

IV 管理并参与大型项目

本部分包括以下内容：

- (1) 在 **GitHub.com** 上找到开源软件（OSS）；
- (2) 发现为 OSS 贡献的有效策略；
- (3) 阅读并理解大型项目的贡献指南；
- (4) 设置一个仓库，以便为 OSS 项目做好准备。
- (5) 在 OSS 中发现对大型私人项目有用的策略；
- (6) 在 **GitHub** 上创建一个团队，以便更好地管理内部源代码项目。

第 9 章：探索和贡献 OSS

本章学习重点：

- (1) 探索 **GitHub** 上的 OSS 项目；
- (2) 为 OSS 做贡献；
- (3) 贡献准则；

对于开发者来说，只要开发者知道去哪里寻找代码，**GitHub** 就是有史以来最伟大的宝藏。每个可能的需求都会有一个对应的仓库。这种丰富的选择一开始可能会让人不知所措。

除利用开源软件（Open Source Software, OSS）之外，为开源软件做贡献也是继续成长为一名软件开发者的绝佳途径。它让你有机会使用在日常工作中可能不会使用的技术。它将与一个致力于解决各种挑战的庞大的开发者社区联系起来。这些开发者中的许多人很乐意分享他们的知识给希望学习新知识的人。

在本章中，我们将探讨如何探索 **GitHub** 所提供的各种开源软件。我们不仅探讨了如何发现你可能想在自己的项目中使用的软件库，而且还提供了寻找你可能想贡献的软件库的技巧。

9.1 探索 **GitHub**

GitHub 的 Explore（探索）页面（见 <https://github.com/explore>）是发现符合你兴趣的仓库的一个很好的起点。它包含人工和算法策划的内容。图 9-1 显示了探索页面的一部分。

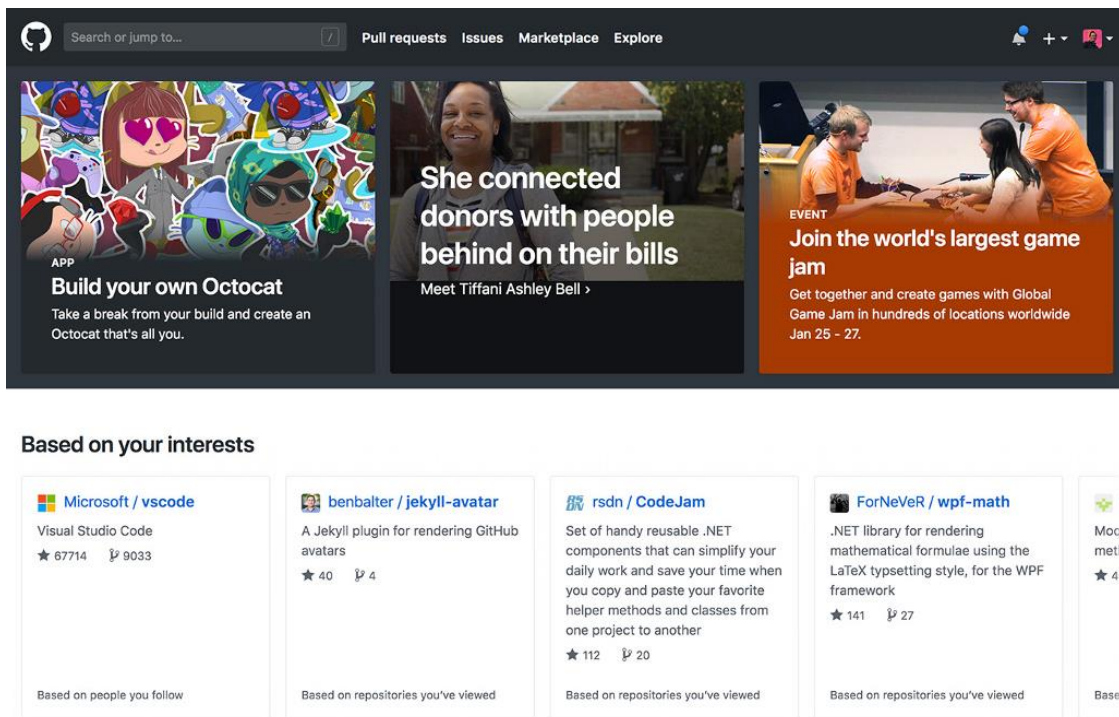


图 9-1 GitHub 上的 Explore（探索）页面

接下来的几节中，我们将描述探索页面的每个部分及其重要性。在必要的时候，我们会介绍你需要做什么来使自己的仓库称为该部分的候选仓库。

9.1.1 探索标题部分

标题（head）部分包含三个特色网站。在撰写本书时，包含以下三个特色网站：

- （1）在 myoctocat.com 上创建你自己的 Octocat 应用程序；
- （2）底特律水利工程（Detroit Water Project）创始人 Tiffany Ashley Bell 的简介；
- （3）关于全球游戏展（The Global Game Jam）的信息。

你可以通过查看此部分去发现 GitHub 认为值得展示的有趣的应用程序、人物和活动。

9.1.2 发现仓库

标题下面的部分展示了根据你的兴趣而选择的仓库。这些仓库是从“discover”页面中提取（见 <https://github.com/discover>），一个通过算法专门为你提供推荐的页面。根据你的兴趣点击相应的标题，可以查看完整的推荐列表（见图 9-2）。

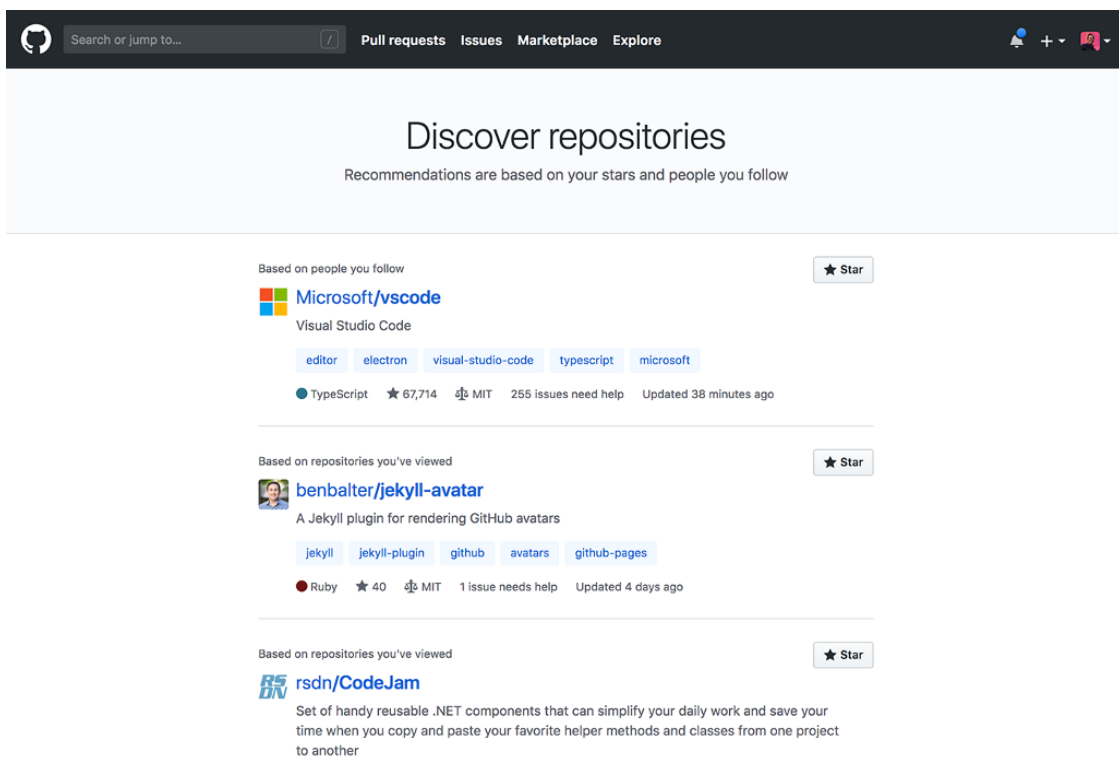


图 9-2 GitHub 上包含仓库建议的 discover 页面

GitHub 使用机器学习技术，根据你所收藏（star）、贡献和查看过的仓库生成一个推荐列表，同时还考虑到你在 GitHub 上关注的人。第 17 章更详细地介绍了这些行为及其重要性。建议：如果要改变推荐，请注意你所收藏的仓库和你关注的人。

9.1.3 趋势仓库

探索页面的下一个部分是一个包括排名前 25 的趋势（trending）仓库列表。单击标题，可以访问 <https://github.com/trending>，该连接页面包括整个 GitHub 上正在流行的仓库和用户。

提示：为什么只排前 25 个仓库？因为拥有超过 25 个仓库将淡化趋势的含义。此外，计算排名的过程需要花费大量时间，因此限制它可以保持低成本。

此页面允许你根据仓库的主要语言筛选仓库。如果你对 JavaScript 仓库感兴趣，可以单击“Other Languages”按钮并选择 JavaScript。

页面顶部的 Trending 按钮默认为筛选时间为今天，但你可以单击它查看本周和本月的排名。

为了确定排名，GitHub 查看各种数据指标，如 star 量、fork 量、commit 量、follow 量和 pageview 等。GitHub 对这些数据进行了适当的权衡，并考虑了活动发生频率等因素，而不仅是总数。

9.1.4 探索主题

热门主题（topics）部分列出了 GitHub 上最流行的主题。主题是用户应用于仓库的类别。主题为人们提供有关仓库的更多信息。仓库所有者可以为仓库指定多个主题。该概念与标签或问题标签非常相似。

图 9-3 显示了 thewecanzone/GitHubForDummiesReaders 仓库的主题列表。仓库管理员可以查看“管理主题（Manage topics）”链接，该链接允许他们添加或删除仓库的主题。

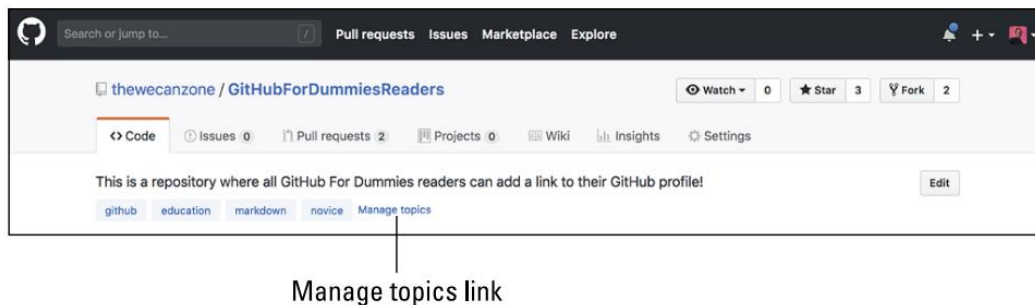


图 9-3 仓库的主题列表

当添加主题时，输入主题名称的一部分，GitHub 会根据其他人在 GitHub 中使用的话题提供建议选项。图 9-4 显示了一个示例，我们在其中输入“novice”，GitHub 推荐了包含“novice”的其他主题。

