**Цели проекта:**

1. В заданные сроки командно реализовать качественную верстку одного из макетов.

* Срок выполнения проекта для групп Online - 7 дней *(например, понедельник вечером начало проекта - в следующий понедельник вечером презентация проекта)*.
* Срок выполнения проекта для групп Bootcamp - 5 дней *(например, понедельник утром начало проекта - пятница вторая половина дня презентация)*.

1. Получение практики работы в командах:

- опыт работы со сборщиком проектов Parcel;  
- командное взаимодействие и работа c Git;

- навык ветвления и слияния проекта;

- решение конфликтов после слияния веток;

- опыт планирования и постановки задач;

- опыт работы с Trello *(*[*https://trello.com/*](https://trello.com/)*);*  
- опыт на позиции team lead команды *(*[*https://dou.ua/lenta/articles/team-lead-position/*](https://dou.ua/lenta/articles/team-lead-position/)*)*;  
- опыт на позиции scrum master команды *(*[*https://dou.ua/lenta/articles/scrum-master-position/*](https://dou.ua/lenta/articles/scrum-master-position/)*)*;

- опыт презентации проектов.

**Роли:**

**Team lead**. Следит за технической реализацией проекта, отвечает за качество кода, пишет код ревью и задачи по проекту для участников команды. Является владельцем репозитория по командному проекту. Напрямую общается с проектным ментором по вопросам реализации проекта и возникающим у команды проблемам.

**Scrum master**. Регулирует сроки выполнения проекта. Проводит ежедневный Stand Up для планирования работы команды. Для проведения Stand Up можно использовать Google Meet, Zoom, Skype, Discord. Следит за наличием задач у всех участников команды и сроками их реализации. Напрямую общается с проектным ментором по вопросам планирования и срокам выполнения проекта.  
  
**Проектный ментор**. Является наставником во время проведения проекта. Задачей проектного ментора является консультирование по вопросам технической реализацией проекта, выбору инструментов, технологий и планированию работы на проекте. Также ментор ревьюирует предварительную версию проекта до его презентации.

**День 0. Планирование.**

1. Студенты распределяются по командам клиент-менеджером.
2. Каждая команда получает отдельный канал в *slack* и ссылку на доску в *trello.*
3. В трелло нужно выполнить карточки маркированные красным.

* выбрать название команды;
* выбрать тимлида;
* выбрать скрам-мастера;
* просмотреть материалы по работе сборщика*;*
* просмотреть материалы по командной работе в GitHub .

**Все необходимые материалы по проекту находятся в trello в колонке “Материалы”:**

* ссылка на инструкцию по работе на проекте ( как добавлять в репозиторий участников проекта, установка сборки и работа с Git);
* видео по командной работе в GitHub;
* видео по работе сборщика;
* ссылка на репозиторий сборки;
* ссылка на макет;
* ссылка на техническое задание к макету.

1. *Посетить вводный вебинар по теме "Проект".*

**День 1. Старт проекта.**

1. Выполнить задачи из trello маркированные оранжевым и желтым. Последующие задачи для команды добавляет тимлид или скрам-мастер. Участники проекта также могут добавлять свои текущие задачи в trello самостоятельно.
2. Каждый участник проекта выбирает себе задачу, отмечает себя на карточке с задачей в trello, указывает предварительные сроки выполнения задачи и перемещает ее в колонку “В работе”. После выполнения задачи, карточка перемещается в колонку “Ожидает проверки”. После проверки задачи тимлидом, карточка перемещается в колонку “Готова”. Если задача требует доработки, участник команды снова возвращает её в колонку “В работе”.
3. Команда проводит ежедневный Stand Up длительностью не более 15 минут.
4. Каждый разработчик на Stand Up отвечает на три вопроса: Что он сделал вчера (сегодня)? Что будет делать сегодня (завтра)? В чем нужна помощь?

**!!!Важно!!!** на Stand Up мы не начинаем разбирать те моменты где нужна помощь, мы говорим о них, чтобы вся команда понимала как проходит процесс выполнения проекта. После проведения Stand Up разработчик, которому нужна помощь может остаться и разобрать свои вопросы с членом команды, который знает ответ или с Тимлидом.

1. Посетить созвон с ментором по проекту.

**!!Важно!!!** Подготовить список вопросов к ментору и описание проблем с которыми не удалось справиться самостоятельно. Задача ментора - в режиме онлайн помочь настроить работу сборщика и закрыть во вопросы по технической реализации макета.

**День 2. Работа над проектом.**

1. Тимлид пишет задачи в трелло для команды.
2. Команда проводит ежедневный Stand Up длительностью не более 15 минут.
3. Каждый участник проекта выбирает себе задачу в работу.

**День 3. Работа над проектом.**

1. Тимлид пишет задачи в трелло для команды.
2. Команда проводит ежедневный Stand Up длительностью не более 15 минут.
3. Каждый участник проекта выбирает себе задачу в работу.

**День 4. Работа над проектом.**

1. Тимлид пишет задачи в трелло для команды.
2. Команда проводит ежедневный Stand Up длительностью не более 15 минут.
3. Каждый участник проекта выбирает себе задачу в работу.

**День 5. Demo.**

1. Команда собирает предварительную версию проекта.
2. Команда проводит ежедневный Stand Up длительностью не более 15 минут.
3. Ссылки на исходные файлы проекта и рабочую страницу (github pages) передаются проектному ментору на ревью.
4. Ревью ментором пишется в trello в колонку “Задачи”.
5. Каждый участник проекта выбирает себе задачу в работу.
6. Вечером проводится созвон с проектным ментором, чтобы разобрать вопросы по результатам ревью.

**День 6. Работа над проектом, подготовка к презентации.**

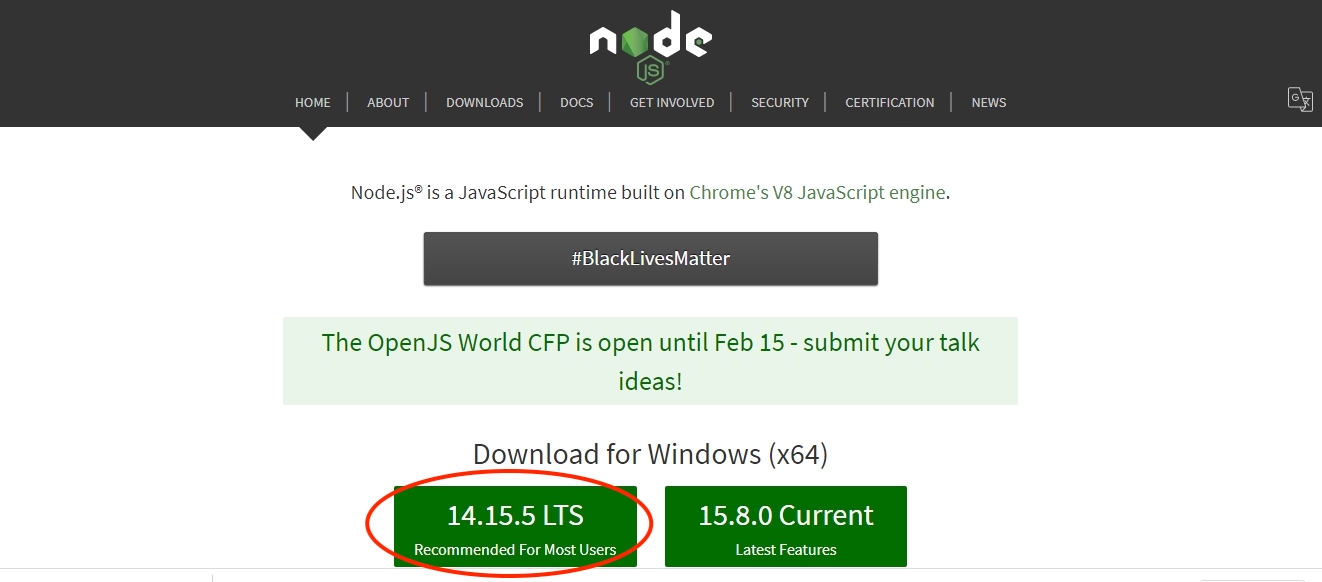
1. Тимлид пишет задачи в трелло для команды.
2. Команда проводит ежедневный Stand Up длительностью не более 15 минут.
3. Каждый участник проекта выбирает себе задачу в работу.
4. Команда готовит презентацию проекта. Длительность презентации не должна превышать 5 минут.

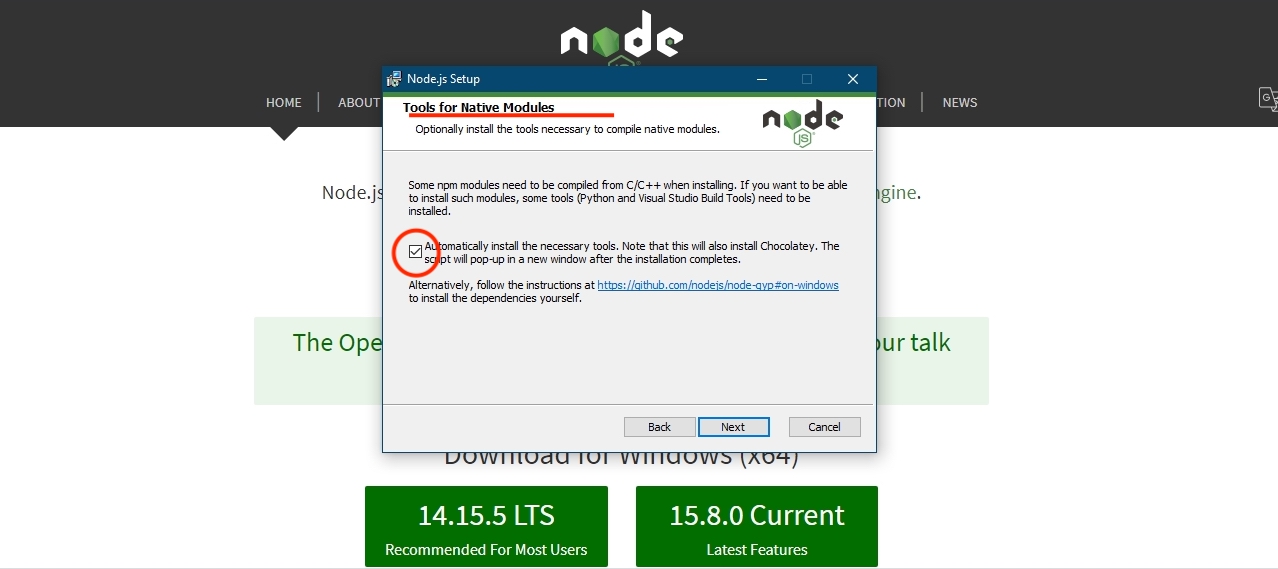
**День 7. Презентация проекта.**

1. Тимлид деплоит финальную версию проекта.
2. Команда проводит ежедневный Stand Up длительностью не более 15 минут.
3. Команда проверяет финальную версию проекта на предмет критических ошибок.
4. Команда репетирует презентацию проекта. Длительность презентации не должна превышать 5 минут.
5. Презентация проекта проводится при участии клиент-менеджера, проектных менторов и всех студентов курса.

