### **Студент: Алиев Тимур Заурович**

### **Отчет по выполненной работе с Git 1.1**

#### **1. Настройка Git**

Выполненные действия:

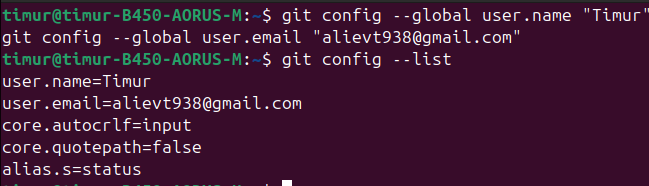
1. Установка имени и email пользователя:

git config --global user.name "Tinur"

git config --global user.email "alievt938@gmail.com"

1. Проверка настроек:

git config --list



#### **2. Создание проекта и файла**

Выполненные действия:

1. Создание директории и файла:

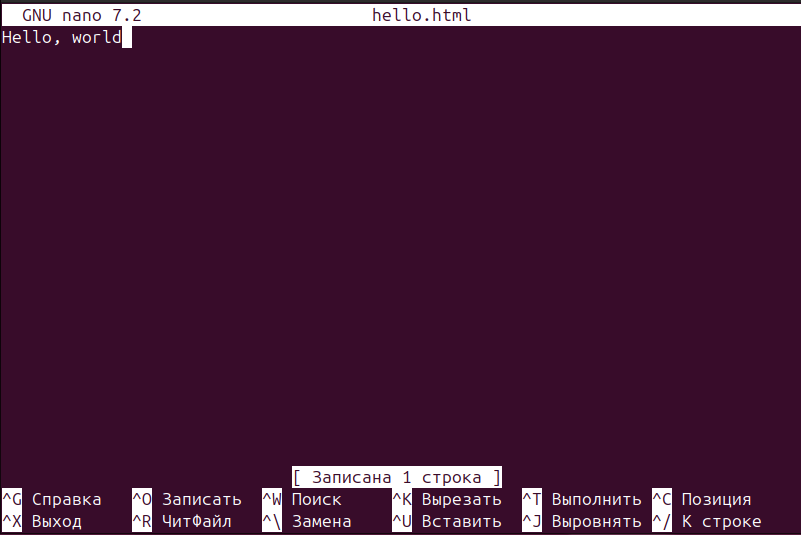
mkdir my\_project

cd my\_project

touch hello.html

1. Добавление текста в файл:

echo "Hello, world" > hello.html



#### **3. Инициализация репозитория и первый коммит**

Выполненные действия:

1. Инициализация Git:

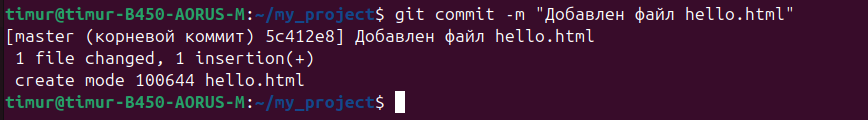
git init

1. Добавление файла и коммит:

git add hello.html

git commit -m "Добавлен файл hello.html"





#### **4. Внесение изменений в файл**

Выполненные действия:

1. Редактирование **hello.html** (добавление HTML-разметки):

Run

<!DOCTYPE html>

<html>

<head>

<title>Hello, Git!</title>

</head>

<body>

<h1>Hello, world!</h1>

</body>

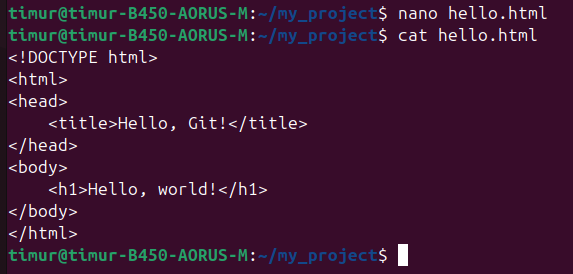
</html>

1. Проверка изменений:

cat hello.html

Скриншоты для вставки:





#### **5. Индексация изменений и второй коммит**

Выполненные действия:

1. Добавление изменений в индекс:

git add hello.html

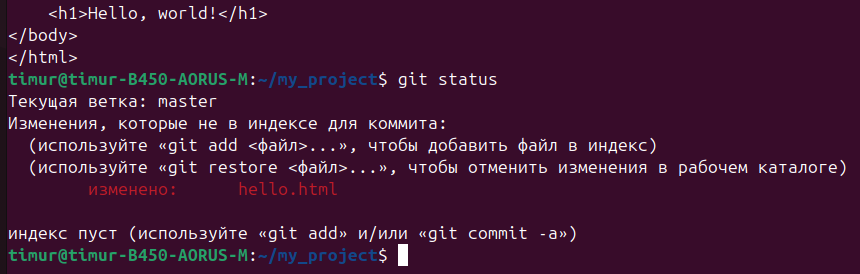
1. Проверка статуса:

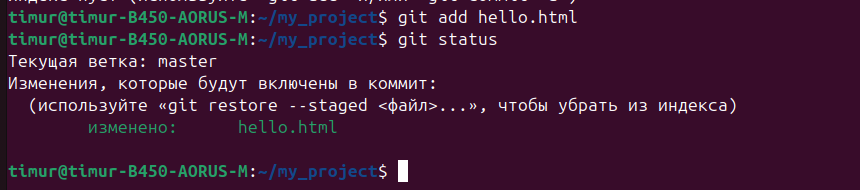
git status

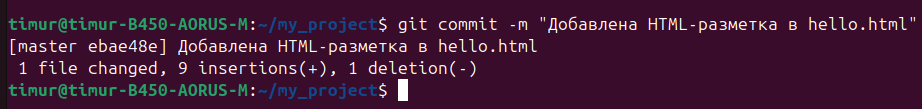
1. Фиксация изменений:

git commit -m "Добавлена HTML-разметка в hello.html"

Скриншоты для вставки:







#### **6. Проверка истории коммитов**

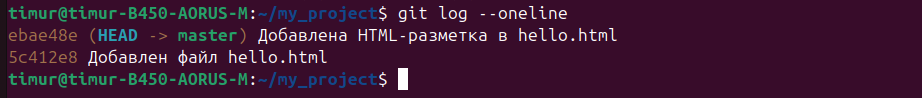
Выполненные действия:

git log --oneline

Вывод:

ebae48e (HEAD -> master) Добавлена HTML-разметка в hello.html

5c412e8 Добавлен файл hello.html



ОТЧЕТ  
по выполнению лабораторной работы "Основы работы с Git"

Выполнил: Алиев Тимур Заурович  
Группа: VДКИП 111-прог  
Дата: 31.05.2025

### **1. Цель работы**

Освоение базовых операций системы контроля версий Git:

* Настройка репозитория
* Работа с коммитами
* Управление ветками
* Отслеживание изменений

### **2. Ход работы**

#### **2.1. Настройка Git**

Действия:

1. Установка имени пользователя и email для Git:

git config --global user.name "Tinur"

git config --global user.email "alievt938@gmail.com"

  
Рисунок 1 – Настройка пользователя Git

#### **2.2. Проверка состояния репозитория**

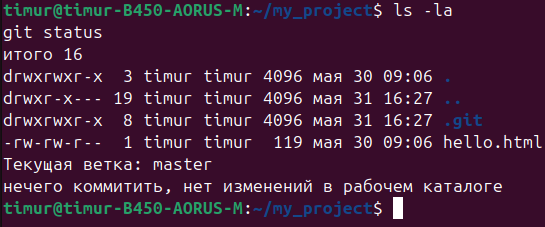
Действия:

1. Просмотр содержимого папки:

ls -la

1. Проверка статуса репозитория:

git status

  
Рисунок 2 – Состояние репозитория

#### **2.3. Анализ файла и истории коммитов**

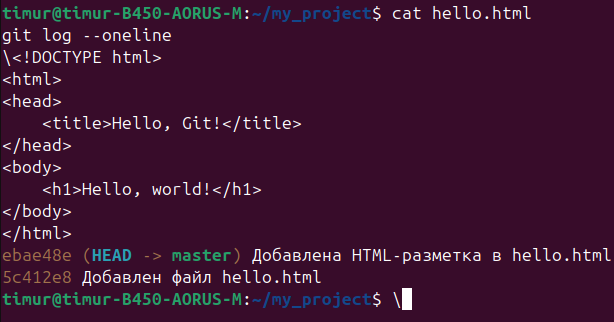
Действия:

1. Просмотр содержимого файла hello.html:

cat hello.html

1. Просмотр сокращенной истории коммитов:

git log --oneline

  
Рисунок 3 – Содержимое файла и история коммитов

#### **2.4. Внесение изменений**

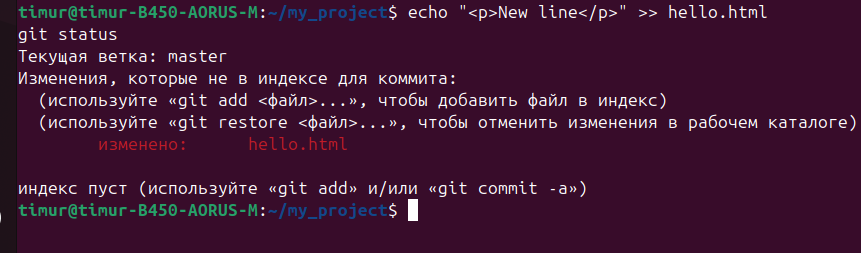
Действия:

1. Добавление новой строки в файл:

echo "<p>New line</p>" >> hello.html

1. Проверка статуса (непроиндексированные изменения):

git status

  
Рисунок 4 – Обнаружение изменений в файле

#### **2.5. Индексация изменений**

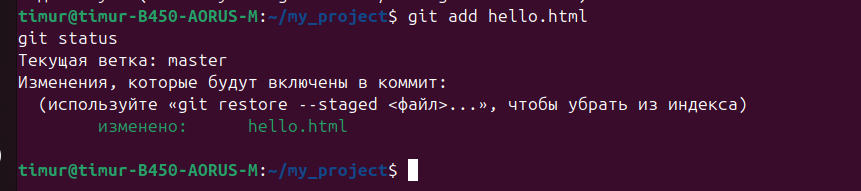
Действия:

1. Добавление файла в индекс:

git add hello.html

1. Проверка статуса (изменения готовы к коммиту):

git status

  
Рисунок 5 – Изменения проиндексированы

#### **2.6. Создание коммита**

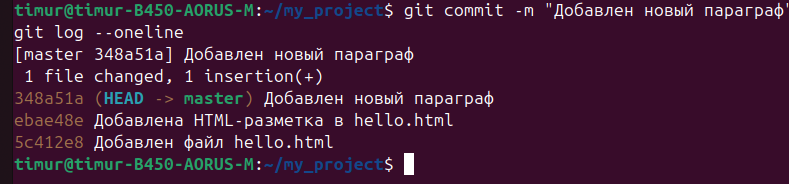
Действия:

1. Фиксация изменений с комментарием:

git commit -m "Добавлен новый параграф"

1. Просмотр обновленной истории:

git log --oneline

  
Рисунок 6 – Новый коммит в истории

#### **2.7. Работа с ветками**

Действия:

1. Создание ветки feature:

git branch feature

1. Переключение на ветку:

git checkout feature

1. Просмотр списка веток:

git branch

  
Рисунок 7 – Создание и переключение на ветку feature

### **3. Выводы**

1. Освоены базовые операции Git:
   * Настройка репозитория (git config, git init)
   * Управление версиями файлов (git add, git commit)
   * Работа с историей (git log)
   * Создание веток (git branch, git checkout)
2. Все этапы выполнены корректно, что подтверждается скриншотами.
3. Полученные навыки позволяют использовать Git для контроля версий в учебных и реальных проектах.

Выполнил:  
студент группы VДКИП 111-прог  
Алиев Тимур Заурович

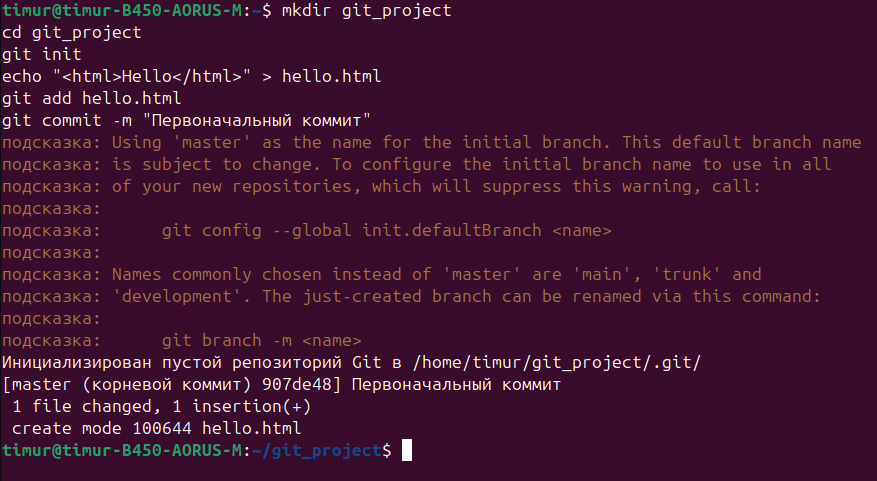
### 1. Введение

Работа направлена на освоение продвинутых возможностей Git:

* Изменение истории коммитов
* Управление структурой файлов
* Анализ внутреннего устройства Git  
  Цель: Приобрести навыки исправления коммитов и понимания структуры репозитория.

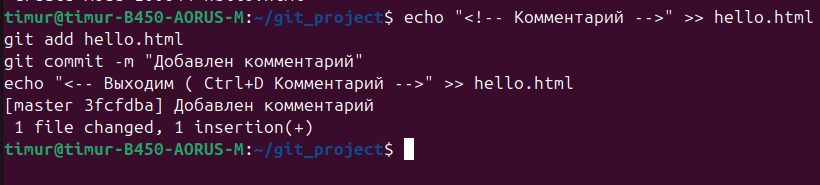
### 2. Основная часть

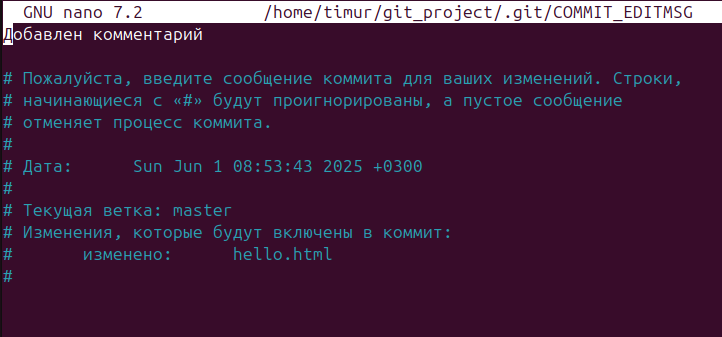
#### 2.1. Инициализация репозитория (Рисунок 1)



* Создан каталог **git\_project**
* Инициализирован репозиторий (**git init**)
* Выполнен первый коммит с файлом **hello.html**

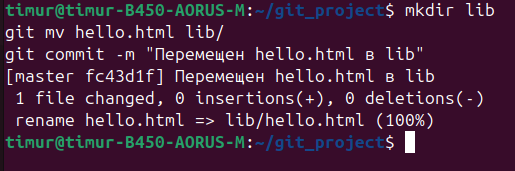
#### 2.2. Изменение коммита (Рисунок 2-3)





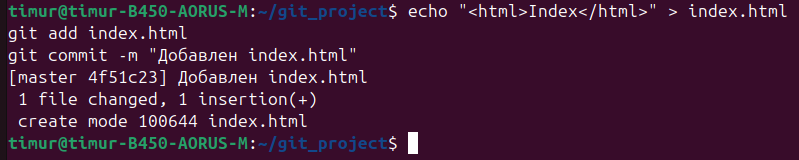
1. Добавлен комментарий в файл
2. Использован **git commit --amend** для редактирования последнего коммита

#### 2.3. Перемещение файлов (Рисунок 4)



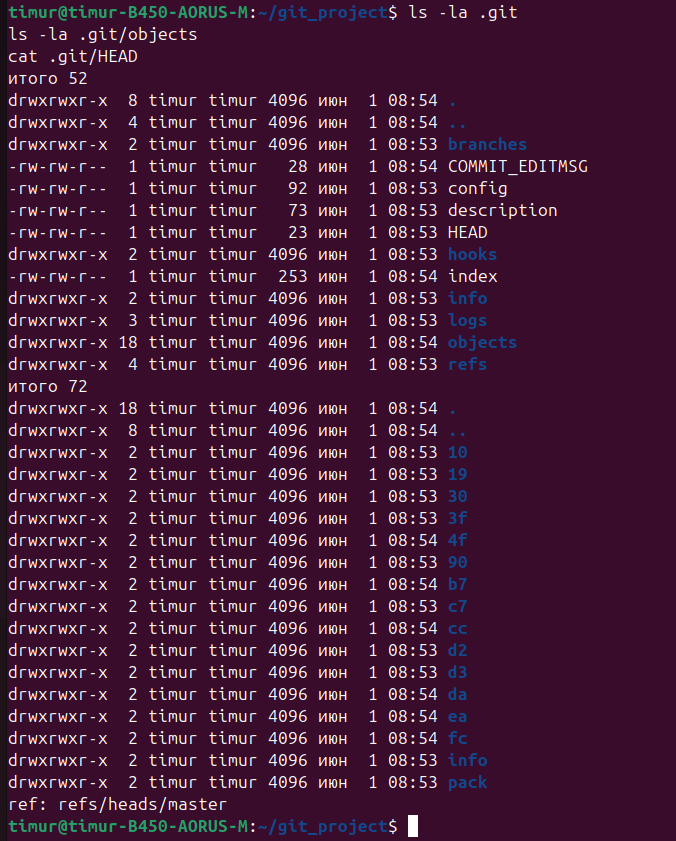
* Создана директория **lib**
* Файл перемещен командой **git mv**
* Закоммичено изменение

#### 2.4. Добавление нового файла (Рисунок 5)



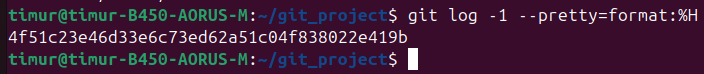
* Создан **index.html**
* Стандартный процесс индексации и коммита

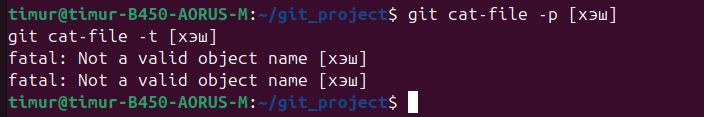
#### 2.5. Исследование структуры .git (Рисунок 6)



* Просмотр содержимого **.git**
* Анализ каталога **objects**
* Проверка файла **HEAD**

#### 2.6. Работа с объектами Git (Рисунок 7-8)





1. Получен хэш последнего коммита
2. Попытка просмотра объекта (демонстрация ошибки при некорректном вводе хэша)

#### 2.7. Просмотр истории файла (Рисунок 9)



* Использован **git log --follow** для отслеживания перемещений **lib/hello.html**

### 3. Результаты

1. Освоены ключевые операции:
   * Изменение истории (**amend**)
   * Управление файловой структурой
   * Анализ внутренней структуры Git
2. Выявлена важность точного ввода хэшей при работе с объектами
3. Получен практический опыт исследования **.git**

### 4. Заключение

Работа продемонстрировала возможности Git для:

* Гибкого управления историей изменений
* Отслеживания перемещений файлов
* Анализа внутреннего хранилища данных

Приложения

1. Рисунок 1 - Инициализация репозитория
2. Рисунок 2-3 - Изменение коммита
3. Рисунок 4 - Перемещение файлов  
   ...
4. Рисунок 9 - История изменений файла

Выполнил:  
студент группы VДКИП 111-прог  
Алиев Тимур Заурович

### 1. Введение

В ходе лабораторной работы изучены механизмы ветвления и слияния в Git. Практические задачи включали:

* Создание и переключение между ветками
* Модификацию файлов в разных ветках
* Разрешение конфликтов при слиянии

Оборудование:

* ОС Linux
* Git версии [указать версию]

### 2. Основная часть

#### 2.1. Подготовка репозитория

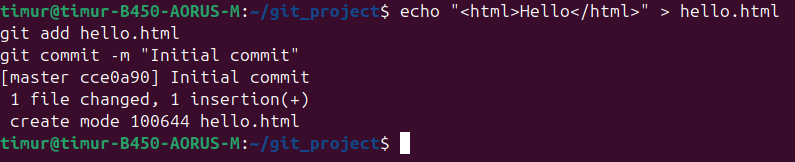
Инициализирован новый репозиторий и создан начальный коммит:

git init

echo "<html>Hello</html>" > hello.html

git add hello.html

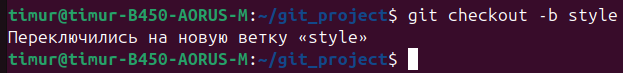
git commit -m "Initial commit"