访问主题页面 <https://github.com/topics> 来查看 GitHub 上最常用的主题。

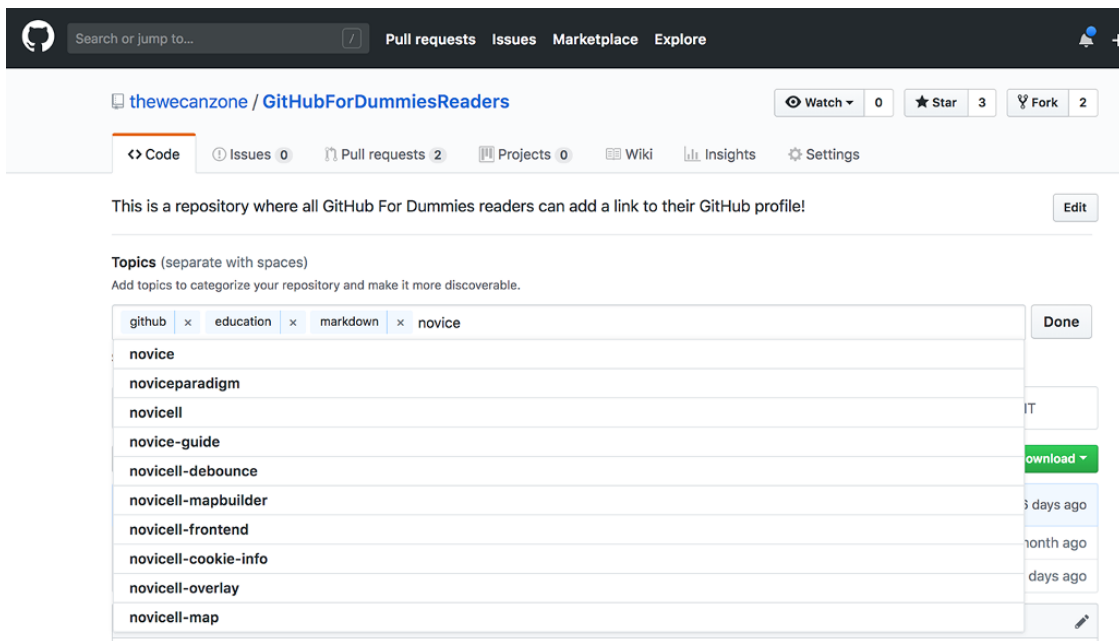


图 9-4 推荐的主题列表

单击一个特定的主题是探索自己感兴趣的主题相关仓库的好方法。例如，如果你对探索 Node.js 仓库感兴趣，可以导航到 <https://github.com/topics/nodejs>。

主题是开放的，因为人们可以将他们想要的任何主题应用到仓库中。然而，热门主题往往是经过筛选的。例如，Node.js 主题页面有一个描述和 logo（见图 9-5）。图 9-5 还展示了其他排序方式（通过扩展 Sort 按钮）。

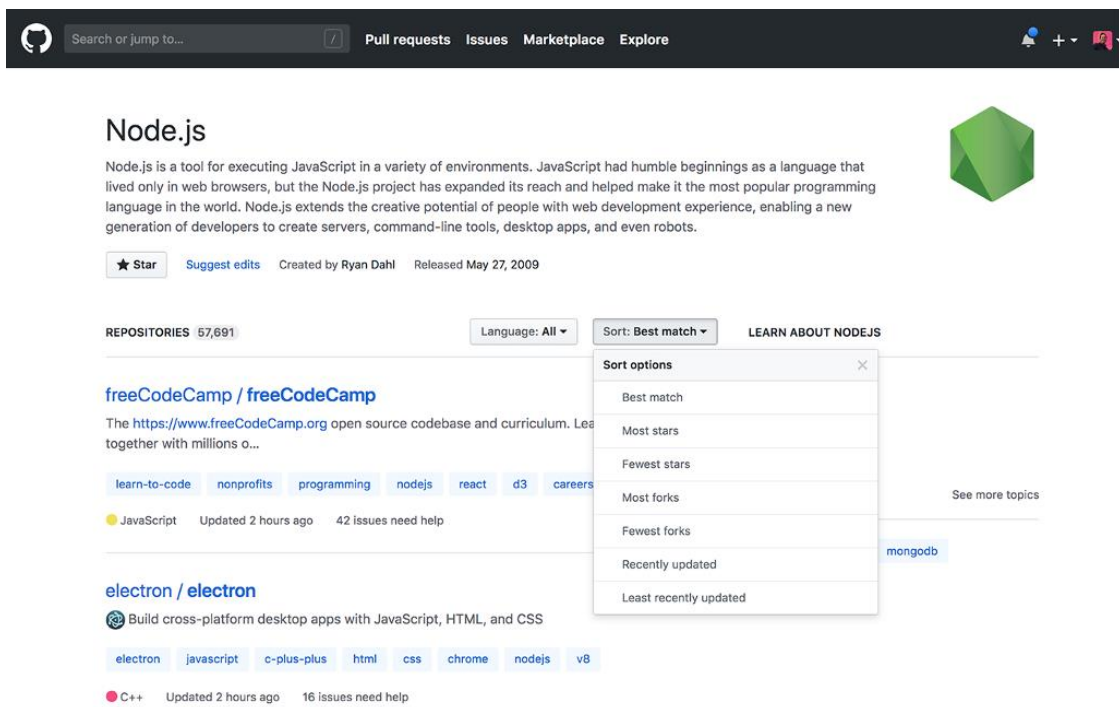


图 9-5 Node.js 的主题页面

主题描述和 logo 本身是在一个开源仓库中指定的。任何人都可以对现有主题描述提建议。任何人都可以为新主题提出主题描述和 logo。主题描述和 logo 位于 <https://github.com/github/explore> 的仓库。例如，Node.js 仓库的 logo 和主题描述位于 <https://github.com/github/explore/tree/master/topics/nodejs>。

9.1.5 探索市场应用程序

探索页面的下一部分列出了流行的市场应用程序（Marketplace apps）。你可以将这些应用程序添加到仓库中，以便在使用仓库时启用更多功能并提高工作效率。

第 15 章更详细地介绍了 GitHub 市场。

9.1.6 探索 Events

“Events”部分列出了一些即将举行的 GitHub 相关活动，如 GitHub Universe（GitHub 的年度旗舰会议）、GitHub Satellite（在世界各地举办的小型 GitHub 社区活动）等。经常查看此页面以了解未来的活动，这也是联系更大的 GitHub 社区的一个好方法。

9.1.7 探索 Collections

接下来的三个部分是精心策划的合集（Collections）。在撰写本书时，它们目前是前端 JavaScript 框架、学习编码和 Probot 应用程序。

每个合集都由人工精心筛选的结果。合集可能包含网页和仓库的列表。其目标是通过列出网站供进一步阅读，以及包含相关代码的仓库，以此作为深入学习特定主题的良好起点。

例如，要编辑学习编码（Learn to Code）合集，建议在 `github/explore` 仓库中编辑此文件：<https://github.com/github/explore/blob/master/collections/learn-to-code/index.md>。

9.1.8 获取好友的帮助

探索 GitHub 上其他人在做什么发现有趣的新仓库的一个好方法。无论是你的好友还是你崇拜的其他人，一定要注意他们在 GitHub 上的活动。一种方法是通过 star（收藏）。

提示：当你在 GitHub 上探索仓库时，一定要 star 那些引起你兴趣的仓库。要查看 star 过的仓库，请转到 <https://github.com/stars>。为仓库添加 star 不仅是为仓库添加书签以供日后探索的好方法，而且也是对仓库表示认可的好方法。

在 star 页面的左下角是 GitHub 上好友的头像。单击一个好友以查看她 star 了的仓库。被 star 的仓库中通常会出现你可能没有注意到的有趣仓库。

你还可以访问你的好友或你欣赏的其他开发者的个人资料页面，并查看他们的仓库。固定仓库（Pinned repositories）是一个人明确选择固定的仓库。图 9-6 显示了本书作者之一 Phil 的固定仓库。

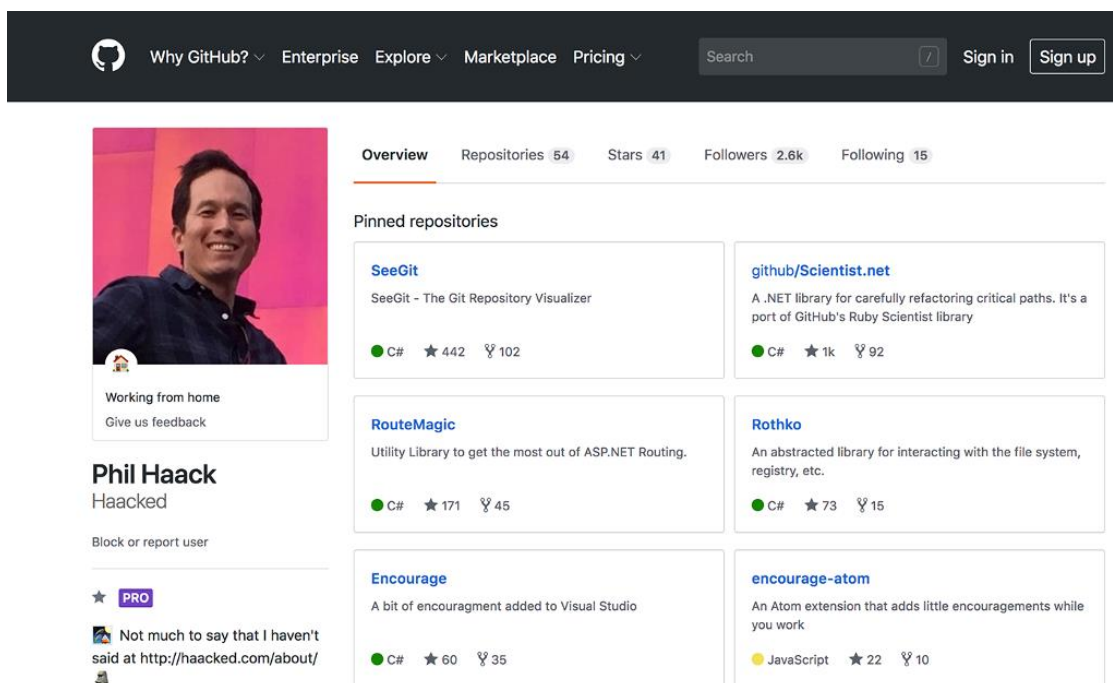


图 9-6:作者之一 Phil 的个人资料页面

9.2 寻找可以贡献的地方

你们中的一些人可能已经准备好不再仅仅使用开源代码，并且准备通过对开源的贡献来磨练自己的技能。那么该如何为一个项目做出贡献？

当你在寻找贡献的地方时，在 GitHub 上探索有趣的仓库发挥了作用。唯一的区别是，当你在寻找一个可以贡献的地方时，你不得不进一步缩小你的探索范围。

首先要问自己，你希望为哪种类型的仓库做出贡献。回答这个问题有助于缩小和指导你的搜索范围。

为开源项目做贡献有很多合理的理由。包括以下一项或多项理由：

- （1）希望全面参与 OSS 并学习如何做出贡献；
- （2）希望帮助改进一个我在日常工作中使用的项目；
- （3）希望支持一个对世界有益的项目；
- （4）希望通过在一个我通常无法接触的技术领域工作来扩展我的技能；
- （5）只是觉得无聊，希望为一些很酷的事情做点贡献。