# **Установка Node.js**

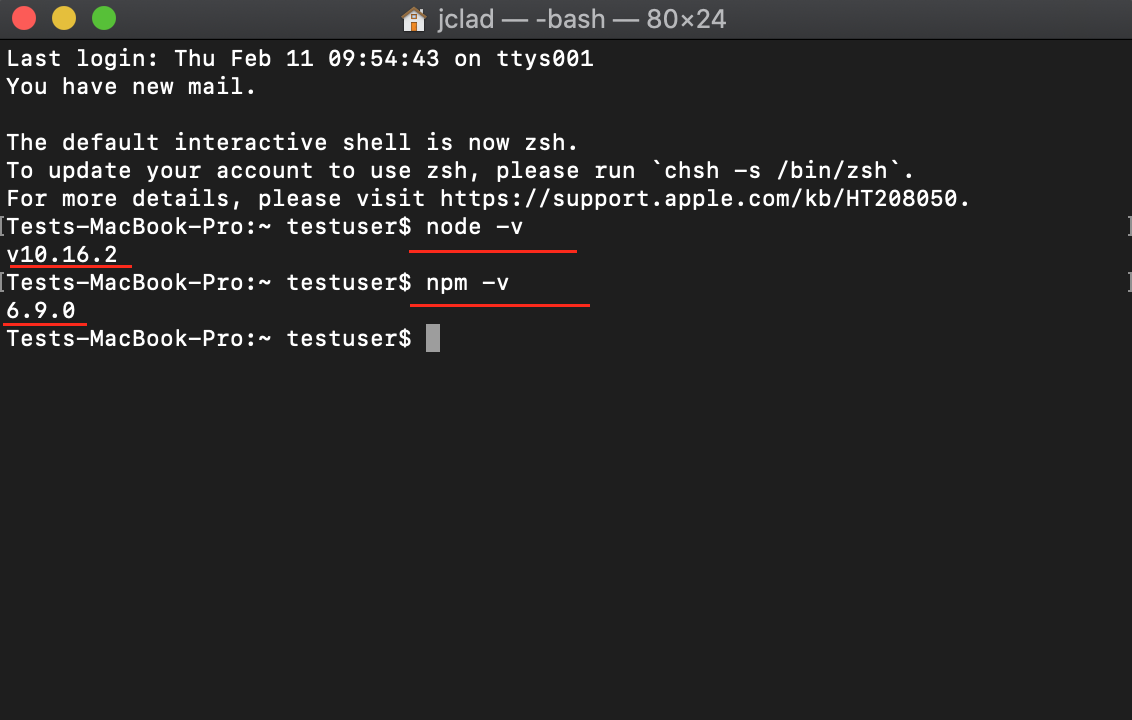
1. Перейти на сайт <https://nodejs.org/>
2. Cкачать стабильную версию Node (LTS).



1. Запустить установку, обязательно поставить галочку *Automatically install the necessary tools*.
2. Если предыдущий пункт был пропущен, пользователям Windows инструменты нужно установить вручную через консоль.

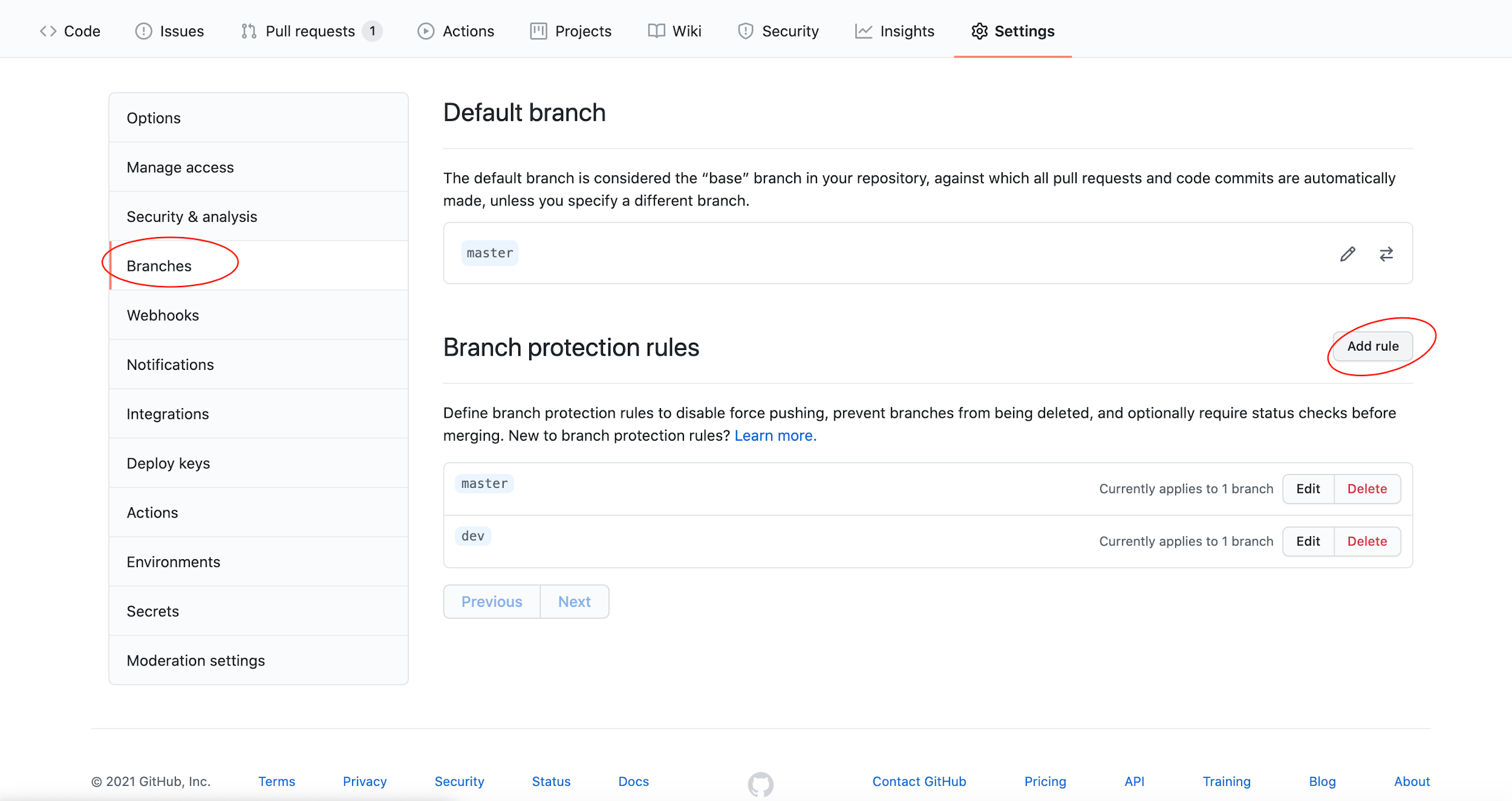
Пользователям Windows, в режиме администратора. [Как запустить Powershell](https://youtu.be/p2tFnxcymwk) в режиме администратора.

**npm install --global --production windows-build-tools**

1. Чтобы проверить установлен ли Node.js введите в консоль команды   
   **npm node -v**  
   и   
   **npm npm -v**  
     
   В ответ на команду отобразиться установленная версия Node и npm.