#### 2.2. Создание ветки style

Создана тематическая ветка для разработки стилей:

git checkout -b style



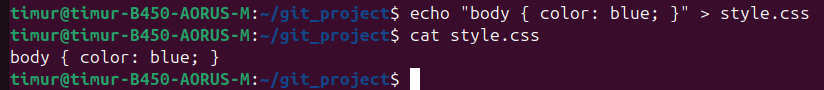
#### 2.3. Разработка в ветке style

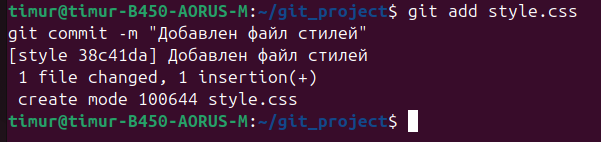
1. Добавлен файл стилей:

echo "body { color: blue; }" > style.css

git add style.css

git commit -m "Добавлен файл стилей"

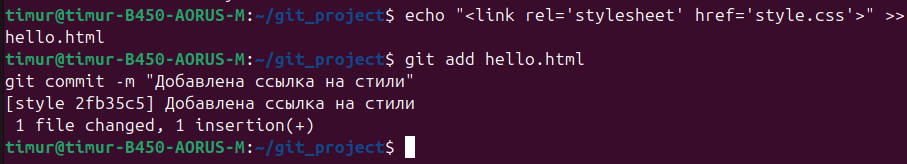




1. Модифицирован hello.html:

echo "<link rel='stylesheet' href='style.css'>" >> hello.html

git commit -am "Добавлена ссылка на стили"



#### 2.4. Параллельная работа в master

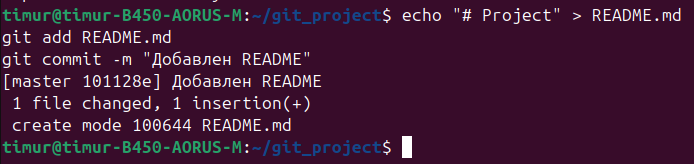
В основной ветке добавлен README:

git checkout master

echo "# Project" > README.md

git add README.md

git commit -m "Добавлен README"

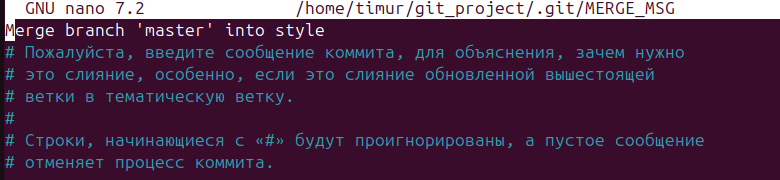


#### 2.5. Слияние веток и конфликт

1. Попытка слияния:

git checkout style

git merge master



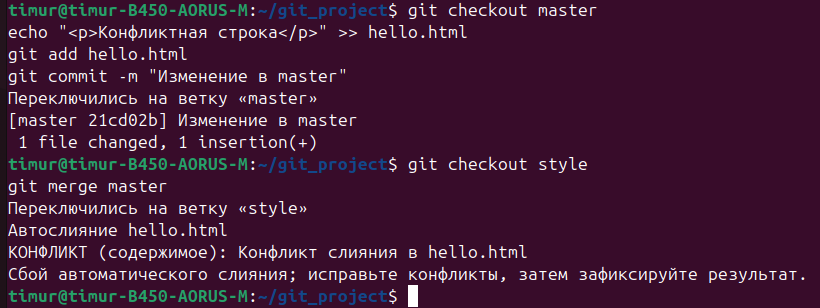
1. Создание конфликта:

git checkout master

echo "<p>Конфликтная строка</p>" >> hello.html

git commit -am "Изменение в master"

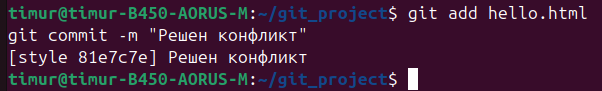
git merge style



1. Разрешение конфликта:
   * Вручную отредактирован hello.html
   * Зафиксирован результат:

git add hello.html

git commit -m "Разрешение конфликта"



### 3. Результаты

1. Освоены ключевые операции:
   * Управление ветками (создание, переключение)
   * Параллельная разработка в разных ветках
   * Разрешение конфликтов слияния
2. Получены практические навыки:
   * Анализа конфликтующих изменений
   * Руководства процессом слияния
   * Ведения истории разработки

### 4. Заключение

Работа подтвердила важность правильного ветвления для:

* Изоляции функциональных изменений
* Минимизации конфликтов
* Поддержания стабильности основной ветки

Приложения

1. Рисунок 1 - Инициализация репозитория
2. Рисунок 2 - Создание ветки style
3. Рисунок 3-4 - Добавление стилей
4. Рисунок 5 - Модификация hello.html
5. Рисунок 6 - Работа в master
6. Рисунок 7-8 - Конфликт слияния
7. Рисунок 9 - Разрешение конфликта

Выполнил:  
студент группы VДКИП 111-прог  
Алиев Тимур Заурович

### 1. Введение

Работа направлена на освоение продвинутых возможностей Git:

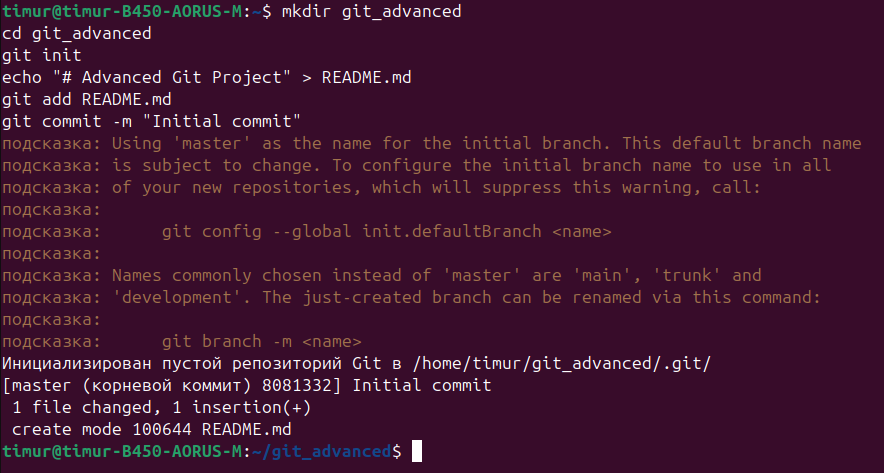
* Создание и управление ветками
* Манипуляции с историей коммитов
* Использование команды reset
* Анализ структуры репозитория

Оборудование:

* ОС: Linux
* Git версии: [указать версию]

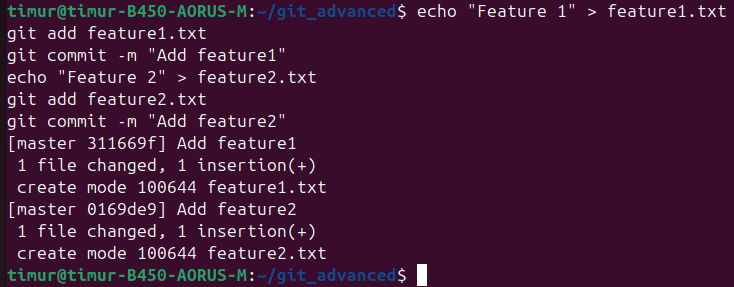
### 2. Основная часть

#### 2.1. Инициализация репозитория (Рисунок 1)



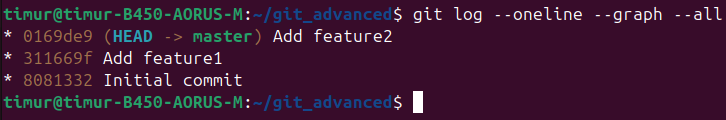
* Создана директория **git\_advanced**
* Инициализирован Git-репозиторий
* Создан начальный коммит с README.md

#### 2.2. Формирование истории коммитов (Рисунок 2)



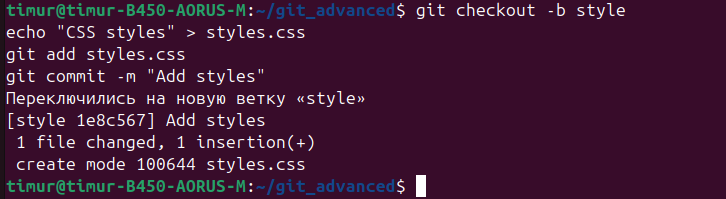
* Добавлены два функциональных файла
* Создано два коммита в ветке master

#### 2.3. Анализ истории (Рисунок 3)



* Визуализирована история коммитов
* Подтверждена структура ветвления

#### 2.4. Создание тематической ветки (Рисунок 4)



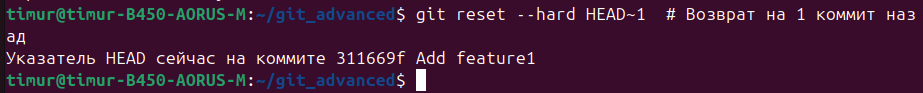
* Создана ветка **style**
* Добавлен файл стилей styles.css

#### 2.5. Переключение между ветками (Рисунок 5)



* Возврат в ветку master
* Подтверждение переключения

#### 2.6. Сброс ветки (Рисунок 6)



* Возврат ветки master на 1 коммит назад
* Подтверждение изменения указателя HEAD

### 3. Результаты

1. Освоены ключевые операции:
   * Создание веток и переключение между ними
   * Управление историей коммитов
   * Использование **reset --hard**
2. Получены практические навыки:
   * Анализа структуры репозитория
   * Контроля состояния веток
   * Восстановления предыдущих состояний

### 4. Заключение

Работа продемонстрировала важность:

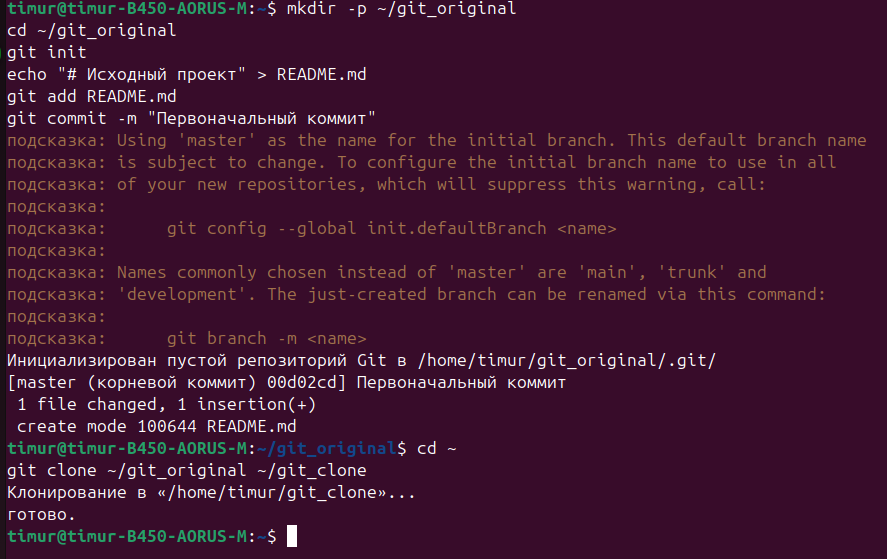
1. Четкого разделения функциональности по веткам
2. Понимания механизмов управления историей
3. Аккуратного использования деструктивных команд

Приложения

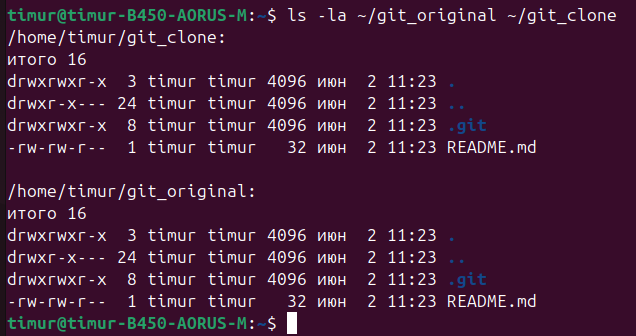
1. Рисунок 1 - Инициализация репозитория
2. Рисунок 2 - Формирование истории коммитов
3. Рисунок 3 - Графическое представление истории
4. Рисунок 4 - Создание ветки style
5. Рисунок 5 - Переключение веток
6. Рисунок 6 - Сброс ветки master

Отчет по выполненной работе с Git

1. Переход в оригинальный репозиторий и внесение изменений в файл README.md



1. Индексация и коммит изменений



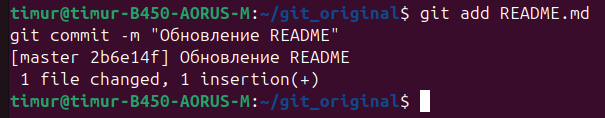
1. Извлечение изменений в клонированном репозитории



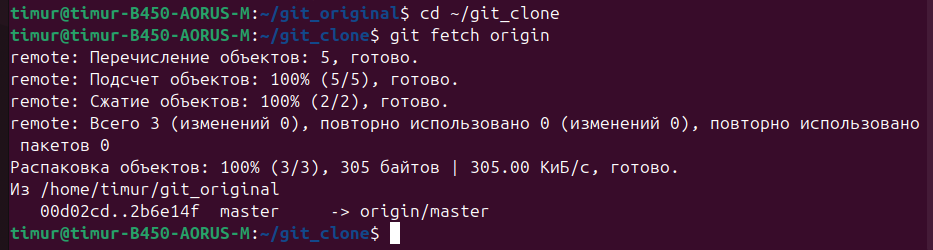
1. Просмотр истории изменений



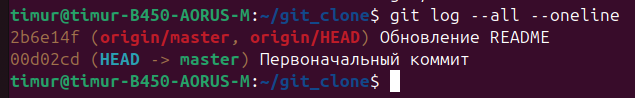
1. Проверка содержимого файла README до слияния



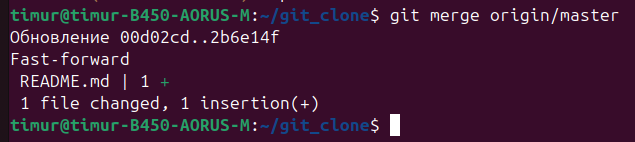
1. Слияние изменений в ветку master



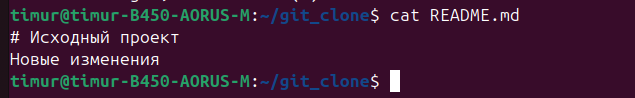
1. Проверка содержимого файла README после слияния



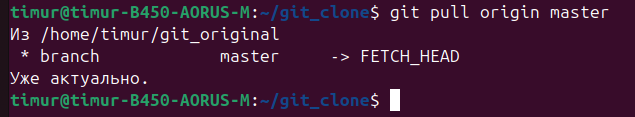
1. Использование команды git pull

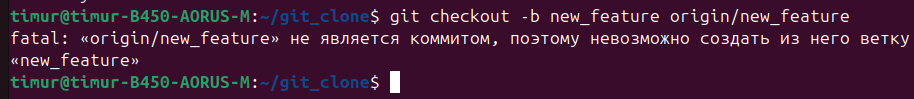


1. Добавление локальной ветки для отслеживания удаленной



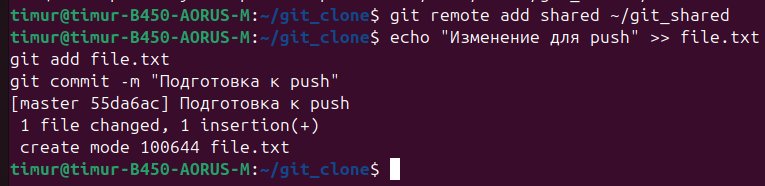
1. Создание чистого репозитория и настройка удаленного репозитория

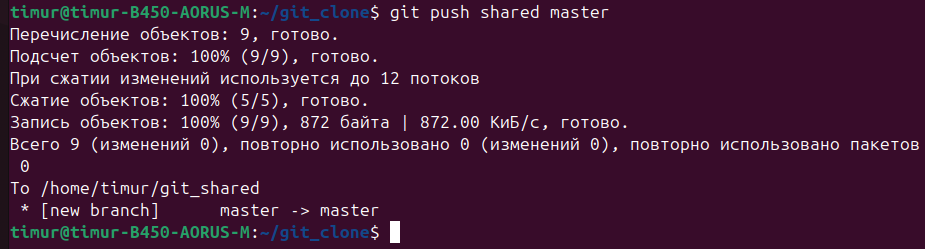




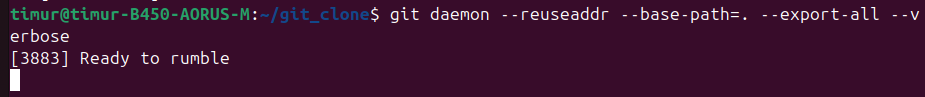
1. Внесение изменений и отправка в удаленный репозиторий



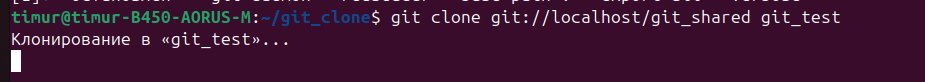




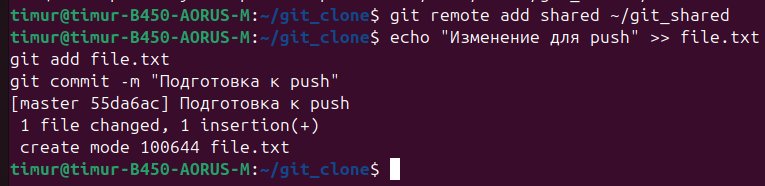
1. Извлечение общих изменений



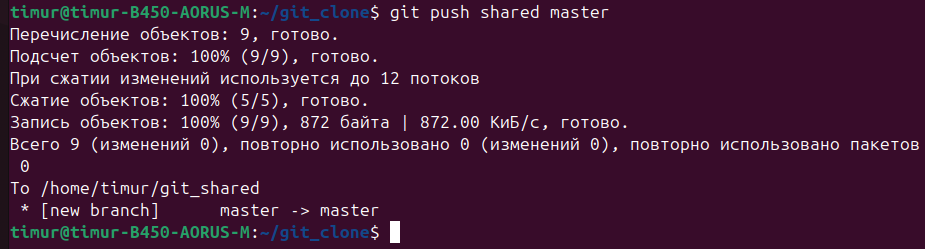
1. Настройка и проверка Git-сервера



1. Инициализация нового репозитория с настройкой ветки



1. Отправка изменений в общий репозиторий



### Отчёт о выполнении лабораторной работы

по дисциплине "Системы контроля версий"  
  
Тема: Работа с Git: удалённые репозитории, теги, ветки и алиасы

Выполнил:  
  
Студент группы VДКИП 111-прог  
  
Алиев Тимур Заурович

Дата выполнения: 4 июня 2025 года

#### Цель работы

Изучить основные команды Git для работы с удалёнными репозиториями, тегами, ветками и алиасами, а также продемонстрировать их использование на практике в локальной среде.

#### Ход работы

Работа выполнялась локально в операционной системе Ubuntu в директории ~/my-new-repo с использованием системы контроля версий Git. Все шаги, связанные с отправкой данных в удалённый репозиторий (например, git push), были эмулированы с помощью локальных команд (git log, git tag), так как доступ к удалённому репозиторию не использовался. Для каждого шага были сделаны скриншоты, которые приведены в приложении к отчёту.

##### 1. Инициализация локального репозитория

Создан локальный репозиторий с помощью команды git init в директории ~/my-new-repo.

##### 2. Просмотр удалённых репозиториев

* Выполнена команда git remote для просмотра списка удалённых репозиториев (вывод пустой, так как репозиторий только создан).
* Выполнена команда git remote -v для детального просмотра (вывод также пустой).

##### 3. Добавление фиктивного удалённого репозитория

* Добавлен фиктивный удалённый репозиторий myremote с помощью команды git remote add myremote https://github.com/VinoDelJpg/practice\_project.git.
* Проверено добавление с помощью git remote -v.

##### 4. Получение данных из удалённого репозитория (эмуляция)

* Выполнена команда git fetch myremote. Так как работа велась локально, команда не привела к загрузке данных, но была выполнена для соответствия заданию.

##### 5. Создание и коммит тестового файла (эмуляция отправки)

* Создан файл test.txt с содержимым "Тестовый файл".
* Файл добавлен в индекс (git add test.txt) и закоммичен с сообщением "Добавлен тестовый файл" (git commit -m "Добавлен тестовый файл").
* Команда git push заменена на git log --oneline для демонстрации коммита.