无论你是出于什么原因要为开源做出贡献，当你接触一个新的仓库时，一个好的开始就是寻找比较容易完成的 issue。许多仓库都有某种方式来表明那些对于

首次贡献者来说是很好的 issue。通常，他们会贴上一个标签，比如 help-wanted 或者 up-for-grabs。

事实上，一些网站会搜索标记为此类的问题，并使那些希望开始使用开源的人可以搜索这些问题。例如，其中一个网站（Up For Grabs）位于 <https://up-for-grabs.net>。

该网站有一个按项目（project）、标签（label）和标记（tag）查找问题的搜索工具。图 9-7 显示了一个示例，其中我们正在查看所有具有初学者标签（label）并使用 javascript 标记（tag）的项目。

网站将显示匹配到的仓库列表。单击仓库结果的标签，查看具有该标签的仓库的特定 issue。筛选 issue 是一种很好的方法，可以帮助初学者找到具体的 issue。

一个名为 help-wanted 的 GitHub 主题（见 <https://github.com/topics/help-wanted>），列出了正在寻求帮助的仓库。当你访问需要帮助的项目时，请访问 issue 页面以查找你可以帮助解决的潜在问题。



图 9-7 Up for grabs 网站的过滤器

例如，图 9-8 显示了 Visual Studio Code 开源项目的问题跟踪器。在项目的顶部有一条消息，其中提到了项目的贡献原则，以及一条有用的提示，用于查找标记为“help wanted”或“good first issue”的 issue。图 9-8 显示了标签“good first issue”所涵盖的问题。

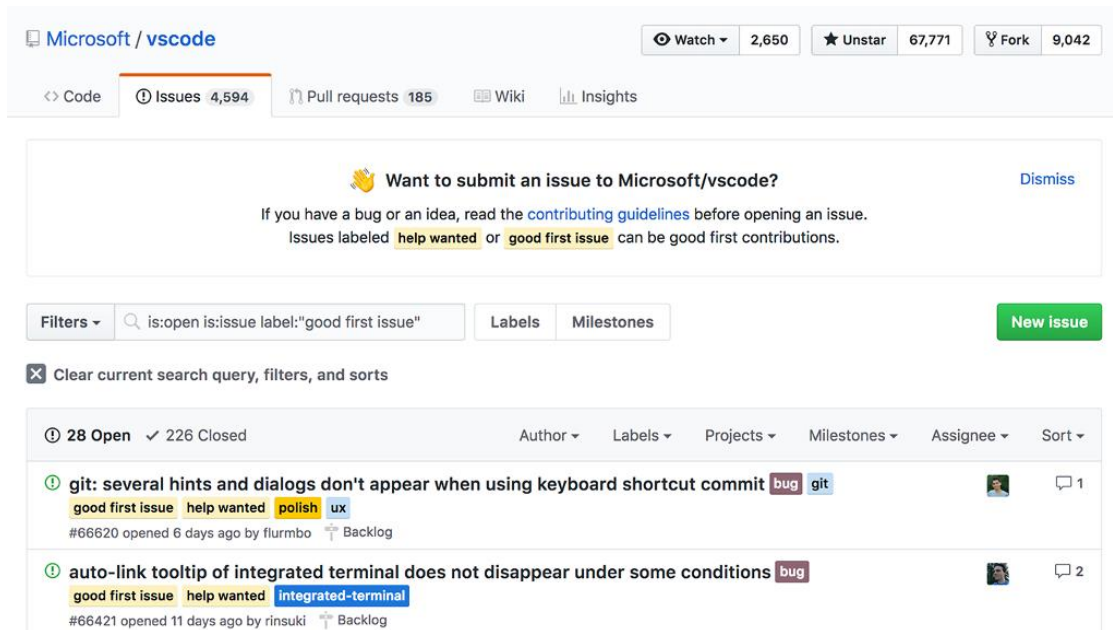


图 9-8 Visual Studio Code 的 Issues，通过标签“good first issue”筛选

9.3 调查你要贡献的项目

假设你决定要为某个特定项目做出贡献。接下来的步骤是什么呢？对于本例，我们查看 Visual Studio Code 仓库（见 <https://github.com/microsoft/vscode>）。

9.3.1 阅读贡献指南

正如 Issue 页面上的建议所指出的那样，首先通读贡献指南。Visual Studio 代码的贡献指南的网址如下：

<https://github.com/Microsoft/vscode/blob/master/CONTRIBUTING.md>。

提示：issue 页面上显示的贡献指南消息遵循一种约定。添加一个名为 contributing.md 的文件到仓库的根目录、添加名为 .docs 的文件夹或名为 .github 的文件夹，以公示本仓库的贡献准则。

与许多贡献指南一样，该指南提供了如何为项目做出贡献的概述。它指出，贡献不仅仅是编写代码。为了强调这一理念，该指南首先让你知道在何处询问有关项目的问题。该指南涉及以下关键主题：

- (1) 在哪里提问；
- (2) 在哪里提供反馈；

- (3) 报告 issue 的地点和方式;
- (4) 有关其自动 issue 管理的详细信息;
- (5) 提供如何贡献代码的指南的链接。

这种格式在贡献指南中非常常见。有时，一个较小的项目还将包括如何在主文档中贡献代码，但对于较大或更复杂的项目，贡献代码是一个大主题。

9.3.2 阅读贡献代码指南

VSCode 项目将其代码贡献指南保存的网址如下 <https://github.com/Microsoft/vscode/wiki/How-to-Contribute>。牢记：如果你打算贡献代码，请仔细阅读本指南。该指南包含了你希望在项目中编写代码的关键信息，其结构在开源项目中非常常见：① 构建代码；② 运行代码；③ 调试代码；④ 运行自动化测试；⑤ 运行自动代码分析，如 linting。

下一节介绍代码贡献工作流程：

- (1) 遵循分支规范；
- (2) 创建 PR；
- (3) 分发包装代码。

9.3.3 阅读行为准则

贡献指南侧重于贡献的机制。许多项目现在还包括一个 CODE_OF_CONDUCT.md 文件。该文件列出了对参与项目人员的期望。它为项目设置行为准则，以及违反行为准则的情况下的解决过程。

VS Code 使用微软开源行为准则，其链如下：（见 <https://opensource.microsoft.com/codeofconduct>）。

提示：在自己的仓库中添加行为准则非常简单。如果使用浏览器在 GitHub 上添加新文件并将文件命名为 CODE_OF_CONDUCT.md，GitHub 将显示一个行为准则，选择下拉列表，其中包含两个选项：贡献者公约和公民行为准则。

你还可以访问任何公共仓库的社区页面来添加行为准则。通过在仓库 URL 的末尾追加 /community 转到社区页面。例如，键入 <https://github.com/microsoft/vscode/community> 来了解项目是否有行为准则。如果没有，页面将为你提供提出建议的机会。

9.4 设定贡献者期望

在阅读了行为准则和贡献指南之后，你就可以开始贡献了。现在你就应该明确流程是如何运作的。

牢记：这些关于贡献者的一般指导原则可能没有在贡献指南中详细说明，更多的是在开源环境中长期工作所积累的集体智慧的结果。

9.4.1 他们不会解决所有问题（Issue）

许多（可能不是全部）项目的资源有限，它们的优先级可能与你的优先级不一致。当你谈论一个 Issue 时，务必牢记这一点。一方面，提出一个好的 issue 需要时间和精力。一个好的 Issue 可以清楚地描述了一个问题，并有清晰的步骤来重现这个 Issue，这对于一个项目来说是非常有价值的，所以可以理解写 Issue 的人是很投入的。

牢记：另一方面，一个 Issue 可能不会被修复，这就是为什么我们注意到撰写一个 Issue 是一种贡献。这不是以物换物，而是一份礼物，所以你不要期望得到太多的回报。好的项目维护者会感谢你提出问题，并标记他们可能知道的任何解决办法，但他们不欠你任何东西。如果你所遇到的 Issue 被标记为 won't fix，请不要将其放在心上。

9.4.2 他们不会合并每个 PR

提交 PR 是大量工作的最终结果。为了提交正确的 PR，开发人员必须花时间让代码在自己的机器上运行，充分理解代码以进行更改，编写更改，然后提交。

当仓库维护人员拒绝 PR 时，你感到失望是可以理解的。但是请记住，他们没有义务接受你的 PR。虽然你在 PR 中花费了很多时间，但是他们必须在项目的生命周期中决定代码的更改与否。

牢记：PR 被拒绝的情况应该非常罕见。你和项目维护人员可以事先做很多事情来避免 PR 被拒绝。第一步从互相沟通开始，在开始解决问题之前，进行有目的的评论，说明你的总体计划。

确保项目相关人员与你的看法达成共识，通过这种交流降低去做与项目背道而驰的可能性。

不过，不要在那里停止交流。一旦有一两个 Commit，就将其推送到 PR，并确保 PR 清楚地标记为正在进行的工作，通过关注 PR 可以在项目进行过程中持续获得反馈，并确保方向正确。这样做最大限度地减少了走弯路。

最后，确保你遵循了他们的所有贡献准则、风格准则等。由于这是他们的项目，如果他们选择合并你的 PR，他们将拥有这段代码。这里不是彰显你的工作方式或个人代码风格的地方。

9.4.3 他们不欠你任何东西

作为一个流行开源项目的维护者，最大的挑战之一是许多使用该软件的人所表达的权利感。

牢记：在大多数情况下，维护人员都是项目的志愿者。但事实并非总是如此，有时，他们是从事开源项目的有偿员工，但很有可能不是你付钱给他们的。他们不欠你，他们不欠你钱，他们不欠你任何东西，你应该相应地对待他们。

