

Лабораторная работа 1.3

Тема: Основы вставки Git.

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

Порядок выполнения работы


1. Создаём аккаунт в GitHub. Затем создаём новый общедоступный репозиторий, в котором будет использована лицензия MIT и язык программирования Python.

Create a new repository

A repository contains all project files, including the revision history. Already have a project repository elsewhere? [Import a repository.](#)

Required fields are marked with an asterisk (*).

Owner *

 RomanGorchakov ▾

Repository name *

/ Test

✓ Test is available.

Great repository names are short and memorable. Need inspiration? How about [automatic-octo-chainsaw](#) ?

Description (optional)

☒  **Public**

Anyone on the internet can see this repository. You choose who can commit.

☐  **Private**

You choose who can see and commit to this repository.

Initialize this repository with:

☐ Add a README file

This is where you can write a long description for your project. [Learn more about READMEs.](#)

Add .gitignore

.gitignore template: Python ▾

Choose which files not to track from a list of templates. [Learn more about ignoring files.](#)

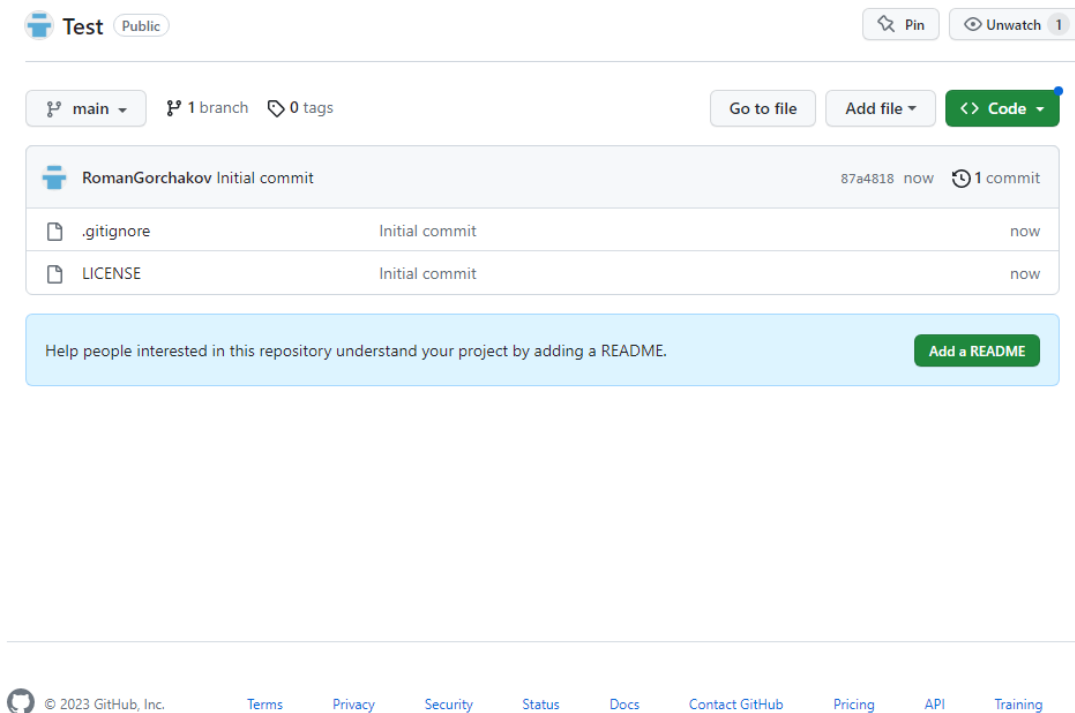
Choose a license

License: MIT License ▾

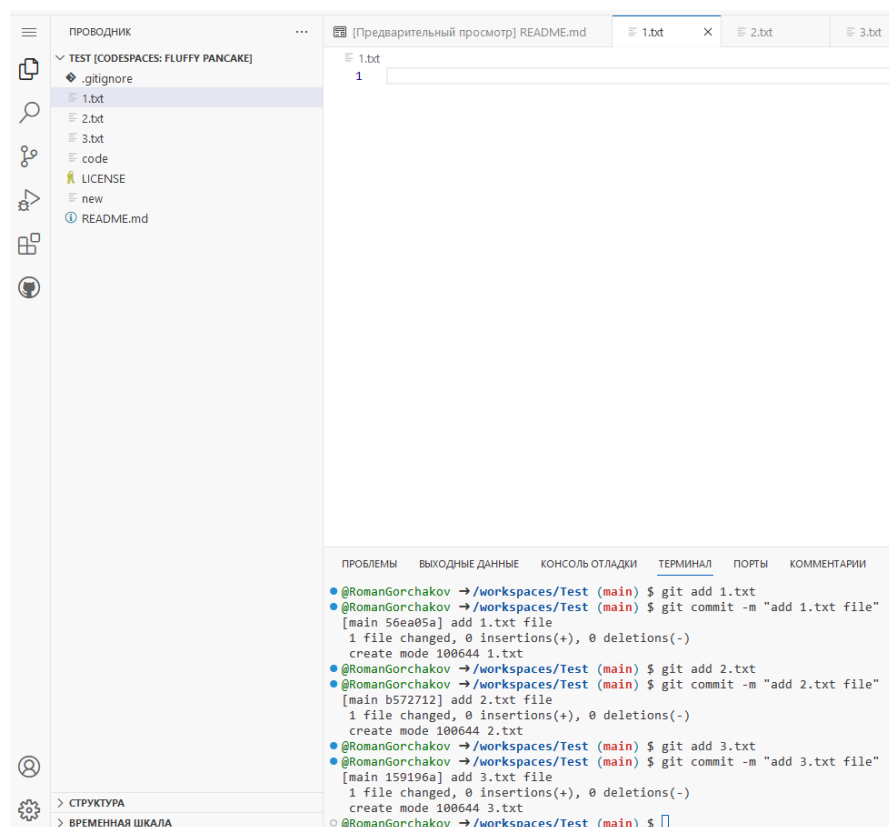
A license tells others what they can and can't do with your code. [Learn more about licenses.](#)

 You are creating a public repository in your personal account.

