Министерство науки и высшего образования Российской Федерации

Федеральное государственное автономное образовательное учреждение

высшего образования

«СЕВЕРО-КАВКАЗСКИЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ»

Институт цифрового развития

Кафедра инфокоммуникаций

**ОТЧЕТ**

**ПО ЛАБОРАТОРНОЙ РАБОТЕ №3**

**дисциплины «****Программирование на python»**

|  |  |  |  |
| --- | --- | --- | --- |
|  | | Выполнил:  Кожуховский Виктор Андреевич  2 курс, группа ИВТ-б-о-22-1,  09.03.01 «Информатика и вычислительная техника», направленность (профиль) «Программное обеспечение средств вычислительной техники и автоматизированных систем  », очная форма обучения  \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  (подпись) | |
|  | | Руководитель практики:  Воронкин Роман Александрович  \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  (подпись) | |
|  | |  | |

Отчет защищен с оценкой \_\_\_\_\_\_\_\_\_\_\_ Дата защиты\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Ставрополь, 2023 г.

**Тема:** Основы ветвления Git.

**Цель работы:** исследование базовых возможностей по работе с локальными и удаленными ветками Git.

# Методика и порядок выполнения работы

1. Создал общедоступный репозиторий на GitHub, в котором будет использована лицензия MIT.

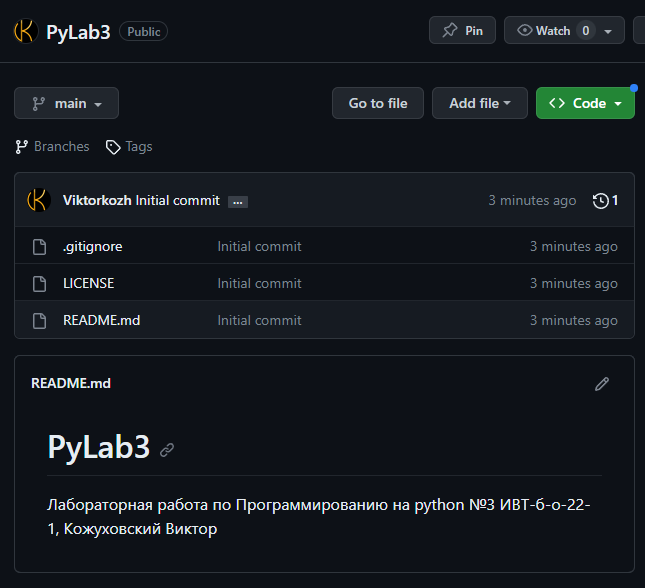


Рисунок 1. Созданный репозиторий

1. Создал три файла: 1.txt, 2.txt, 3.txt.

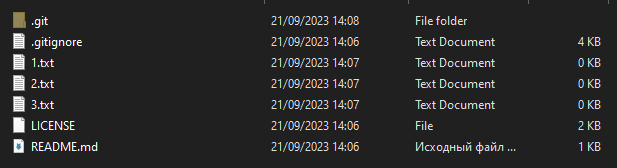


Рисунок 2. Созданные текстовые файлы в папке репозитория

1. Проиндексировал первый файл и сделал коммит с комментарием "add 1.txt file", проиндексировал второй и третий файлы, перезаписал уже сделанный коммит с новым комментарием "add 2.txt and 3.txt."

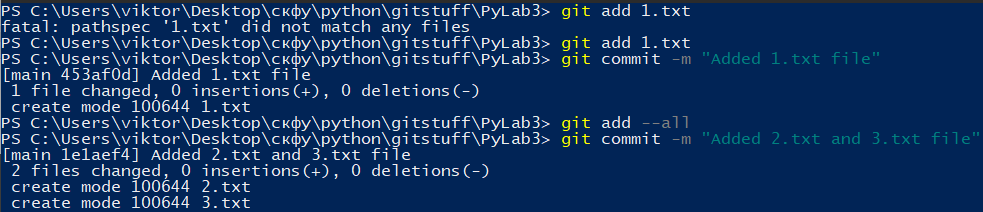


Рисунок 3. Перезаписанный коммит

1. Создал новую ветку my\_first\_branch, перешел на ветку и создал новый файл in\_branch.txt, закоммитил изменения.