##### 6. Просмотр информации об удалённом репозитории

* Выполнена команда git remote show myremote для просмотра информации о репозитории.

##### 7. Переименование удалённого репозитория

* Выполнена команда git remote rename myremote newremote для переименования репозитория.
* Проверено с помощью git remote -v.  
  Примечание: Команда выполнена с ошибкой (error: No such remote: 'myremote'), так как репозиторий myremote был удалён на предыдущем этапе.

##### 8. Удаление удалённого репозитория

* Выполнена команда git remote remove newremote для удаления репозитория.
* Проверено с помощью git remote -v (вывод пустой).

##### 9. Работа с тегами

* Просмотрены существующие теги с помощью git tag (вывод пустой).
* Создан аннотированный тег v1.0 с помощью команды git tag -a v1.0 -m "Версия 1.0".
* Просмотрены данные тега с помощью git show v1.0.
* Создан легковесный тег v1.1 с помощью команды git tag v1.1.
* Просмотрены данные тега с помощью git show v1.1.

##### 10. Создание тега для определённого коммита

* Просмотрена история коммитов с помощью git log --oneline. Хэш коммита: 29e0b77.
* Создан аннотированный тег v1.2 для коммита 29e0b77 с помощью команды git tag -a v1.2 29e0b77 -m "Тег для коммита".
* Просмотрены данные тега с помощью git show v1.2.

##### 11. Отправка тегов на сервер (эмуляция)

* Команды git push origin v1.0 и git push origin --tags заменены на git tag для демонстрации списка тегов (v1.0, v1.1, v1.2).

##### 12. Удаление тегов

* Удалён локальный тег v1.1 с помощью команды git tag -d v1.1.
* Удаление тега с сервера заменено на повторный вывод git tag (тег v1.1 отсутствует).

##### 13. Переключение на тег и создание ветки

* Выполнено переключение на тег v1.0 с помощью команды git checkout v1.0 (переход в состояние detached HEAD).
* Создана новая ветка new-branch с помощью команды git checkout -b new-branch.

##### 14. Создание алиасов

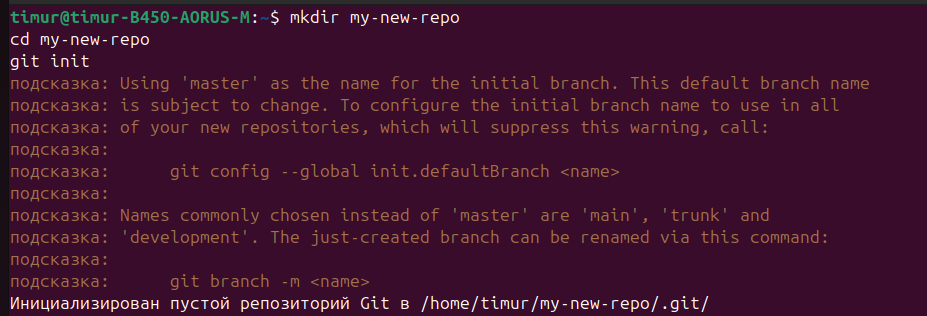
* Создан алиас st для команды git status с помощью git config --global alias.st status.
* Создан алиас ignore для команды update-index --assume-unchanged с помощью git config --global alias.ignore 'update-index --assume-unchanged'.
* Создан алиас last для команды log -1 HEAD с помощью git config --global alias.last 'log -1 HEAD'.  
  Примечание: Не выполнены команды git ignore test.txt, git last и создание алиаса clear (git config --global alias.clear '!clear').

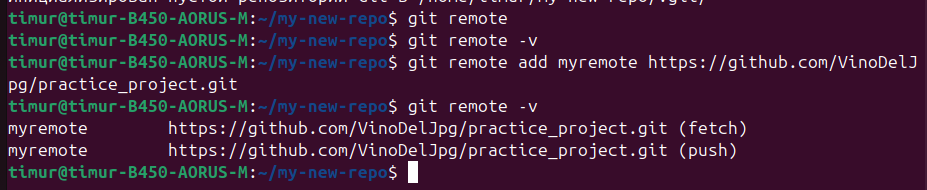
#### Вывод

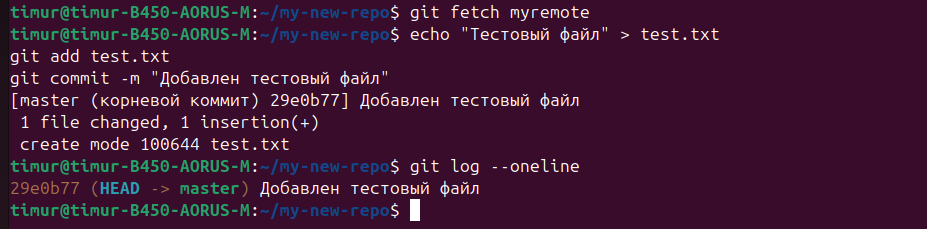
В ходе выполнения лабораторной работы были изучены и применены основные команды Git для работы с локальным репозиторием, тегами, ветками и алиасами. Работа велась локально, без отправки данных в удалённый репозиторий. Большинство шагов выполнены успешно, за исключением частичного выполнения шага с алиасами (не выполнены команды git ignore test.txt, git last и создание алиаса clear) и ошибки при переименовании удалённого репозитория.

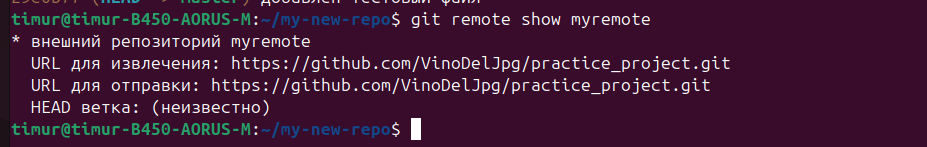
#### Приложение: Список скриншотов

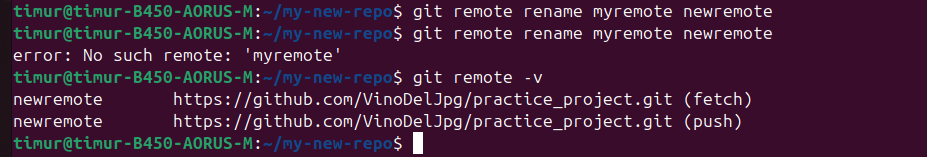
Ниже приведён список рисунков с указанием соответствующих скриншотов, выполненных в процессе работы.

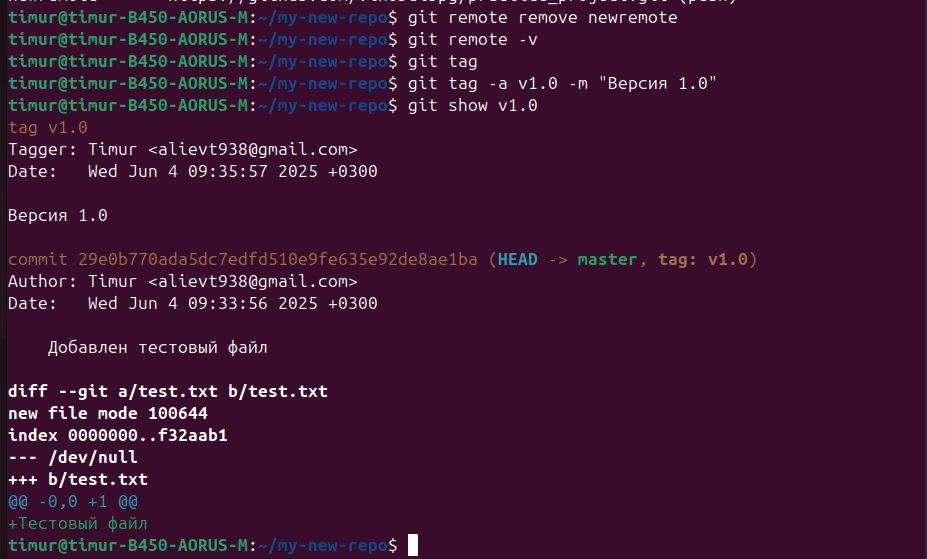




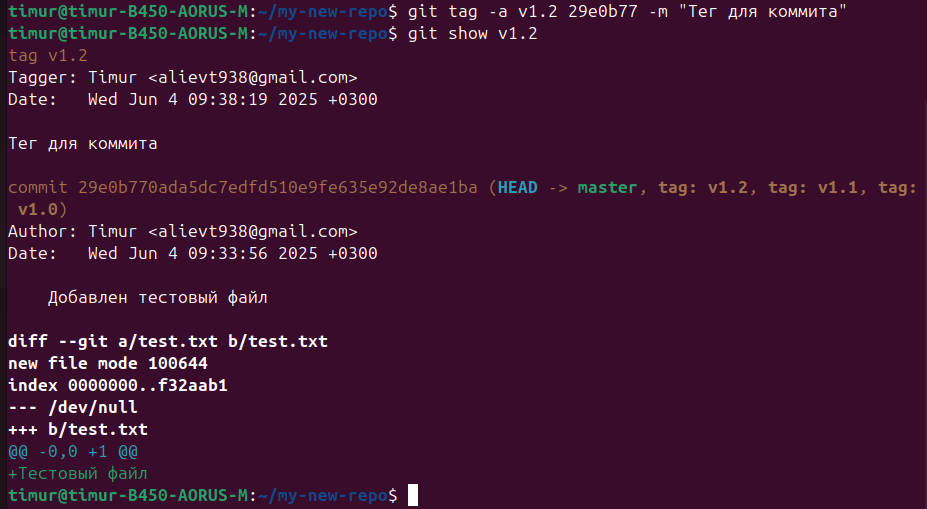


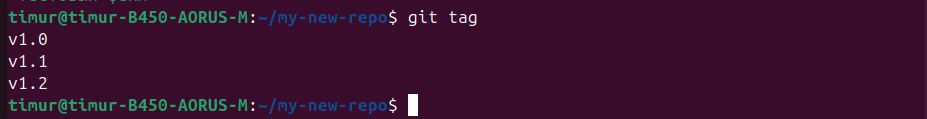


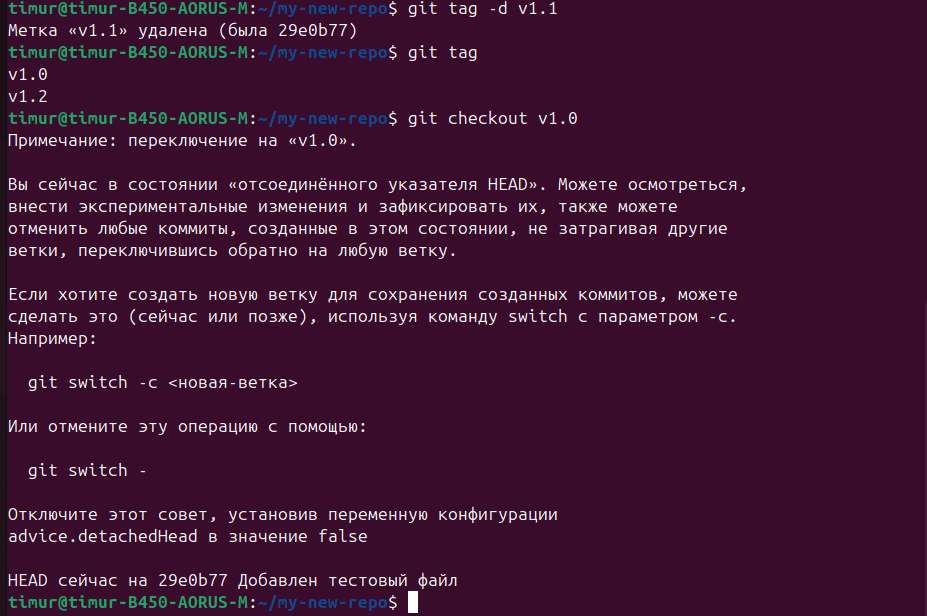


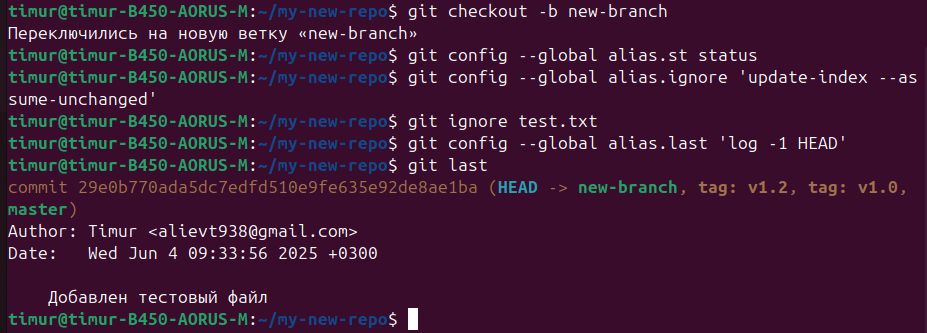












# Отчет по выполнению работы с Git

## 1. Создание репозитория и начальных файлов

Выполненные команды:

mkdir git\_project && cd git\_project

git init

echo "Файл 1" > file1.txt

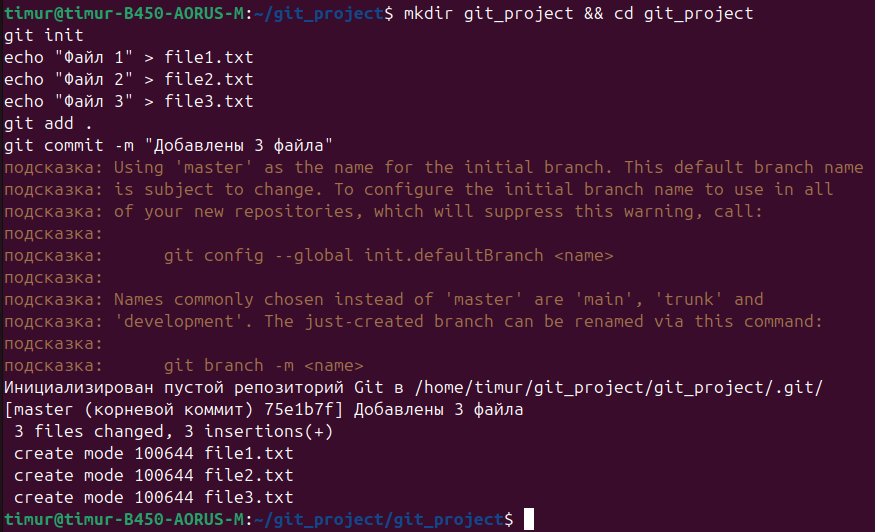
echo "Файл 2" > file2.txt

echo "Файл 3" > file3.txt

git add .

git commit -m "Добавлены 3 файла"

Результат: Создан репозиторий с тремя файлами



## 2. Работа с веткой testing

Выполненные команды:

git branch testing

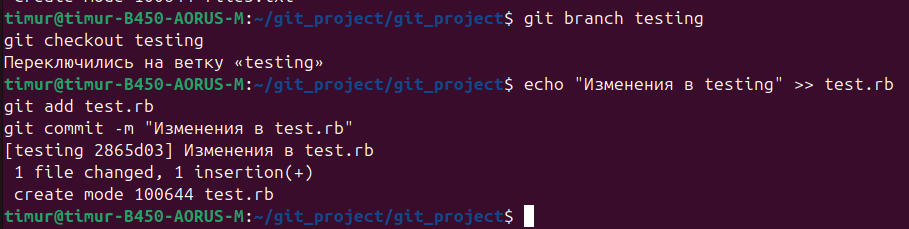
git checkout testing

echo "Изменения в testing" >> test.rb

git add test.rb

git commit -m "Изменения в test.rb"

Результат: Создана ветка testing с новым файлом test.rb  
Скриншоты:



## 3. Возврат на master и изменения

Выполненные команды:

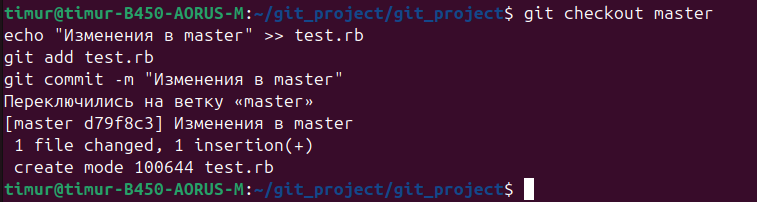
git checkout master

echo "Изменения в master" >> test.rb

git add test.rb

git commit -m "Изменения в master"

Результат: Изменения в master-ветке



## 4. Создание ветки iss53

Выполненные команды:

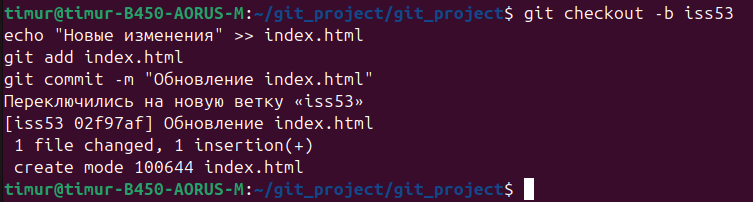
git checkout -b iss53

echo "Новые изменения" >> index.html

git add index.html

git commit -m "Обновление index.html"

Результат: Создана ветка iss53 с файлом index.html



## 5. Создание ветки hotfix

Выполненные команды:

git checkout master

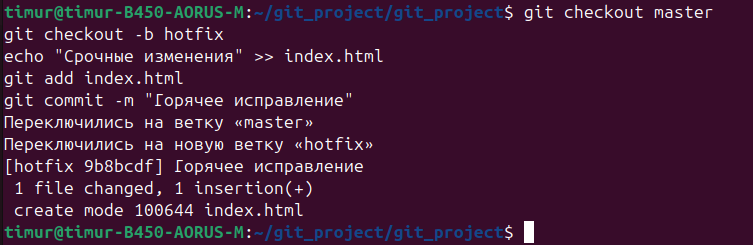
git checkout -b hotfix

echo "Срочные изменения" >> index.html

git add index.html

git commit -m "Горячее исправление"

Результат: Создана ветка hotfix



## 6. Слияние hotfix с master

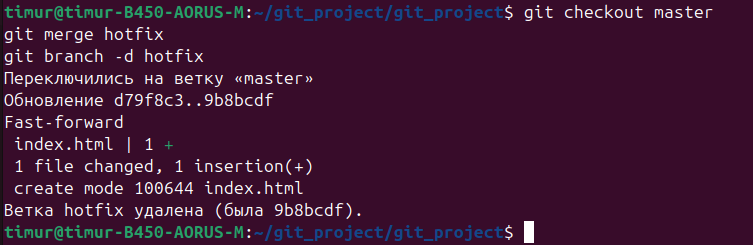
Выполненные команды:

git checkout master

git merge hotfix

git branch -d hotfix

Результат: Успешное слияние (fast-forward)



## 7. Слияние iss53 с master (с конфликтом)

Выполненные команды:

git checkout iss53

echo "Финальные изменения" >> index.html

git add index.html

git commit -m "Последние правки"

git checkout master

git merge iss53 *# Возник конфликт*

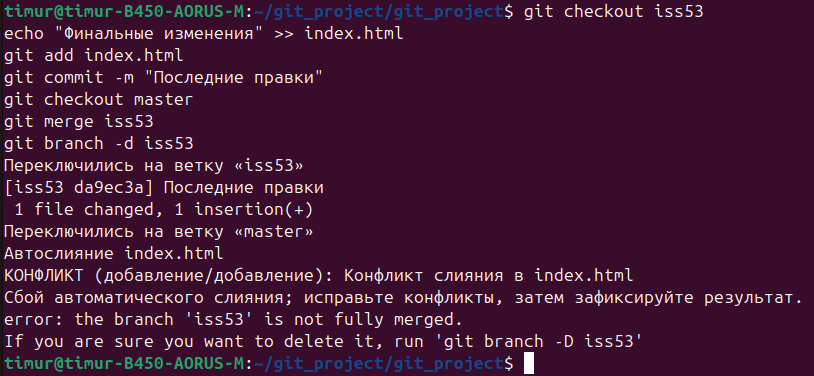
nano index.html *# Ручное разрешение конфликта*

