实践项目：第二部分 – Git CLI

预计时间: 45分钟

概述

祝贺您通过开源运输成本计算器 Python 脚本，开启了您公司的 Git 和 GitHub 之旅。所有开发人员都已对该存储库做出了贡献，并且他们的更改已被接受并合并到新的全球存储库中。现在，您被要求编辑一些代码并添加一些文件。为此，您将分叉此存储库，在提供的实验环境中使用 Git CLI 进行必要的编辑，添加文件，并打开一个拉取请求。

目标

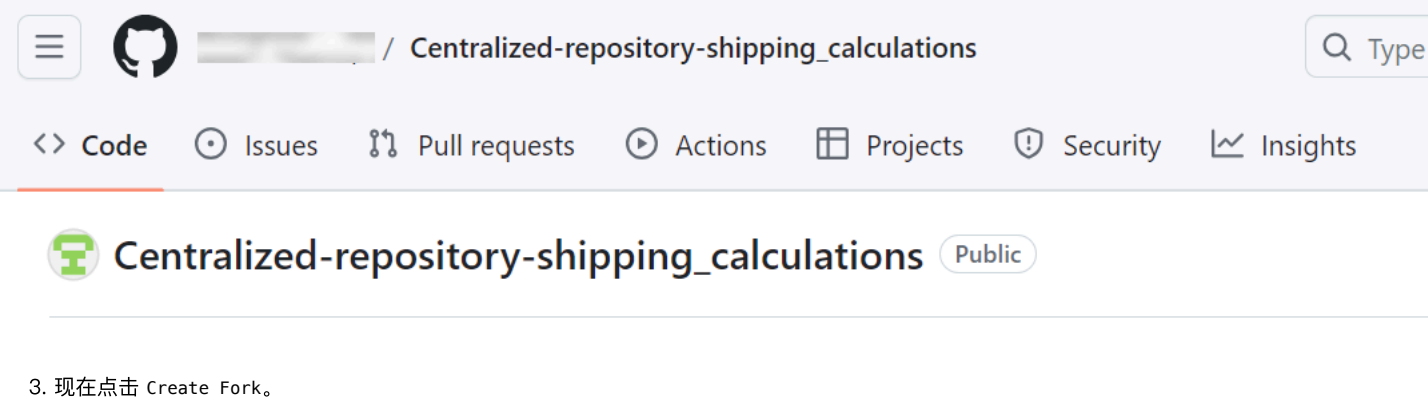
完成本实验后，您将能够展示您可以：

1. 将上游存储库分叉到您自己的账户中。
2. 在实验环境中本地克隆代码。
3. 在存储库中创建一个分支。
4. 将分支合并回 `main` 分支。
5. 从分叉的存储库向上游存储库创建一个拉取请求。
6. 撤销您之前所做的更改。