当然，一个运行良好的项目的维护人员会尽力为参与项目的人员提供便利。他们希望大多数参与其中的人都快乐，这是很自然的。开源项目得益于思想共享和更多的贡献者，因此，如果符合他们的路线图和优先事项，并且不至于太麻烦的话，不采取强硬的态度，努力为你提供这个功能，往往是最符合他们的利益。所以，向别人索取东西并没有错。为你想要的东西提出有力的理由是没有错的。但最终，你不付他们钱，他们也不欠你任何东西。

9.5 密切关注项目

当你是一个项目的贡献者时，关注项目的进展情况非常重要。一种方法是通过 GitHub 通知。第 1 章介绍如何管理通知设置。GitHub 可以发送新问题、PR 的新注释等。根据你对项目的兴趣级别管理通知设置非常重要。

虽然 GitHub 可能是开源的枢纽，但除了 GitHub，还有许多与开源项目相关的资源。你可能想浏览 GitHub 之外的开源项目。例如，许多项目都有一个你可以关注的 Twitter 帐户。有些项目有一个公共 Slack 实例，你可以在其中与维护人员和使用该项目的其他人聊天。你可能还想阅读这些项目的维护人员的博客，以了解最新的发展。许多项目和项目维护人员仍在发布 RSS 提要，虽然这是一项古老的技术，它仍能使人们了解网站的进展。

第 10 章：启动自己的 OSS

本章学习重点：

- （1）为 OSS 建立仓库；
- （2）管理 OSS 工作流；
- （3）结束 OSS 项目。

许多开发者在一生中总有想要分享自己代码的时候。分享代码的途径和动机因人而异。一些开发者觉得其他人对自己的代码感兴趣，他们仅仅就“丢出”源码出来，并不期望他人的后续贡献。而有些人则想发起一场有大量贡献者参与的真正的运动。

无论动机是什么，开源软件的唯一要求是选择符合开源定义（Open Source Definition, OSD）的许可证。可以在 <https://opensource.org/osd> 中找到开源定义。开放源代码计划（Open Source Initiative, OSI）有一个批准许可证的流程，其网站列出了大量许可证。

但对于大多数开源软件来说，选择许可证只是一个起点。管理一个开源项目需要做更多的工作。在本章中，我们将介绍如何在 GitHub 上启动一个开源项目、对其进行维护，以及在必要将它废弃。

10.1.创建开源仓库

当创建仓库并期望将它开源时，通常会在仓库创建过程中将仓库公开并选择一个许可证。

有时候，将上述选择推迟到以后反而更合适。因为也许想把一切弄妥当了之后再公开仓库，或者还想对许可证选择上深思熟虑一番。以下各节重点介绍了如何将现有私有仓库转换为开源仓库。我们假设已经创建了一个包含了 README.md 的仓库，但没有公开该仓库，也没有选择许可证（如果还未创建，请参阅第 3 章）。

10.1.1.添加许可证

软件被视为开源的唯一要求是它拥有开源许可证，而 GitHub 使向现有项目添加许可证变得容易。

图 10-1 显示了没有许可证的私有仓库主页。要添加许可证，请执行以下操作：

- （1）点击 Create newfile 按钮。

单击按钮后，GitHub 会提示命名文件。

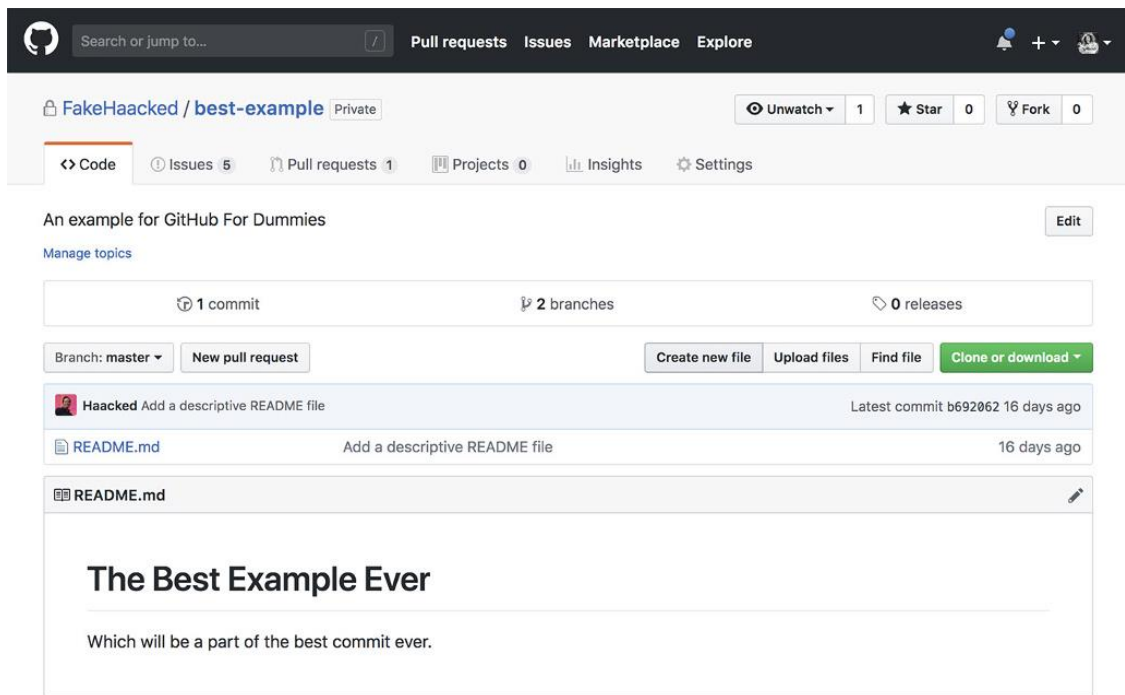


图 10-1 即将开放源代码仓库的主页

(2) 键入文件名。

例如，命名为 LICENSE。GitHub 注意到正试图创建一个许可证文件，并在右侧添加了一个标记为“选择许可证模板（Choose a license template）”按钮，如图 10-2 所示。

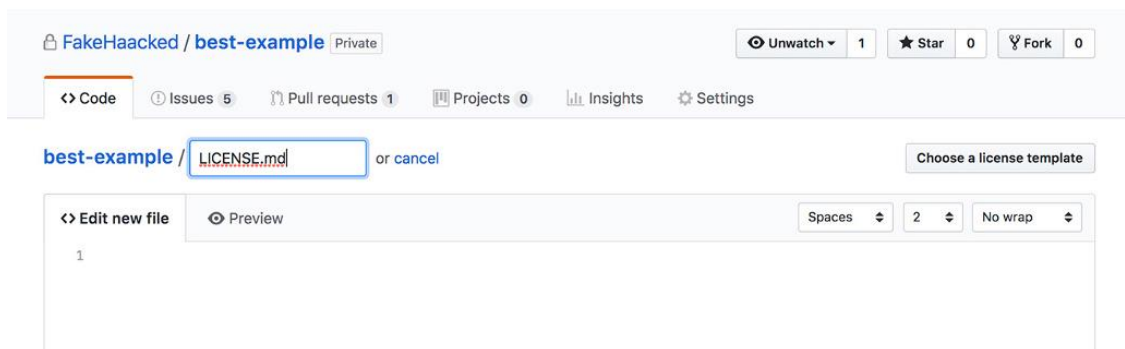


图 10-2 选择许可证模板（Choose a license template）按钮

(3) 单击“Choose a license template”按钮查看许可证模板列表。

如图 10-3 所示，GitHub 上使用的三个最常见的许可证以粗体显示在前面。第 3 章提供了选择许可证的简要指南，但本页有一个链接指向更详细的指南，帮助选择最适合项目的许可证。

The three most popular licenses on GitHub

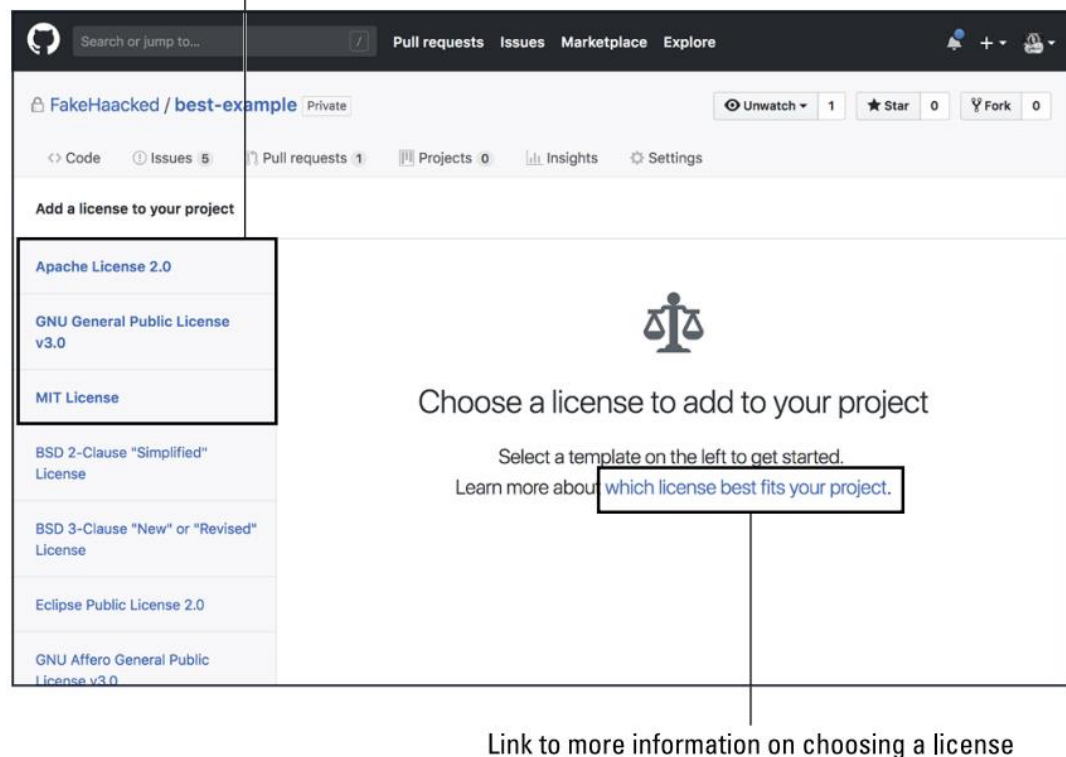


图 10-3 许可证选择器页面

(4) 确认需要哪个许可证后，单击该许可证以查看有关该许可证的信息。

此页面包括一个高级摘要，其中包含有关许可证关键特征的要点。此概述有助于了解许可证，而无需费力阅读所有法律术语。

当然，该页面还为那些喜欢法律术语的人展示了许可证的全文，右侧是许可证所需填写的字段。

(5) 在字段中输入信息，以根据情况自定义许可证。

可以在图 10-4 中看到此页面的示例。

High-level summary of the license attributes

The screenshot shows a web interface for selecting a license. On the left, under the heading "Add a license to your project", there is a list of licenses: Apache License 2.0, GNU General Public License v3.0, MIT License, BSD 2-Clause "Simplified" License, BSD 3-Clause "New" or "Revised" License, Eclipse Public License 2.0, GNU Affero General Public License v3.0, and GNU General Public License v2.0. The MIT License is selected. In the center, a detailed view of the MIT License is shown, including a description, a table of permissions, limitations, and conditions, and a note that this is not legal advice. On the right, a form titled "To adopt MIT License, enter your details. You'll have a chance to review before committing a LICENSE file to a new branch or the root of your project." contains fields for "Year" (with a dropdown set to 2019) and "Full name" (with a text input containing "Phil Haack"). A green "Review and submit" button is at the bottom of the form.

