giПрактическое занятие №3

Работа заключается в изучении Git-команд посредством прохождения тестов на обучающем сайте Learn Git Branching, и последующая выгрузка работы из ПЗ №3 на GitHub

ВВЕДЕНИЕ

В задачах №1-2 расписывались команды для создания коммитов и ветвей (git commit и git branch соответственно).

В задаче №3 с помощью git checkout -b [branchName] Создали ветку и переключились на неё, создали коммит (git commit), перелечились на ветку main, создали еще один коммит, командой git branch слили ветку bugfix с веткой main тем самым получили коммит с двумя родителями. (Смотреть рисунок 1).

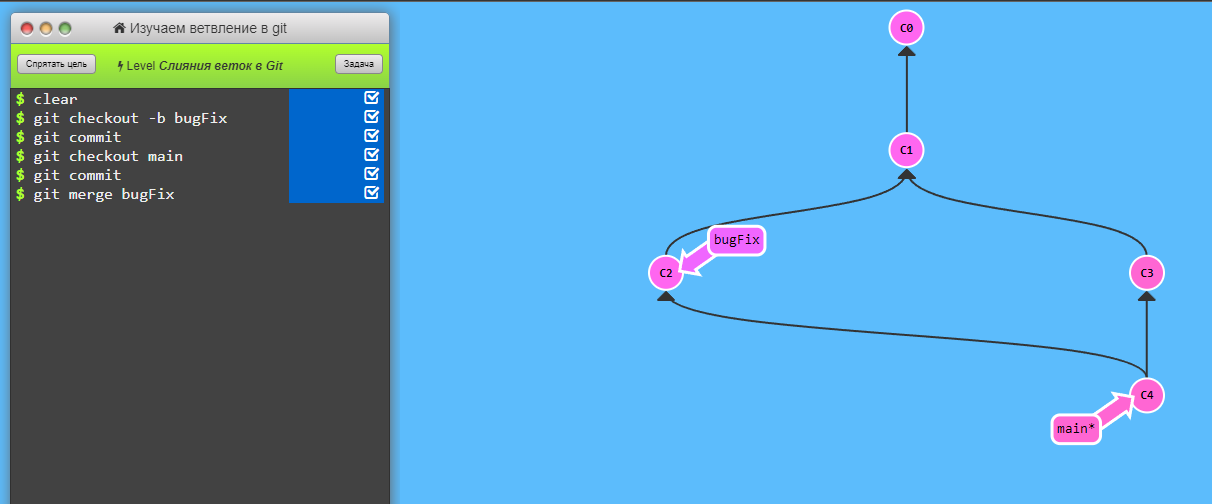


Рисунок 1 – Слияние веток в git

Задача №4 знакомит пользователя с командой rebase, которая, по сути, копирует коммит в другое место, цель задачи была выполнена командами из прошлой задачи, но вместо “слияния” мы использовали rebase.

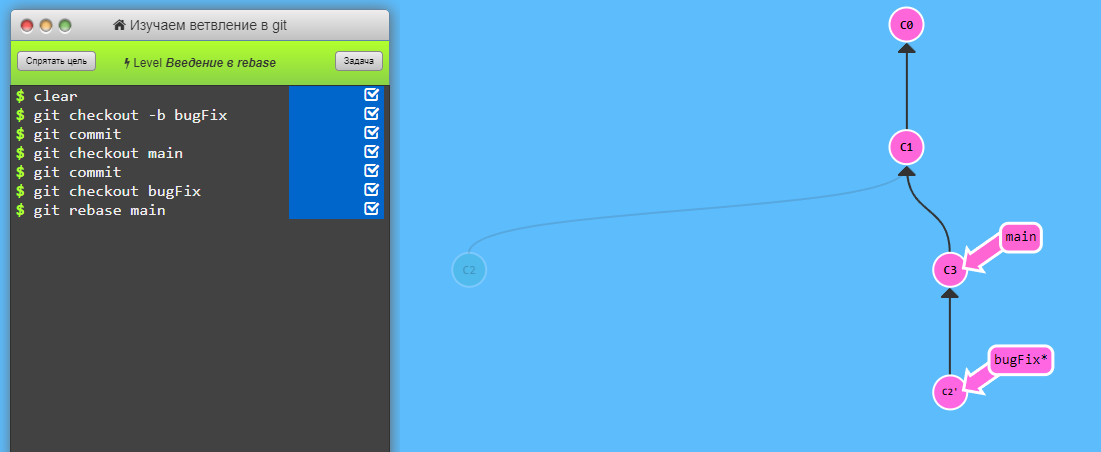


Рисунок 2 – Введение в rebase

Едем дальше

Задача №1 описывает HEAD – имя текущего выбранного коммита, если не выбран коммит, HEAD указывает на выбранную ветку.

Задача №2 описывает работу относительных ссылок, знакомит с оператором “^”, возвращающий родителя коммита так как не всегда удобно вводить хэш коммита

В задаче №3 необходимо переместить указатель HEAD и некоторые ветки на другие коммиты (Смотреть рисунок 3).