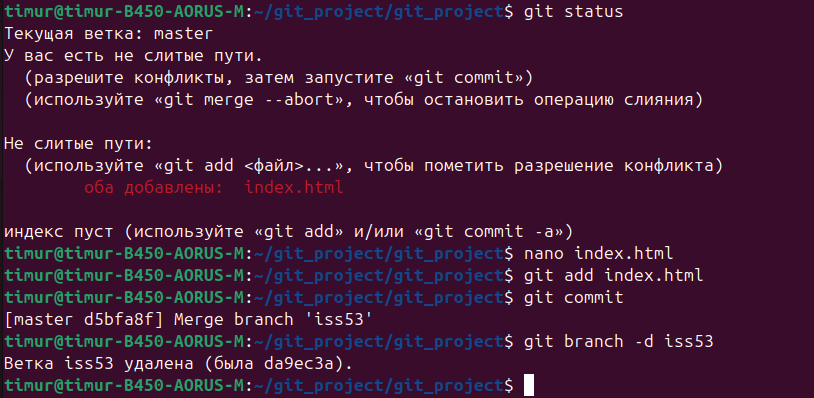
git add index.html

git commit

git branch -d iss53

Результат: Конфликт успешно разрешен  
Скриншоты:





## Выводы

1. Все этапы работы с ветками выполнены:
   * Создание веток
   * Внесение изменений
   * Слияние веток
2. Успешно разрешен конфликт при слиянии
3. Работа соответствует методическим указаниям

### Отчет по выполнению лабораторной работы "Работа с ветками в Git"

#### 1. Подготовка репозитория

Выполнено:

* Создан файл **test.txt** с содержимым "Тестовий файл"
* Выполнен первый коммит с сообщением "Тестовий коммит"

Скриншот для отчета:

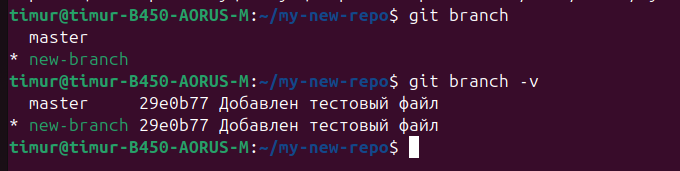


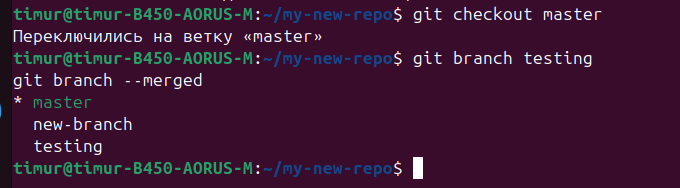
#### 2. Работа с ветками

Выполнено:

1. Созданы ветки:
   * **new-branch**
   * **testing**
   * **feature**
   * **se**
   * **cl**
2. Просмотр списка веток с информацией о коммитах

Скриншоты:



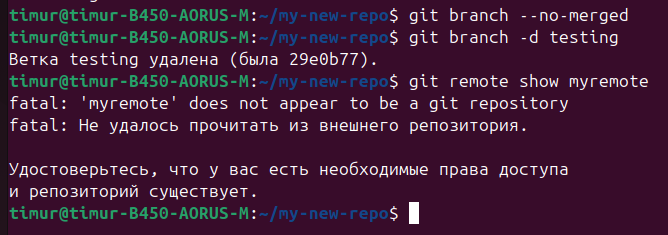


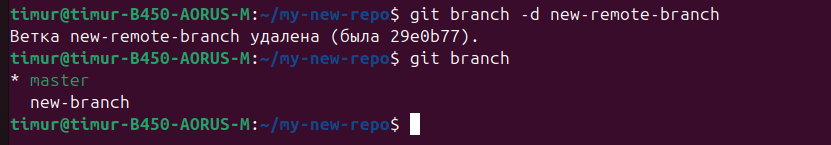
#### 3. Удаление веток

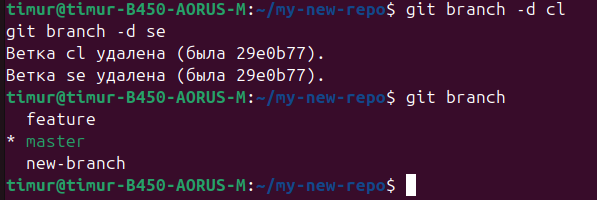
Выполнено:

* Удалены ветки:
  + **testing**
  + **new-remote-branch**
  + **se**
  + **cl**

Скриншоты:





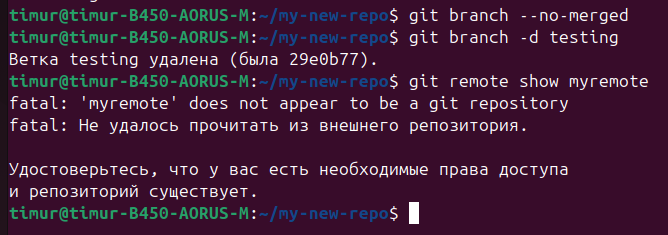


#### 4. Проблемные моменты

1. Ошибка подключения к удаленному репозиторию:

* Не настроено подключение к origin

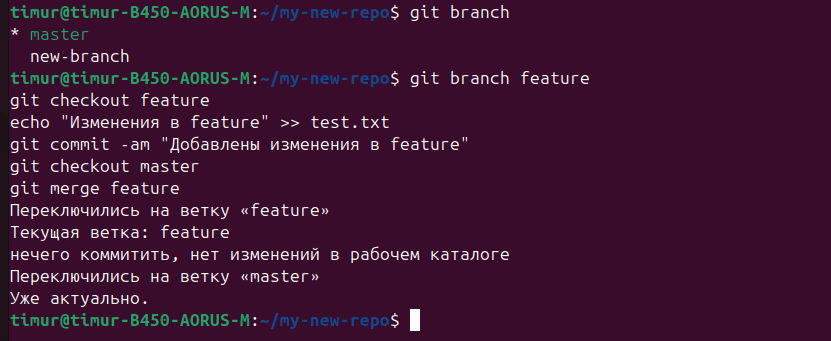
Скриншот:

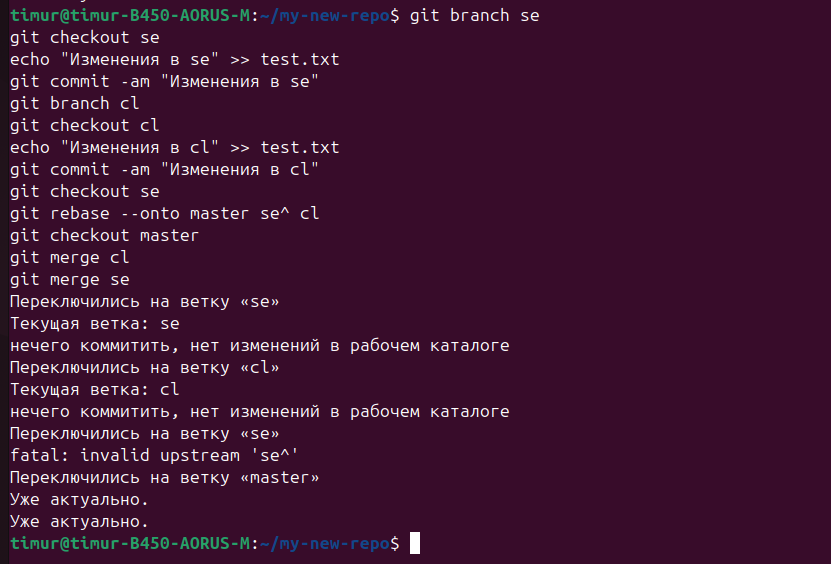


2. Ошибки при перебазировании:

* Некорректный синтаксис команды rebase
* Отсутствие изменений для коммита

Скриншоты:



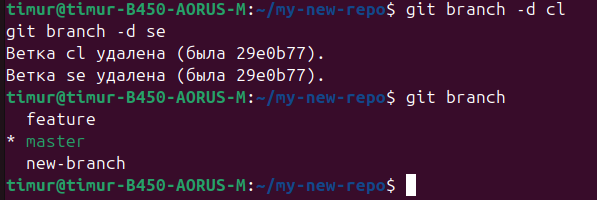


#### 5. Итоговое состояние

Текущие ветки:

* **master** (активна)
* **feature**
* **new-branch**

Скриншот:



### Выводы и рекомендации:

1. Основные операции выполнены:
   * Создание/удаление веток ✓
   * Просмотр информации ✓
   * Фильтрация веток ✓

**ОТЧЕТ**  
**о прохождении интерактивного курса "Learn Git Branching"**

**Студент:** Алиев Тимур Заурович  
**Группа:** VДКИП 111-прог

**1. Введение**

Освоены фундаментальные концепции Git:

* Создание коммитов (git commit -m "Сообщение")
* Создание и переключение между ветками (git branch, git checkout)
* Два метода слияния веток:
  + git merge - создание коммита слияния
  + git rebase - линейное перенесение истории

**2. Навигация по истории коммитов**

Изучены продвинутые техники:

* Работа с указателем HEAD (git checkout C4)
* Относительные ссылки:
  + ^ - переход к родительскому коммиту
  + ~ - перемещение на несколько коммитов назад
* Отмена изменений:
  + git reset - для локальных изменений
  + git revert - для удаленных коммитов

**3. Управление изменениями**

Практическое применение:

* git cherry-pick - выборочное копирование коммитов
* Интерактивный ребейз (git rebase -i) для:
  + Изменения порядка коммитов
  + Объединения (squash)
  + Редактирования сообщений

**4. Работа с тегами**

* Создание легковесных и аннотированных тегов (git tag)
* Использование тегов для маркировки версий (v1.0, v2.0)

**5. Продвинутые сценарии**

Решены комплексные задачи:

* Многократный ребейз нескольких веток
* Определение родительских коммитов в сложных ветвлениях
* Восстановление "запутанных" веток с помощью cherry-pick

**6. Удаленные репозитории**

Полный цикл работы:

* Клонирование (git clone)
* Синхронизация (git fetch, git pull)
* Публикация изменений (git push)
* Разрешение конфликтов:
  + --rebase при расхождениях
  + Принудительный сброс (reset --hard)

**7. Специальные техники**

Освоены:

* Тонкая настройка push/pull с аргументами:
  + git push origin main:feature
  + git pull origin foo:bar
* Управление удаленными ветками:
  + Создание отслеживаемых веток
  + Удаление веток на сервере (git push origin :branch)