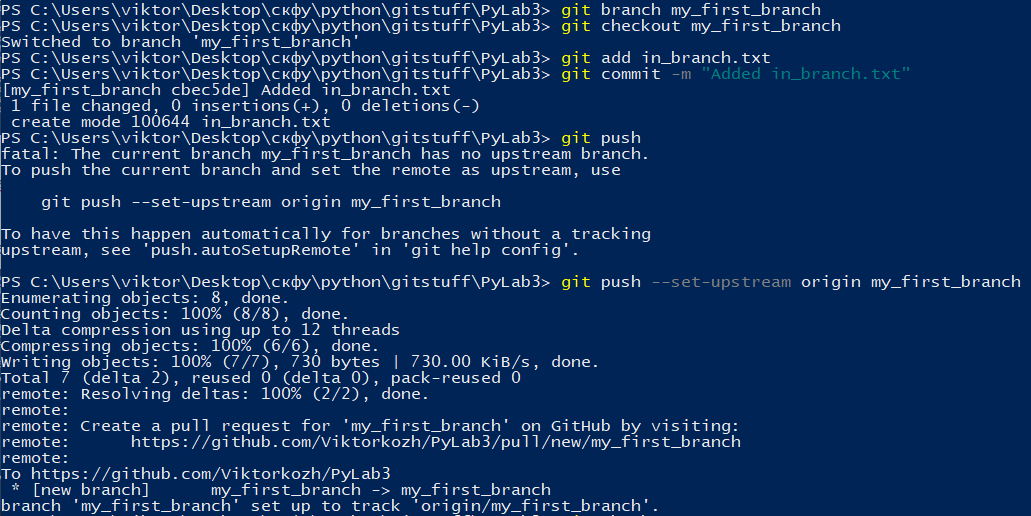


Рисунок 4. Коммит запушен

1. Вернулся на ветку origin.

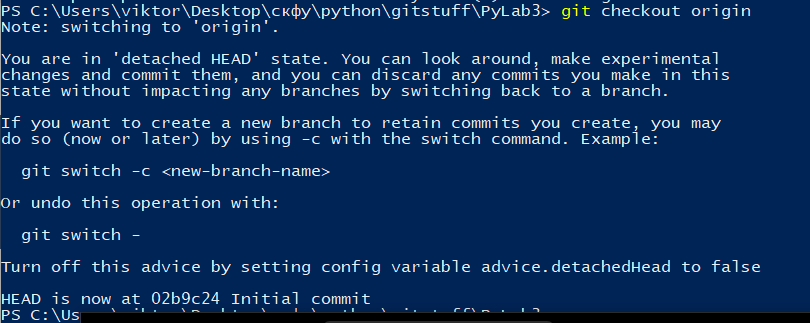


Рисунок 5. Возврат на ветку origin

1. Создал и сразу перешел на ветку new\_branch.

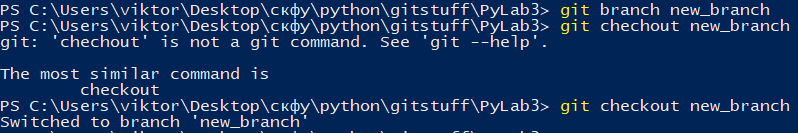


Рисунок 6. Создание новой ветки и переход на нее

1. Сделал изменения в файле 1.txt, добавил строчку “new row in the 1.txt file”, закоммитил изменения.

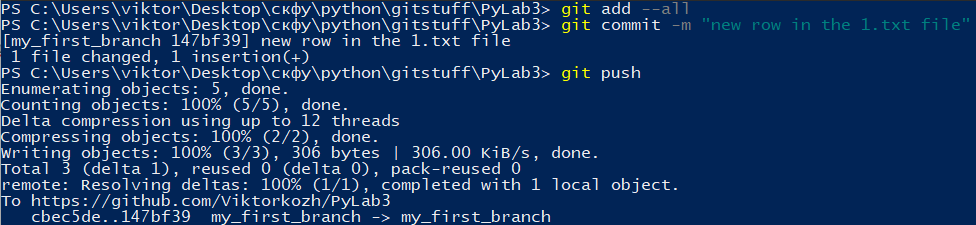


Рисунок 7. Пуш на новую ветку

1. Перешел на ветку origin и слил ветки origin и my\_first\_branch, после чего слил ветки master и new\_branch.