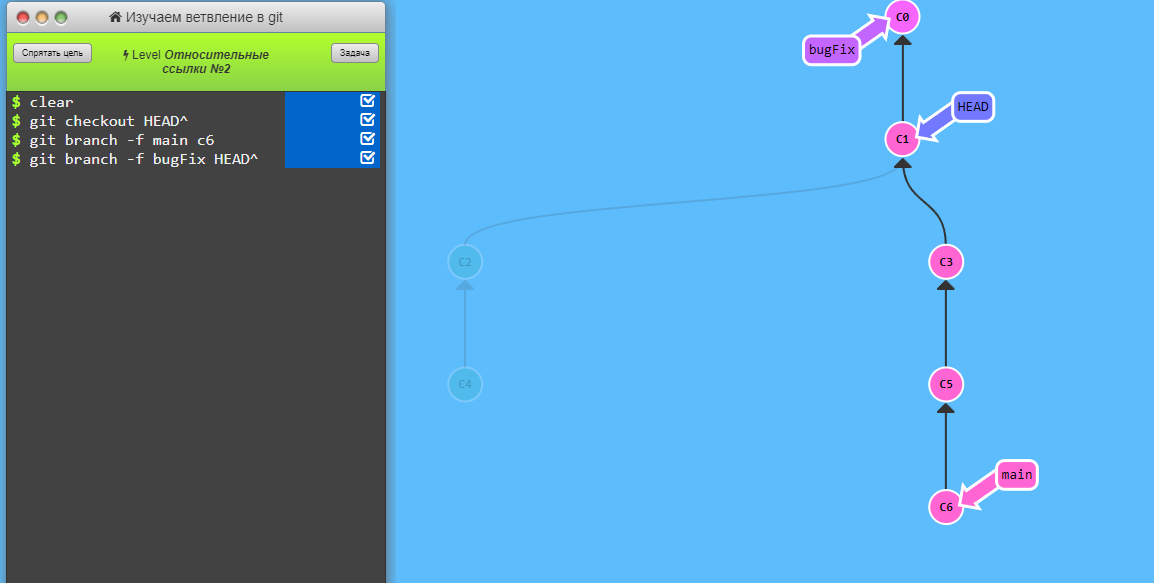


Рисунок 3 – Относительные ссылки №2

Задача №4 ставит цель отменить изменения на условно локальном и условной удалённом коммите, для локальных коммитов используется git reset, для которого необходимо указать до какого момента отменить изменения, для удалённого используется git revert, с указанием коммита, который необходимо изменить (Смотреть рисунок 4).

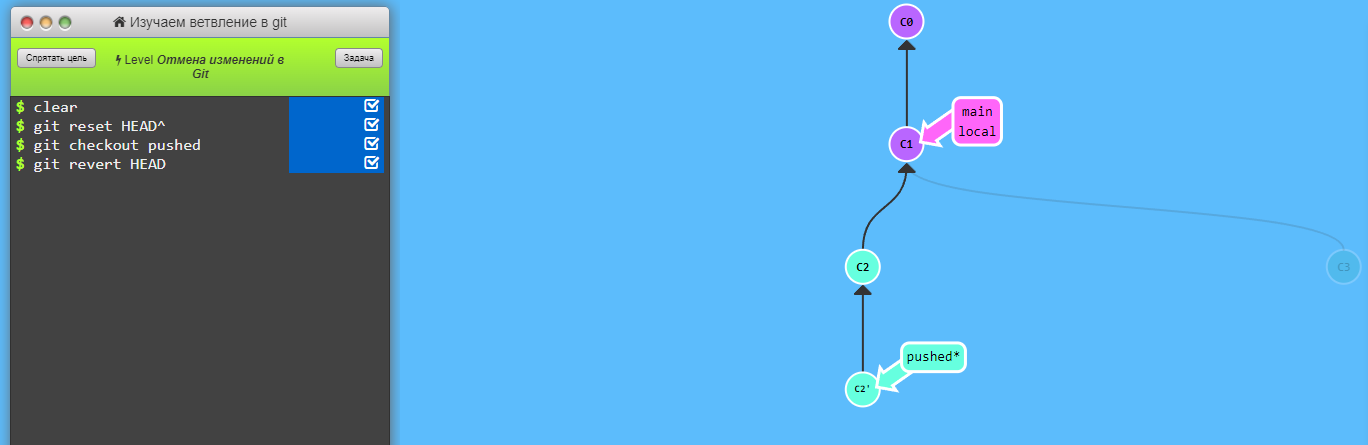


Рисунок 4 – отмена изменений в git

Перемещаем труды туда-сюда

Задача №1 описывает работу команды cherry-pick, которая переносит указанные коммиты в нужное место (HEAD). (Смотреть рисунок 5).



Рисунок 5 – Введение в cherry-pick

Задача №2 объясняет поведение команды rebase с опцией -I, а именно – позволяет менять порядок коммитов, удалять их и сливать.

Сборная солянка

Задача №1 требует выбора одного коммита с помощью ранее изученных инструментов (rebase / chery-pick) (Смотреть рисунок 6).