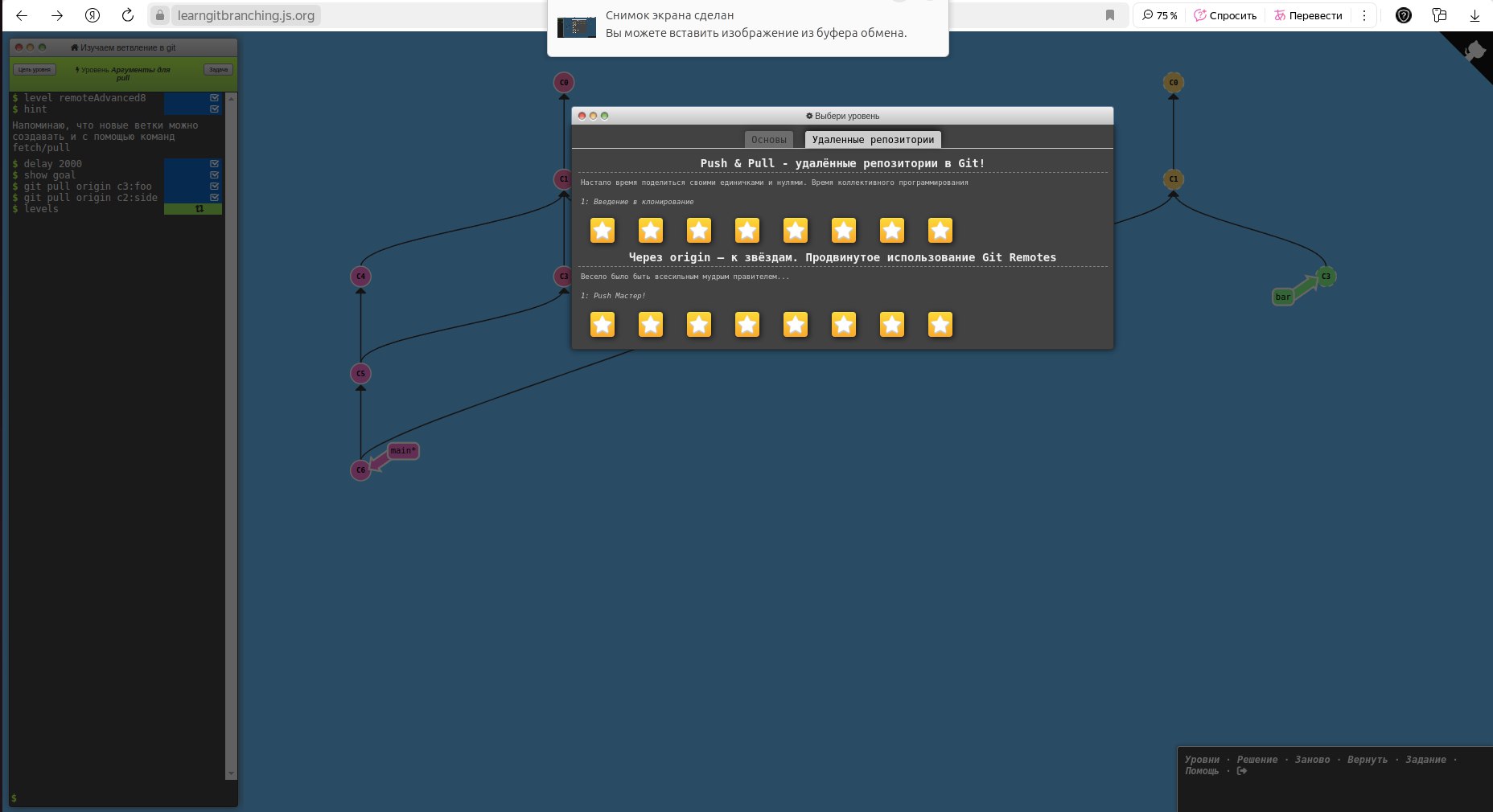
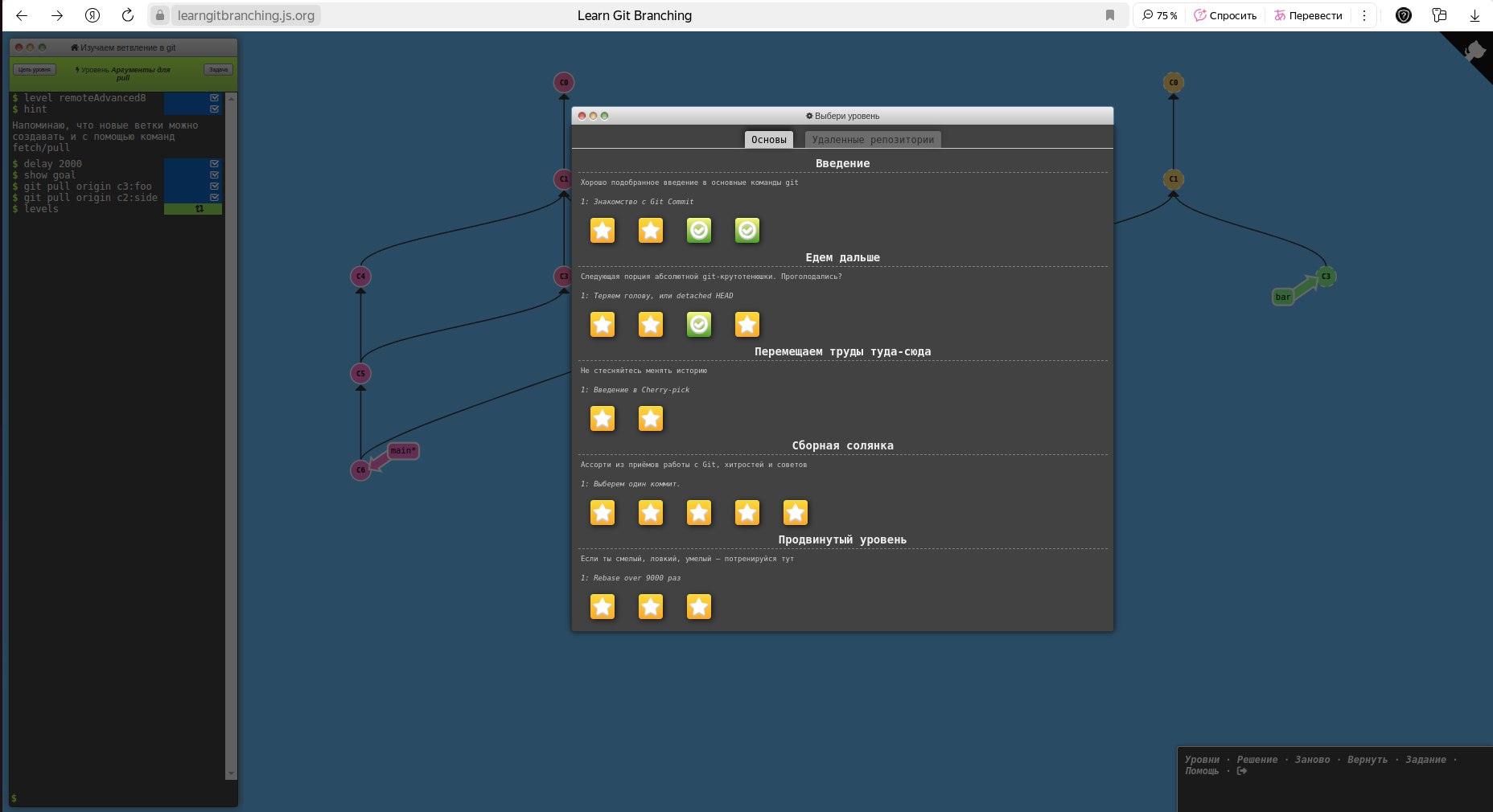
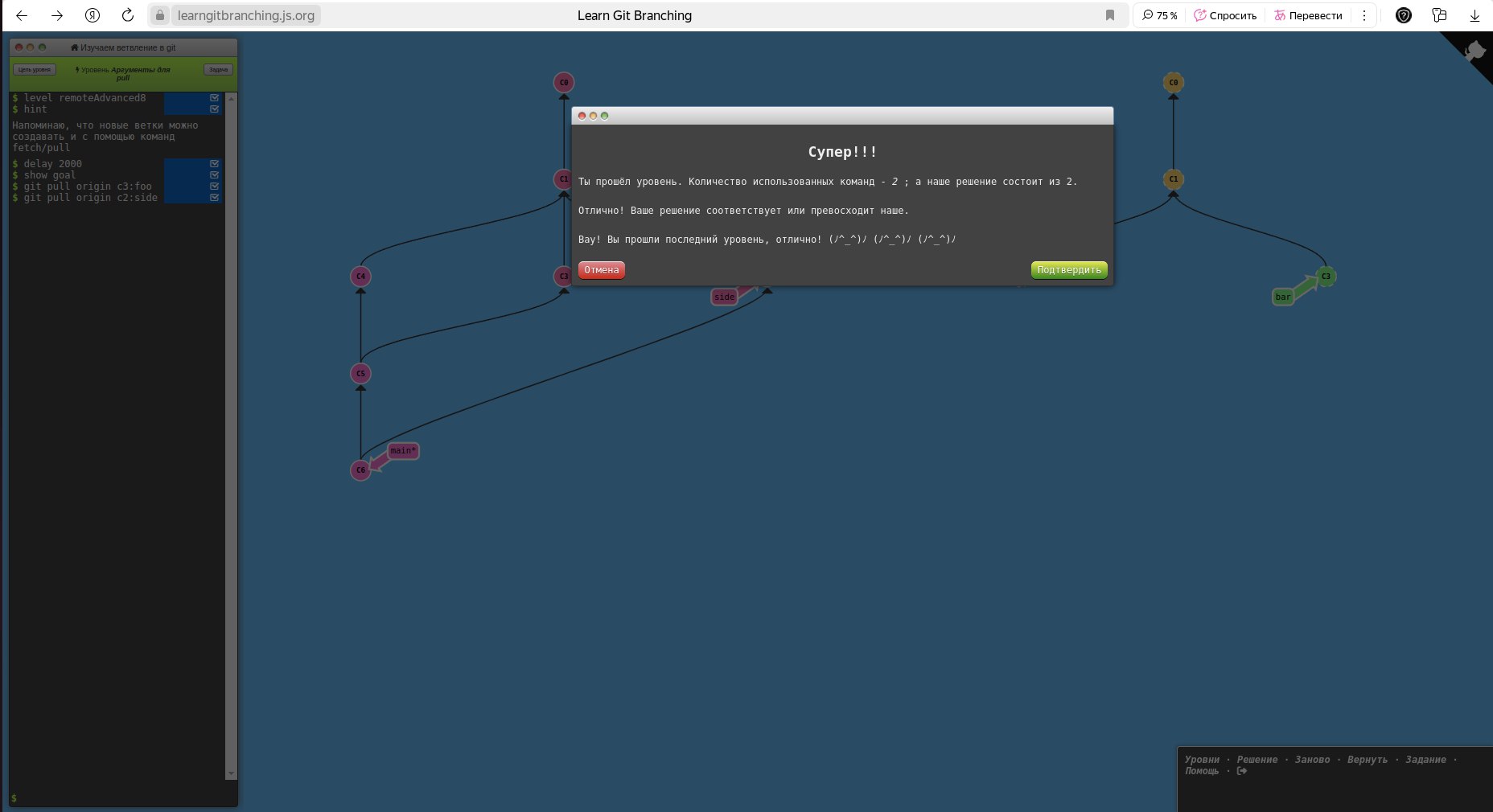
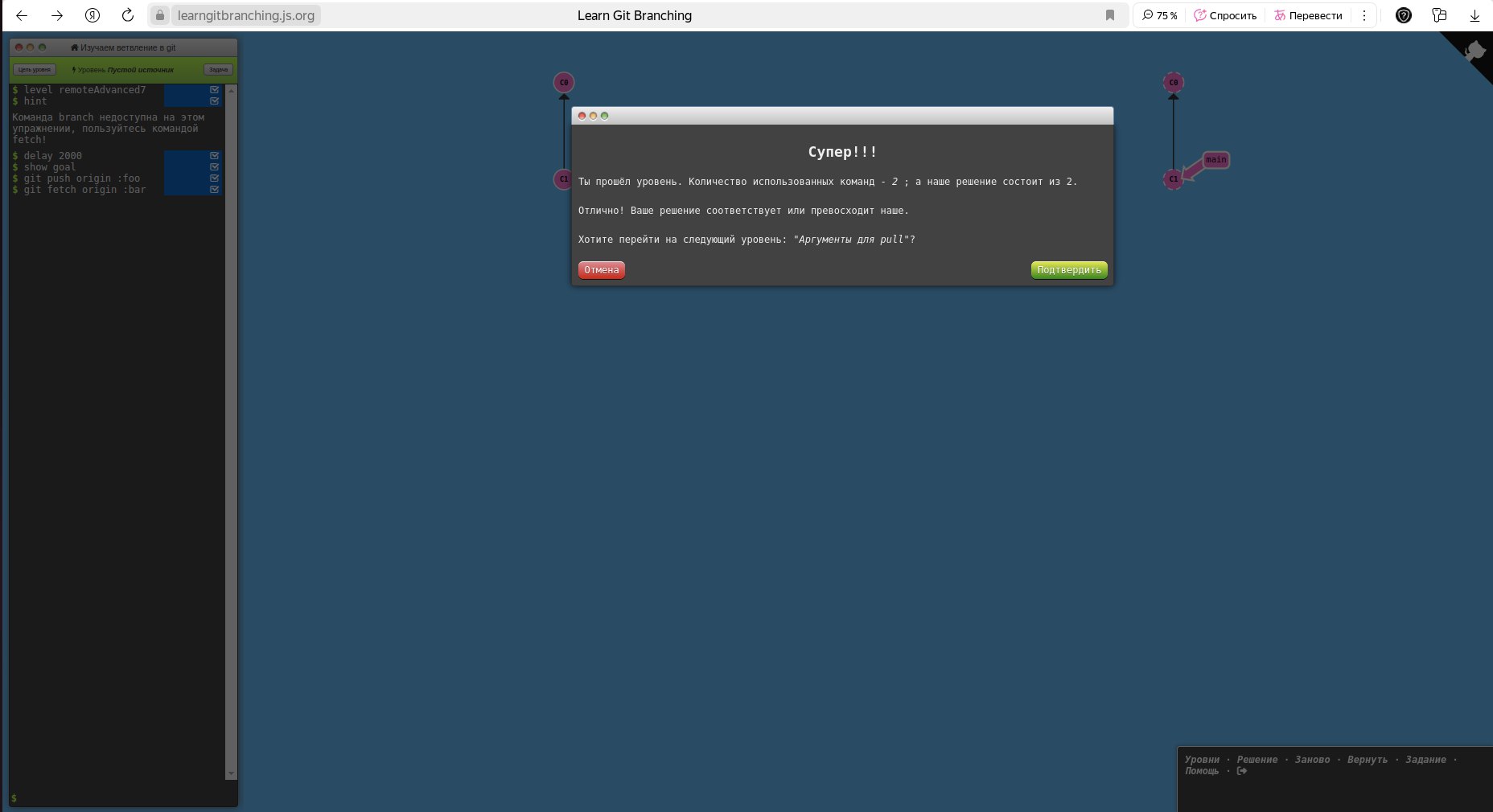
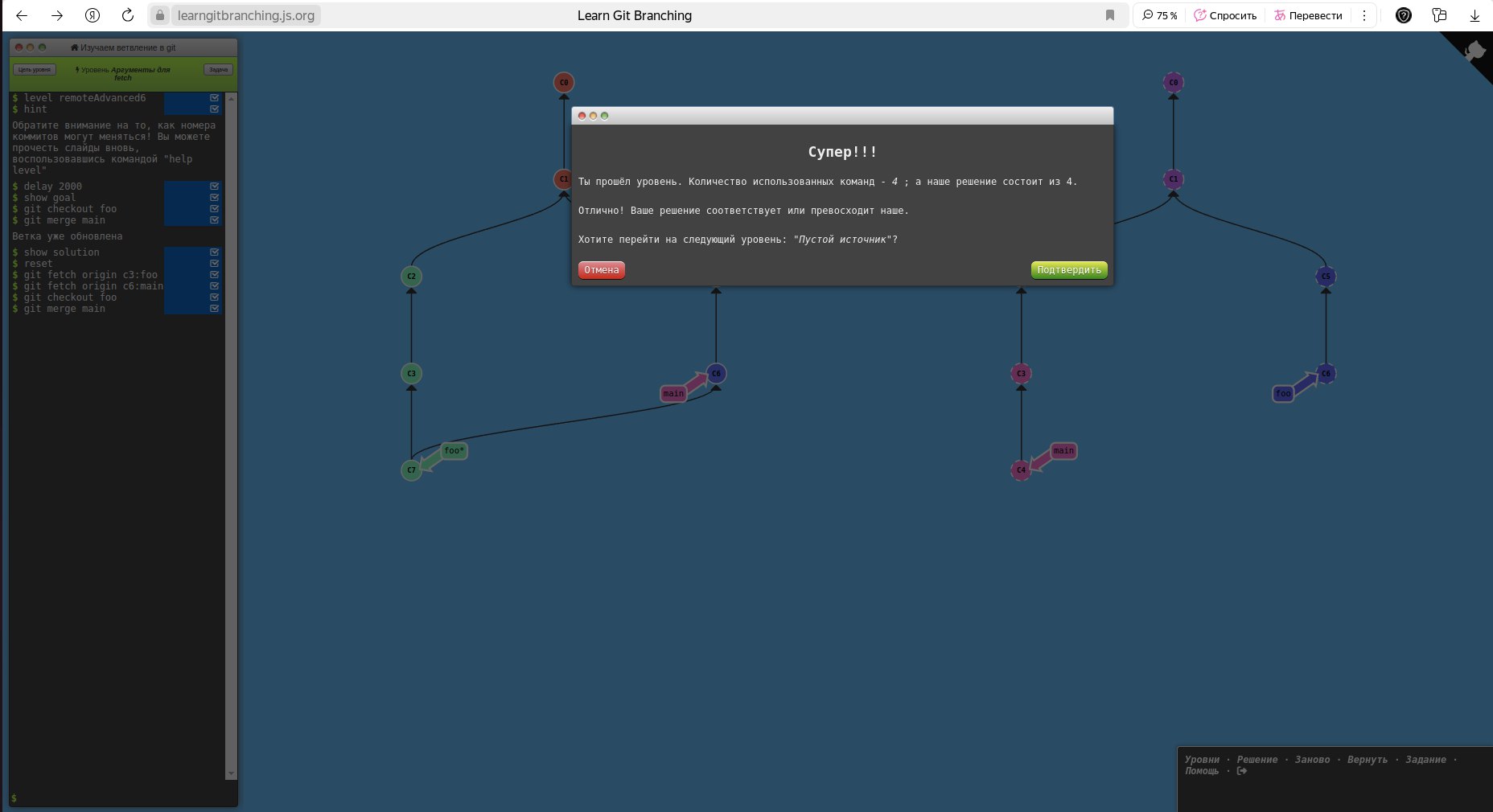
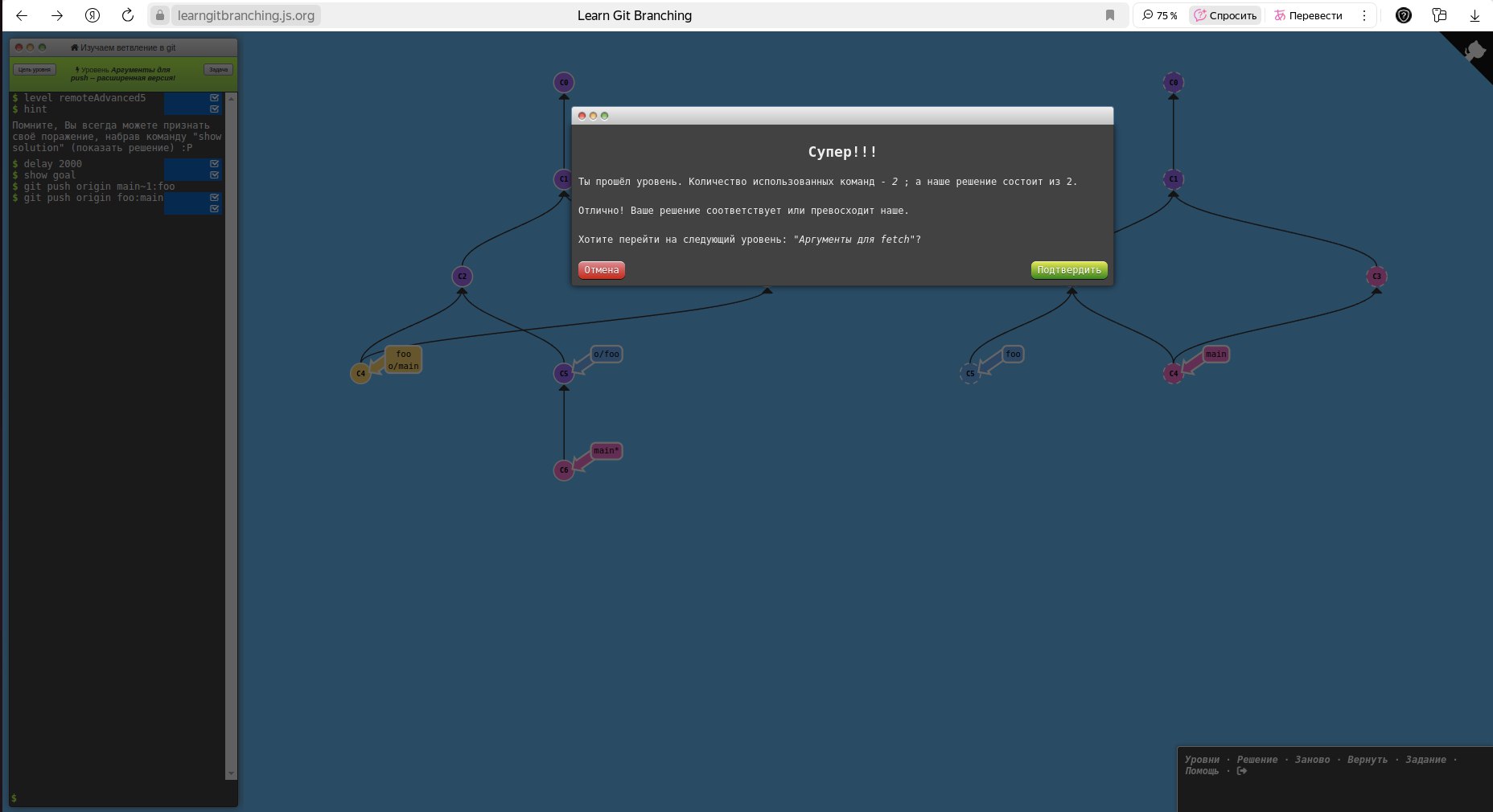
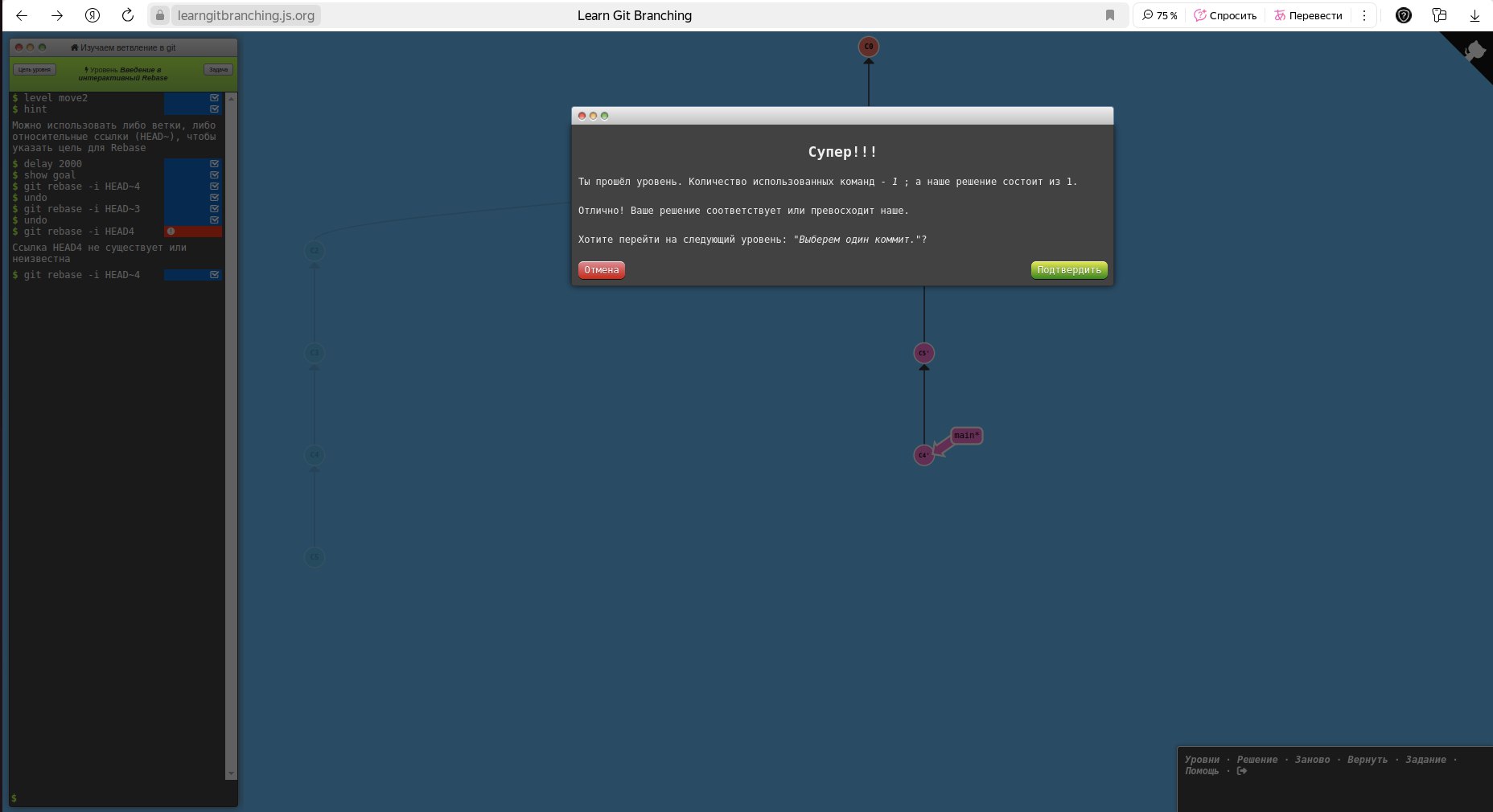