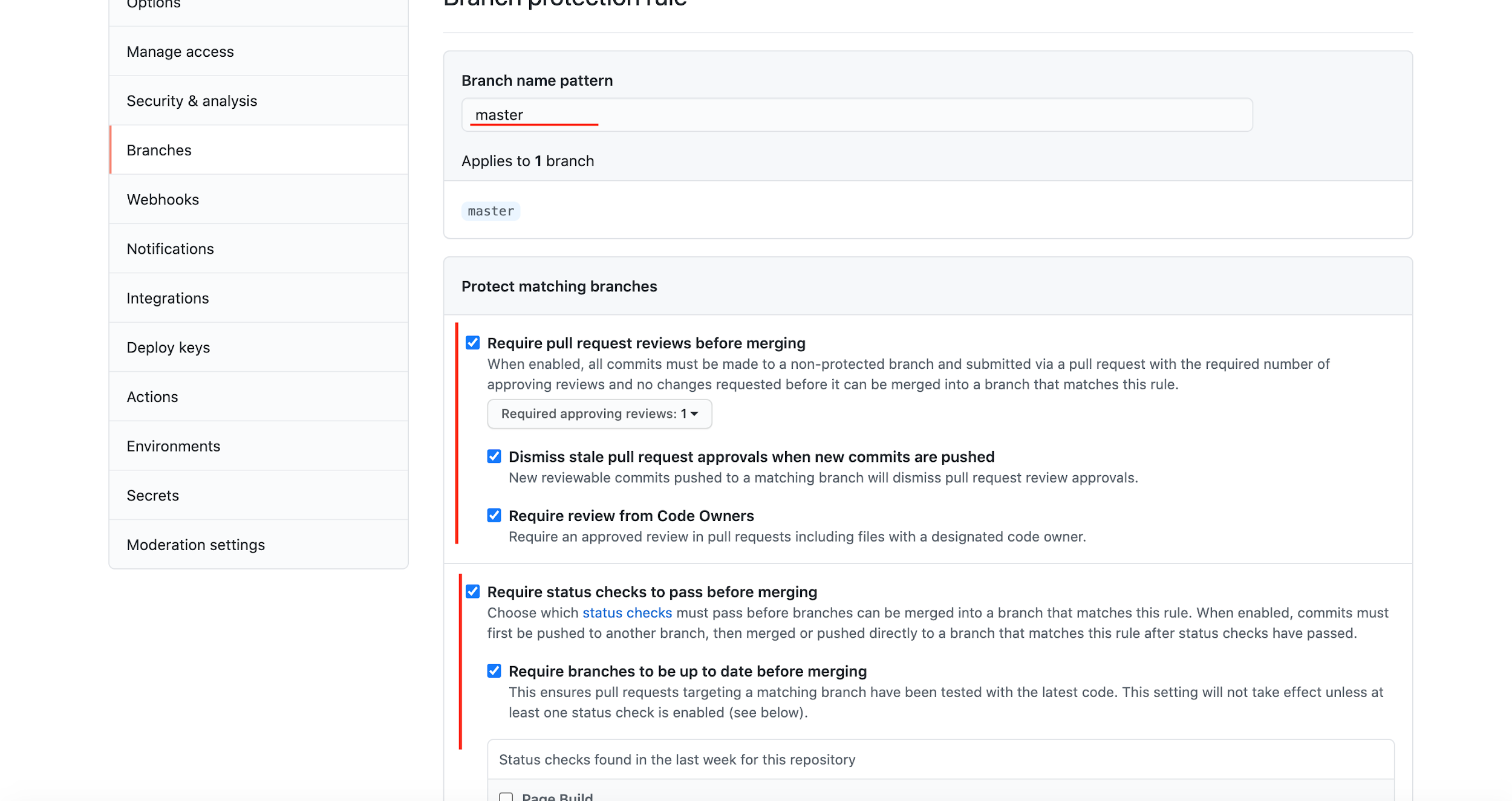
# **Создать репозиторий проекта.**

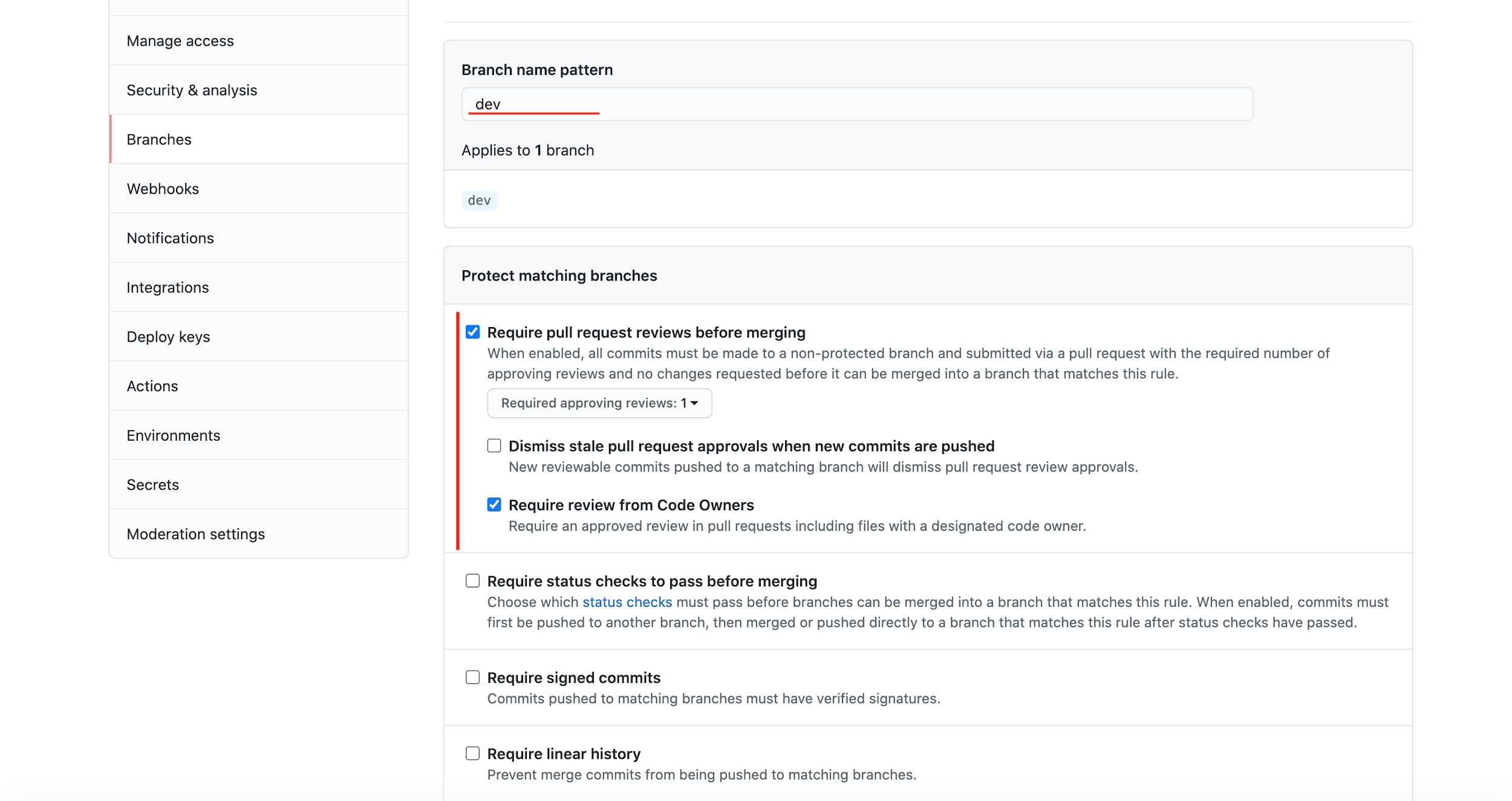
1. Тимлид клонирует сборку для проекта и создает новый репозиторий. <https://github.com/goitacademy/parcel-project-template>
2. Тимлид создает новую ветку dev в репозитории проекта.
3. Тимлид закрывает ветки master(main) и dev от возможности вносить изменения участниками команды без Pull Requests.



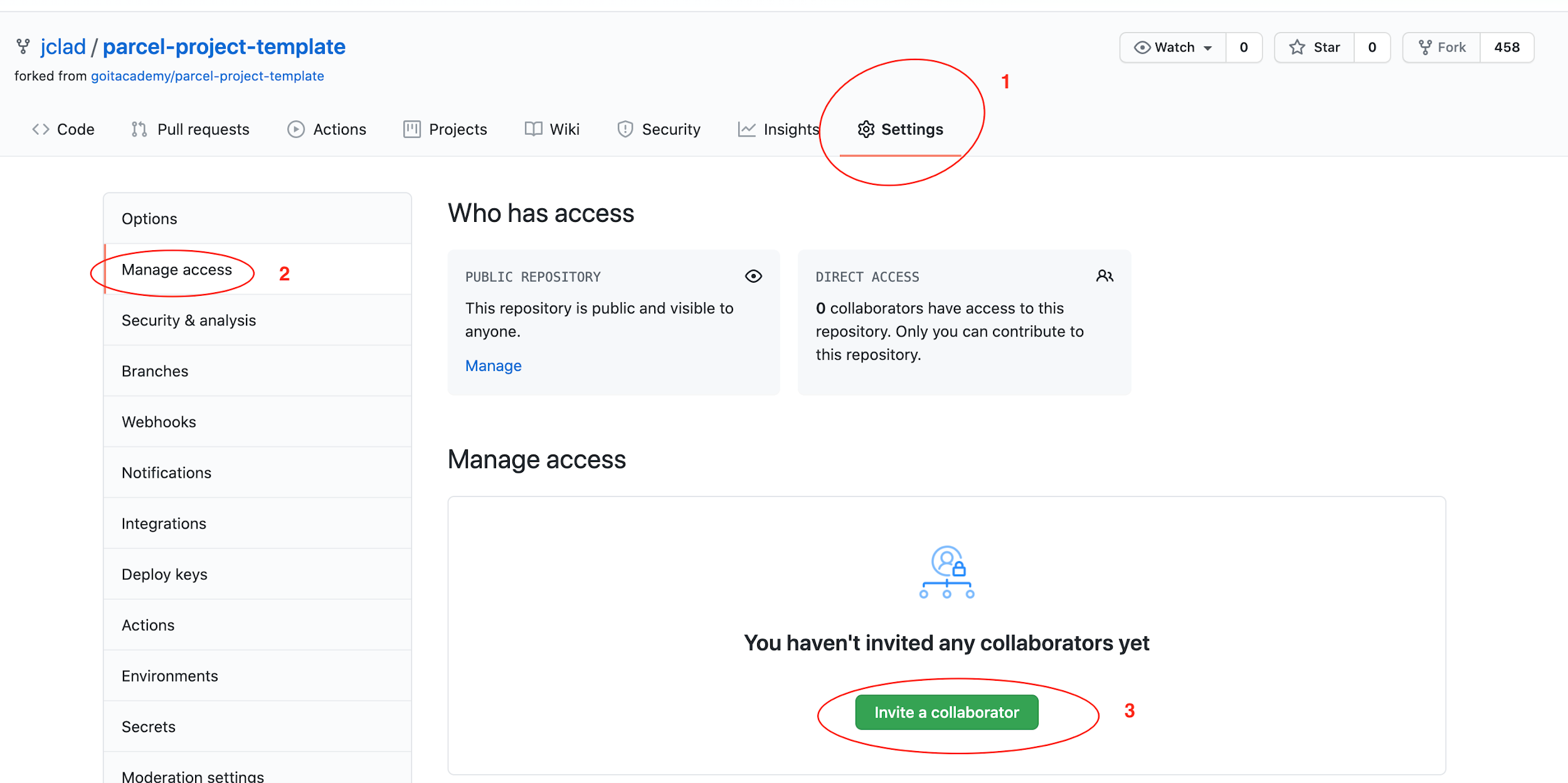
Для этого нужно добавить правила по каждой ветке отдельно.