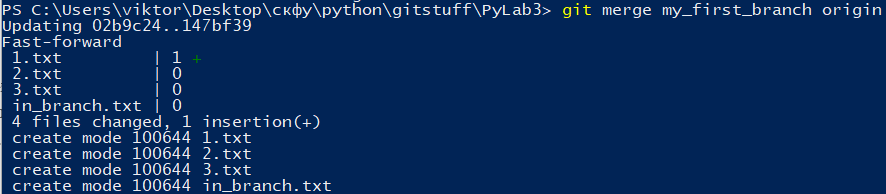


Рисунок 8. Мердж веток

1. Удалил ветки my\_first\_branch и new\_branch.

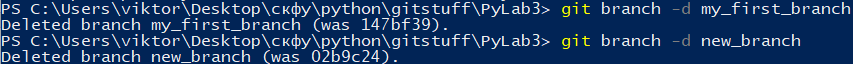


Рисунок 9. Удаление веток

1. Создал ветки branch\_1 и branch\_2.



Рисунок 10. Создание веток branch\_1 и branch\_2

1. Перешел на ветку branch\_1 и изменил файл 1.txt, удалил все содержимое и добавил текст “fix in the 1.txt”, изменил файл 3.txt, удалил все содержимое и добавил текст “fix in the 3.txt”, закоммитил изменения.

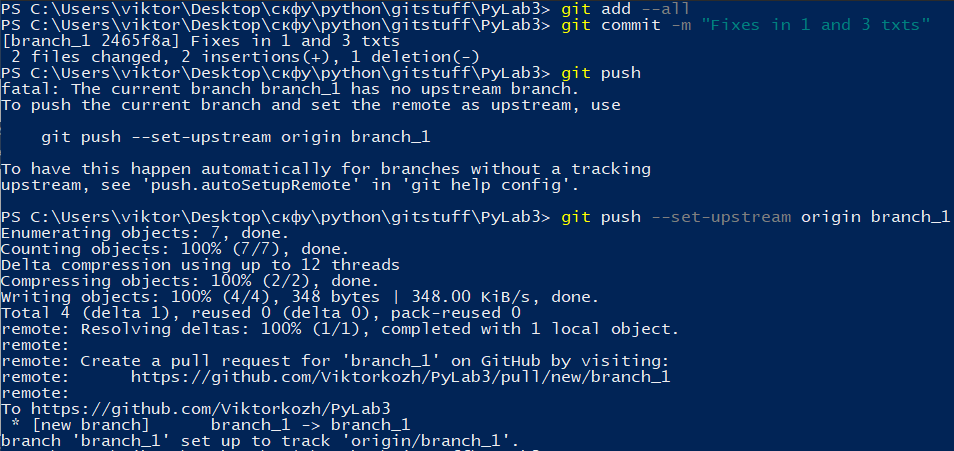
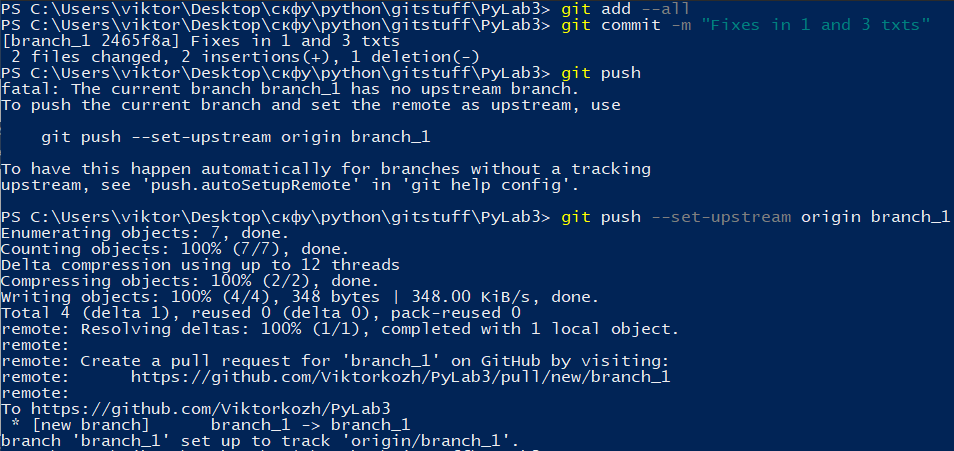


Рисунок 11. Пуш изменений на ветку branch\_1

1. Перешел на ветку branch\_2 и также изменить файл 1.txt, удалил все содержимое и добавил текст “My fix in the 1.txt”, изменил файл 3.txt, удалил все содержимое и добавил текст “My fix in the 3.txt”, закоммитил изменения.