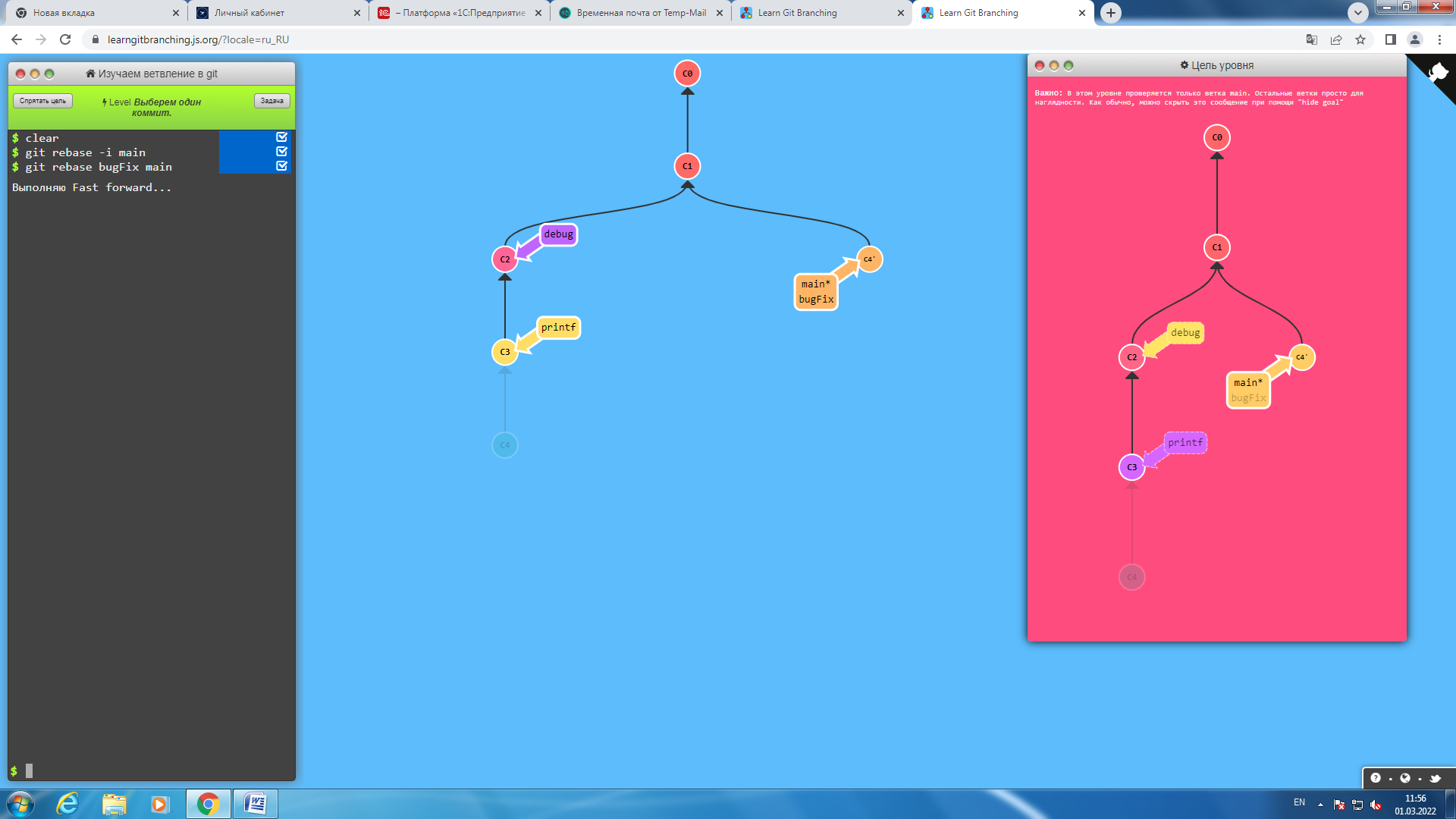


Рисунок 6 – Выберем один коммит

Задача №2 требует внесение изменений в коммит, выступающий родительским для последнего в ветке, для этого необходимо пересобрать ветку (git rebase -i) так, что бы необходимый коммит был вершиной ветки, внести изменения (git commit --amend), снова пересобрать в правильном порядке. (Смотреть рисунок 7).