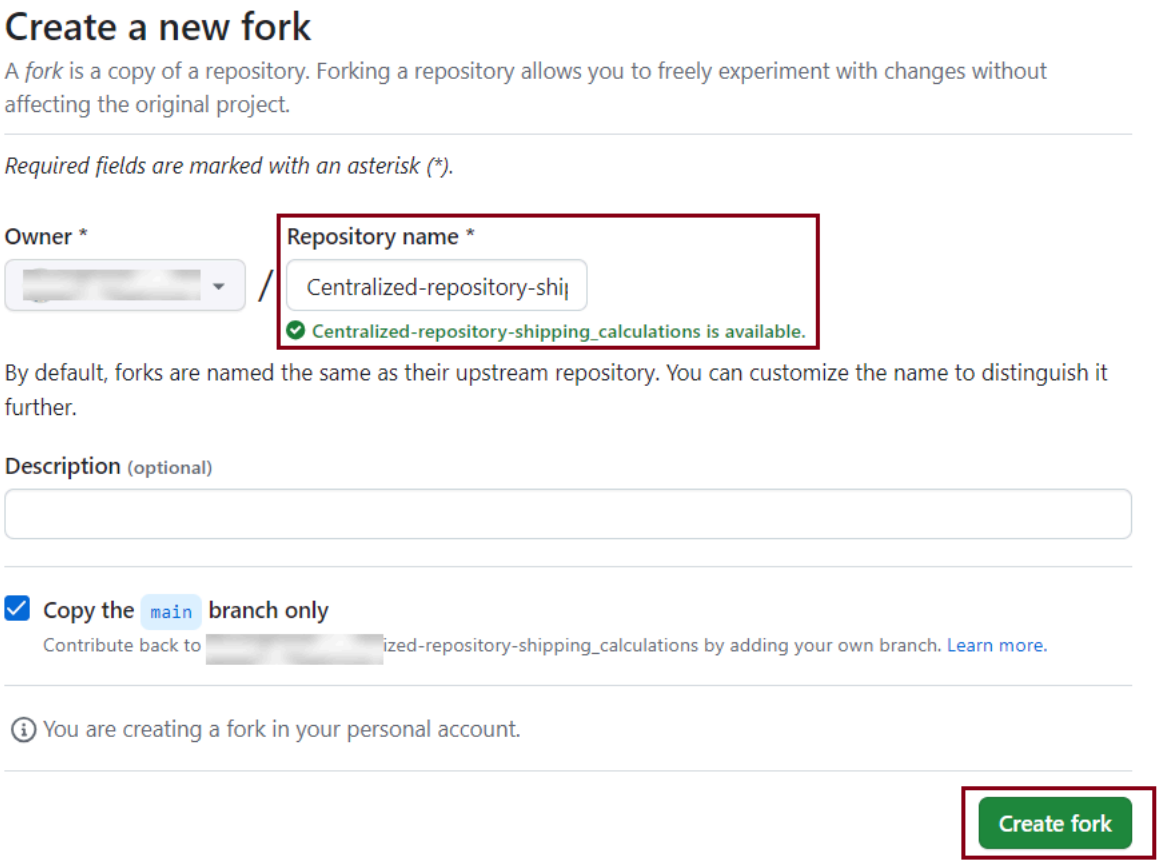
任务 1：分叉仓库

1. 登录 GitHub 并访问 中央仓库

2. 点击 `Fork` 在你的账户中创建一个 GitHub 仓库的个人副本。这允许你独立进行更改。



3. 现在点击 `Create Fork`。



注意：在实际场景中，你将有灵活性更改仓库名称。然而，在这个实践项目中，建议保持相同的仓库名称，以确保你正确完成项目并增强你的理解。

任务 2：创建一个新分支并提交第一个拉取请求

1. 打开终端并在实验环境中克隆你的分叉仓库。确保将 <your GitHub username> 替换为你的实际 GitHub 用户名。

   ```
   git clone https://github.com/<your GitHub username>/Centralized-repository-shipping_calculations.git
   ```

2. 使用 `cd` 命令切换到 **Centralized-repository-shipping_calculations** 目录。

   ```
   cd Centralized-repository-shipping_calculations
   ```

3. 使用 `git checkout -b` 命令创建并切换到一个名为 **shipping_calculator_fixes** 的新分支。

   ```
   git checkout -b shipping_calculator_fixes
   ```

- 验证当前分支并使用命令 `git branch` 查看所有分支。

```
theia@theia-ksundararaja:/home/project$ cd Centralized-repository-shipping_calculations
theia@theia-ksundararaja:/home/project/Centralized-repository-shipping_calculations$ git checkout -b shipping_calculator_fixes
Switched to a new branch 'shipping_calculator_fixes'
theia@theia-ksundararaja:/home/project/Centralized-repository-shipping_calculations$ git branch
  main
* shipping_calculator_fixes
theia@theia-ksundararaja:/home/project/Centralized-repository-shipping_calculations$
```

4. 将以下行插入到 `Shipping_Cost_Calculator.py` 文件中，将 <your GitHub username> 替换为您的真实 GitHub 用户名。

   ```
   # Here is a new update by <your GitHub username>
   ```

5. 要配置您的 Git 用户信息，请执行以下命令：

   ```
   git config --global user.email "your-email@example.com"
   git config --global user.name "YourName"
   ```