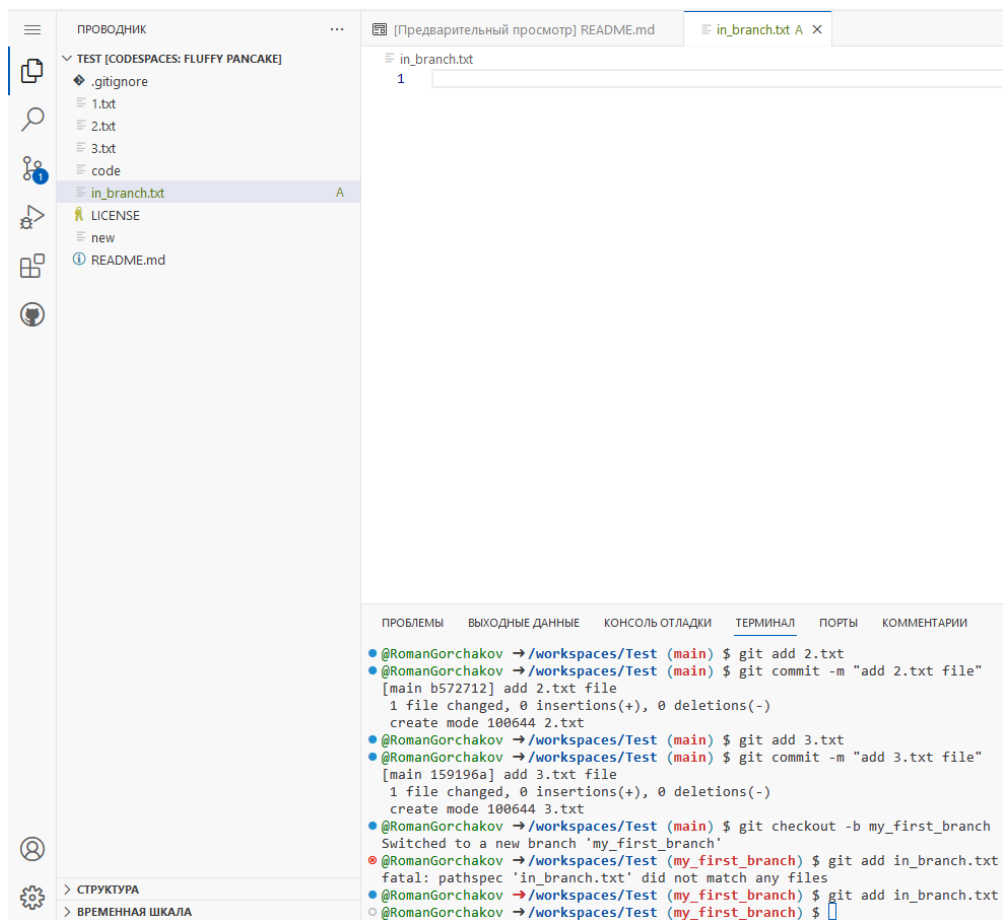
Create repository



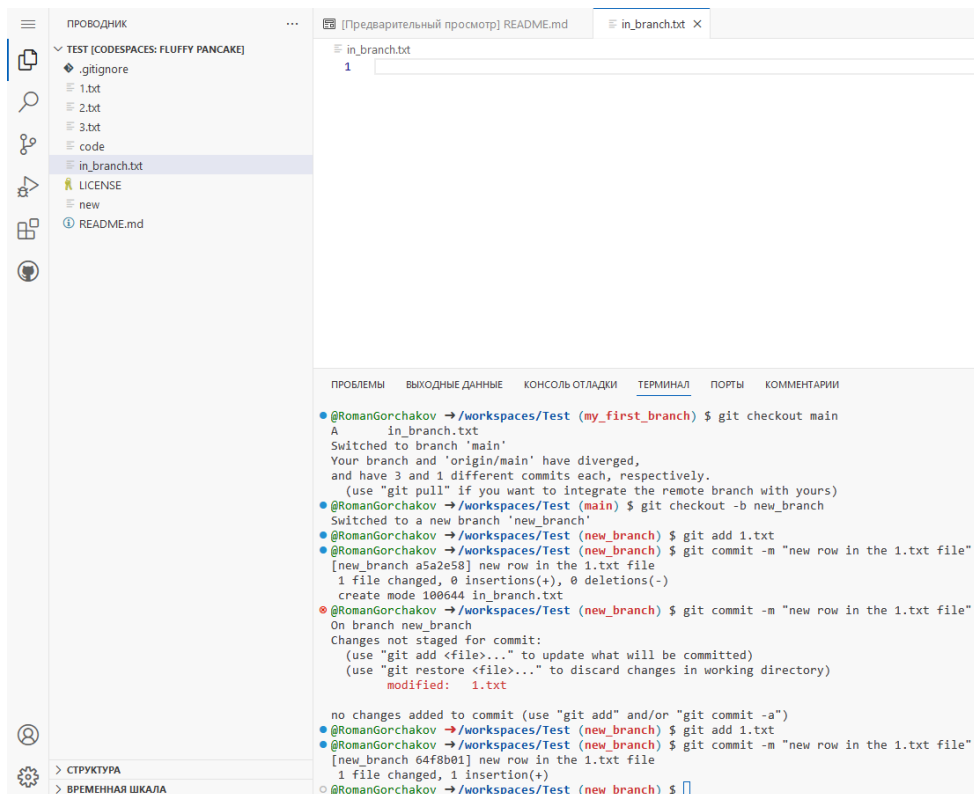
2. Создаём три файла: «1.txt», «2.txt» и «3.txt». Проиндексируем каждый файл через команды «git add» и делаем коммиты с