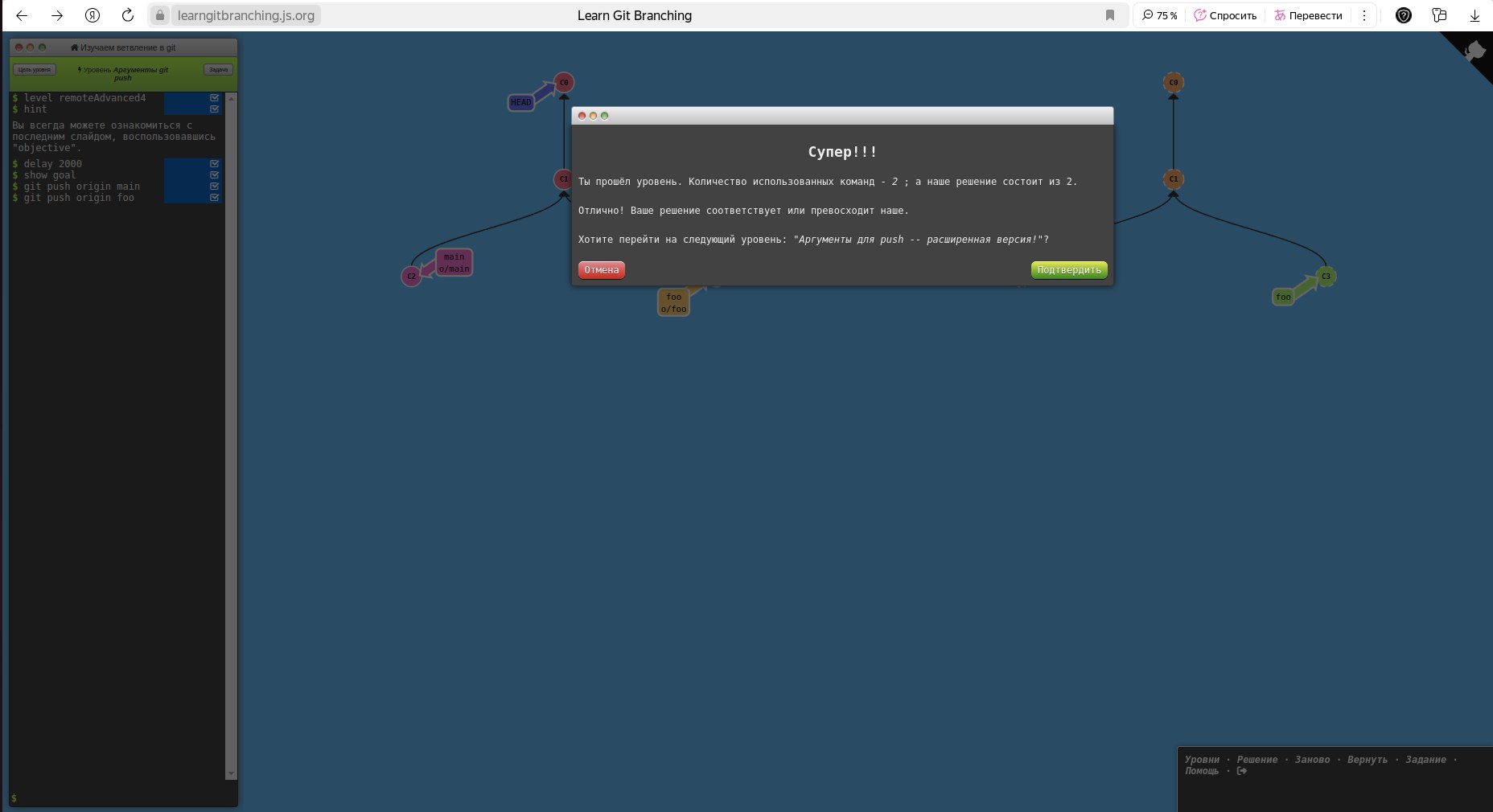
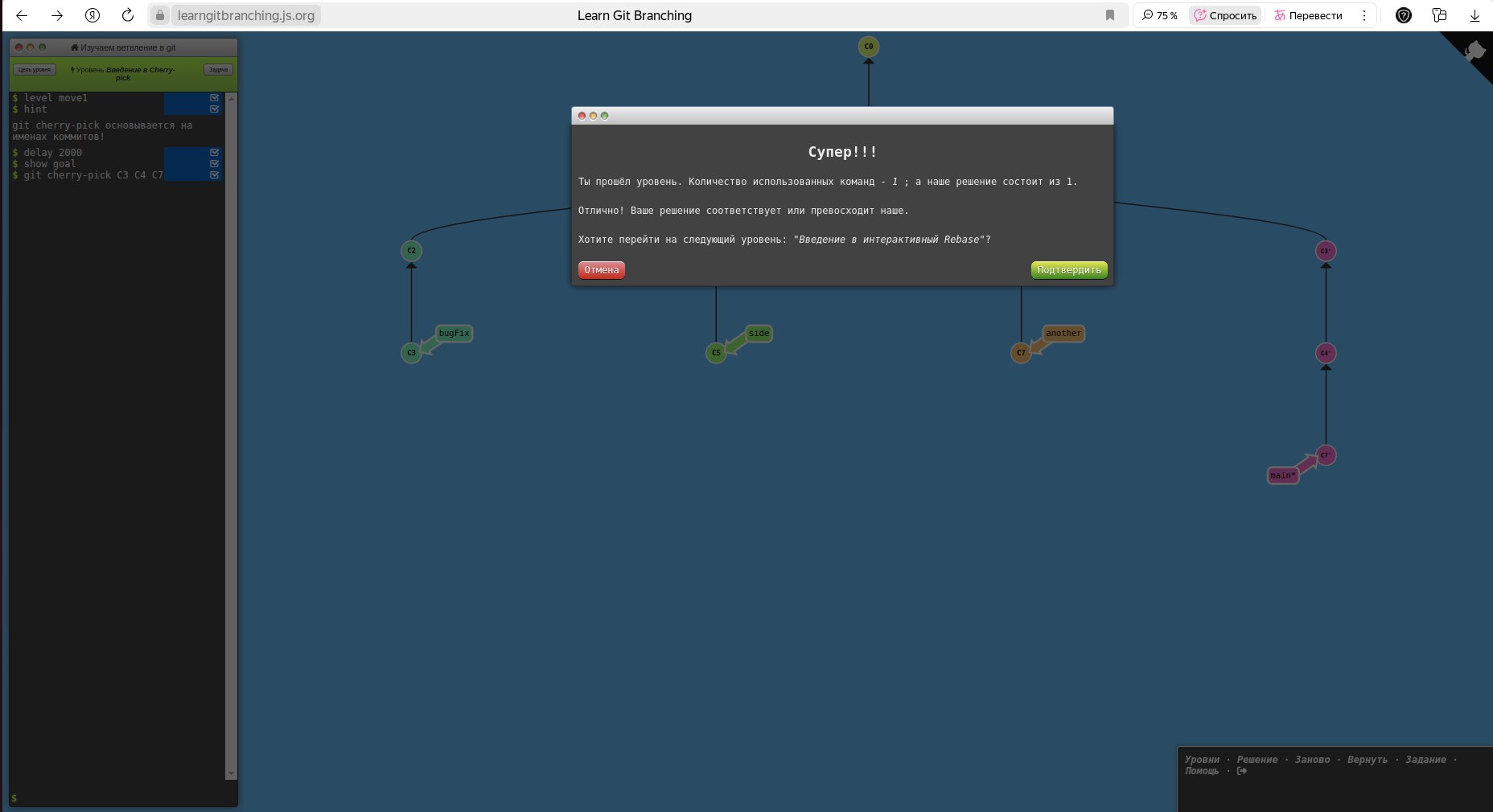
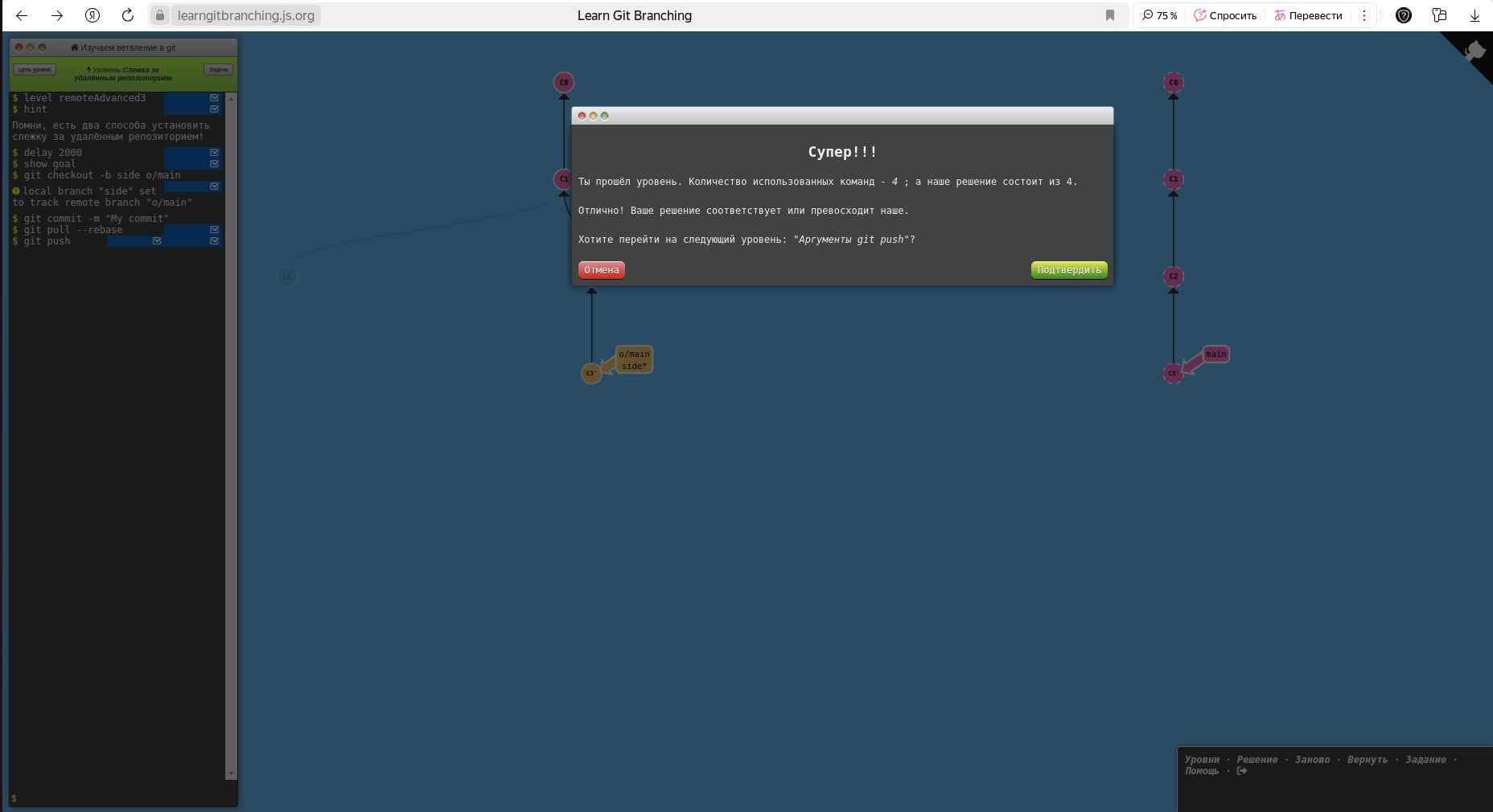
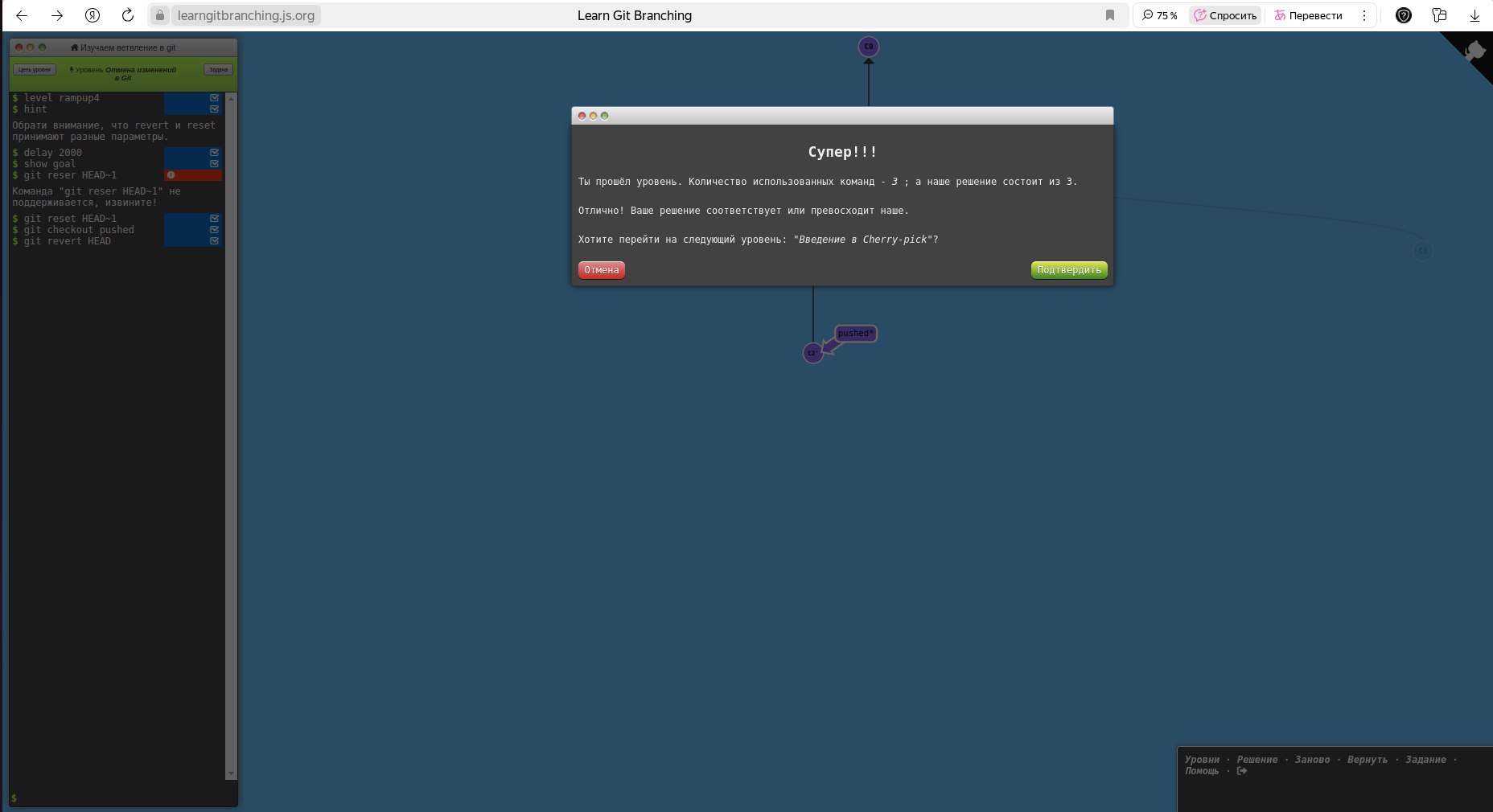
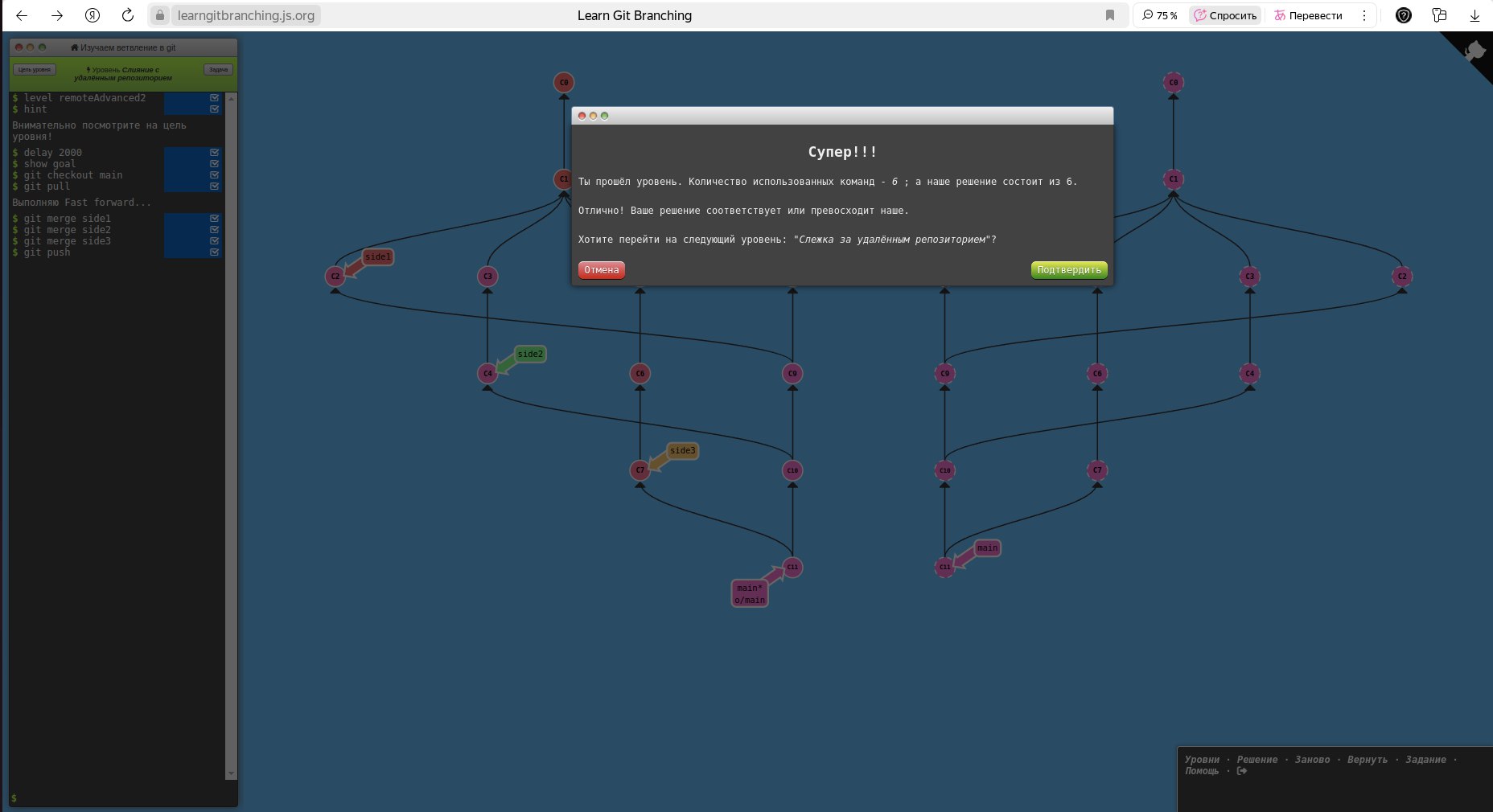
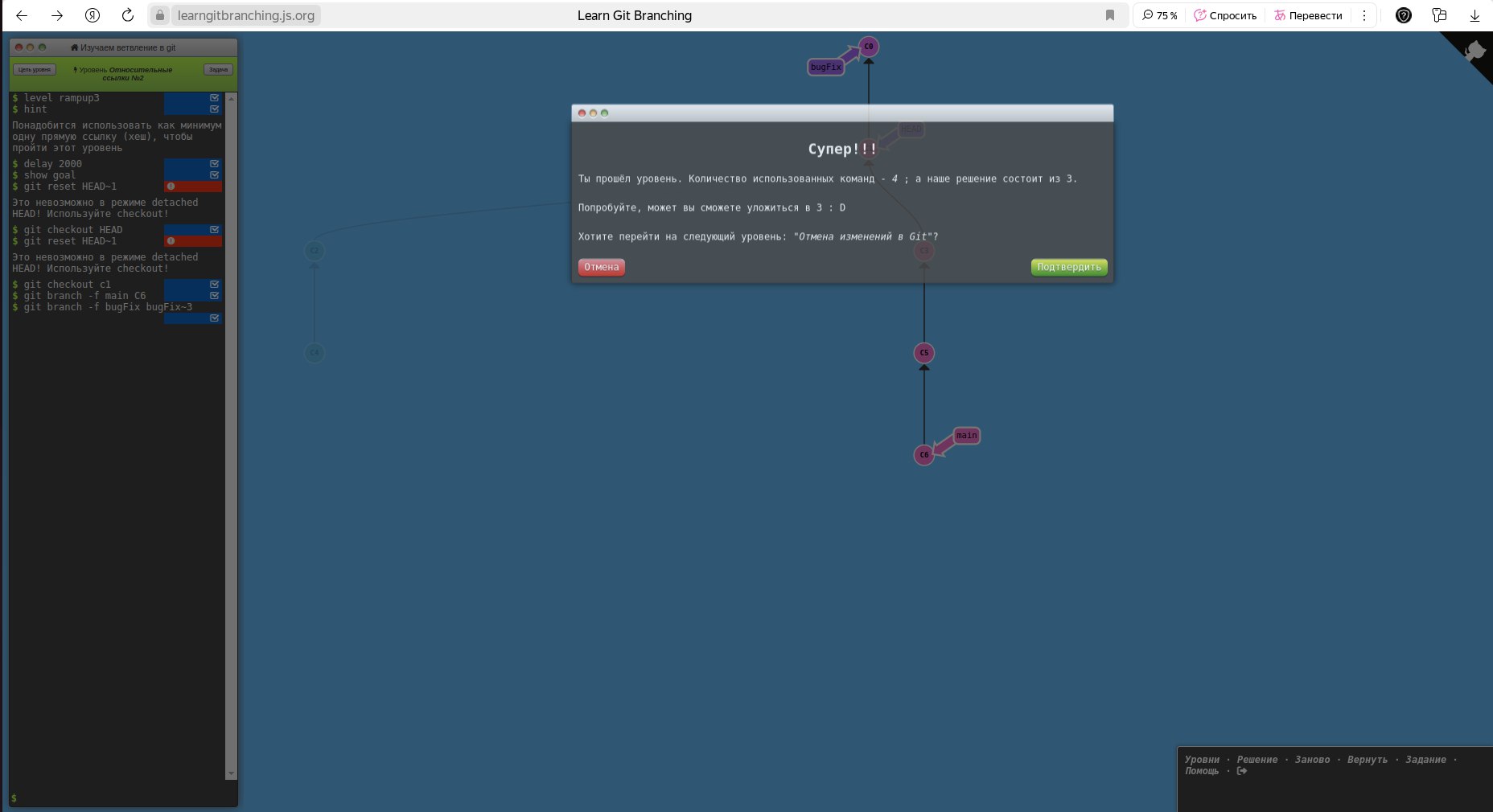
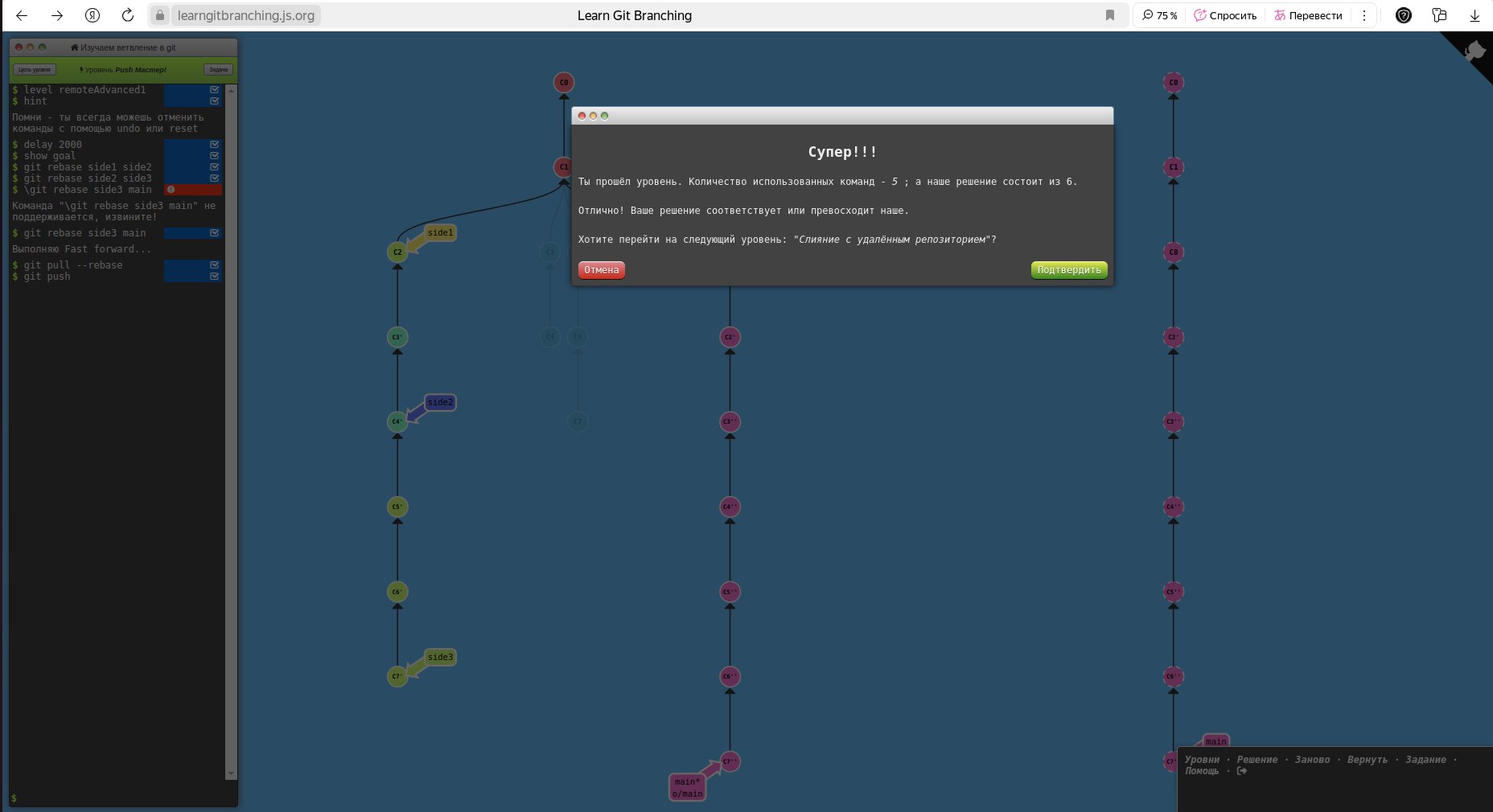