**Ветка master(main)**

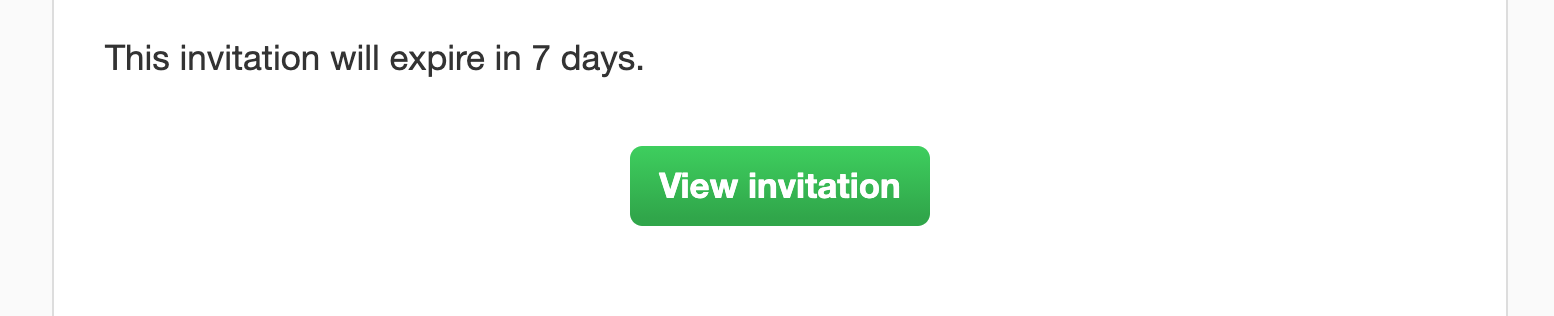


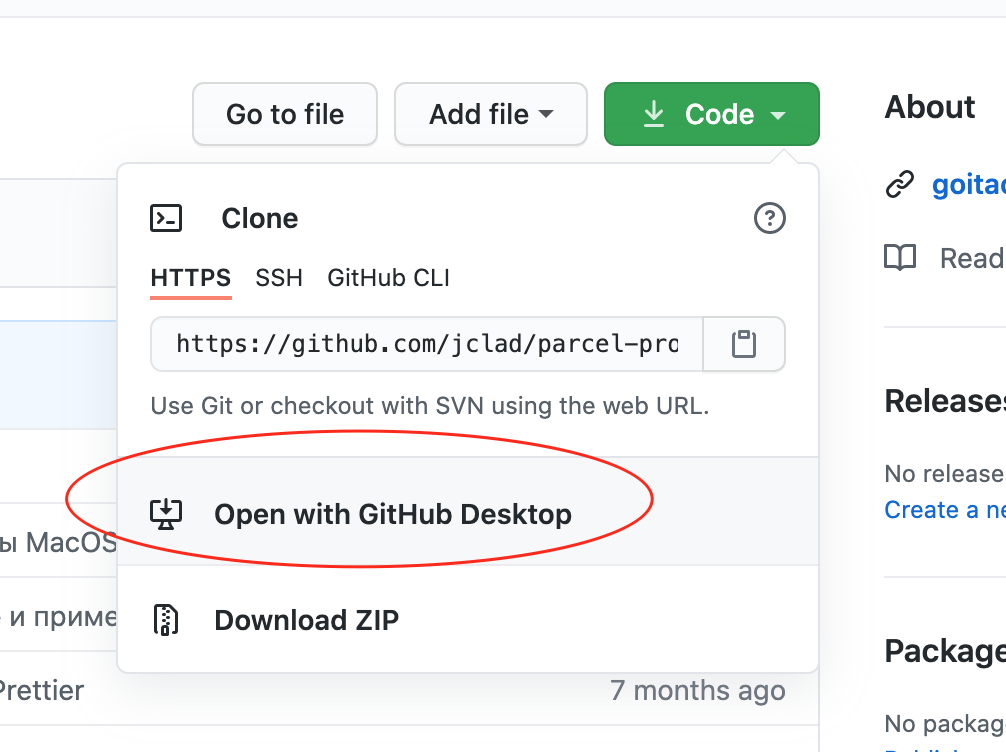
**Ветка dev**

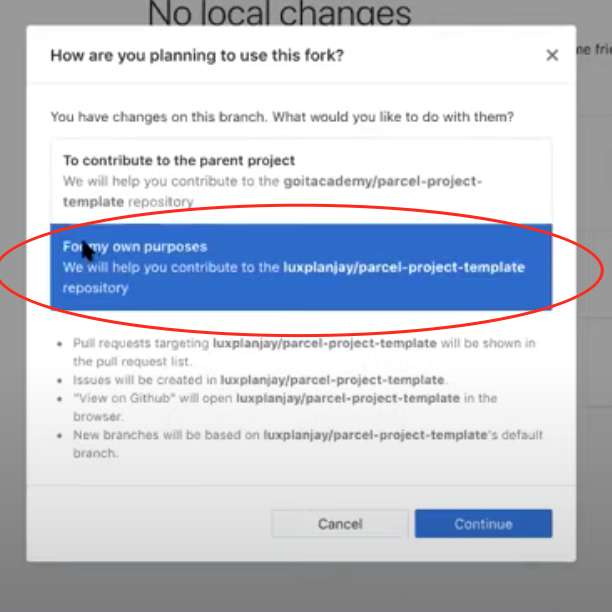
1. Тимлид переходит в свой репозиторий и добавляет участника команды в коллабораторы.

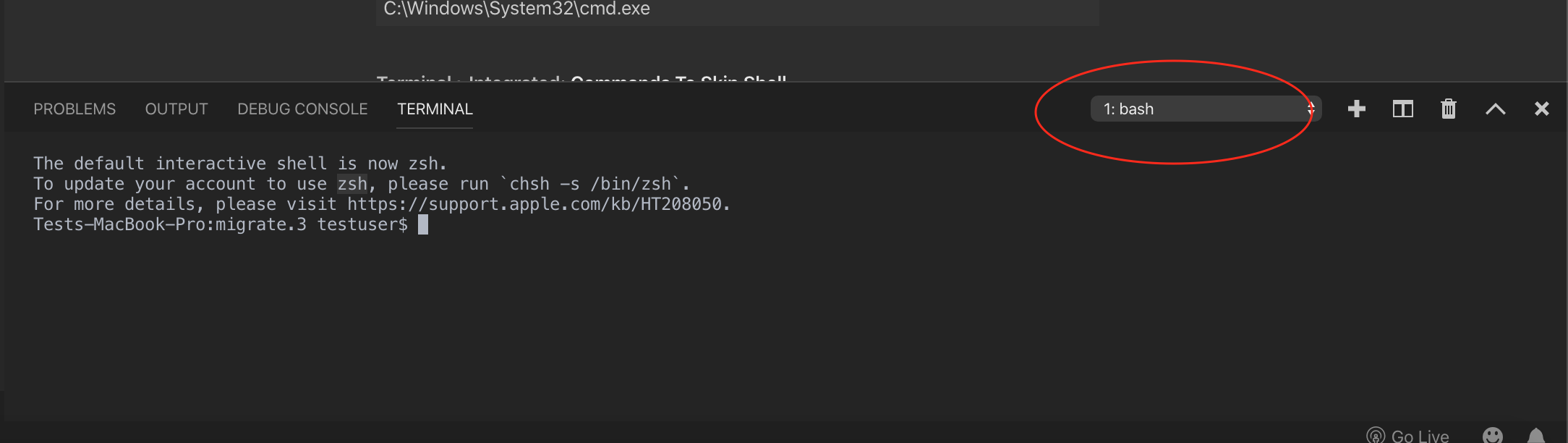


1. Участникам команды на email придут письма с предложением присоединиться к проекту. Нажмите на кнопку в письме, чтобы присоединиться.

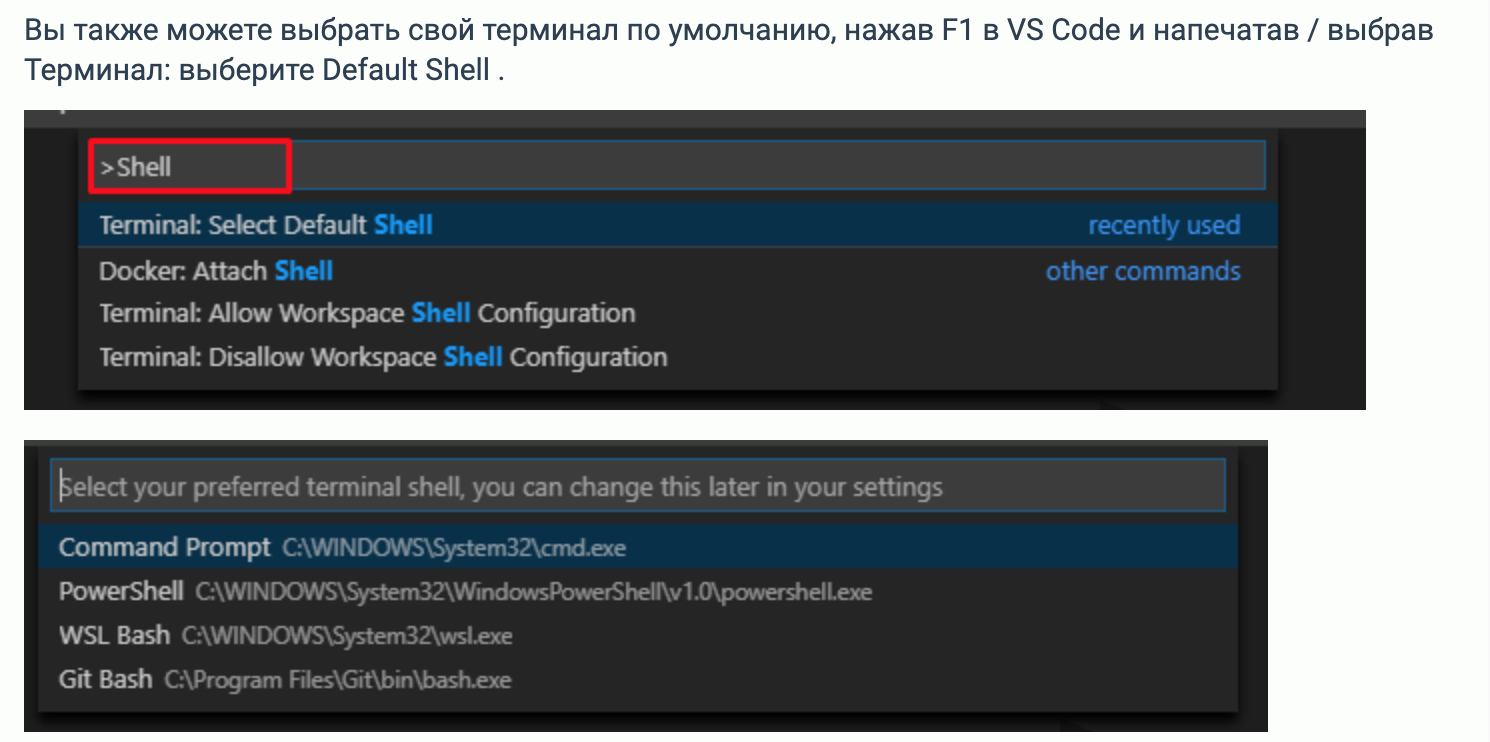


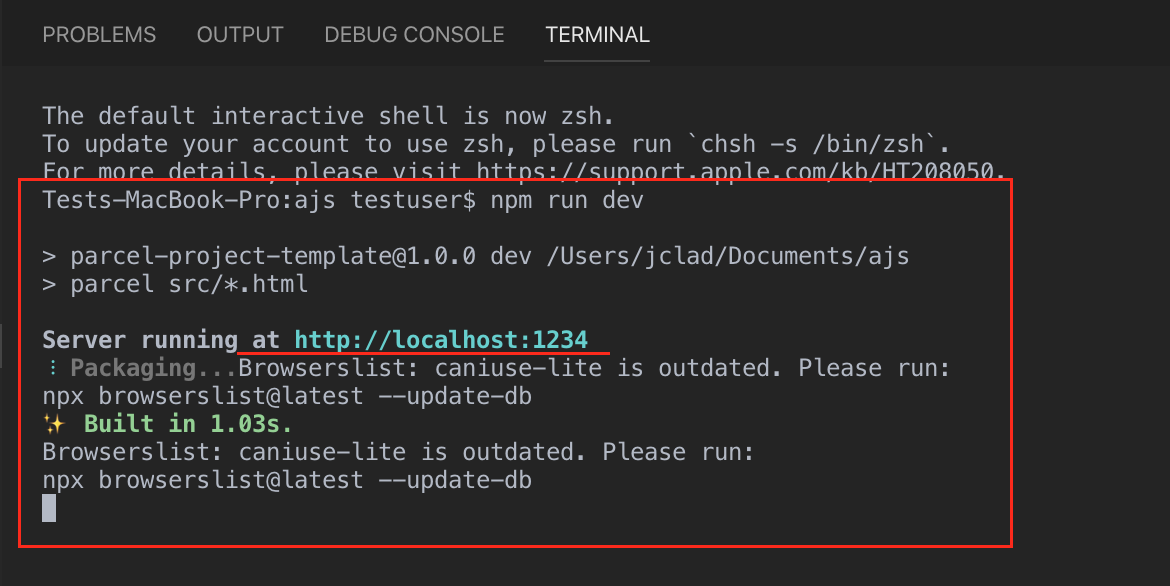
1. Участники команды клонируют себе репозиторий локально.
2. Во время клонирования в GitDesktop появится уведомление с вопросом, как вы хотите использовать проект. Выбирайте пункт “For my own purposes”.