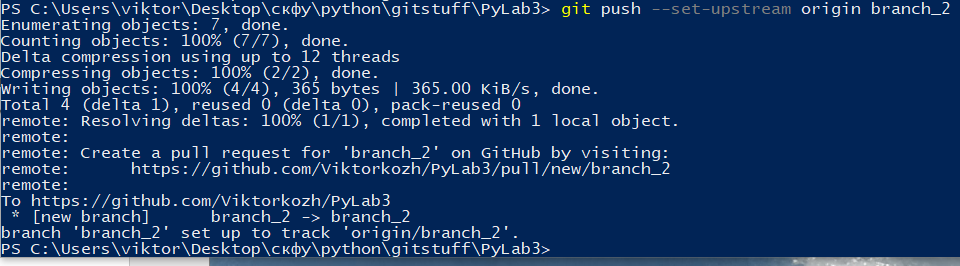


Рисунок 12. Коммит с изменениями запушен

1. Слил изменения ветки branch\_2 в ветку branch\_1.

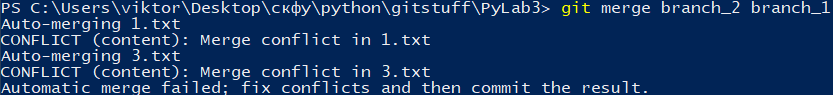


Рисунок 13. Попытка слива изменений

1. Решил конфликт файла 1.txt в ручном режиме, а конфликт 3.txt используя команду git mergetool c помощью одной из доступных утилит, например Meld.

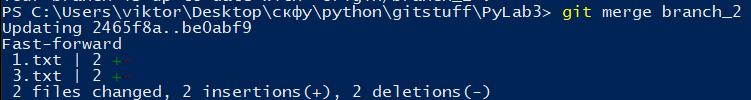


Рисунок 14. Решил конфликт и выполнил git merge

1. Отправил ветку branch\_1 на GitHub.

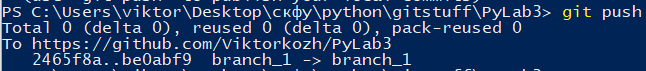


Рисунок 15. Ветка branch\_1 отправлена

1. Создал средствами GitHub удаленную ветку branch\_3.

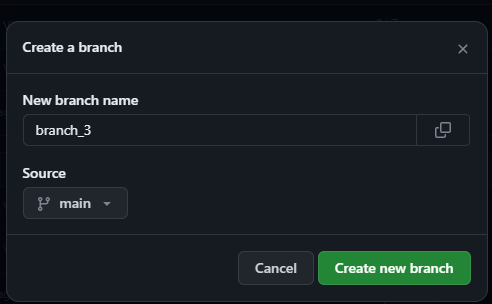


Рисунок 16. Создание ветки branch\_3 средствами GitHub

1. Создал в локальном репозитории ветку отслеживания удаленной ветки branch\_3.

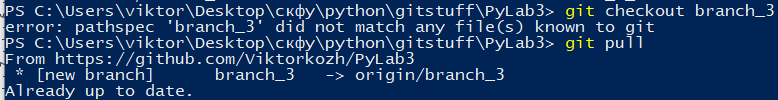


Рисунок 17. Выполнение git pull

1. Перешел на ветку branch\_3 и добавил файл файл 2.txt строку "the final fantasy in the 4.txt file". Выполнил перемещение ветки origin на ветку branch\_2.

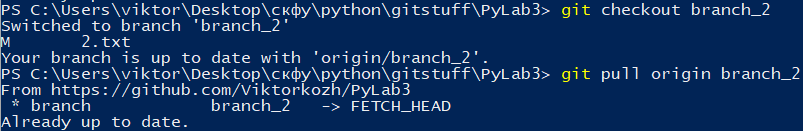


Рисунок 18. Перемещение ветки оригин на ветку branch\_2

1. Отправил изменения веток origin и branch\_2 на GitHub.