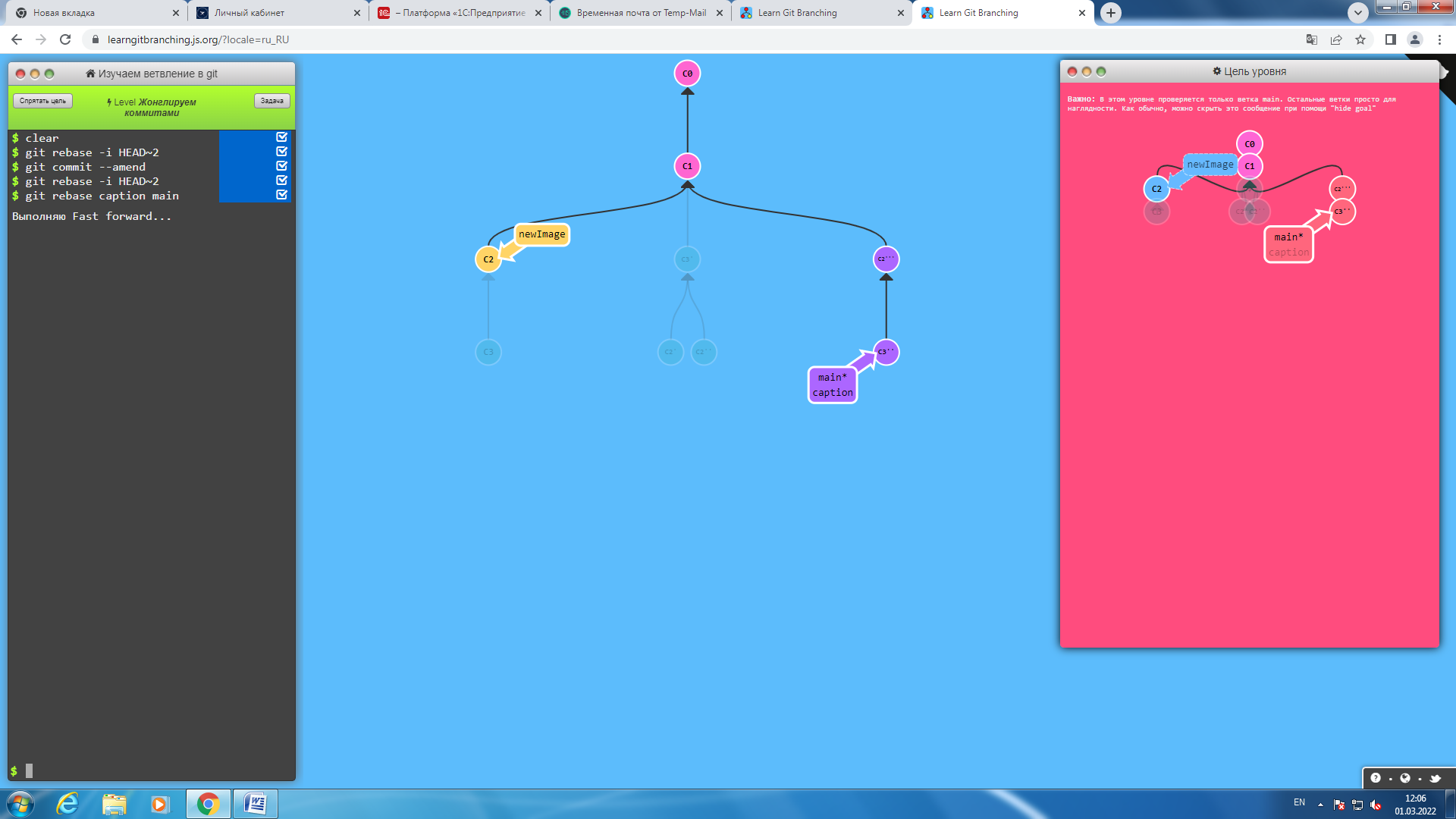


Рисунок 7 – Жонглируем коммитами

В задаче №3 необходимо решить проблему из второй задачи средствами git cherry-pick (смотреть рисунок 8)..

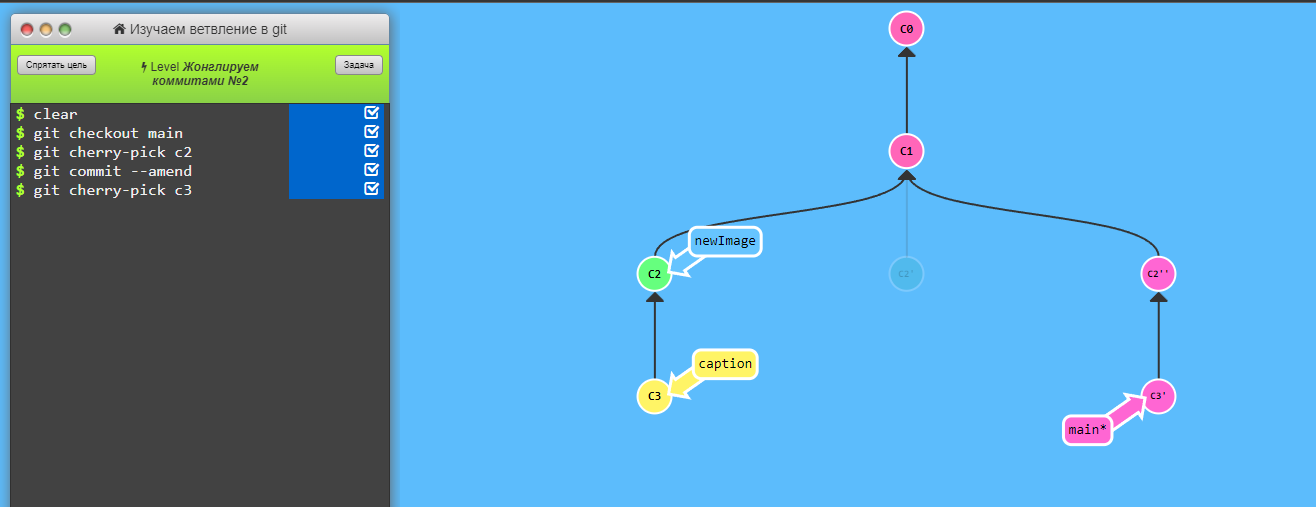


Рисунок 8 – Жонглируем коммитами №2

Задача №4 знакомит пользователя с Git tag, элемент, который будет всегда ссылаться на один коммит. Git tag [tag name] [commit name].

Задача №5 описывает команду git describe, которая позволяет понять как далеко от коммита находится тэг, и что это за тэг.

Продвинутый уровень

Задача №1 предполагает перенос различных коммитов из разных веток в main, для этого используем команду git rebase (смотреть рисунок 9).

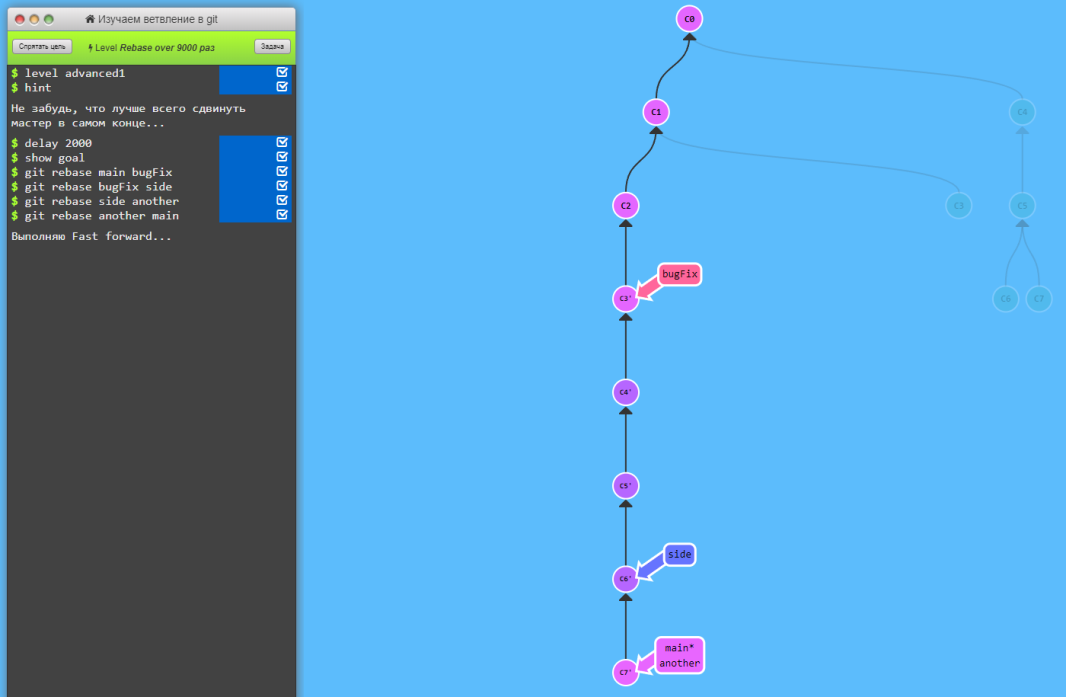


Рисунок 9 – Rebase over 9000 раз

Задача №2 расписывает функционал оператора «^», в случае если после оператора следует номер, он указывает на какого из родителей необходимо перейти.

