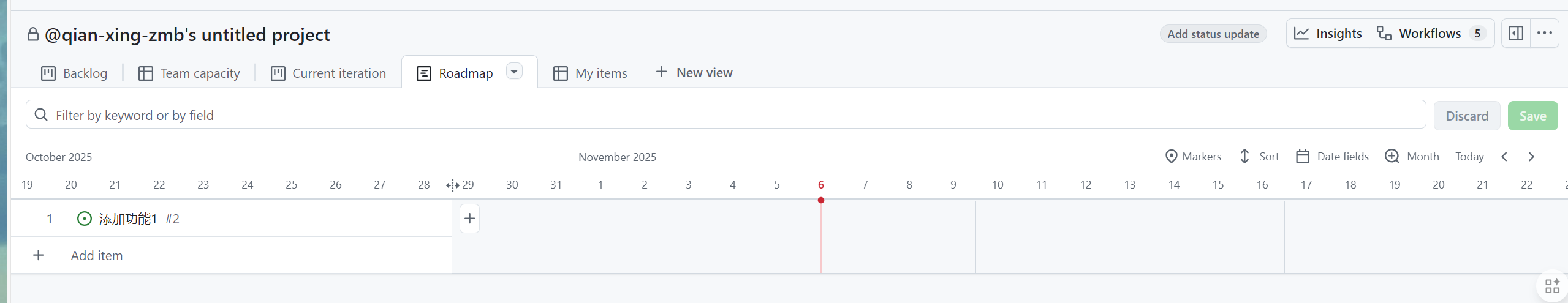


右侧找到这里的priority，这里可以给他们设一个。

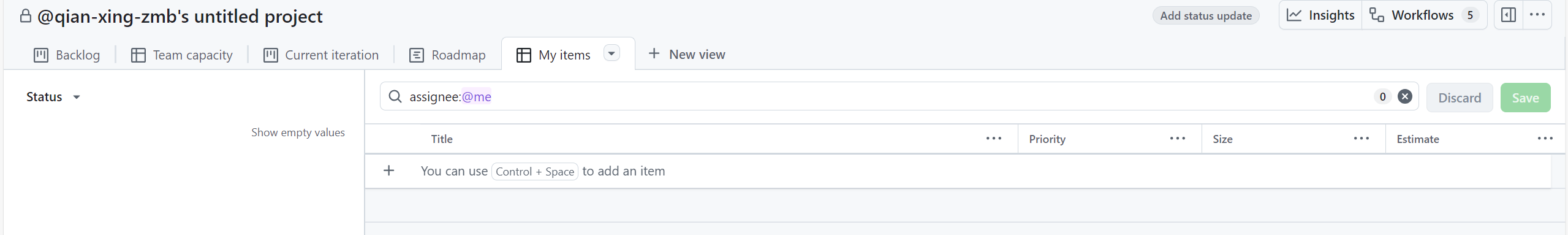
我们可以根据优先级对工单进行分组，这样每个人手里有多少个工单，各自是什么优先级，什么状态，在这个表格里可以说是一目了然