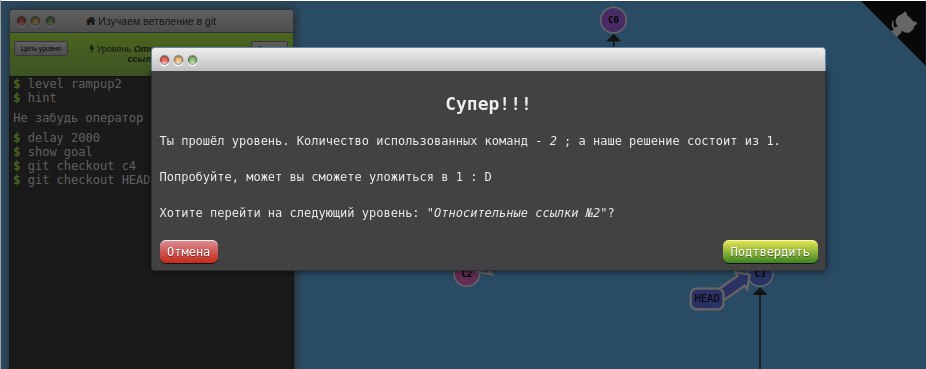
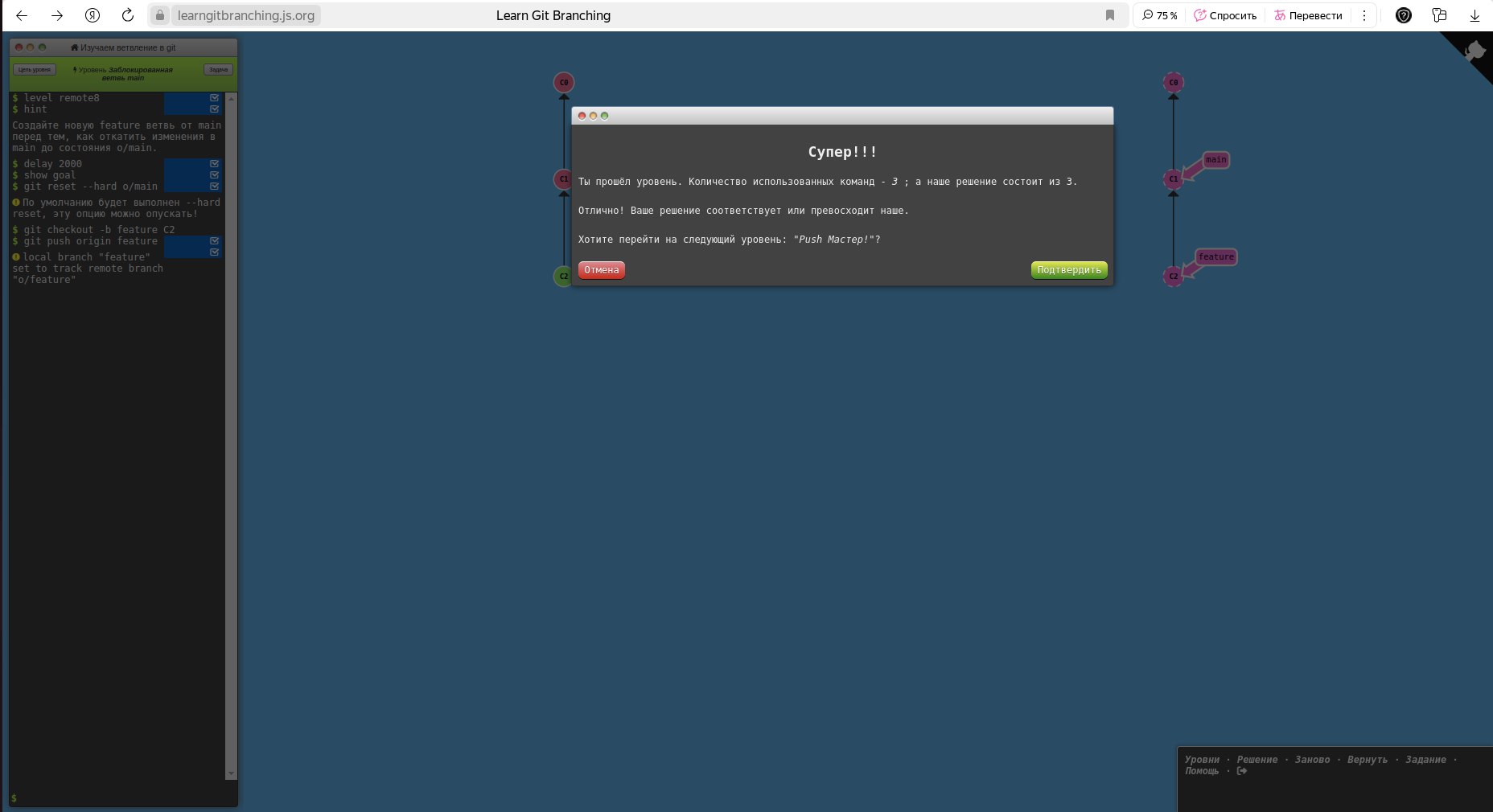
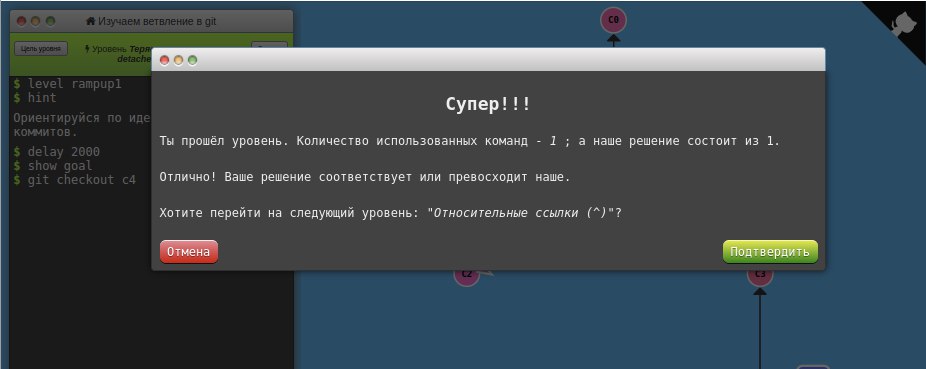
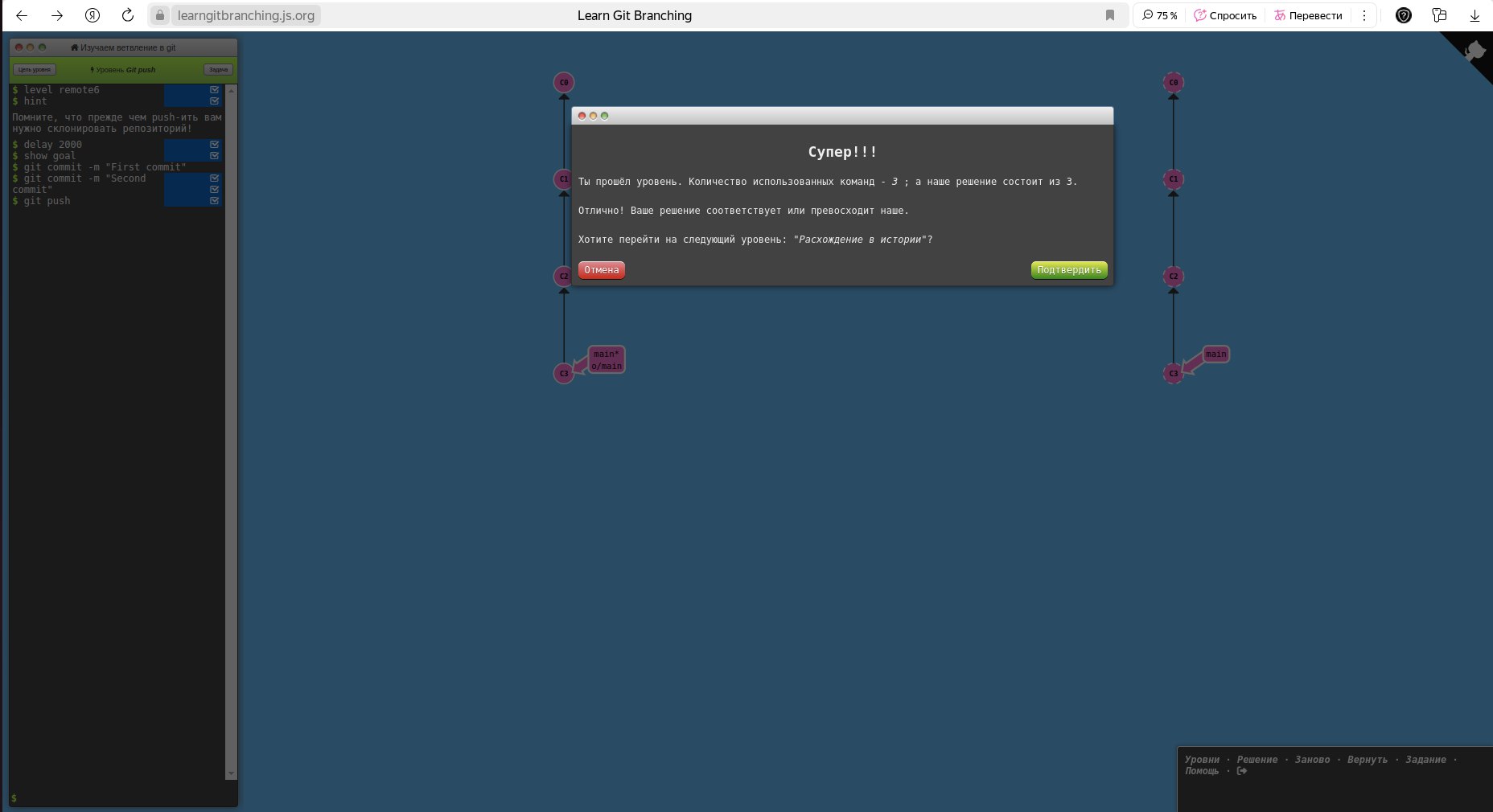
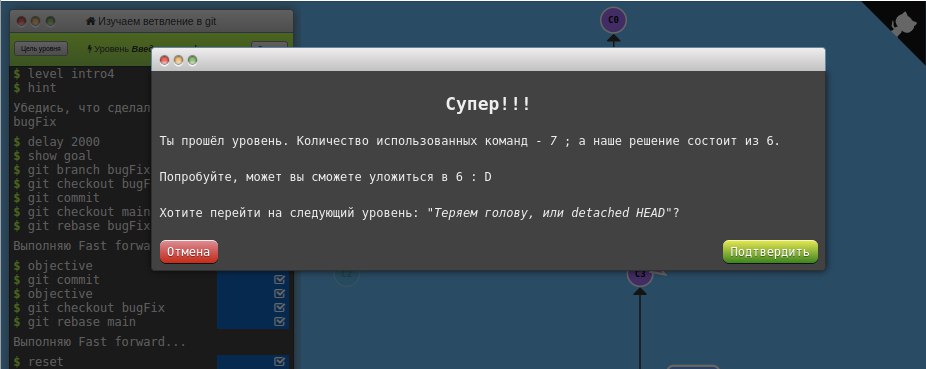
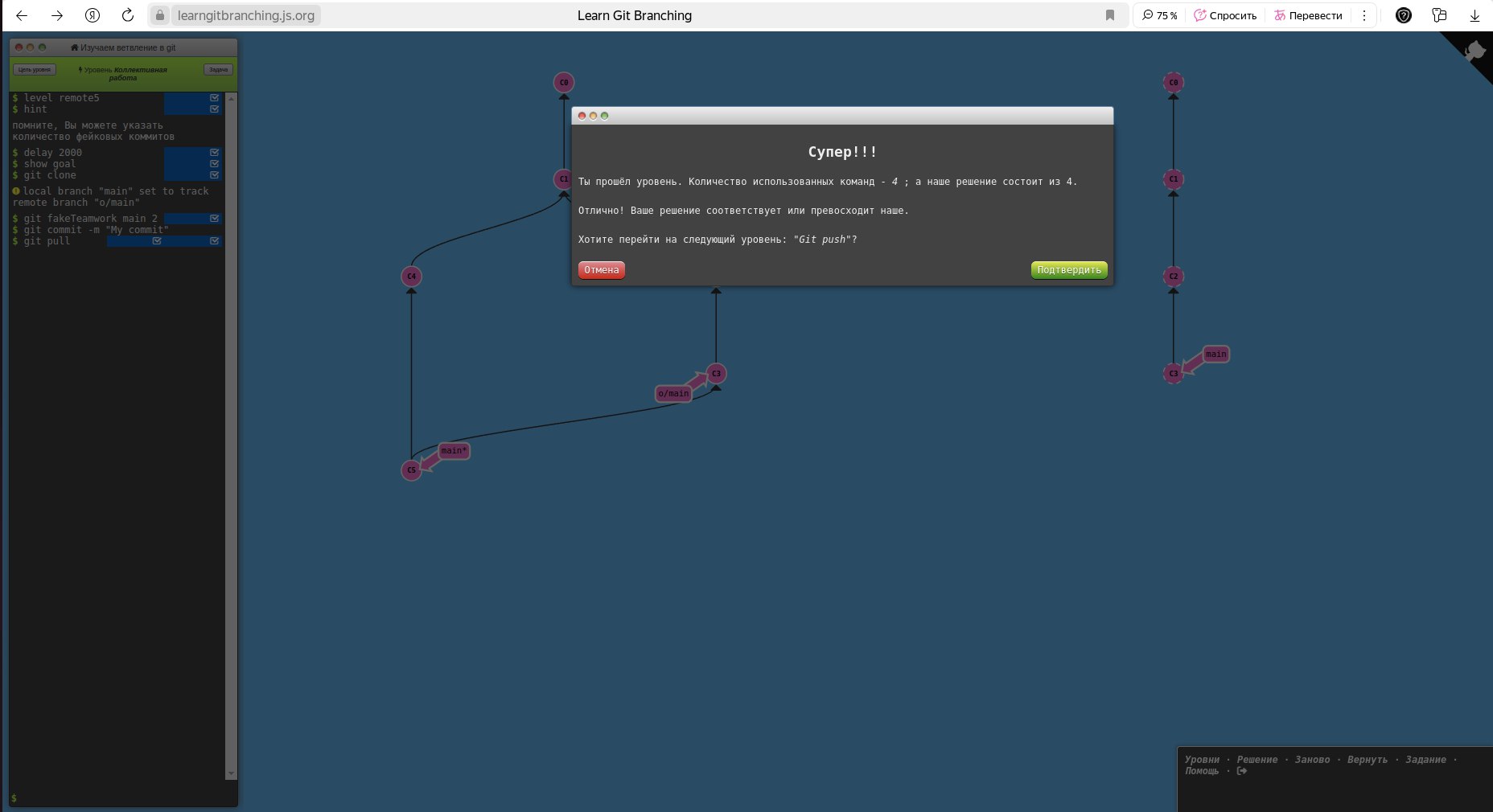
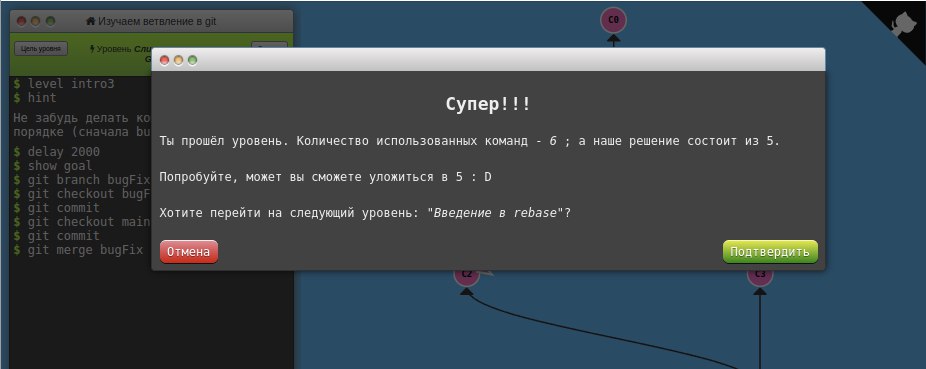