Задача №3 подразумевает распределение различных коммитов из main в некоторые ветки, используем для этого git cherry-pick, git rebase (смотреть рисунок 10).

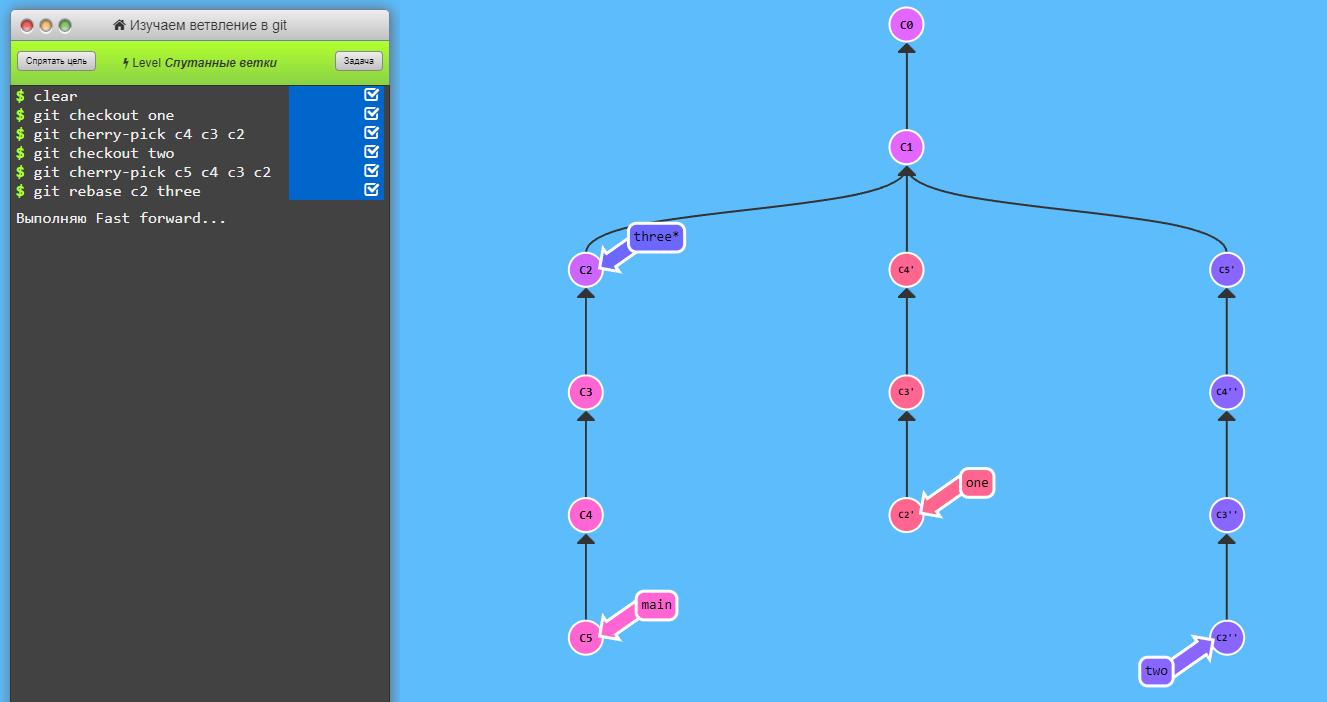


Рисунок 10 – спутанные ветки

Удалённые репозитории в Git

Задача №1 описывает работу с удаленными репозиториями, а конкретно, выгрузку локального коммита на удалённый сервер.

Задача №2 описывает поведение и особенности работы с удалёнными ветками, а именно, при создании коммита на удалённой ветке origin/main локального репозитория, ветка не переместиться на последний коммит, так как origin/main показывает состояния удалённого репозитория, и обновится тогда, когда обновится локальный репозиторий.

Задача №3 описывает команду Git fetch, которая скачивает данные из удалённого репозитория в локальный, но не сливает коммиты.

Задача №4 обесценивает команду fetch командой pull, что скачивает удалённый репозиторий и сливает (merge) его с локальным. (Смотреть рисунок 11).