   注意：请确保将 "your-email@example.com" 替换为您实际的 GitHub 邮箱 ID，将 "YourName" 替换为您实际的 GitHub 用户名。

6. 添加并提交您所做的代码更改，并检查提交状态：

   ```
   git add .
   git commit -m "added a new line to Shipping_Cost_Calculator.py"
   git status
   ```

- `git add`: 暂存对 Shipping_Cost_Calculator.py 所做的更改以待下次提交。

- `git commit`: git commit 记录在 Shipping_Cost_Calculator.py 中暂存的更改以及指定的消息（'向 Shipping_Cost_Calculator.py 添加了一行'）。
- `git status`: git status 显示仓库中 Shipping_Cost_Calculator.py 的当前状态和更改。

7. 使用下面的命令将您在实验环境或本地系统 IDE 中所做的更改推送到您的 Github 仓库中的 `shipping_calculator_fixes` 分支，该分支被跟踪为上游分支。

   ```
   git push --set-upstream origin shipping_calculator_fixes
   ```

   `--set-upstream`:
   这个标志用于建立本地分支与远程分支之间的跟踪关系，允许你在每次使用 *git push* 和 *git pull* 时无需指定远程分支。

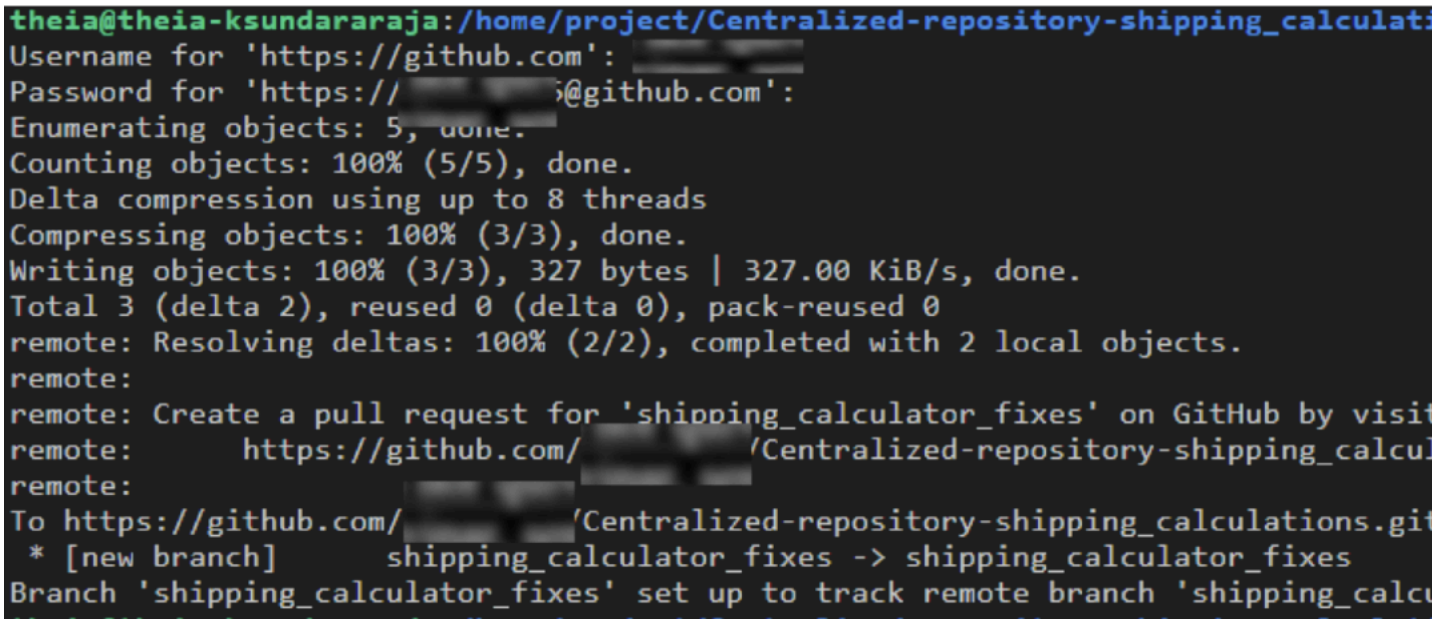
   `origin`:
   这是默认给远程仓库的名称，你最初从中克隆本地仓库。