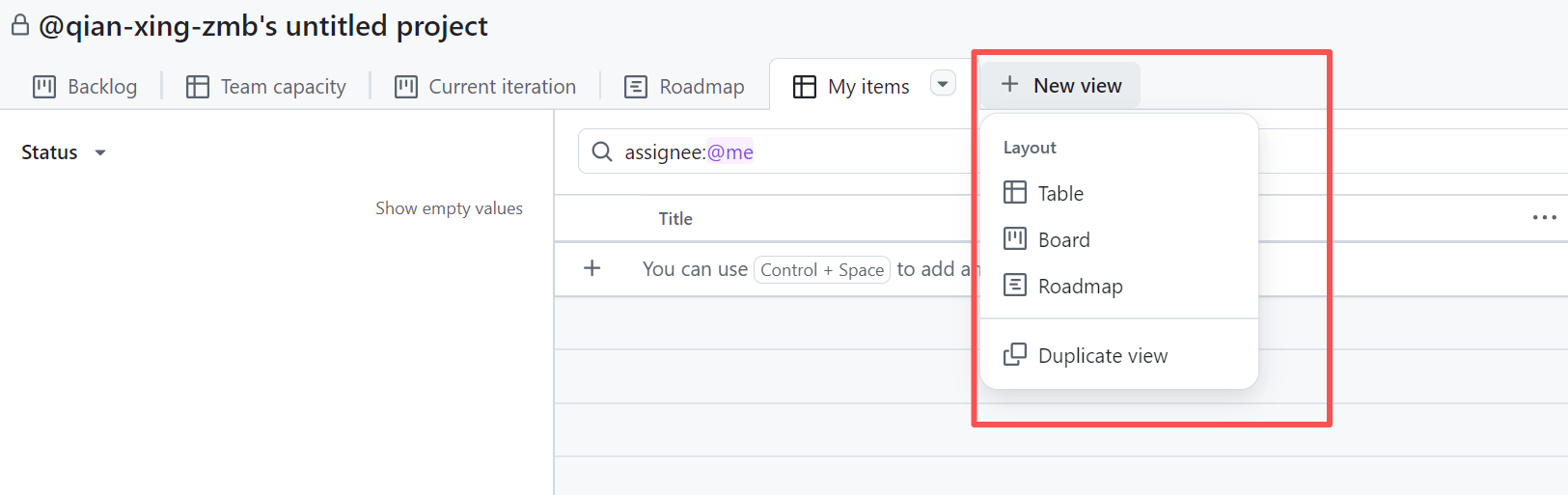
下一个选项卡是current iteration是处于当前选代的工单的状态。迭代其实是敏捷开发里面的一个专用术语，一般两个星期是一个选代，每个选代上线一批功能，然后经过QA测试，然后推上线。也可以自己定义迭代的时间周期等等的。也都是在这个右上角的设置里面调节。