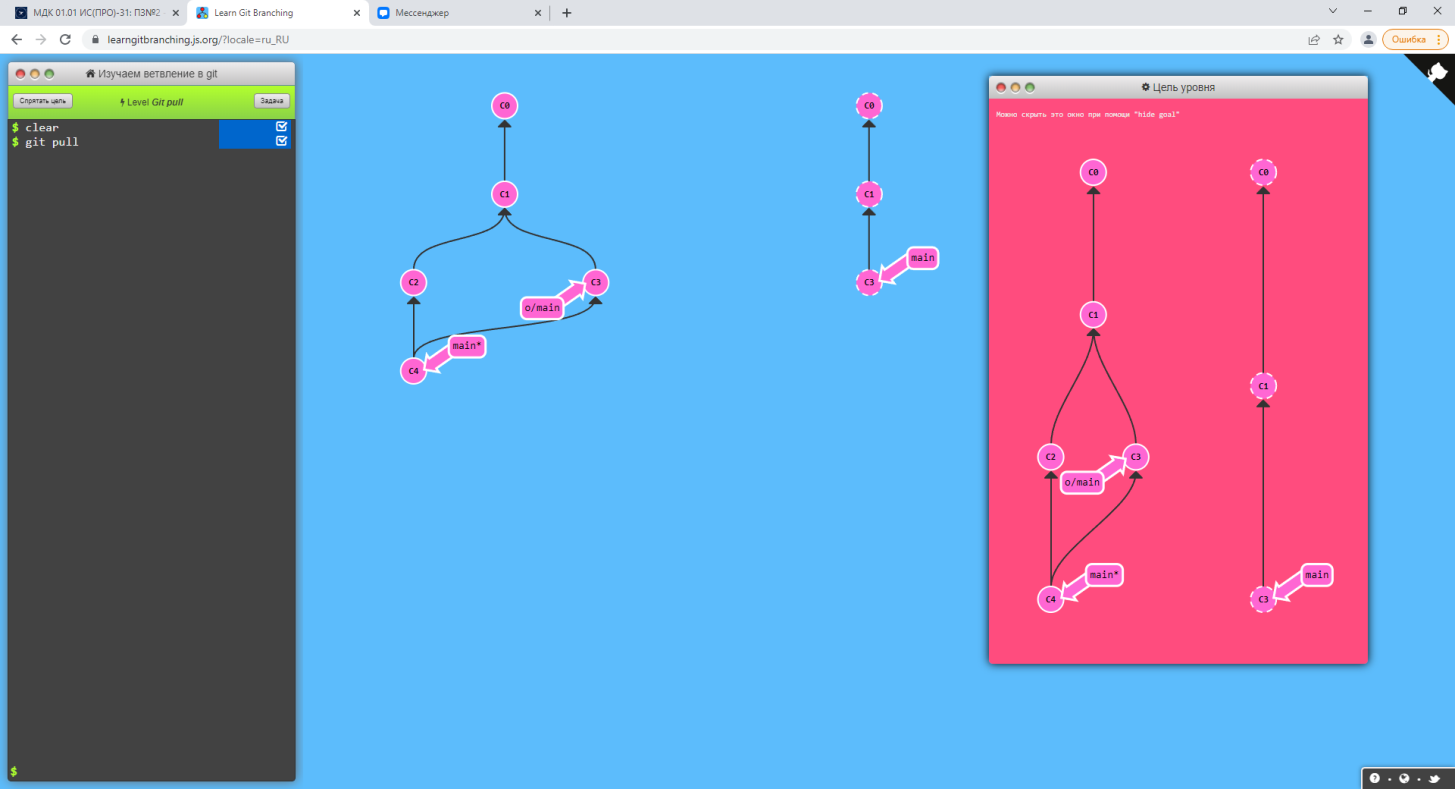


Рисунок 11 – Git pull

Задача №5 представляет команды из прошлых задач в “реальной” работе. С помощью команд git clone, git pull симулировали работу нескольких людей.(Смотреть рисунок 12)