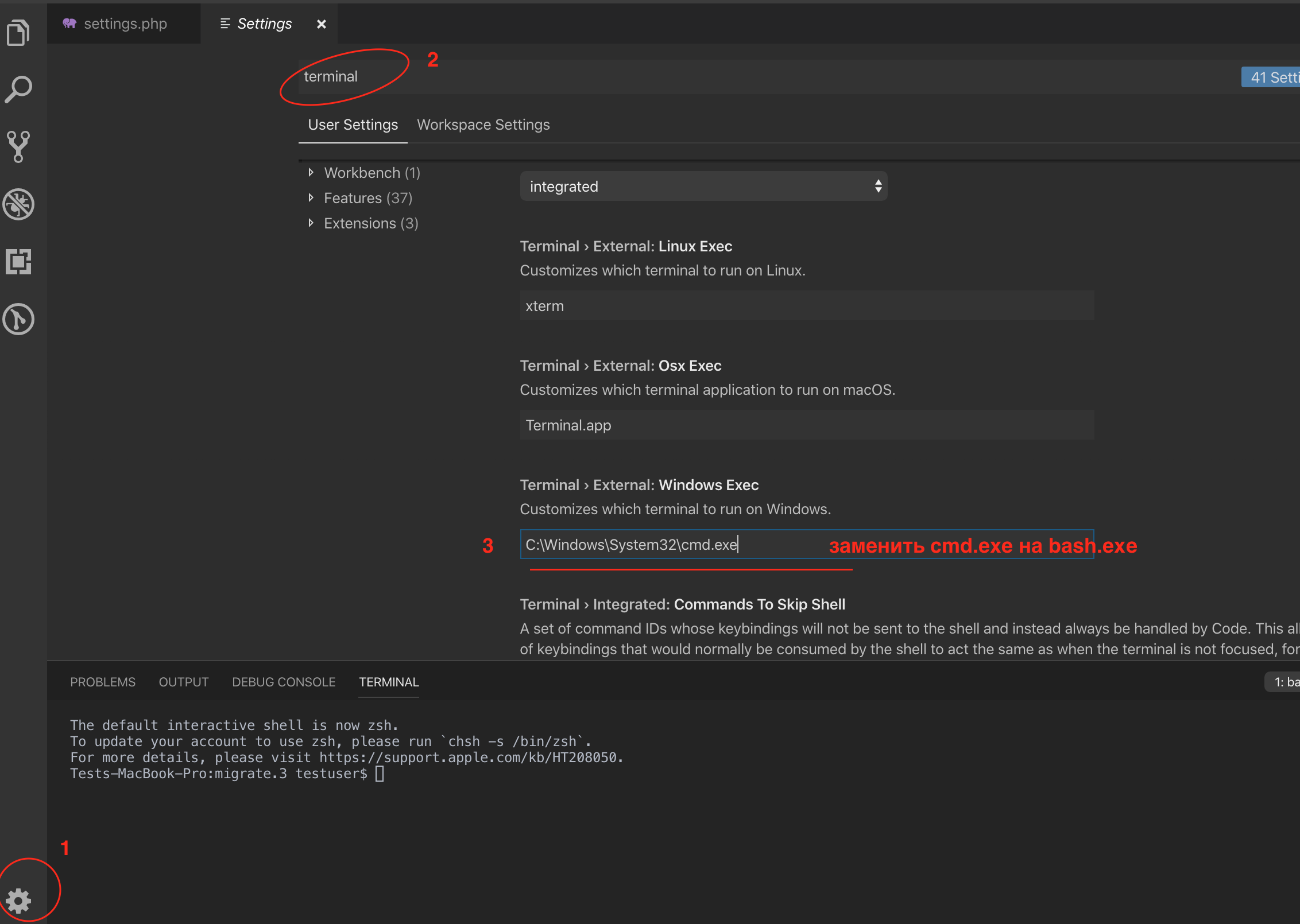
1. **Установка npm пакетов**. Открыть проект в VS Code и запустить команду npm сi. Это нужно сделать один раз, после того как проект был склонирован локально.  
     
   Если установка прошла успешно, в папке проекта появится новая папка node\_modules  
     
   **Примечание: Убедитесь, что вы используете терминал bash или zsh.   
     
   Внимание! При попытке инсталяции npm-пакетов терминал Power Shell может выдавать ошибку. Не используйте его.**

**Изменить терминал по-умолчанию**

Вариант 1.

Вариант 2.  


1. Запустить работу сборщика. Введите в терминал команду **npm run dev.**

В терминале должен появится локальный адрес страницы.  
  
Эта команда является эквивалентом Live Server и Watch Sass. После запуска команды, сборщик будет отслеживать изменения в файлах, обновлять страницу в браузере и компилировать sass.  
  
После окончания работы, чтобы остановить работу сборщика дважды нажмите **ctrl+С**.

1. **Полная инструкция по работе сборки в репозитории** [**https://github.com/goitacademy/parcel-project-template**](https://github.com/goitacademy/parcel-project-template)

## **Основные ветки проекта**

### **Ветка master (main)** В ветке master хранится рабочий исходный код готового проекта.

### **Ветка dev**

Для работы над проектом тимлид создает ветку dev. В эту ветку будут сливаться все изменения по проекту.

### **Ветка gh-pages**

Уже есть в сборке проекта. Самостоятельно создавать её не нужно. В ветке gh-pages хранится готовый проект, который будет отображаться на статической странице gh-pages . Файлы в эту ветку выгружаются автоматически, каждый раз когда обновляется ветка main. Например, после прямого пуша или принятого пул-реквеста. Для этого необходимо в файле package.json отредактировать поле homepage и скрипт *build*, заменив имя\_пользователя и имя\_репозитория на свои.

"homepage": "https://имя\_пользователя.github.io/имя\_репозитория",

"scripts": {

"build": "parcel build src/\*.html --public-url /имя\_репозитория/"

},

*Новая ветка создается под каждую задачу. После того как ветка была смерджена, её нужно удалить локально и на удаленном репозитории*.

## **До старта работы над проектом нужно сделать тестовый раунд, чтобы все разобрались как работать с Git.**

1. Каждый участник команды создает новую ветку

2. Добавляет новый фрагмент в папку partials, в файлы можно вставить любую надпись и подключить фрагмент на страницу index.html

3. Отправить Pull Request  
4. Тимлид принимает все Pull Request

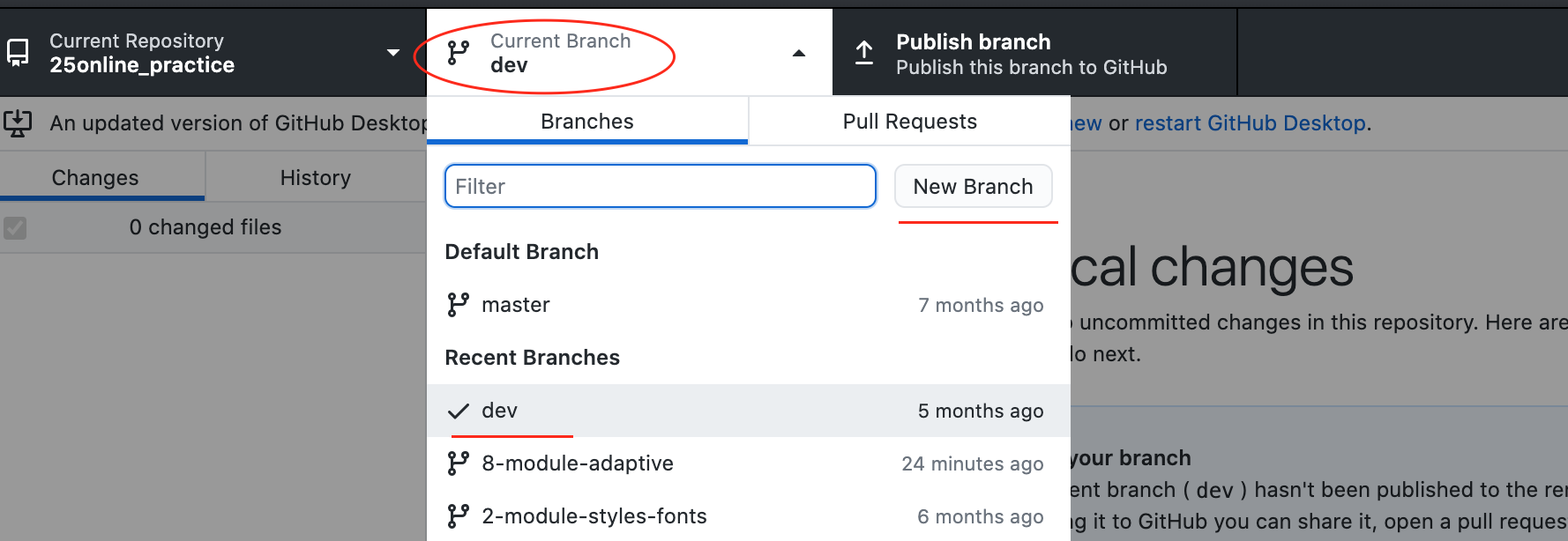
5. После этого нужно вернуться в ветку dev и стянуть изменения с удаленного репозитория.