   `shipping_calculator_fixes`:
   这是你想要推送到 origin（即远程）仓库的本地分支名称。它代表你在本地所做的更改，并希望在远程服务器上更新。

- 在提供的命令中，你正在将 *shipping_calculator_fixes* 分支（在你的实验环境或本地系统 IDE 中创建）中的更改推送到你在任务 1 中分叉的远程仓库 (`https://github.com/<your GitHub username>/Centralized-repository-shipping_calculations.git`)。
- 命令中的 `--set-upstream` 标志在 *shipping_calculator_fixes* 分支（在实验环境或你的本地系统中）与 origin 仓库中同名的远程分支之间建立了跟踪链接。这简化了未来的操作，消除了每次运行 `git pull` 或 `git push` 时显式提及远程分支名称（即 *shipping_calculator_fixes*）的要求。

▶ **请按照此处提到的适当流程进行操作：**

这确保了你成功将更改推送到远程 origin 仓库。



8. 生成并合并拉取请求，以将 `shipping_calculator_fixes` 分支中的更改合并到你的 `main` 分支中。

   注意： 在过去，你的 GitHub 仓库中的默认分支使用名称 `master`。自 2020 年 10 月 1 日起，所有新的 GitHub 仓库使用更具包容性的术语 `main` 作为默认分支的名称，而不是 `master`。

- 请参考 实践项目第一部分 的任务 7 和 8 来完成此过程。

▶ 你的拉取请求应如下所示：

任务 3：提交另一个更新并撤回它

1. 将以下注释添加到 `Shipping_Cost_Calculator.py` 文件中。

   ```
   # Here is another update by <your GitHub username>
   ```

2. 添加并提交新更改：

▶ 点击这里查看解决方案

然而，你现在认为这条最近添加的行是不必要的，你可以稍后再添加。

- 一种方法是删除你添加的行，然后再次包含并提交更改。
- **相反，你现在将使用 `git revert` 命令来实现相同的功能。**

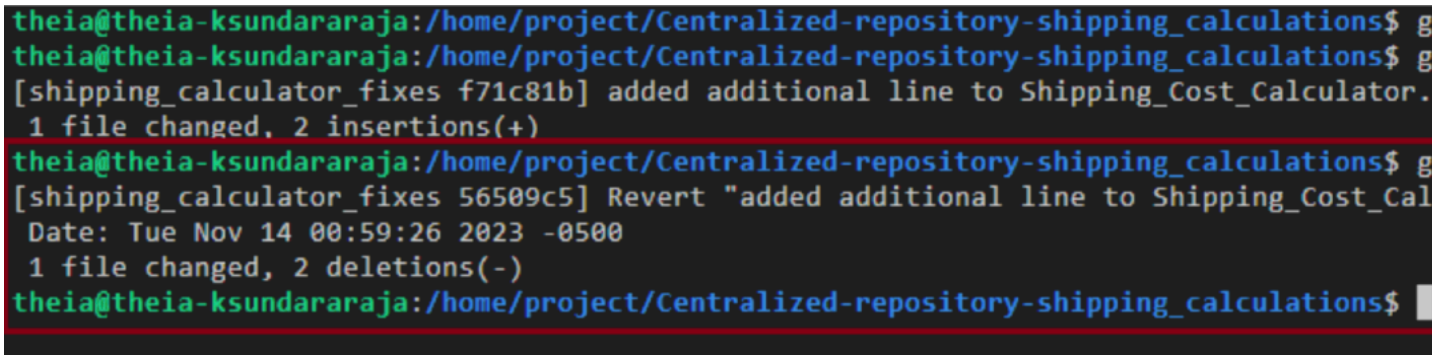
3. 执行如下的 `git revert` 命令：

```
git revert HEAD --no-edit
```