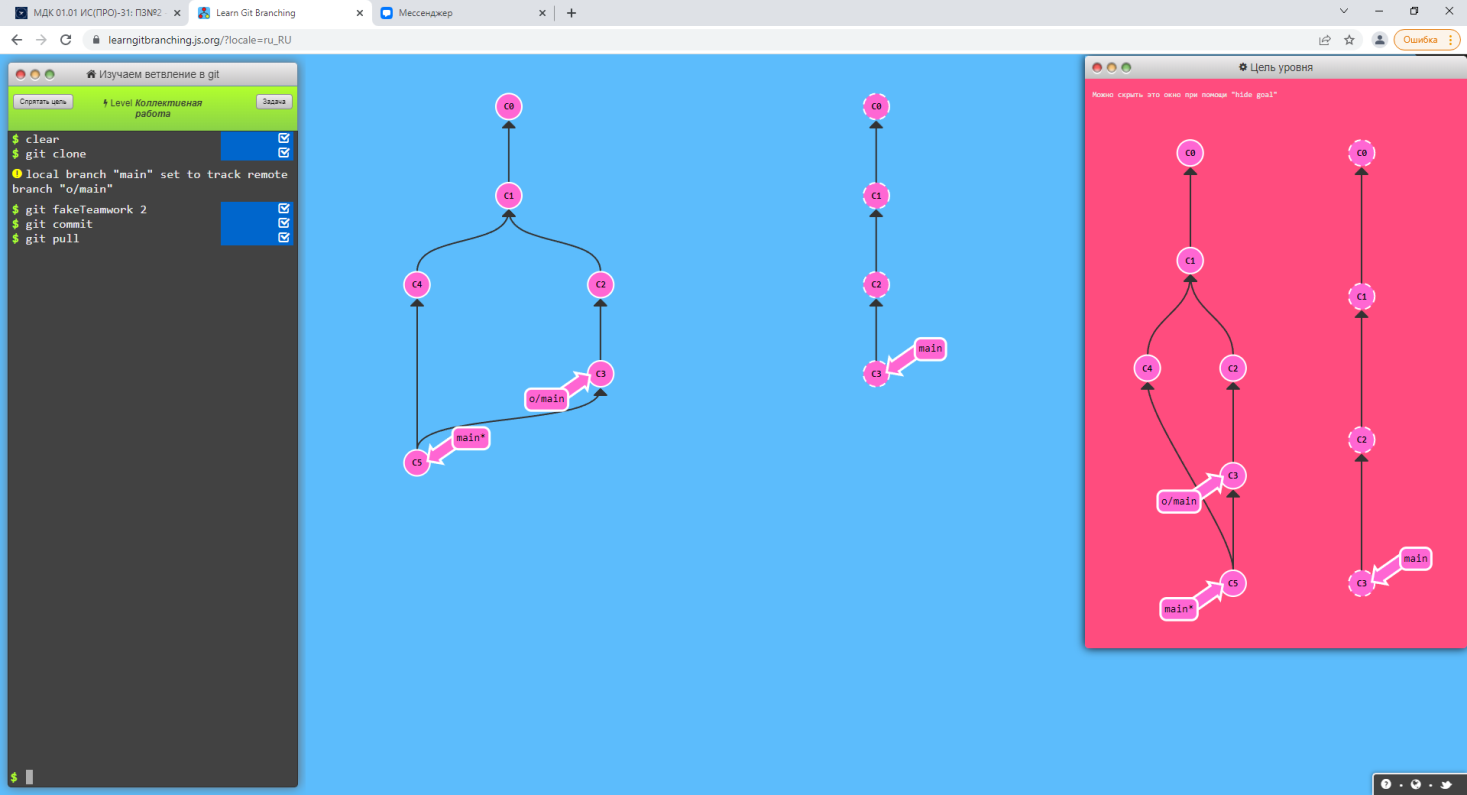


Рисунок 12 - коллективная работа

Задача №6 описывает поведение команды Git push, что выгружает данные из локального коммита в удаленный. (Смотреть рисунок 13).

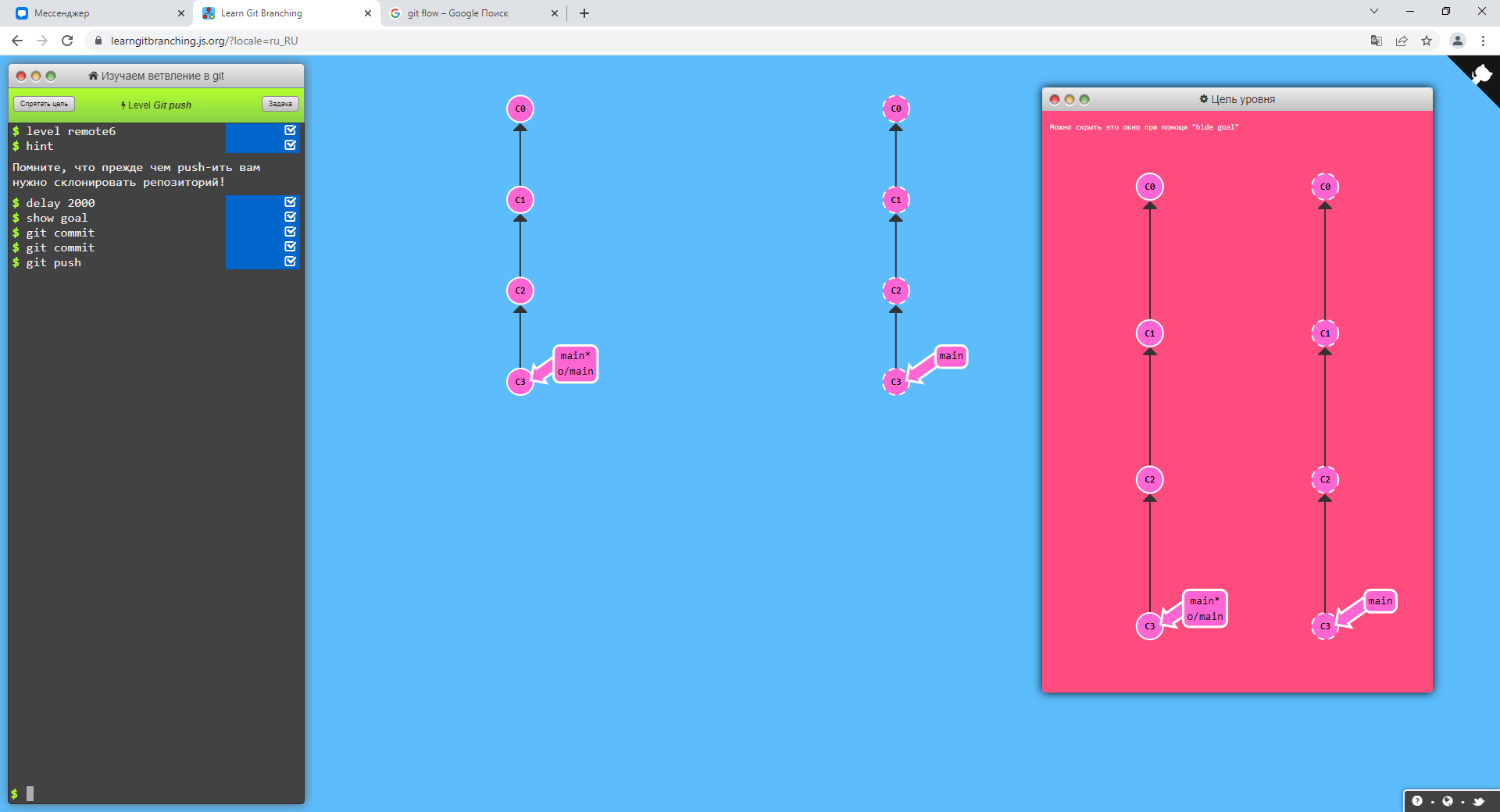


Рисунок 13 – Git Push

Задача №7 требует фабриковки работы группы, посредством использования команд fetching и pushing. (смотреть рисунок 14).