Permissions	Limitations	Conditions
✓ Commercial use	✗ Liability	① License and copyright notice
✓ Modification	✗ Warranty	
✓ Distribution		
✓ Private use		

The text of the license with your information filled in

License template fields and button to move to next step

图 10-4 许可证信息页面

(6) 填写完成所有内容后，单击“Review and submit”按钮。

返回到文件创建页面，包括填写的许可证文本。

(7) 向下滚动以输入提交消息，然后单击“提交新文件（Commit new file）”，以创建许可证文件。

10.1.2 添加贡献者指南

包含在仓库中的另一个有用文档是名为 **CONTRIBUTING.md** 的文件。此文件提供有关如何为项目做出贡献的信息。它应该包括诸如人们应该在哪里提问以及在哪里提供反馈等信息。

第 9 章提供了更多关于贡献者可以在 **CONTRIBUTING.md** 文档中期望找到什么的详细信息。这一章是自己的项目贡献者指南的好起点。

一个健壮的 **CONTRIBUTING.md** 文档将为贡献者节省大量的时间。一定要不时地检查该文档，以确保它是最新的。

10.1.3 增加行为准则

行为准则（code of conduct）明确了参与项目的每个人的行为期望，无论他们是提供代码、对问题发表评论，还是与仓库中的其他人进行交互。另外，行为准则也为自己的项目设定开放和良好环境类型的基调。

要添加行为准则，请遵循与添加许可证时相同的过程，但将文件命名为 `CODE_OF_CONDUCT.md`（请参阅本章中的“添加许可证”部分）。第 9 章也介绍了这个过程。

10.2 公开仓库

至此，仓库已经具备了公开所需的基本项。要将仓库从专用更改为公用，请执行以下操作。

- （1）访问仓库的设置（Settings）页面。
- （2）滚动至设置页面底部，进入“Danger Zone”部分，如图 10-5 所示。

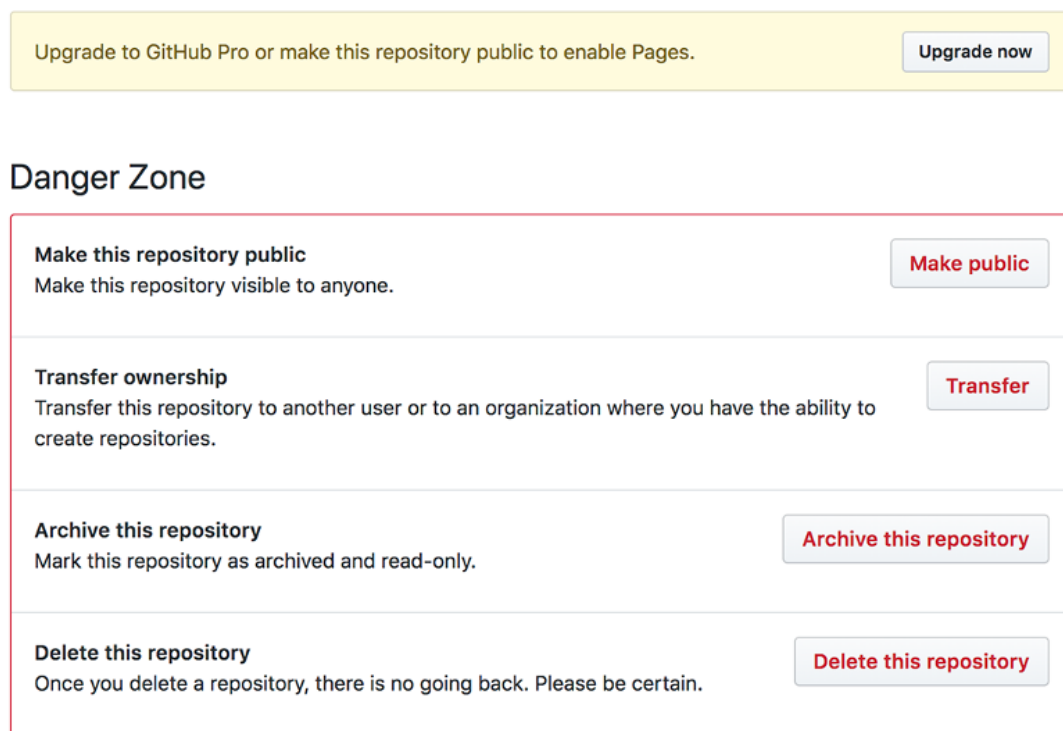


图 10-5 “Danger Zone”：可以将私人仓库公开

- （3）单击“Make Public”按钮以启动公开仓库的过程。

警告：此步骤是一个潜在的危險操作，因为可能会无意中将希望保密的信息暴露给外界。因此，请确保已经准备好公开此仓库。如果创建它的目的是从一开始就将其公开，那么仓库不应该包含任何机密信息。

(4) GitHub 在单击“Make public”按钮后显示一个确认对话框，要求键入仓库的名称，并单击“I understand, make this repository public”按钮，以确认公开仓库，如图 10-6 所示。

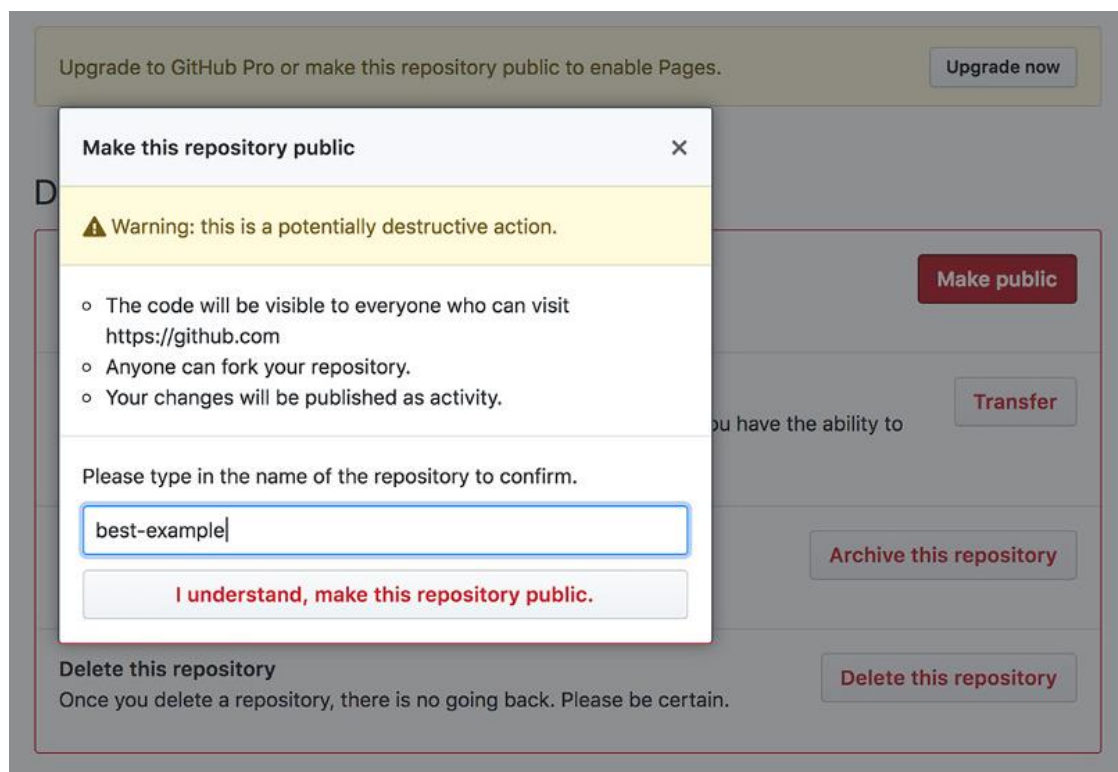


图 10-6 确认将专用仓库转为公开仓库

(5) 键入仓库名称，然后单击“I understand,make this repository public”按钮，将仓库设置为公开仓库。设置页面将出现一个部分，方便用户管理新的开源项目。

10.3 执行行为准则

行为住着呢是沟通社区行为期望的一种良好的方式。但是一个行为准则本身并不足以创建一个健康的社区；必须依照准则执行。

警告：如果从来不必强制执行行为准则，那就最好不过了。大多数参与项目的人都会乐于助人，而且很友善。然而，随着项目的发展，越来越多的人参与进来，几乎不可避免地要介入强制执行行为准则。

10.3.1.善意的回应

请记住，即使有人的行为不符合行为准则精神，他们也不一定是恶意的或是个坏人。也许他们今天过得不好，就发泄到了你的仓库上。这并不是他们行为的借口，但这就是事实，我们每个人都会犯错。

提示：如果有人仓库中的某个 Issue 上，发表了一些带有偏见的或愤怒的评论，而这些评论有悖于期望培养出的社区类型。请用善意的回应来表达对他们处境的同情，但也要明确指出该评论不符合社区行为准则。具体说明他们违反了哪一条行为准则，如果可以的话，向他们建议可以采取的替代方法。

例如，如果有人仓库上评论说，维护人员肯定讨厌某个人，因为他们没有修复某个特定的 bug，那么可以采用类似于以下的回应：“此仓库中不欢迎人身攻击。我们都是利用自己业余时间工作的志愿者。我们理解您的困惑。与其进行人身攻击，不如写下这个 bug 是如何影响您并给你造成困扰的原因。”

牢记：时刻铭记在心，你不是其他人发泄的沙袋。如果你厌倦海量的负面评论，那就休息一下，切忌在愤怒时回应。可以选择远离，也可以向他人寻求帮助。

10.3.2 利用封禁功能

如果一种善意的劝告在每种情况下都能奏效那就好了（见前一节），但有时还是会爆发脾气。在这种情况下，临时交互限制就派上了用场。

在公共仓库的“Settings”页面的最底部（对于私有仓库，此设置不存在）是一个“Moderate（缓和）”部分，其中有指向“Interaction limits（交互限制）”的链接。单击此链接可查看可用的限制集。图 10-7 显示了启用“Limit to existing users（限制现有用户）”选项后的页面。