комментариями с помощью команд «git commit -m».



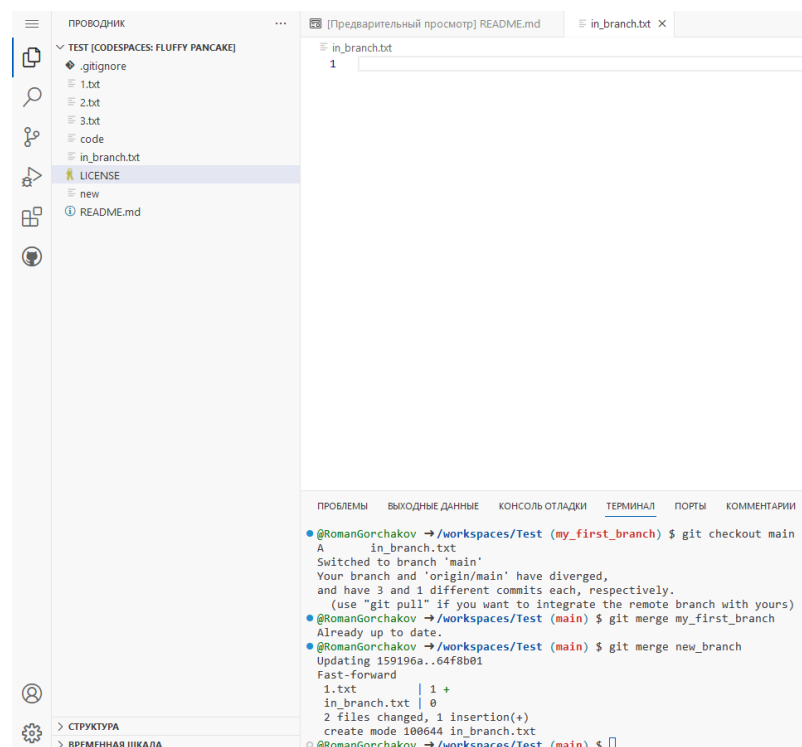
3. Создаём новую ветку «my_first_branch» с помощью команды «git checkout -b». Переходим на неё и создаём на ней новый файл «in_branch.txt».