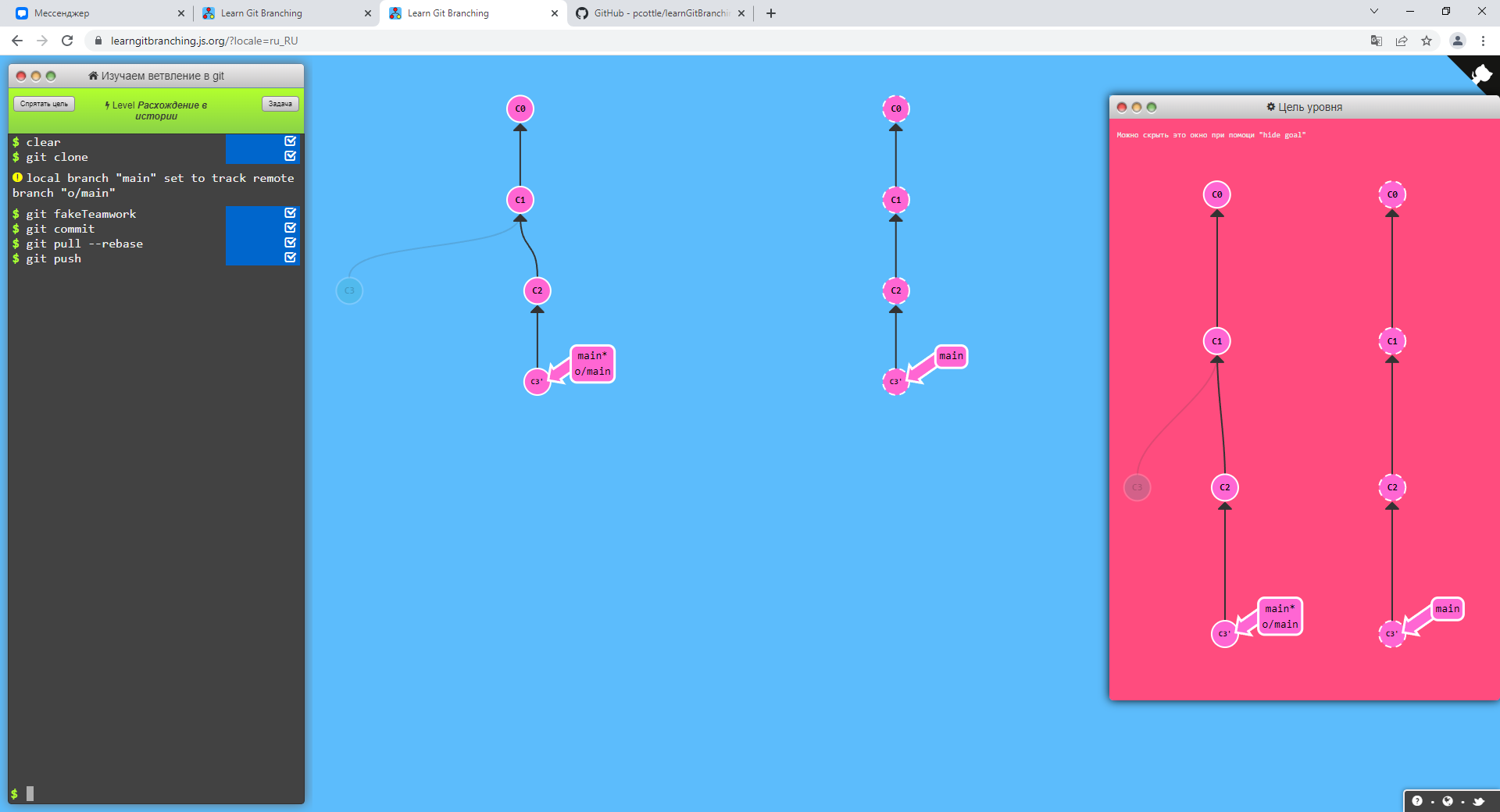


Рисунок 14 – Расхождение в истории

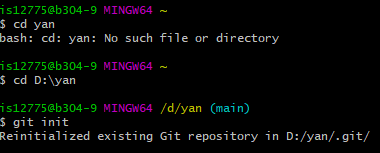


Рисунок 15 – Создание репозитория

git init – создает новый репозиторий.

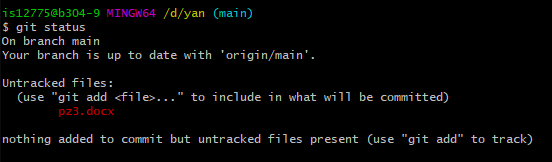


Рисунок 16 – Просмотр состояния файлов

git status – отображает состояние рабочего каталога с помощью ее можно проверить индексацию и изменений и увидеть файлы.



Рисунок 17 – Добавление каталога в индекс

git add- добавляет содержимое рабочего каталога в индекс.

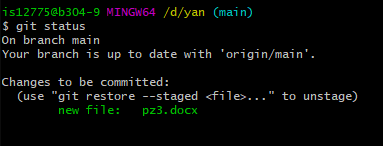


Рисунок 18 – Проверка файлов

Написал git status еще раз чтобы проверить что файлы добавлены.

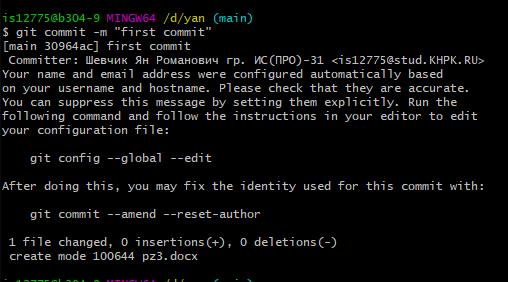


Рисунок 19 – Создание коммита

git commit –m “first commit” создаем коммит и называем его первый коммит.



Рисунок 20 – Создание ветки

git branch –M main- создаем ветку.



Рисунок 21 – Создание записи о подключении

git remote add origin- создание записи о новом подключении к удаленому репозиторию.

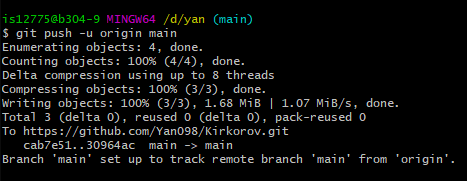


Рисунок 22 – Выгрузка в репозиторий

git push - для выгрузки содержимого локального репозитория в удаленый репозиторий.

Ссылка на репозиторий:

<https://github.com/Yan098/Kirkorov\Kirkorov.git>