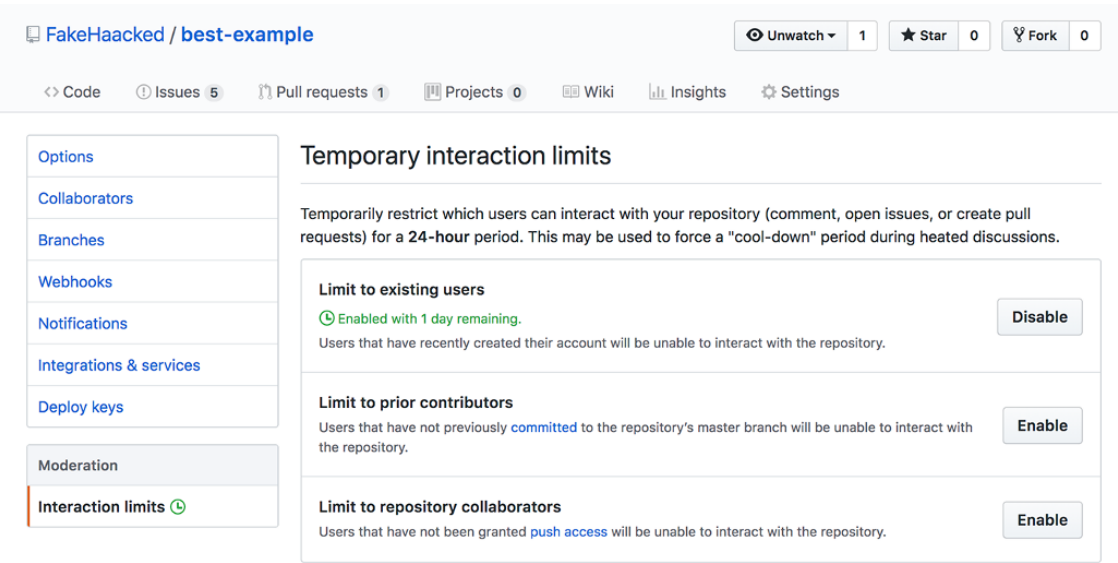


图 10-7 交互限制的内容

牢记：这些限制不是永久性的禁令，而是提供冷静期。当讨论或问题变得有点过于激烈时，这个工具非常适合使用，可以让每个人都有时间冷静下来，防止讨论进一步失控。

10.3.3 屏蔽用户

每隔一段时间，可能会在项目中遇到一个真正的滥用者。如果已经尝试过用仁慈的方式来对待他们，并且也已经尝试过 24 小时的冷静期，但他们仍然坚持无礼或对立的行为，那么可以考虑屏蔽或举报该用户了。

屏蔽或举报用户步骤如下。

(1) 单击其评论旁边的用户名以访问其评论页面。

在他们的个人资料图片下面是“block or report user（屏蔽或举报用户）”的链接，如图 10-8 所示。

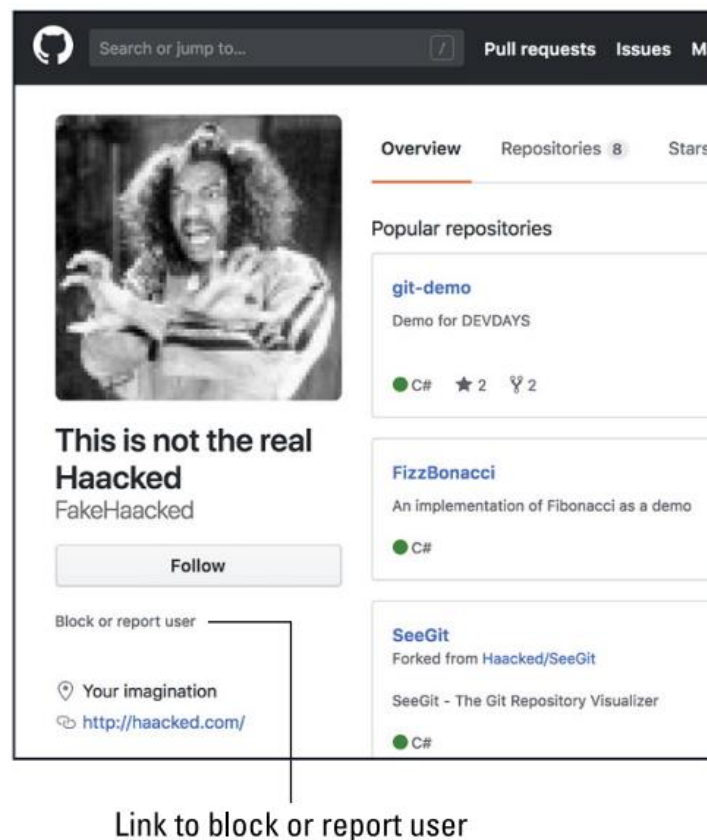


图 10-8 个人资料图片下面的“block or report user（屏蔽或举报用户）”的链接

(2) 单击该链接以屏蔽或举报用户。

出现一个对话框，其中有两个选项：“Block user（屏蔽用户）”或“Report abuse（报告滥用）”，如图 10-9 所示。

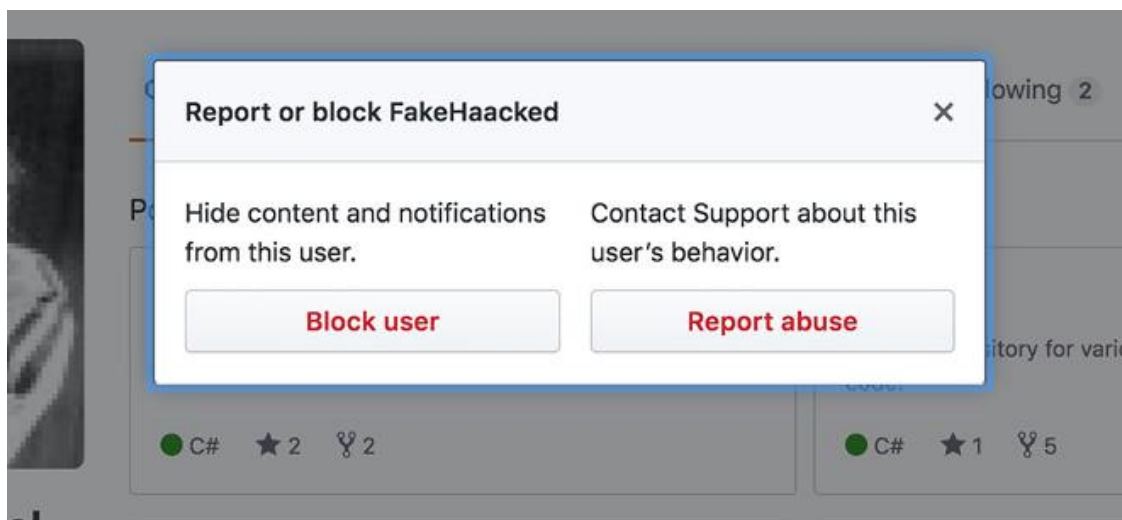


图 10-9 “Block user（屏蔽用户）”或“Report abuse（报告滥用）”按钮

（3）单击“Block user（屏蔽用户）”立即屏蔽该用户。

屏蔽用户会拒绝用户访问活动和仓库。它还会屏蔽用户向发送通知，例如当他们@提到用户名时。有关屏蔽用户时发生的情况的更多详细信息，请访问 <https://help.github.com/articles/blocking-a-user-from-yourpersonal-account/>。

提示：从帐户设置页面（通过 https://github.com/settings/blocked_users），也可以屏蔽用户、查看当前已屏蔽的用户列表、取消屏蔽用户。

10.4 编写 README.md 文件

README.md 文件在 GitHub 上具有特殊意义。当访问仓库的主页时，GitHub 会在主页上显示 Markdown 标记的 README.md 的内容。README.md 内容是访问仓库的访问者希望看到的内容，以了解有关项目的更多信息。

第 3 章介绍了 README.md 文件中应包含的内容。它还应包括指向 CONTRIBUTING.md 文件和 CODE_OF_CONDUCT.md 文件的链接。阅读此文件后，访问者应该了解项目是做什么的，如何参与，以及到哪里去查找更多信息。

10.5 编写规范的文档

开源最被忽视的一个方面是文档。编写规范的文档是将项目与其他项目区分开来的好方法。此外，文档还为其他人在开源领域的应用提供了一个良好的途径。本书的一位作者通过为一个项目贡献文档，而获得了开源领域的成功。

提示：GitHub 支持 wiki 功能，但我们建议使用另一个功能，从仓库主分支的 docs 文件夹中为文档站点提供服务。这个特性有很多好处，但其中一个最大的好处是能够将文档及其所记录的代码一起进行版本控制。

第 4 章介绍如何构建 GitHub 页面站点。从 docs 文件夹提供文档与此过程非常相似。以下步骤假设拥有的仓库不是 GitHub Pages 网站。如果需要复习，第 3 章讨论了如何创建常规仓库。

确保将名为 `index.md` 的文件提交到仓库主分支中的 `docs` 文件夹下。此功能要求在启用之前已经有一个 `docs` 文件夹。如果不清楚如何做到这一点，第 7 章详细地介绍编写和提交代码。

导航到仓库设置，然后向下滚动到 GitHub 页面部分。默认情况下，“Source（源）”设置为“None（无）”。点击按钮，选择如图 10-10 所示的“master branch/docs”文件夹选项，然后点击“Save（保存）”按钮。

GitHub Pages

GitHub Pages is designed to host your personal, organization, or project pages from a GitHub repository.