## **Ветвление и слияние проекта**

**Получить изменения с удаленного репозитория.**

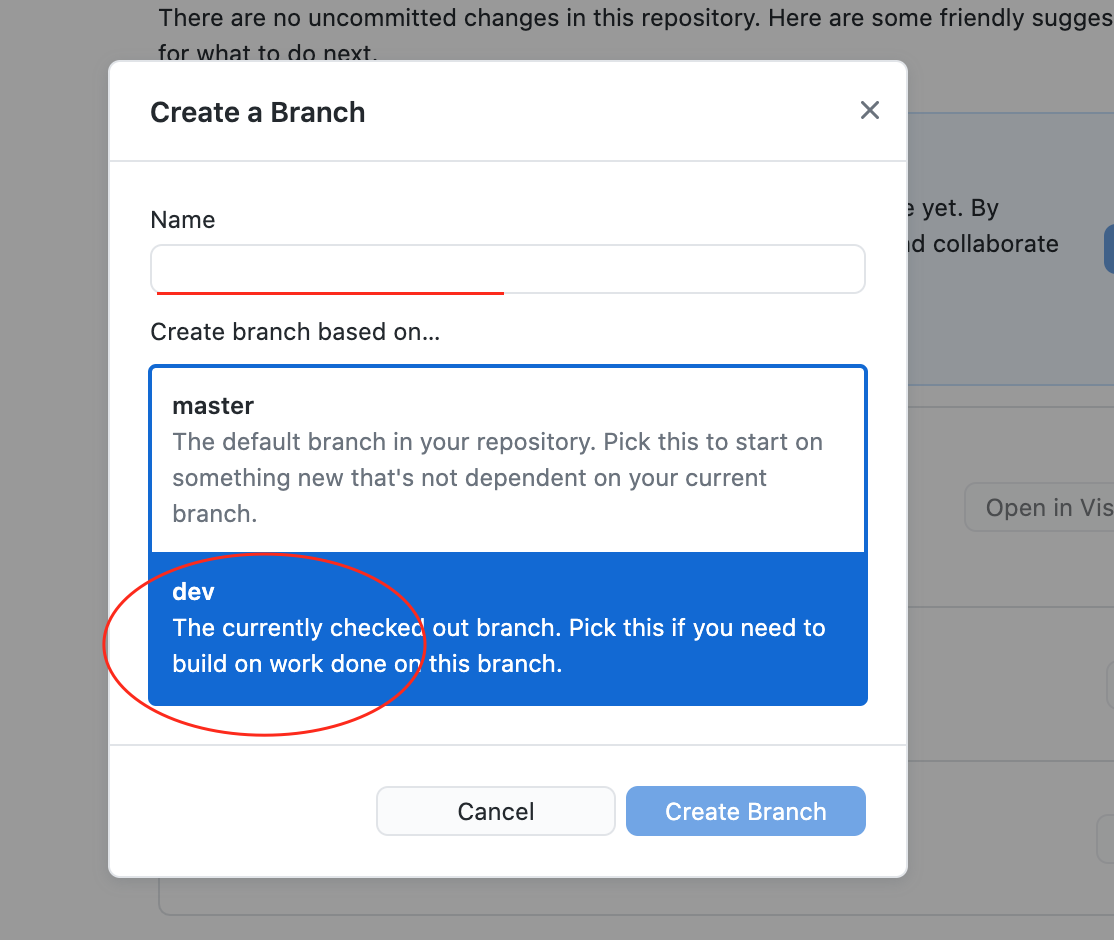
  
  
  
Для тех кто использует консольные команды:  
  
- *git fetch* - стягивает все изменения из удаленного репозитория  
- *git pull* - стягивает изменения с удаленного репозитория по текущей ветке

**Создать новую ветку**

**-** Перейдите в ветку dev (или в ветку от которой вы хотите создать новую).  
- Нажмите *Fetch origin*, чтобы стянуть все изменения с удаленного репозитория.

- Нажмите *New Branch*

- Выберите ветку от которой вы хотите создать новую, введите имя новой ветки и нажмите *Create Branch*

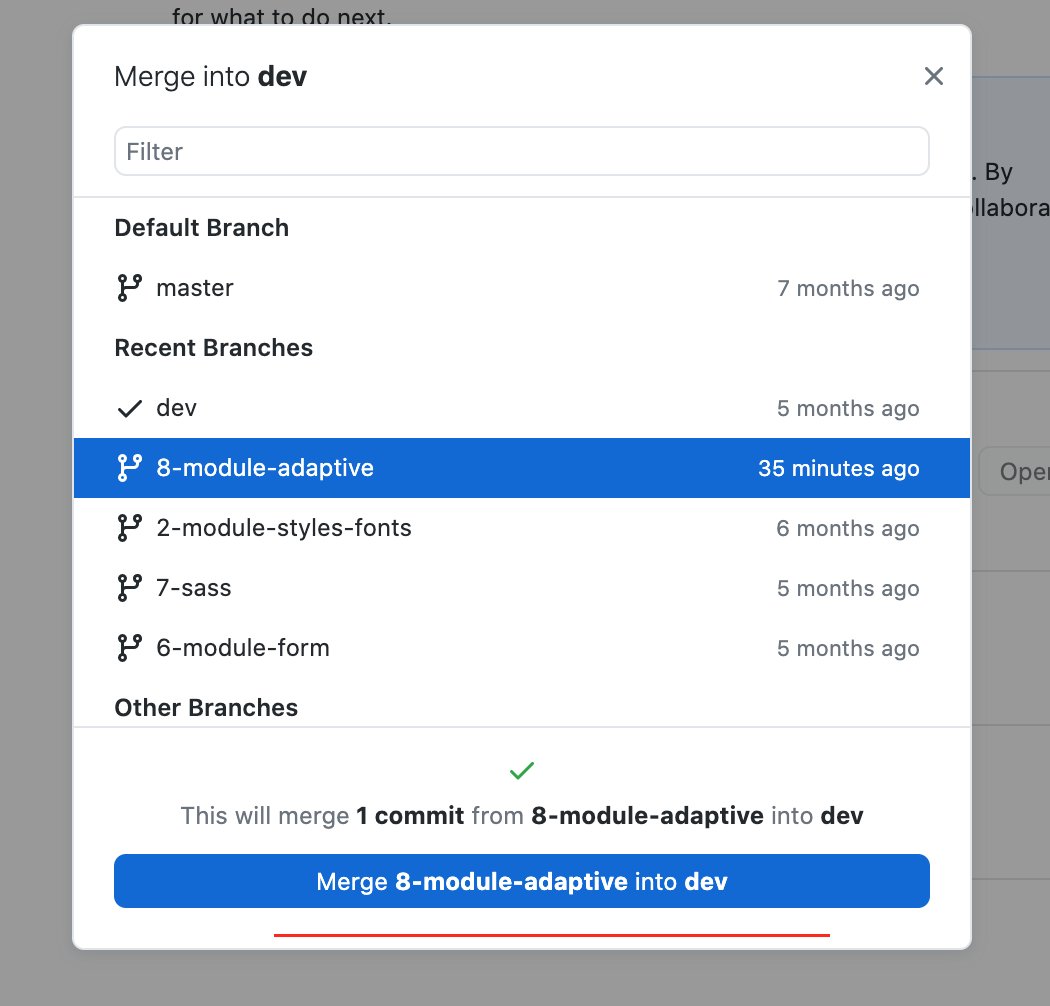
Для тех кто использует консольные команды:

* *git branch* - отображает список доступных веток. Текущая ветка помечена звездочкой.
* *git checkout* <название\_ветки> - перейти на указанную ветку
* *git checkout -b* <название\_ветки> - создать новую ветку и перейти в неё.

**Слияние веток (merge)**

Перейдите в ветку в которую вы хотите слить другую.

Нажмите *Choose a branch to merge into* 

Выберите ветку, которую нужно слить и нажмите *Merge*

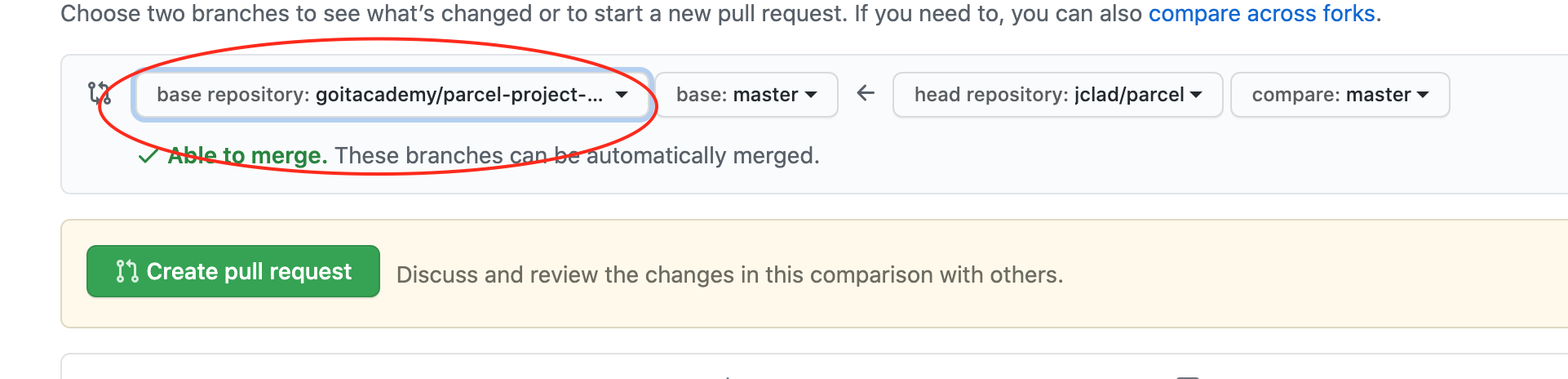
После мерджа вы можете предложить *Pull Request*

Для тех кто использует консольные команды:

* перейти в ветку, в которую нужно слить другую
* *git merge <название\_ветки\_которая\_сливается>*