- `git revert`: 是一个 Git 命令，用于通过创建一个表示指定提交逆操作的新提交来撤销之前的提交。
- `HEAD`: 这是指当前分支中的最新提交。
- `--no-edit`: 此选项用于执行撤销操作，而不打开默认文本编辑器来编辑提交信息。

运行此命令后，Git 将自动生成一个提交信息，表明这是一次撤销操作，并将创建一个新的提交以撤销更改。`--no-edit` 选项防止 Git 打开编辑器供你修改提交信息；相反，它使用默认的撤销信息。

4. 这将导致以下输出：



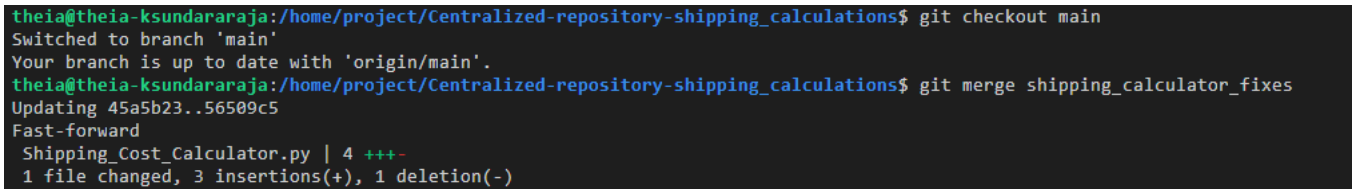
- 你将观察到额外的提交已有效撤销。如果检查 `Shipping_Calculator.py` 文件，你添加的额外行将不再存在。
- 现在，你已回到初始提交，并可以继续将其推送到你的代码库。

5. 使用 `git checkout` 命令切换到主分支：

```
git checkout main
```

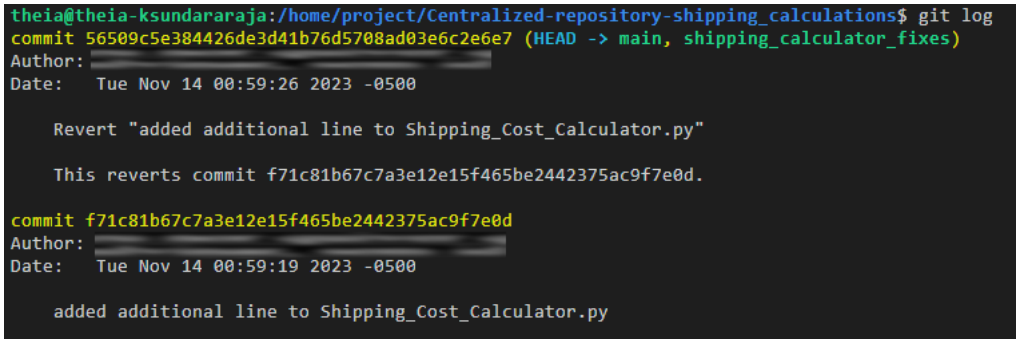
6. 执行 `git merge` 命令将 `shipping_calculator_fixes` 分支的更新推送到你的 `main` 分支。

```
git merge shipping_calculator_fixes
```



7. 运行 `git log` 命令以获取所有提交的时间顺序记录。这将显示有关通过 `git revert` 命令添加并随后撤销的行的信息。

```
git log
```