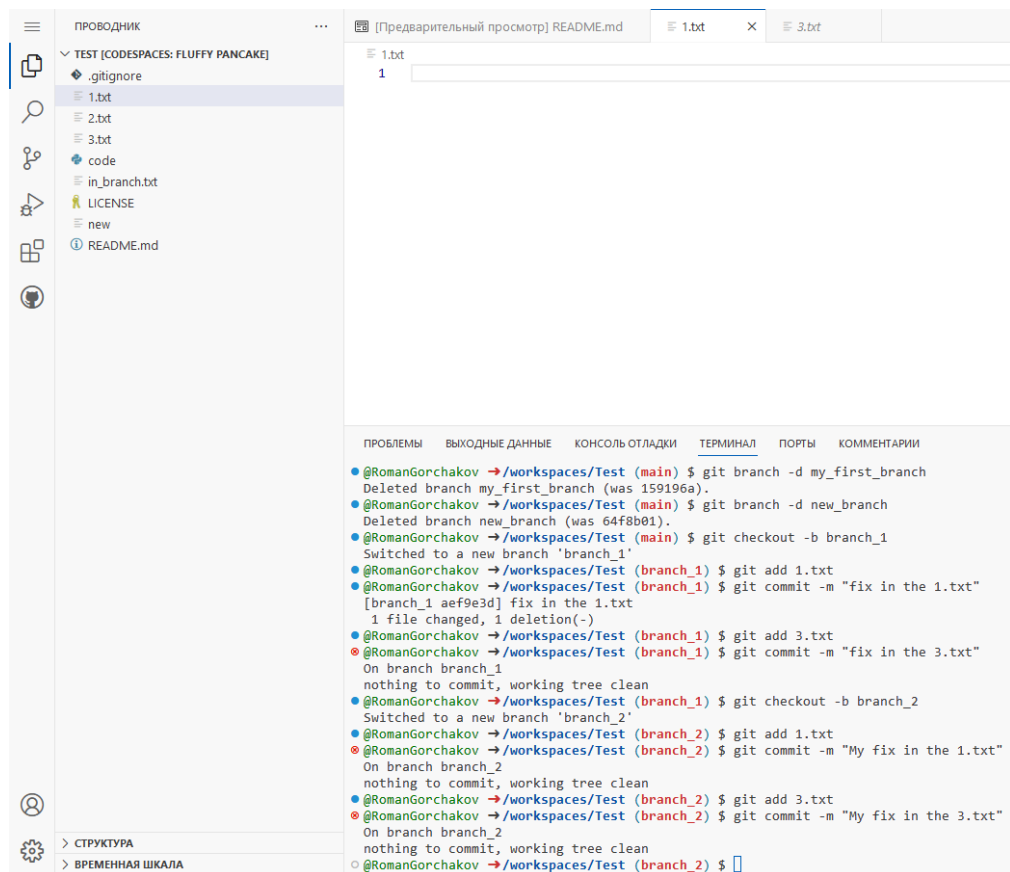
4. Возвращаемся на основную ветку «main» с помощью команды «git checkout main». Создаём новую ветку «new_branch» с помощью команды «git checkout -b». Сохраняем изменения с помощью команды «git commit -m».



5. Возвращаемся на основную ветку «main» с помощью команды «git checkout main». После этого сливаем ветки «main» и «my_first_branch», а затем «main» и «new_branch» с помощью команды «git merge».



6. Удаляем ветки «my_first_branch» и «new_branch» через команду «git branch -d». Создаём ветки «branch_1» и «branch_2». Переходим на ветку «branch_1», удалим содержимое файлов «1.txt» и «3.txt» и сохраняем изменения с помощью команды «git commit -m». Делаем то же самое в ветке «branch_2».



7. Сливаем изменения ветки «branch_2» в «branch_1» с помощью команды «git merge». Отправляем ветку «branch_1» на GitHub с помощью команды «git push». Создаём удалённую ветку «branch_3» с помощью команды «git push origin». Создаём в локальном репозитории ветку отслеживания «searcher» удалённой ветки «branch_3» с помощью команды «git remote add». Переходим на ветку «branch_3» через «git checkout» и добавляем в файл «2.txt» строку «the final fantasy in the 4.txt file». Сохраняем изменения через «git commit -m». Переносим ветку «branch_2» на ветку «main» через «git rebase» и обе ветки – на удалённый репозиторий в GitHub с помощью команды «git push».