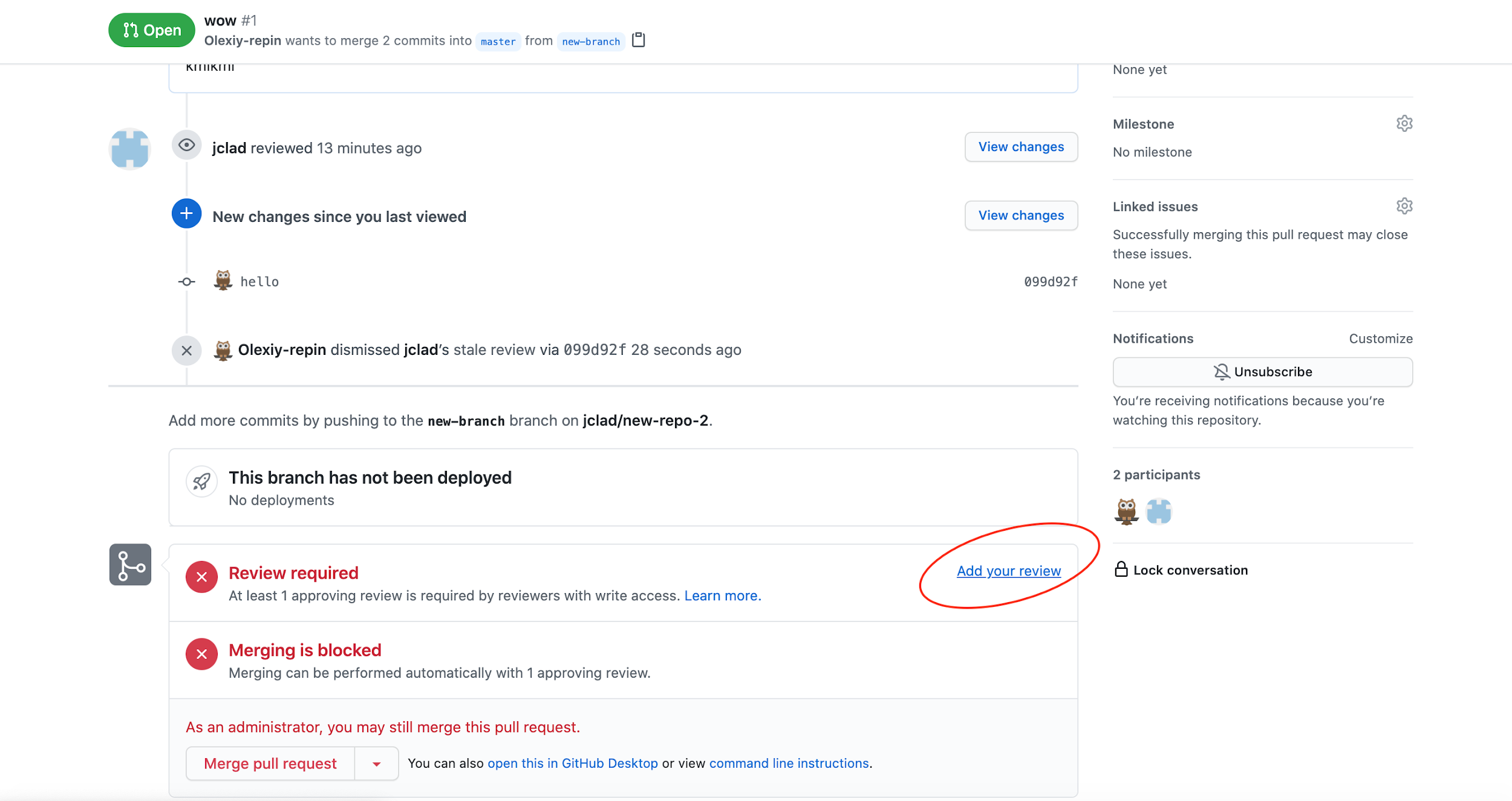
### **Pull Requests**

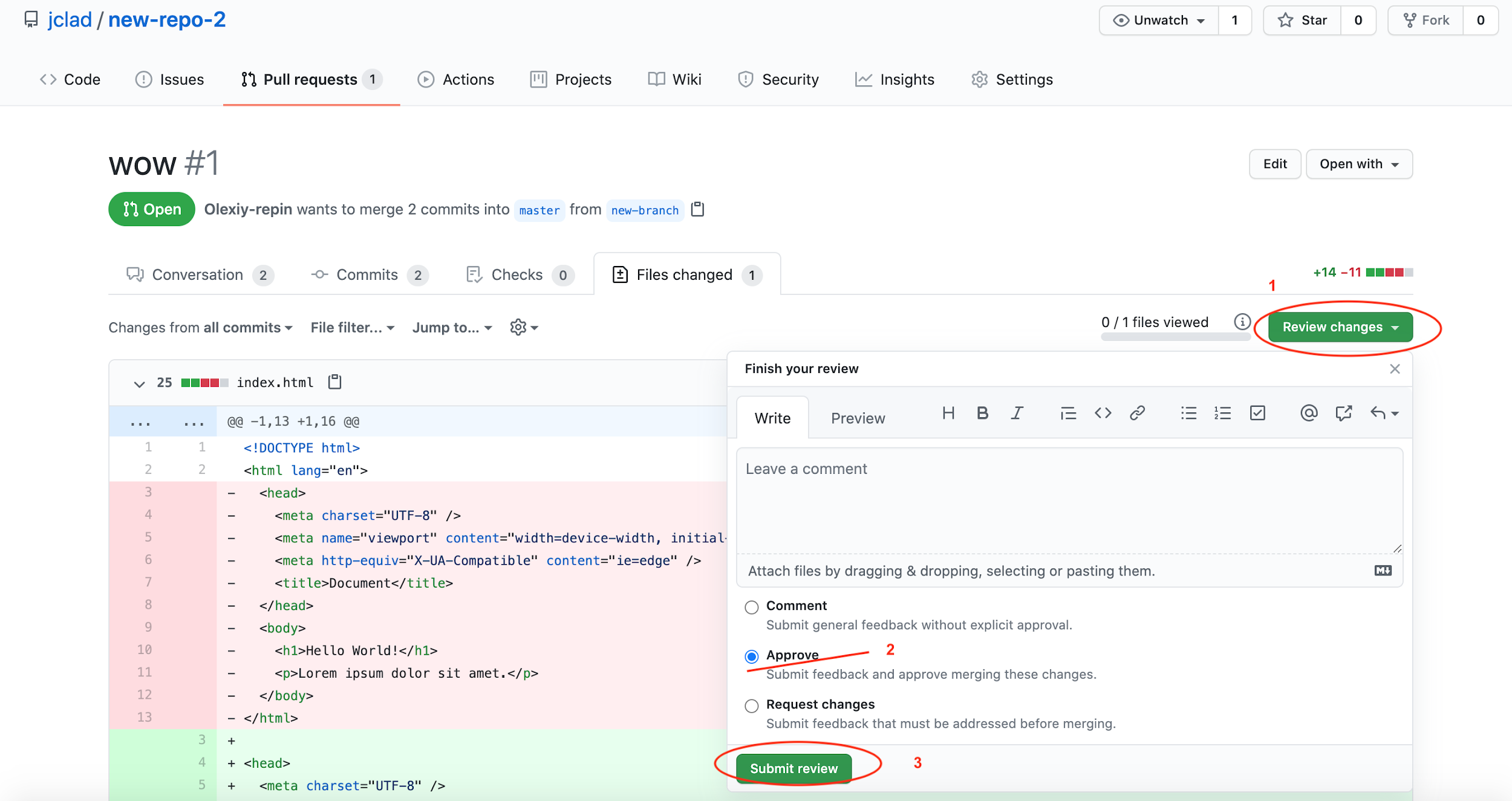
В открывшемся GitHub внимательно выберите репозиторий вашего тимлида и нажмите Create pull request.



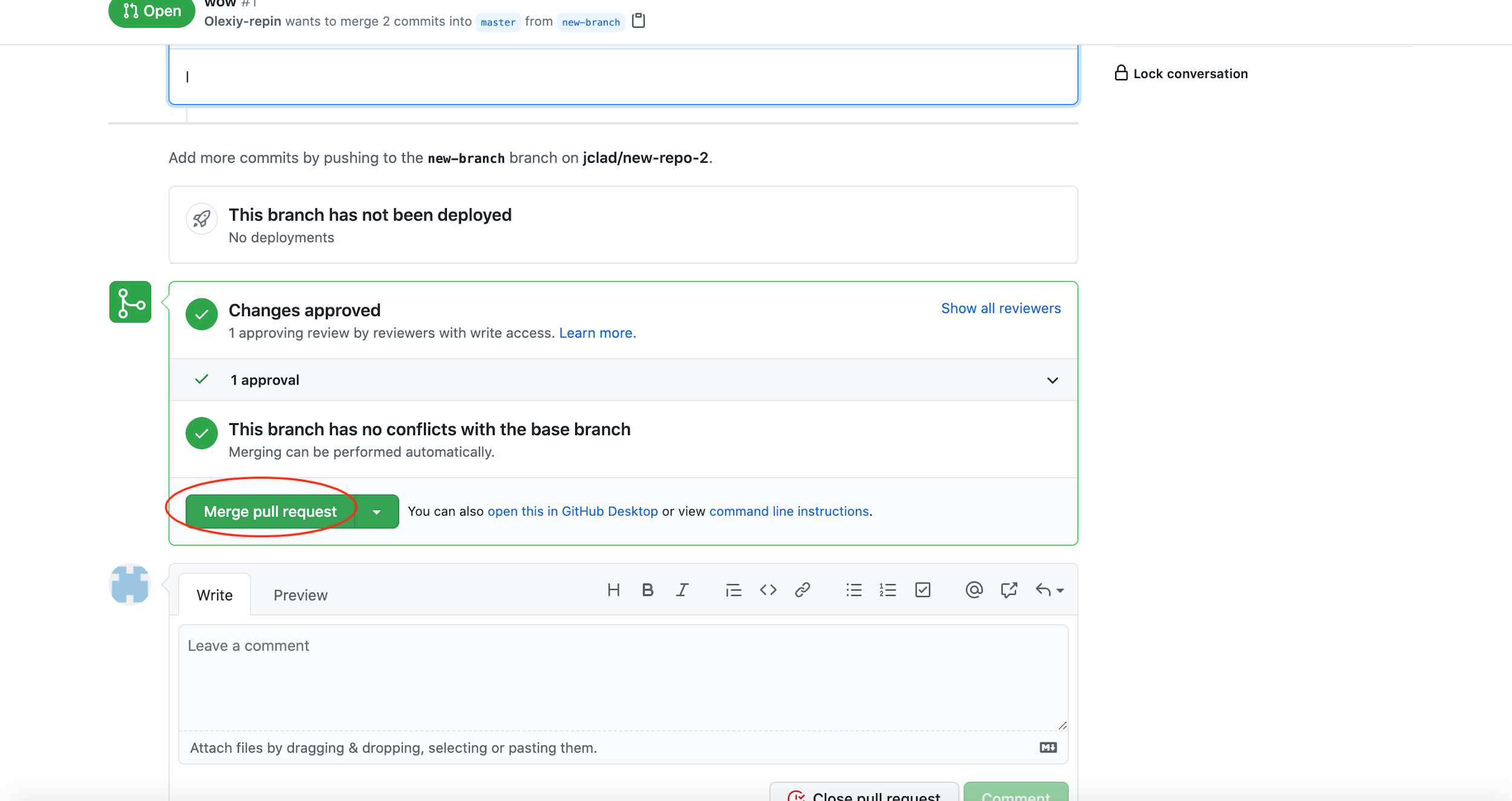
### **Принять Pull Requests**

Перейдите на вкладку Pull Requests. Выберите Add your review.