roadmap这是一个时序图。如果某些功能之间，有前后顺序的依赖差异，可以通过时序图进行查看。

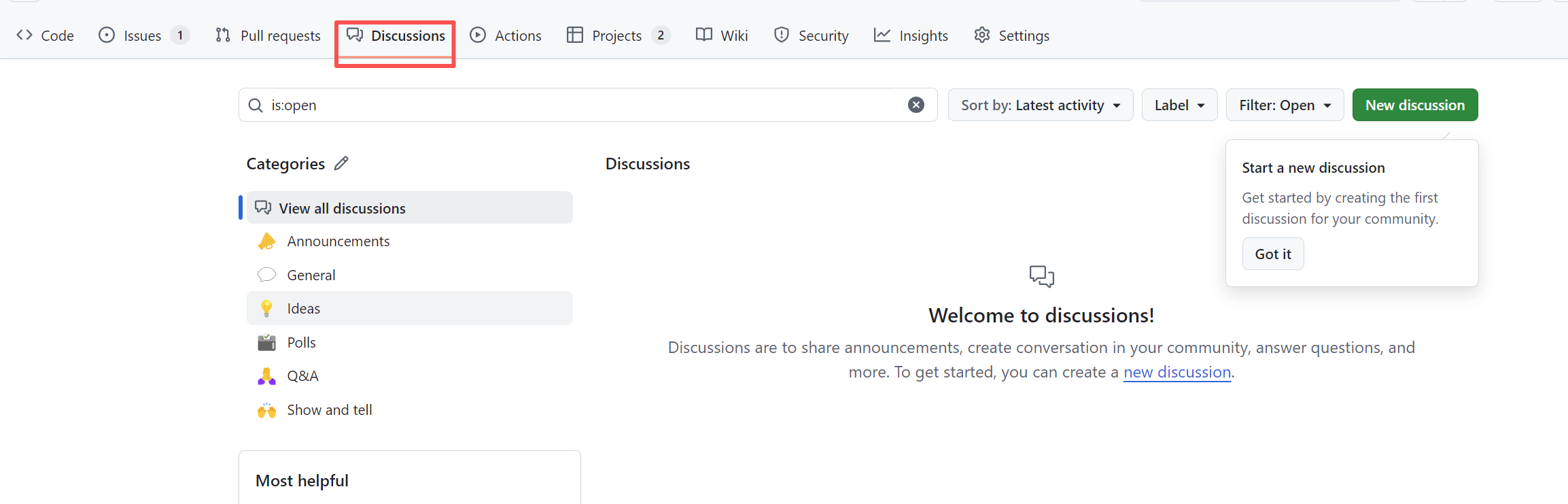


这里的myitems也很简单，就是查看我自己手里的工单状态



还可以添加新的功能，可以添加表格看板，还有时序图，可以进行自己的自定义，从这里可以看出project是一套完善的项目管理工具。

# Github Discussions

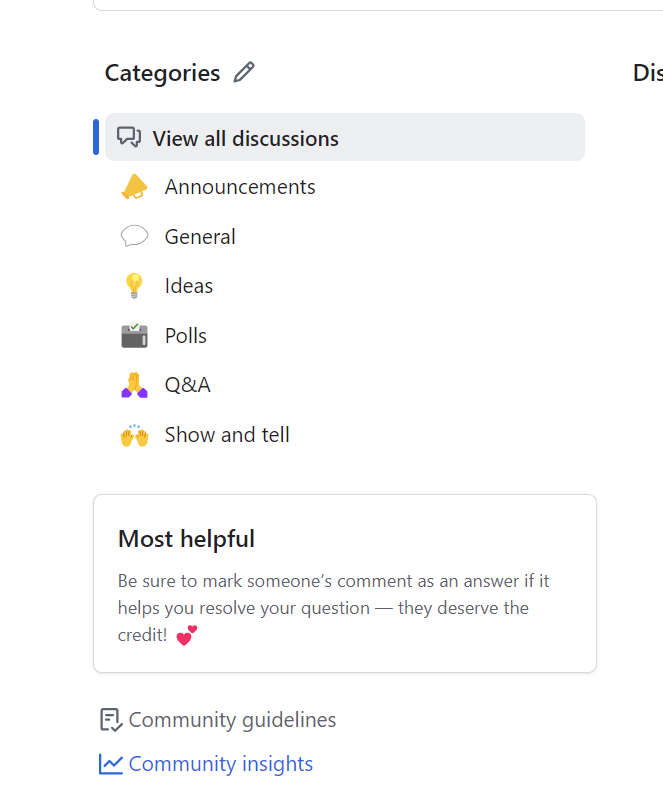


这个就是项目的discussion，就是一个围绕这个项目的论坛

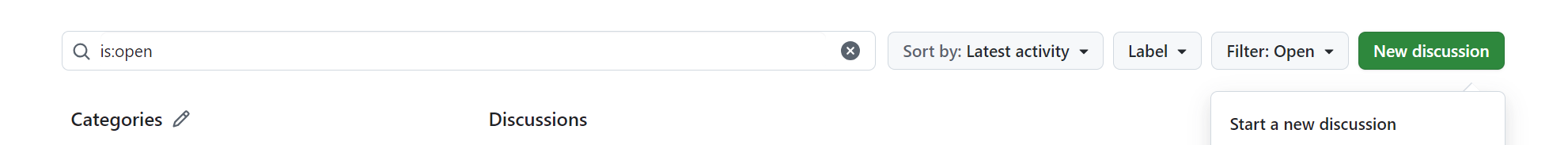
这个功能有点类似于传统的论坛，就是社区里的人们在这里讨论问题，然后可以在这里创建帖子，发表自己的意见，还有询问的问题等等。

discussion一般是比较大的项目才会启用，一般的小型项目使用issue就足够了

在这个论坛里我们可以发帖，点击这个new discussion就可以发帖，也可以回帖（找到一个最近的帖子，在下面发表你的评论）



这里还可以按照主题，对帖子进行分类查看



可以搜索，可以排序，可以按照标签筛选