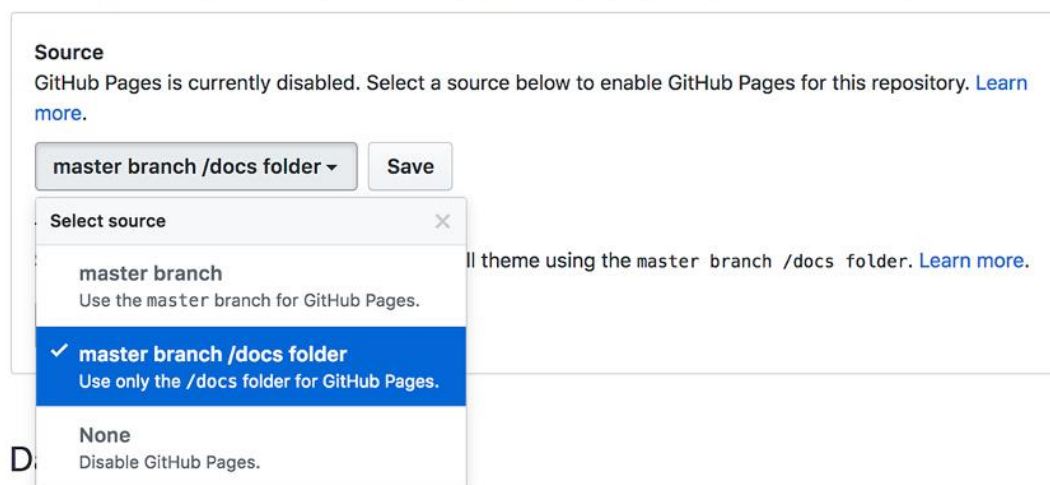


图 10-10 从 docs 文件夹为 GitHub 页面站点提供服务

现在可以访问文档站点 `https://{USERNAME}.github.io/{REPO-NAME}/`。要查看这方面的示例，请访问 `https://fakehaacked.github.io/best-example/`。

正如在第 4 章中介绍的，如果需要，可以选择一个主题并添加自定义域名。

牢记：别忘了在 `README` 文件中添加指向文档站点的链接，以便人们可以找到它。

10.6 管理 Issue

随着你的开源项目的消息传出，人们将开始尝试它。在某个时间点，人们会提出一些 Issues。祝贺！当一个 Issue 被提出时，它意味着有人非常关心你的开源项目，开始报告一个 bug、提出一个 Issue 或提出一个功能请求。这种兴趣对于开源项目来说是件好事。

但是，如果项目非常成功，新 Issue 的涌入可能会开始变得势不可挡。在以下部分中，将介绍一些管理新 Issue 的提示和技巧。

10.6.1 Issue 标签

GitHub 在创建网站时提供了一组默认的问题标签。

bug
duplicate
enhancement
good first issue
help wanted
invalid
question
ready for review
won't fix

可以通过转到“Issue”选项卡并单击“Labels”按钮来管理这些标签。确保标签集对项目有意义。分配标签有助于在问题出现时进行管理。

10.6.2 分类 Issue

随着新 Issue 的产生，对 Issue 进行分类很重要。单词 triage 来自于医疗领域。Merriam Webster 将 triage 定义为“根据患者护理需求的紧急程度对患者进行分类（如在急诊室）。”

分类 Issue 类似于在医院进行分流。分类应用于 Issue 时，是指审查新 Issue 并分配标签以指示 Issue 类型（错误报告、功能请求、等等）、确定其优先级并将问题分配给人员的过程。在某些情况下，可能会关闭在分类过程中不打算解决的 Issue。

提示：每个分类 Issue 都应该赋予一些标签。如有必要，甚至可以为此目的创建分类标签。将分类标签分配给已分类的 Issue，可以使用未标记过滤器查看所有尚未分类的 Issue，如图 10-11 所示。

为了保持条理，通常有必要定期进行分类，而不是在人们产生新 Issue 时才这样做。

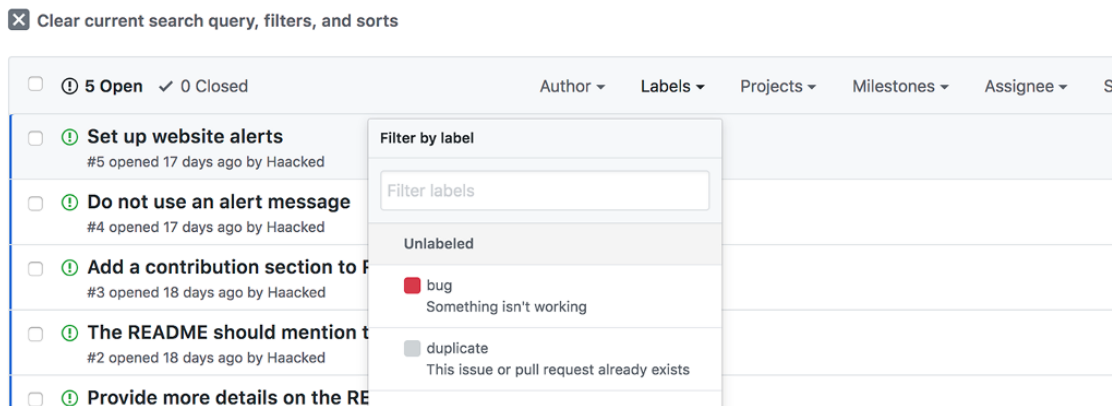


图 10-11 未标记过滤器

10.6.3 Issue 模板

如果提交问题的每个人都了解如何编写适当的 bug 报告，或者他们了解什么是好的特性请求，那就太完美了。仅仅提及“东西坏了”的 bug 报告对仓库维护人员没有太大帮助。

为了帮助提出规范的 Issues，可以设置 Issue 模板，引导人们在仓库中提出 Issues，以提供所需的信息。

在仓库的“Settings（设置）”页面的主窗格的“Issues”部分中，有一个较大的绿色按钮，标记为“Set up templates（设置模板）”。单击该按钮将进入模板设置页面，在该页面中，可以从一组预先存在的问题模板中进行选择或创建自己的模板，如图 10-12 所示。

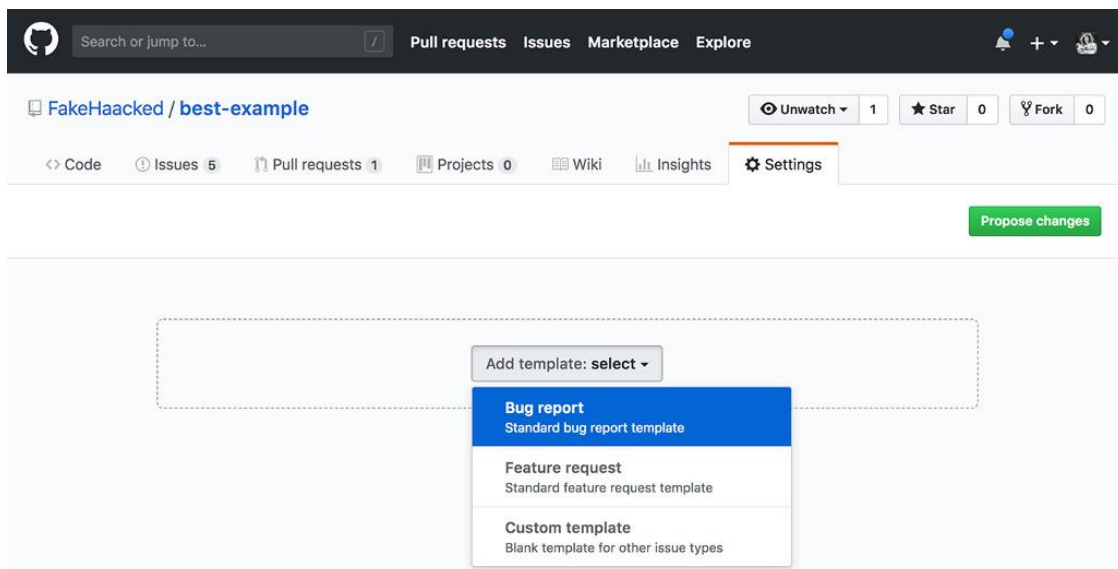


图 10-12 选择问题模板

在“模板设置”页面中，执行以下步骤：

(1) 选择 “Bug report template (Bug 报告模板)”，然后选择 “Feature request template (功能请求模板)”。

结果将在页面上显示它们的列表。请注意，它们尚未处于激活状态。它们仍然需要提交到仓库。

(2) 单击 “Propose changes” 按钮，输入这些模板的提交消息，如图 10-13 所示。

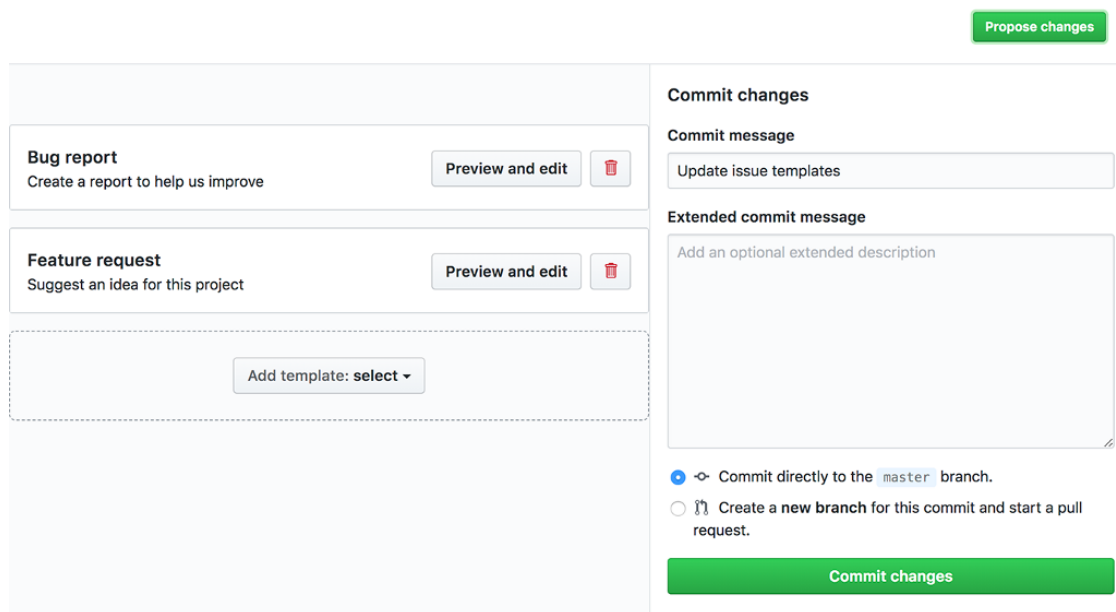


图 10-13 选择一个 issue 模板

(3) 单击 “Commit changes” 按钮将这些模板提交到仓库。

它们存储在一个特殊的.github/ISSUE_TEMPLATE 模板文件夹中。如果导航到该文件夹，将看到两个文件：bug_report.md 和 feature_request.md。查看这些文件的内容，以了解问题模板的结构。

可以通过执行前面的步骤或向.github/ISSUE_TEMPLATE 目录添加与现有问题模板文件具有相同基本结构的文件来创建新模板。

现在，当有人在仓库中创建 issue 时，他们首先会看到一组 issue 模板，如图 10-14 所示。

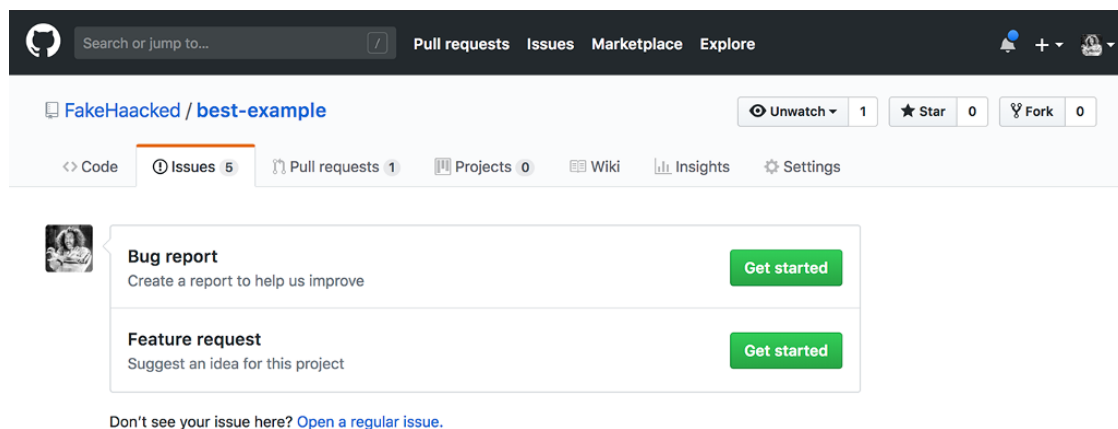


图 10-14 选择一个 Issue 模板来创造 Issue

单击其中一个模板旁的“Get started（开始）”按钮，然后显示问题创建表单，但其内容预先填充了模板信息。图 10-15 显示了使用默认错误报告模板预填充的问题。

Issue: Bug report

Create a report to help us improve. If this doesn't look right, [choose a different type](#).