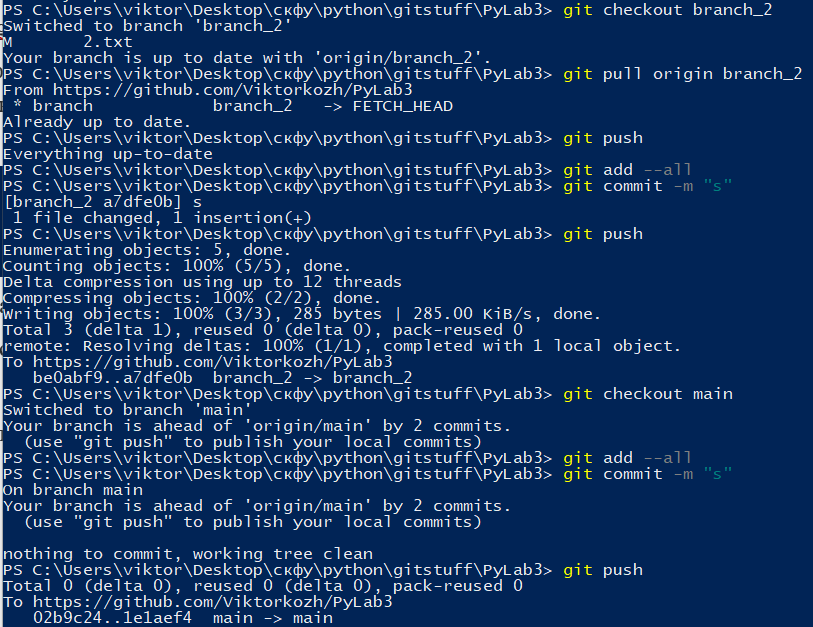


Рисунок 19. Пуши origin и branch\_2

# Вопросы для защиты работы

1. Что такое ветка?

Ветка в Git — это простой перемещаемый указатель на один из коммитов.

2. Что такое HEAD?

HEAD – это указатель, задача которого ссылаться на определенный коммит в репозитории.

HEAD – это указатель на коммит в вашем репозитории, который станет родителем следующего коммита.

HEAD указывает на коммит, относительного которого будет создана рабочая копия во время операции checkout.

3. Способы создания веток.

Чтобы создать новую ветку, необходимо использовать команду git branch.

Чтобы создать ветку и сразу переключиться на нее, можно использовать команду git checkout -b.

4. Как узнать текущую ветку?

Увидеть текущую ветку можно при помощи простой команды git log, которая покажет куда указывают указатели веток. Эта опция называется -decorate. HEAD будет стоять рядом с текущей веткой. Также можно использовать git branch -v, рядом с текущей веткой будет значек \*.

5. Как переключаться между ветками?

Для переключения на существующую ветку выполните команду git checkout.

6. Что такое удаленная ветка?

Удалённые ссылки — это ссылки (указатели) в ваших удалённых репозиториях, включая ветки, теги и так далее. Полный список удалённых ссылок можно получить с помощью команды git ls-remote <remote> или команды git remote show <remote> для получения удалённых веток и дополнительной информации.

7. Что такое ветка отслеживания?

Ветки слежения — это ссылки на определённое состояние удалённых веток. Это локальные ветки, которые нельзя перемещать; Git перемещает их автоматически при любой коммуникации с удаленным репозиторием, чтобы гарантировать точное соответствие с ним.

8. Как создать ветку отслеживания?

При клонировании репозитория, как правило, автоматически создаётся ветка master, которая следит за origin/master. Однако, при желании можно настроить отслеживание и других веток — следить за ветками на других серверах или отключить слежение за веткой master. Сделать это можно с помощью команды git checkout -b <branch> <remote>/<branch>. Это часто используемая команда, поэтому Git предоставляет сокращённую форму записи в виде флага --track.

9. Как отправить изменения из локальной ветки в удаленную ветку?

Когда вы хотите поделиться веткой, вам необходимо отправить её на удалённый сервер, где у вас есть права на запись. Ваши локальные ветки автоматически не синхронизируются с удалёнными при отправке — вам нужно явно указать те ветки, которые вы хотите отправить.