[Overview](#)
[Yours](#)
[Active](#)
[Stale](#)
[All branches](#)
[New branch](#)

Default branch
↻

🔍 main
📄 Updated last week by RomanGorchakov
Default
🔗
✎

Your branches

branch_2	📄 Updated last week by RomanGorchakov	6 11	#2	🔍 Open	🔗	✎	🗑
branch_3	📄 Updated last week by RomanGorchakov	5 0		New pull request	🔗	✎	🗑
branch_1	📄 Updated last week by RomanGorchakov	6 6	#1	🔍 Open	🔗	✎	🗑

Active branches

branch_2	📄 Updated last week by RomanGorchakov	6 11	#2	🔍 Open	🔗	✎	🗑
branch_1	📄 Updated last week by RomanGorchakov	6 6	#1	🔍 Open	🔗	✎	🗑
branch_3	📄 Updated last week by RomanGorchakov	5 0		New pull request	🔗	✎	🗑

© 2023 GitHub, Inc.

[Terms](#)
[Privacy](#)
[Security](#)
[Status](#)
[Docs](#)
[Contact GitHub](#)
[Pricing](#)
[API](#)
[Training](#)
[Blog](#)
[About](#)

Test
Public

📌 Pin
👁 Unwatch 1
🍴 Fork 0
⭐ Star 0

🔍 main
🔗 4 branches
🏷 0 tags

Go to file
Add file
<> Code

RomanGorchakov
Add files via upload

e4b0fad · 37 minutes ago · 19 commits

📄 .gitignore	Initial commit	3 weeks ago
📄 LICENSE	Initial commit	3 weeks ago
📄 README.md	Create README.md	2 weeks ago
📄 code	Create code	2 weeks ago
📄 ЛР1.3_ГорчаковРВ.docx	Add files via upload	37 minutes ago

README.md
✎

Информация обо мне [🔗](#)

Имя: Роман.

Фамилия: Горчаков.

Отчество: Владимирович.

Название группы: ПИЖ-6-о-22-1.

About
⚙

No description, website, or topics provided.

📖 Readme
📄 MIT license
🔗 Activity
⭐ 0 stars
👁 1 watching
🍴 0 forks

Releases

No releases published
[Create a new release](#)

Packages

No packages published
[Publish your first package](#)

Контрольные вопросы

1. Что такое ветка?

Ветка в Git – простой перемещаемый указатель на один из таких коммитов.

2. Что такое HEAD?

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

3. Способы создания веток.

Создание веток происходит с помощью команды «git branch».

4. Как узнать текущую ветку?

Текущую ветку можно узнать с помощью команды «git log –oneline –decorate».

5. Как переключаться между ветками?

Между ветками можно переключаться с помощью команды «git checkout».

6. Что такое удалённая ветка?

Удалённые ссылки – ссылки (указатели) в ваших удалённых репозиториях, включая ветки, теги и так далее.

7. Что такое ветка отслеживания?

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

8. Как создать ветку отслеживания?

Ветку слежения можно создать с помощью команды «git remote add».

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

Отправить изменения из локальной ветки в удалённую можно с помощью команды «git push».

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

Команда «git pull» автоматически сливает коммиты, не давая пользователю просмотреть их сразу. При использовании «git fetch» Git собирает все коммиты из целевой ветки, которых нет в текущей ветке, и сохраняет их в локальном репозитории.

11. Как удалить локальную и удалённую ветки?

Удалить локальную и удалённую ветки можно с помощью команды «git push --delete».

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

Git-flow – устаревшая версия рабочего процесса Git, в свое время ставшая принципиально новой стратегией управления ветками в Git.

Основные типы веток: главные, вспомогательные, функциональные, ветви релизов, ветви исправлений

Первый шаг рабочего процесса заключается в создании ветки develop от стандартной ветки main. Под каждую новую функцию нужно выделить собственную ветку, которую можно отправить в центральный репозиторий для создания резервной копии или совместной работы команды. Когда в ветке develop оказывается достаточно функций для выпуска (или приближается назначенная дата релиза), от ветки develop создается ветка release. Создание этой ветки запускает следующий цикл релиза, и с этого момента новые функции добавить больше нельзя – допускается лишь исправление багов, создание документации и решение других задач, связанных с релизом.

Git Flow может замедлять работу, когда приходится ревьюить большие пулл реквесты, когда вы пытаетесь выполнить итерацию быстро. Релизы сложно делать чаще, чем раз в неделю. Большие функции могут потратить дни на мерж и резолв конфликтов и форсировать несколько циклов тестирования.

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.