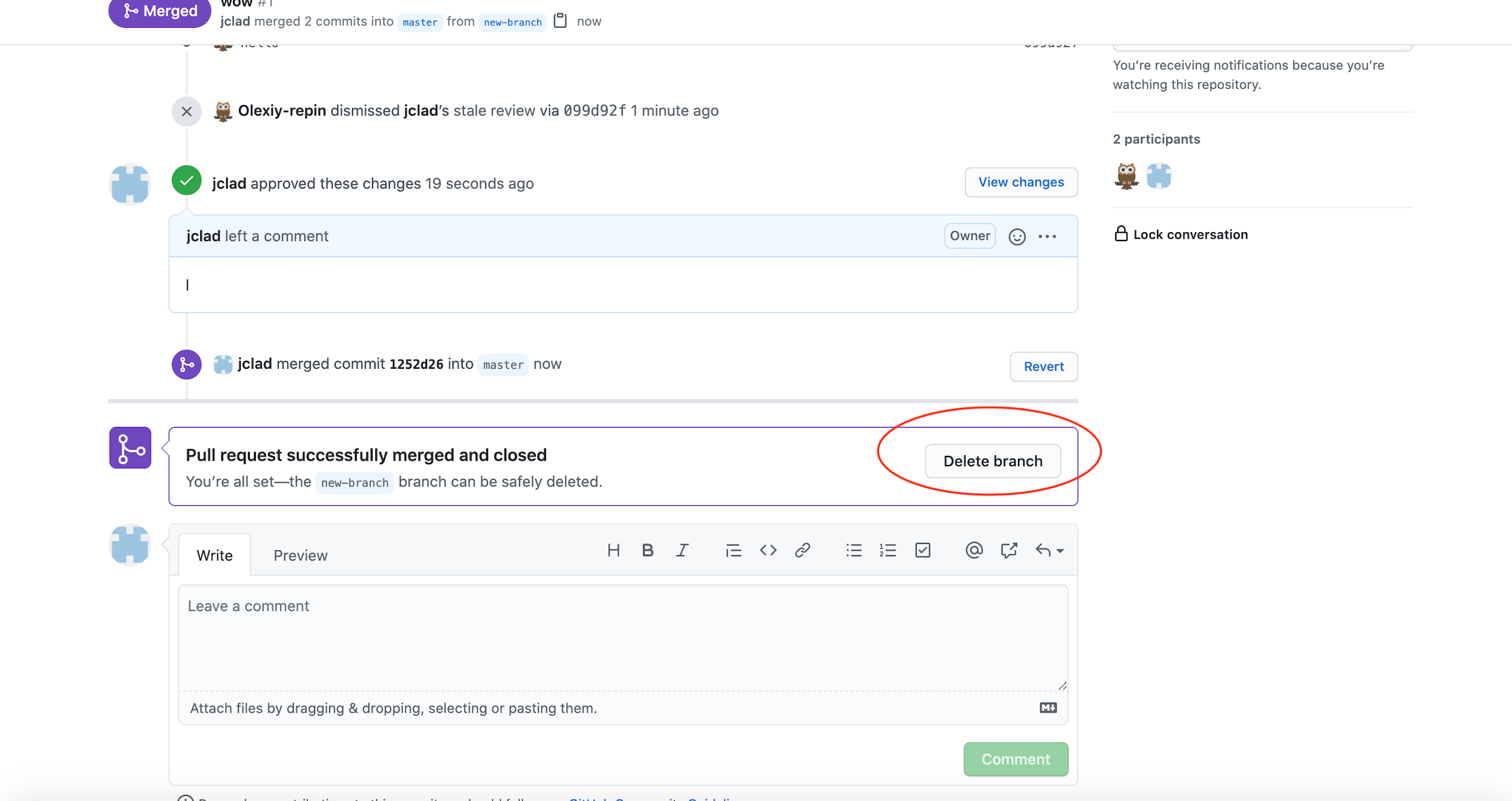


Просмотрите код на наличие замечаний и нажмите Review changes. Если всё впорядке, поставьте маркер Approve.

Теперь можно смерджить изменения и принять pull request.

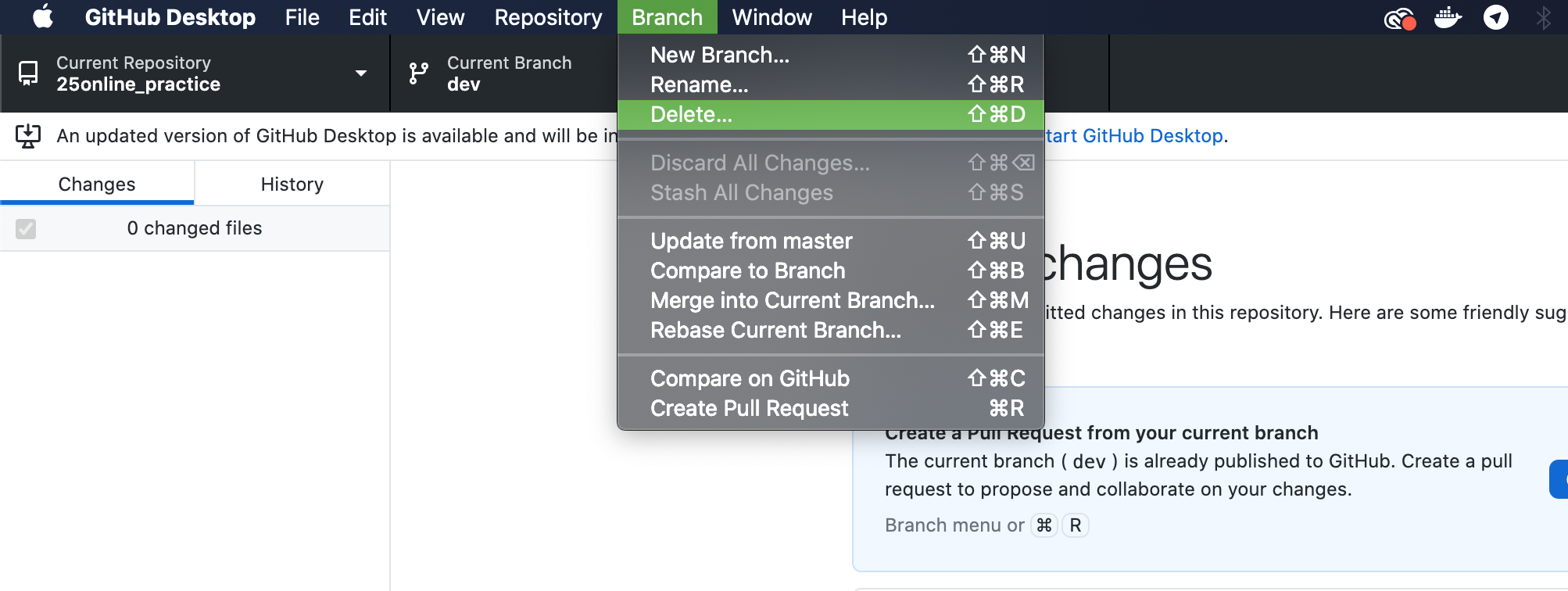


Не забывайте удалять ветки после мерджа!

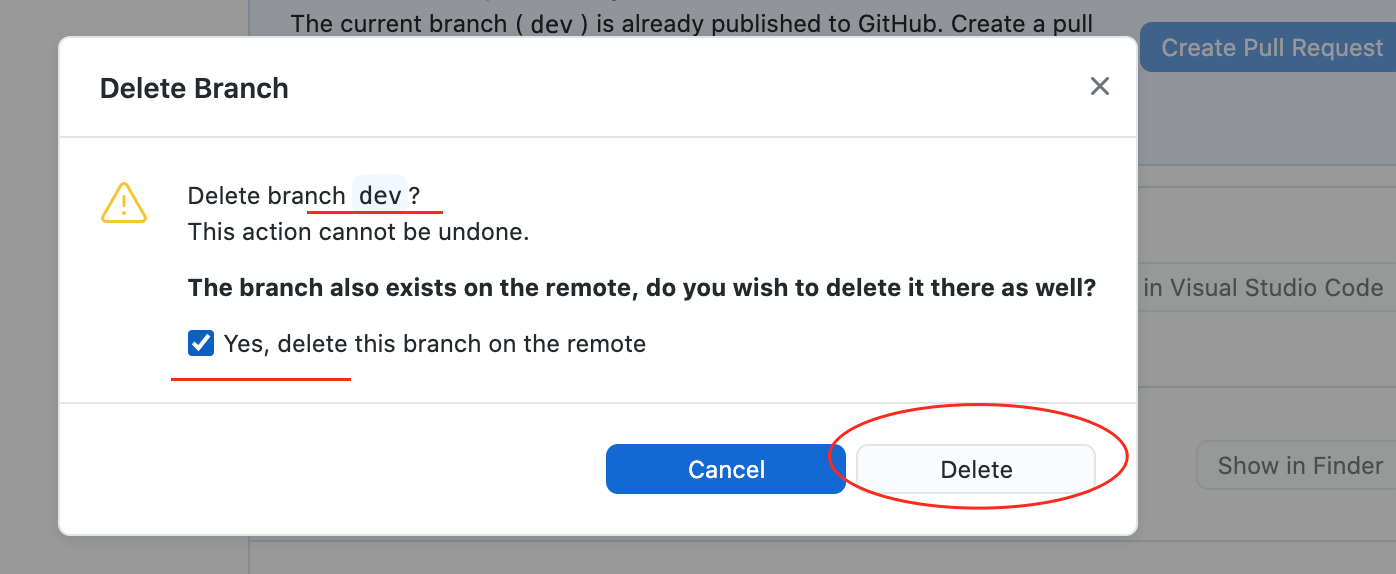


### **Удаление веток**

Чтобы удалить ветку перейдите в пункт меню Branch > Delete.



* Убедитесь, что вы хотите удалить правильную ветку.
* Поставьте галочку, если ветку нужно удалить не только локально, но и в удаленном репозитории.
* Нажмите Delete



Для тех кто использует консольные команды:  
  
git branch -d <название\_ветки> - удалить ветку

git push origin --delete <название\_ветки> - удалить ветку на удаленном репозитории