注意：要退出 git log 命令，只需按“Q”键。此操作将关闭日志视图并返回命令提示符。

通过此操作，`shipping_calculator_fixes` 分支的所有提交已成功推送到您分叉的仓库中的 `main` 分支。

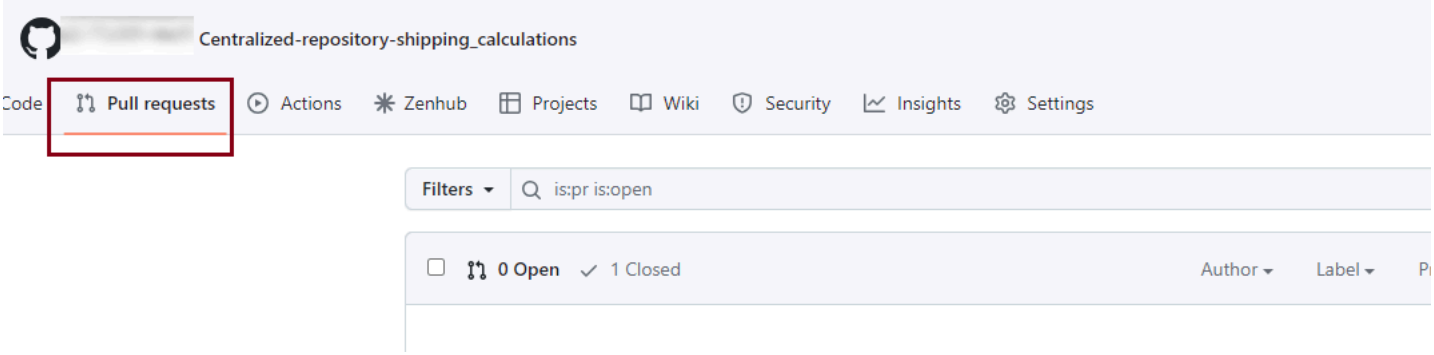
任务 4：从您的仓库向集中协作仓库提交拉取请求

现在，您将把您的项目贡献给公司当局考虑并接受的集中协作仓库。

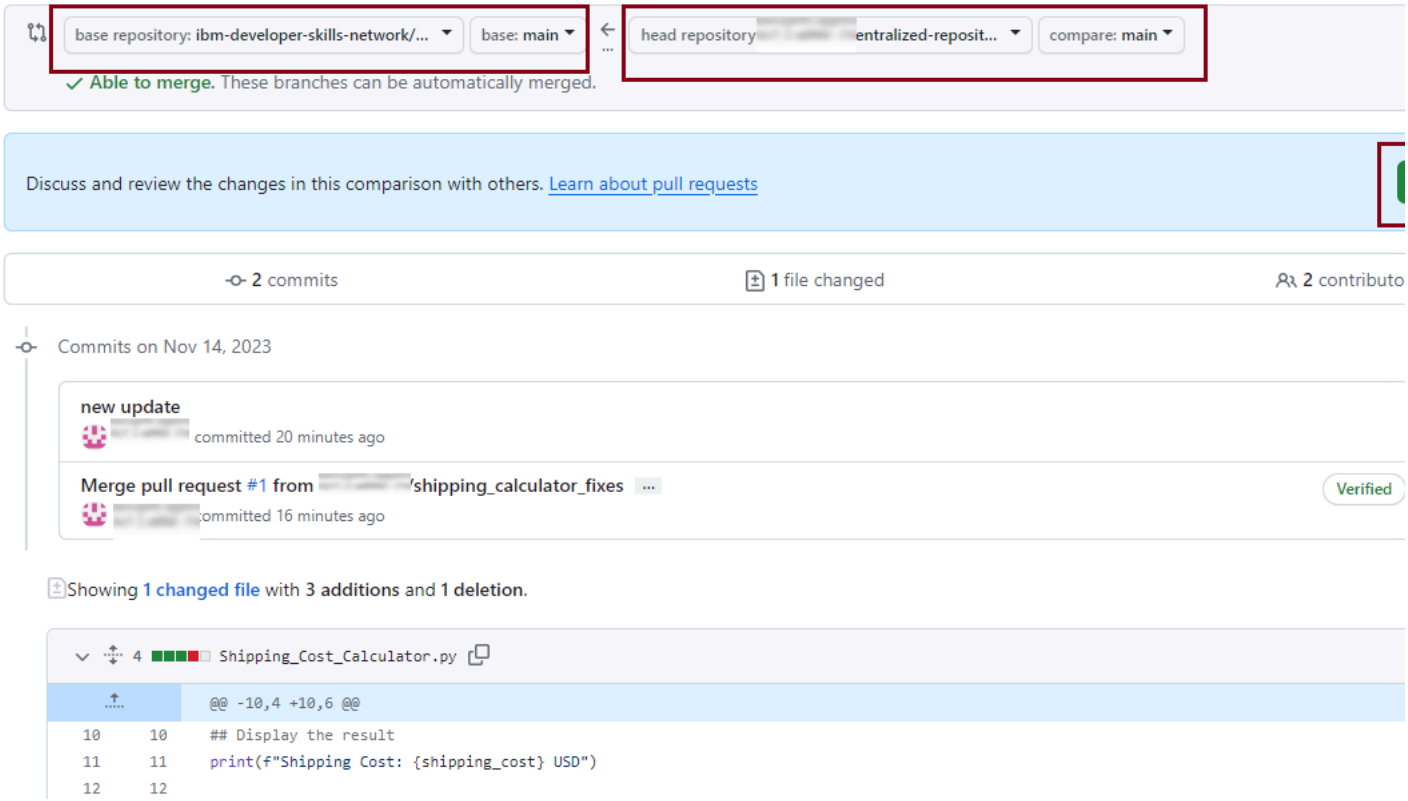
由于这是集中仓库，只有公司当局拥有 `write` 和 `Admin` 访问权限。