Таким образом, вы можете использовать свои личные ветки для работы, которую не хотите показывать, а отправлять только те тематические ветки, над которыми вы хотите работать с кем-то совместно. Отправка осуществляется командой git push <remote> <branch>.

10. В чем отличие команд git fetch и git pull ?

Команда git fetch получает с сервера все изменения, которых у вас ещё нет, но не будет изменять состояние вашей рабочей директории. Эта команда просто получает данные и позволяет вам самостоятельно сделать слияние. Тем не менее, существует команда git pull, которая в большинстве случаев является командой git fetch , за которой непосредственно следует команда git merge .

Если у вас настроена ветка слежения, или она явно установлена, или она была создана автоматически командами clone или checkout, git pull определит сервер и ветку, за которыми следит ваша текущая ветка, получит данные с этого сервера и затем попытается слить удалённую ветку.

11. Как удалить локальную и удаленную ветки?

Для удаления локальной ветки выполните команду git branch с параметром -d.

Вы можете удалить ветку на удалённом сервере используя параметр -- delete для команды git push .

12. Изучить модель ветвления git-flow (использовать материалы статей <https://www.atlassian.com/ru/git/tutorials/comparing-workflows/gitflow-workflow>, https://hab1r.com/ru/post/106912/). Какие основные типы веток присуствуют в модели git-flow? Как организована работа с ветками в модели git-flow? В чем недостатки git-flow?

В модели ветвления git-flow, основные типы веток это:

master - главная ветка проекта, которая содержит только стабильный код

и используется для создания релизов.

develop - ветка, в которой ведется основная разработка проекта. Она

содержит все изменения, которые были сделаны разработчиками.

feature - ветки для добавления новой функциональности в проект. Они ветвятся от ветки develop, и после завершения работы ветки объединяются с develop веткой.

release - ветки для подготовки новой версии проекта. Они ветвятся от ветки develop, содержат минимальный набор изменений и используются для подготовки релиза. После тестирования и отладки, ветка объединяется с веткой master и develop.

hotfix - ветки для исправления критических ошибок на

производственной версии проекта. Они ветвятся от ветки master, и после исправления ошибок объединяются с master и develop ветками.

Работа с ветками в модели git-flow организована следующим образом:

Начало разработки новой функциональности начинается с ветвления от ветки develop ветки feature, на которой работает разработчик.

После завершения работы, все изменения ветки feature тестируются, затем вливаются обратно в ветку develop.

В момент подготовки новой версии программного продукта ветка release создается из develop, и она используется для проведения основных тестов наиболее важных функций.

Ветка hotfix создается из ветки master, если в производственной версии обнаружена критическая ошибка, и на этой ветке выполняется исправление.

Недостатки git-flow включают в себя:

Сложность и необходимость управления множеством веток, что может быть трудным для маленьких команд.

Не слишком хорошо подходит для быстрой разработки и быстрой реализации необходимых исправлений или функций в связи с наличием большого количества ветвлений.

Накладывает значительный набор процедур и правил для разработки и управления релизами.

13. На прошлой лабораторной работе было задание выбрать одно из программных средств с GUI для работы с Git. Необходимо в рамках этого вопроса привести описание инструментов для работы с ветками Git, предоставляемых этим средством.

Codeberg.org предоставляет доступ к репозиториям Git и предоставляет возможности для работы с ветками, аналогично большинству других хостингов репозиториев Git. Вот некоторые из основных инструментов и команд, которые можно использовать для работы с ветками на Codeberg.org:

1. `git branch`: Эта команда позволяет просматривать список существующих веток в вашем локальном репозитории.

2. `git checkout`: С помощью этой команды можно переключаться между существующими ветками в локальном репозитории.

3. `git checkout -b new-branch-name`: Эта команда создает новую ветку с указанным именем и переключается на нее.

4. `git push origin branch-name`: Для отправки изменений из вашей локальной ветки в ветку на Codeberg.org, можно использовать эту команду, где `branch-name` - это имя ветки.

5. Codeberg.org также предоставляет веб-интерфейс для работы с репозиториями, который включает в себя возможность просмотра, создания и удаления веток.