图 10-15 使用错误报告模板创建新 issue

Issue 的内容将引导 issue 创建者浏览他们期望的所有信息。当然，创建 Issue 的人总是可以选择忽略模板，在 Issue 中放入他们想要的任何内容。

10.6.4 保存答复

通常，随着仓库越来越受欢迎，Issue 开始浮出水面，你会发现自己一遍又一遍地重复回答问题。例如，如果有人报告一个 **bug**，可能需要他们提供一个日志文件。如果每个 **bug** 报告都应该包含一个日志文件，那么应该在问题模板中提到这一点。





但即使这样做了，很多人也会忘记把它包括进去。在这种情况下，保存的答复会派上用场。保存的回复是一个可以反复使用的固定回复。

在撰写本书时，固定响应仅适用于用户，而不是特定于仓库。因此，可以创建自己保存的回复，但它们不会立即提供给仓库的其他成员。

通过 <https://github.com/settings/replies> 的页面，可以管理已保存回复的列表。要创建新的已保存回复，请填写底部的表单，如图 10-16 所示。

Saved replies

Saved replies are re-usable text snippets that you can use throughout GitHub comment fields. Saved replies can save you time if you're often typing similar responses. [Learn more.](#)

Log File Hey please send me your log file	 
MORE INFO Please provide more details about what you expected and what actually happens. Also, if there's a l...	 

Add a saved reply

Saved reply title


Write

Preview

AA B i “ < > 🔗 ☰ ☷ ☰ ☷ ☷ @ 🚩 ↶

Hello, IT. Have you tried turning it off and on again?

Attach files by dragging & dropping, selecting them, or pasting from the clipboard.

 Styling with Markdown is supported

Add saved reply

图 10-16 创建新的已保存回复

要在响应问题或请求评论时访问保存的回复，请单击评论框右上角的箭头，如图 10-17 所示。这将显示保存的回复列表。单击要用于在评论框中填充该回复的选项。

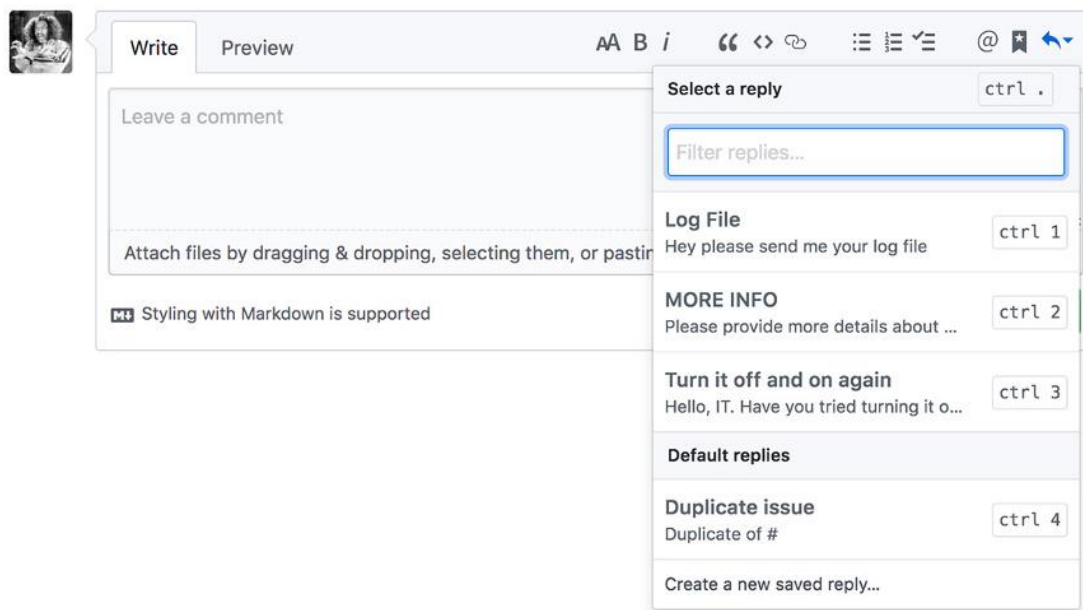


图 10-17 在评论框中选择已保存回复

10.7 结束项目

在仓库生命周期的某个阶段，你可能想离开它。这种离开可能有很多原因。也许这个项目已经不再有用，已经被其他更好的项目所取代。也许这个项目取得了巨大的成功，但没有更多的时间和精力继续投入。

不管是什么原因，重要的是要以启动仓库时所表现出的同样谨慎来，来处理自己与仓库的关系。

10.7.1 归档项目

当项目对其他人不再那么有用时，归档项目是一个不错的选择。或者，即使它仍然有用，但基本上是完整的，归档也可能是一个不错的选择。归档项目表示该项目不再处于活动维护状态。代码仍然可以在全世界使用，人们可以分叉并启动项目，但没有人可以创建新 Issue、新 PR 或将代码推送到归档的仓库。

要归档仓库，请转到仓库设置页面。

(1) 一直向下滚动到 **Danger Zone**，然后单击“Archive this repository（归档此仓库）”按钮，如图 10-18 所示。

Danger Zone

Make this repository private Hide this repository from the public.	Make private
Transfer ownership Transfer this repository to another user or to an organization where you have the ability to create repositories.	Transfer
Archive this repository Mark this repository as archived and read-only.	Archive this repository
Delete this repository Once you delete a repository, there is no going back. Please be certain.	Delete this repository

图 10-18 “Archive this repository（归档此仓库）”按钮

（2）单击此按钮将显示一个详细的确认框，该对话框描述了如果归档仓库将发生的情况，如图 10-19 所示。

Archive repository

×

Unexpected bad things will happen if you don't read this!

This will make the **FakeHaacked/best-example** repository, issues, pull requests, labels, milestones, projects, wiki, releases, commits, tags, branches, reactions and comments read-only and disable any future comments. The repository can still be forked.

Ensure you've changed any repository settings, considered closing all open issues and pull requests and updated your README before you archive this repository.

Once archived, you can unarchive the repository at any time.

Please type in the name of the repository to confirm.

I understand the consequences, archive this repository

图 10-19 “Archive repository（归档仓库）”对话框

（3）键入仓库的名称，然后单击“I understand the consequences, archive this repository”按钮进行归档。

10.7.2 转让所有权

如果项目受到一组维护人员的欢迎，可能希望将所有权转移到另一个账户，而不是归档项目。通过转移所有权，新所有者可以拥有仓库的全部权限并继续维护它。

转让所有权的步骤如下：

（1）转到仓库的“Settings（设置）”页面并向下滚动到“Danger Zone（危险区域）”。

（2）点击“Transfer（转让）”按钮，弹出“Transfer（转让）”对话框，如图 10-20 所示。

（3）确认当前仓库的名称并键入新所有者的用户名或组织名称。

仓库的 URL 会更改，但当有人试图访问该仓库时，GitHub 会将旧 URL 重定向到新 URL。

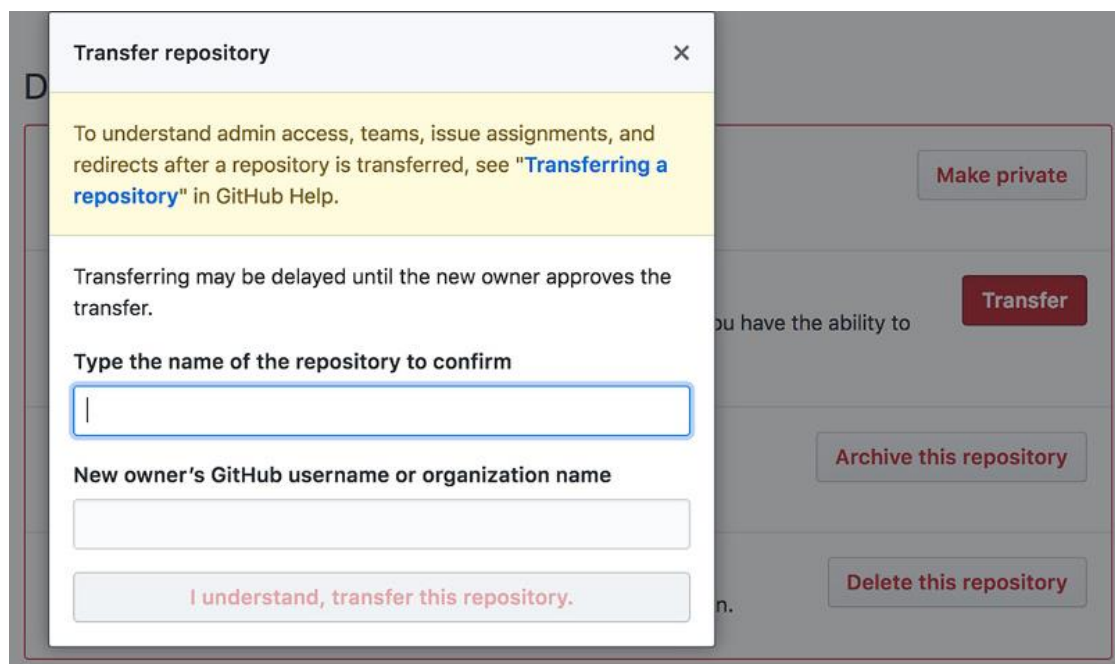


图 10-20 “Transfer repository（转让）”对话框