您只能提交拉取请求，没有合并的权限。

1. 在您的 GitHub 仓库中，导航到 `Pull requests` 并点击 `New pull request`。

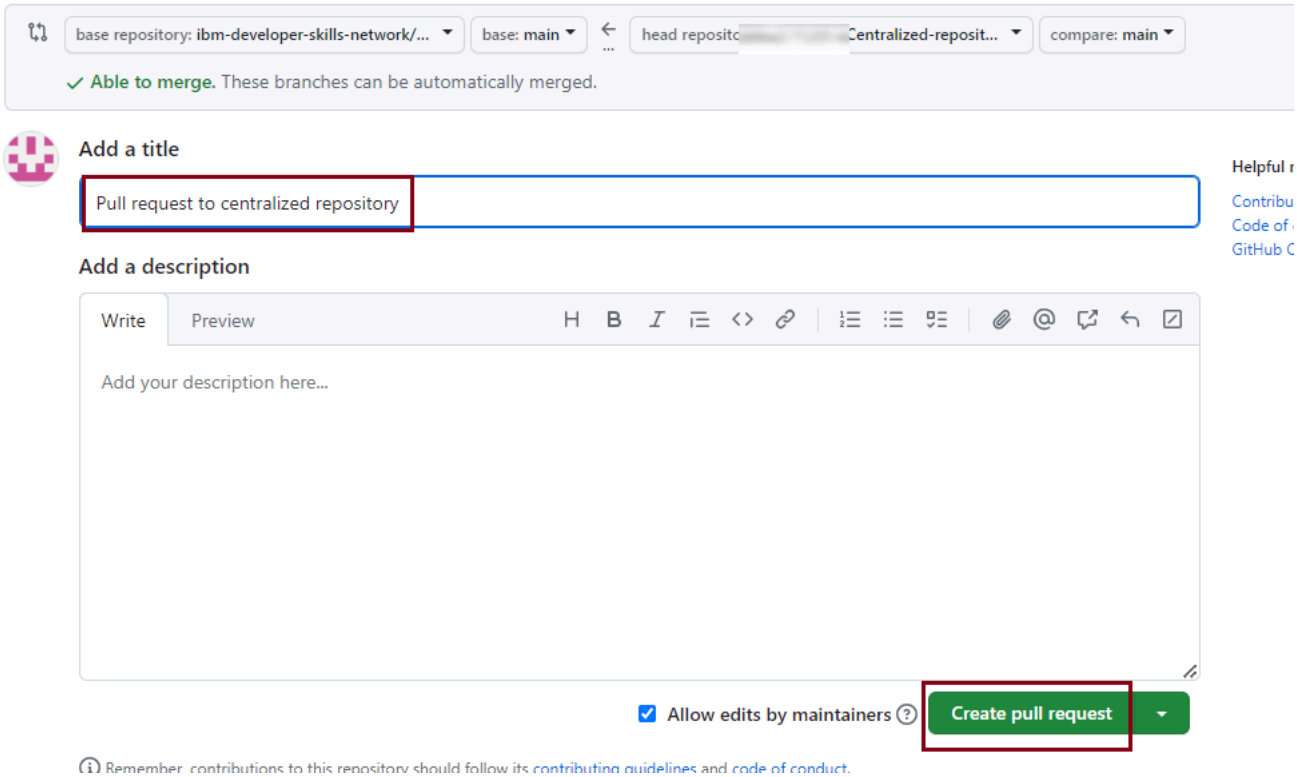


2. 选择 `compare across forks`，从下拉菜单中选择相关字段，然后点击 `Create pull request`。

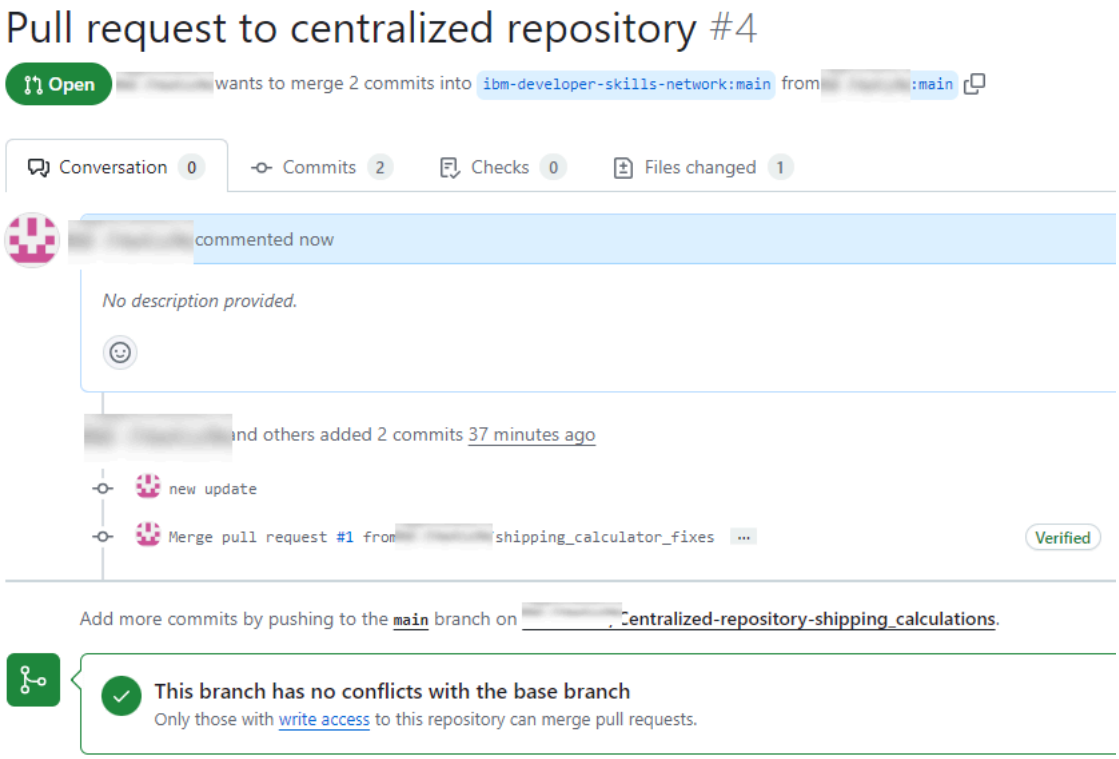


- `head repository` 和 `compare` 字段分别指您希望发起拉取请求的仓库 URL 和分支。在这个实例中，它是您的 GitHub 仓库。
- `base repository` 和 `base` 字段分别指您打算提交拉取请求的上游仓库的 URL 和分支。在这个实例中，它是您在任务 1 中分叉的集中仓库。

3. 添加适当的评论，然后点击 `Create pull request`。



4. 显示的界面表明您已成功将最新更新提交到公司的集中仓库以进行协作。



5. 拉取请求将接受审核，并随后由仓库管理员合并。

总结

在这个实验中，您将代码库分叉到您的账户并在本地克隆。然后，您创建了一个新分支以进行更改。在完成修改后，您将该分支合并回 `main`。您还学习了如何创建拉取请求以进行协作。最后，您探索了在需要时撤销更改的过程。