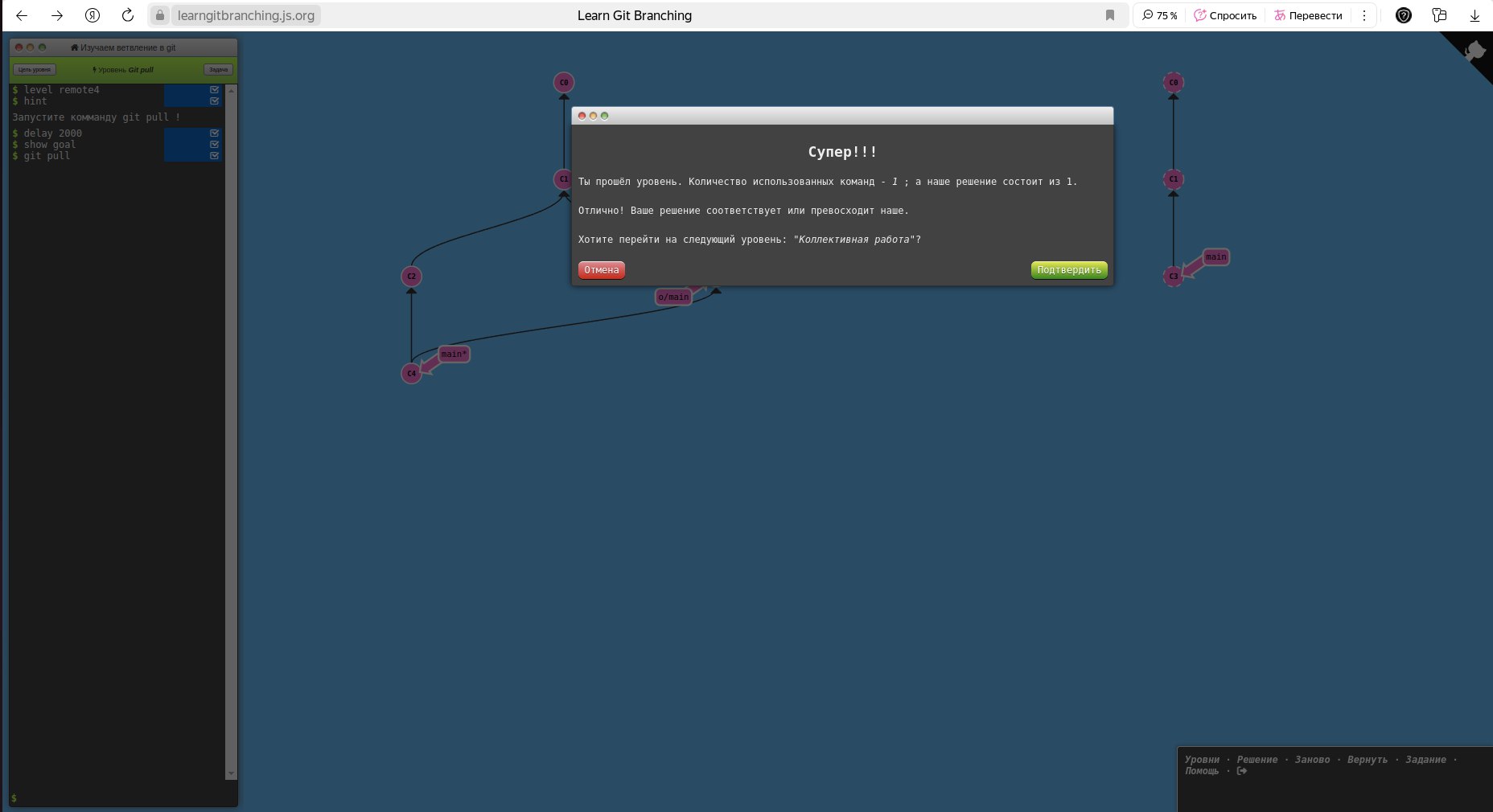
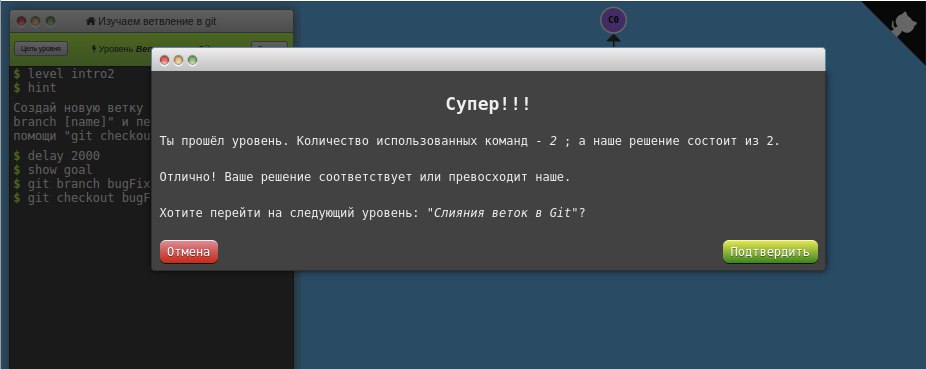
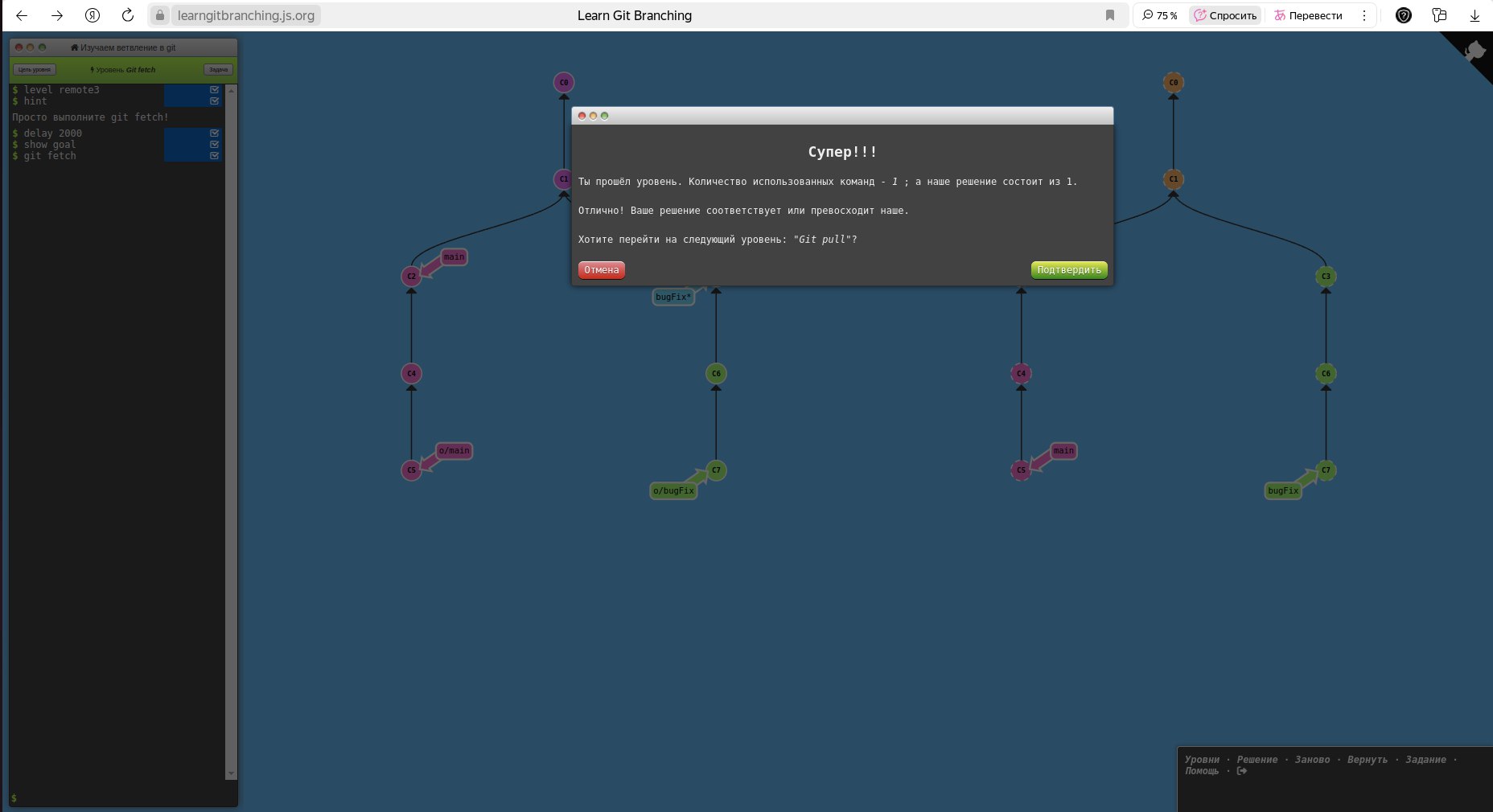
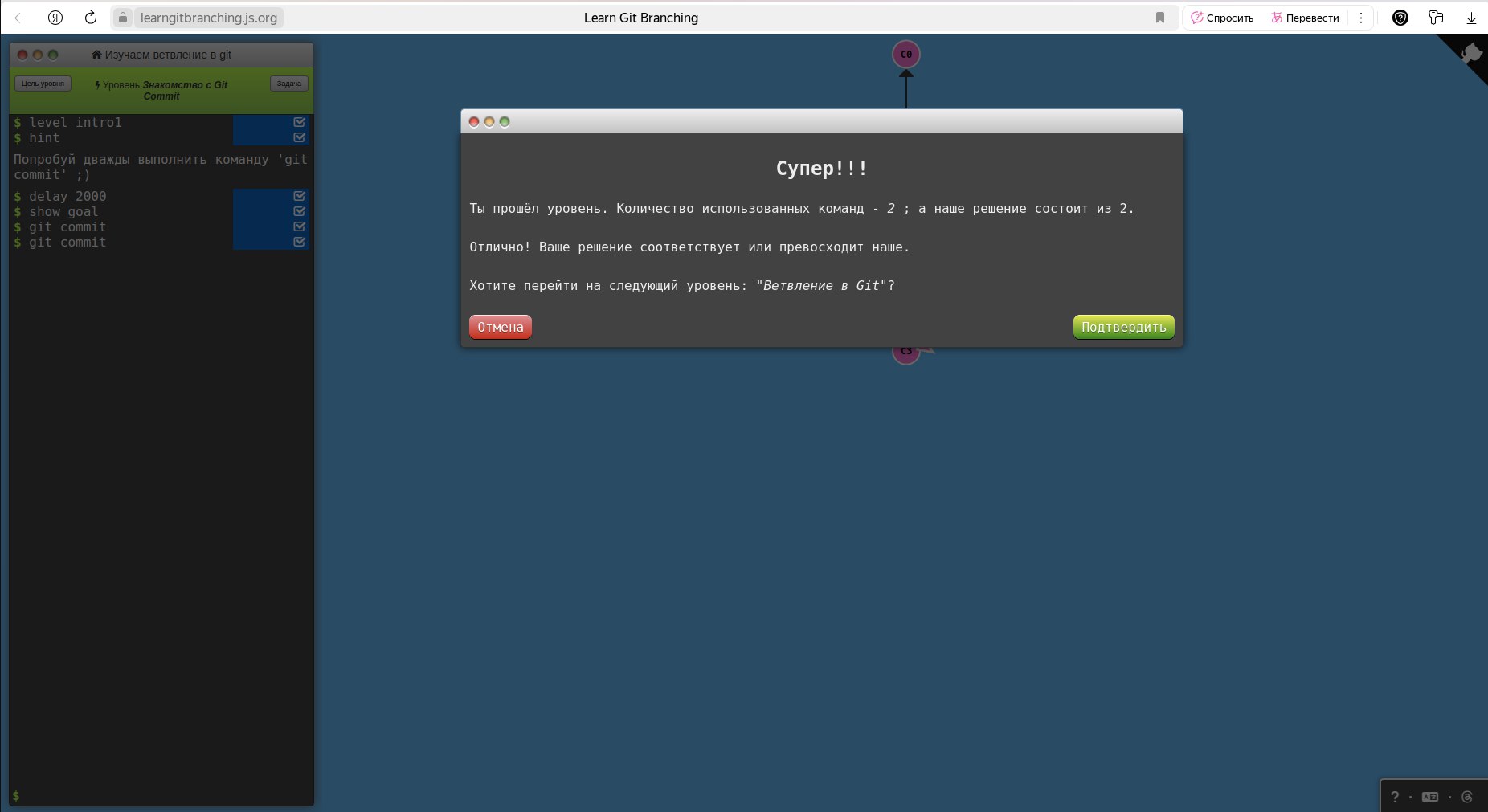
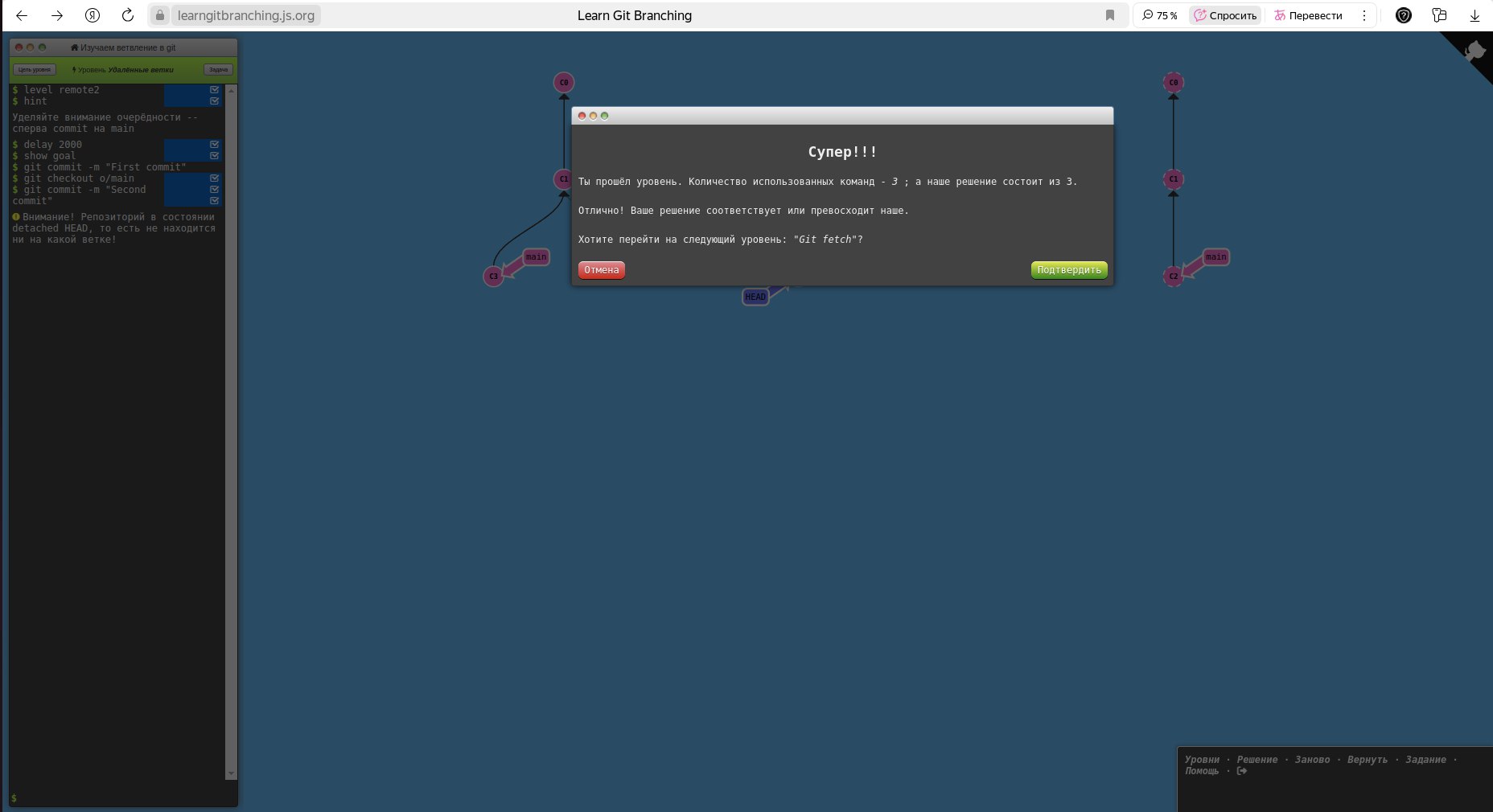
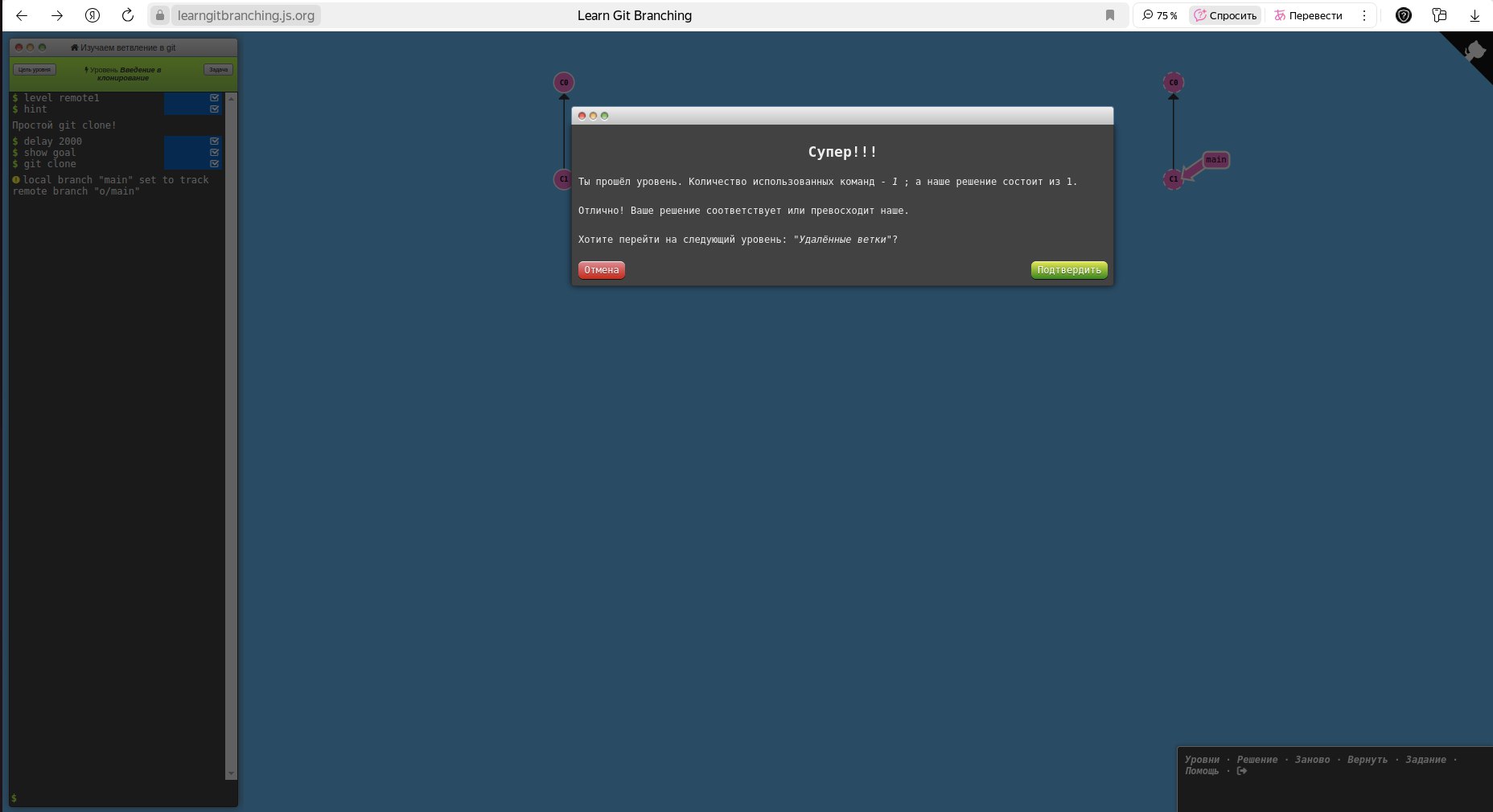
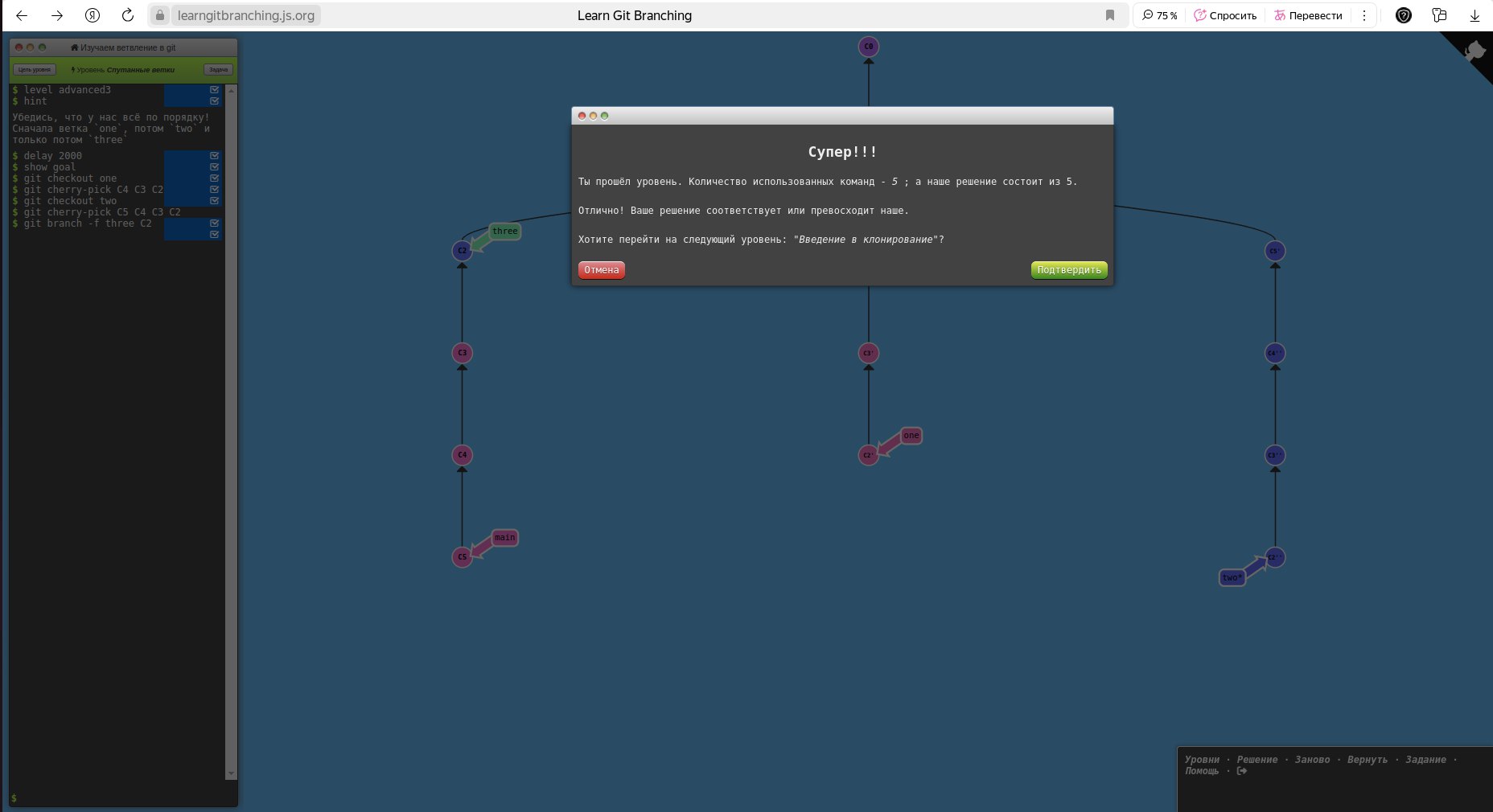
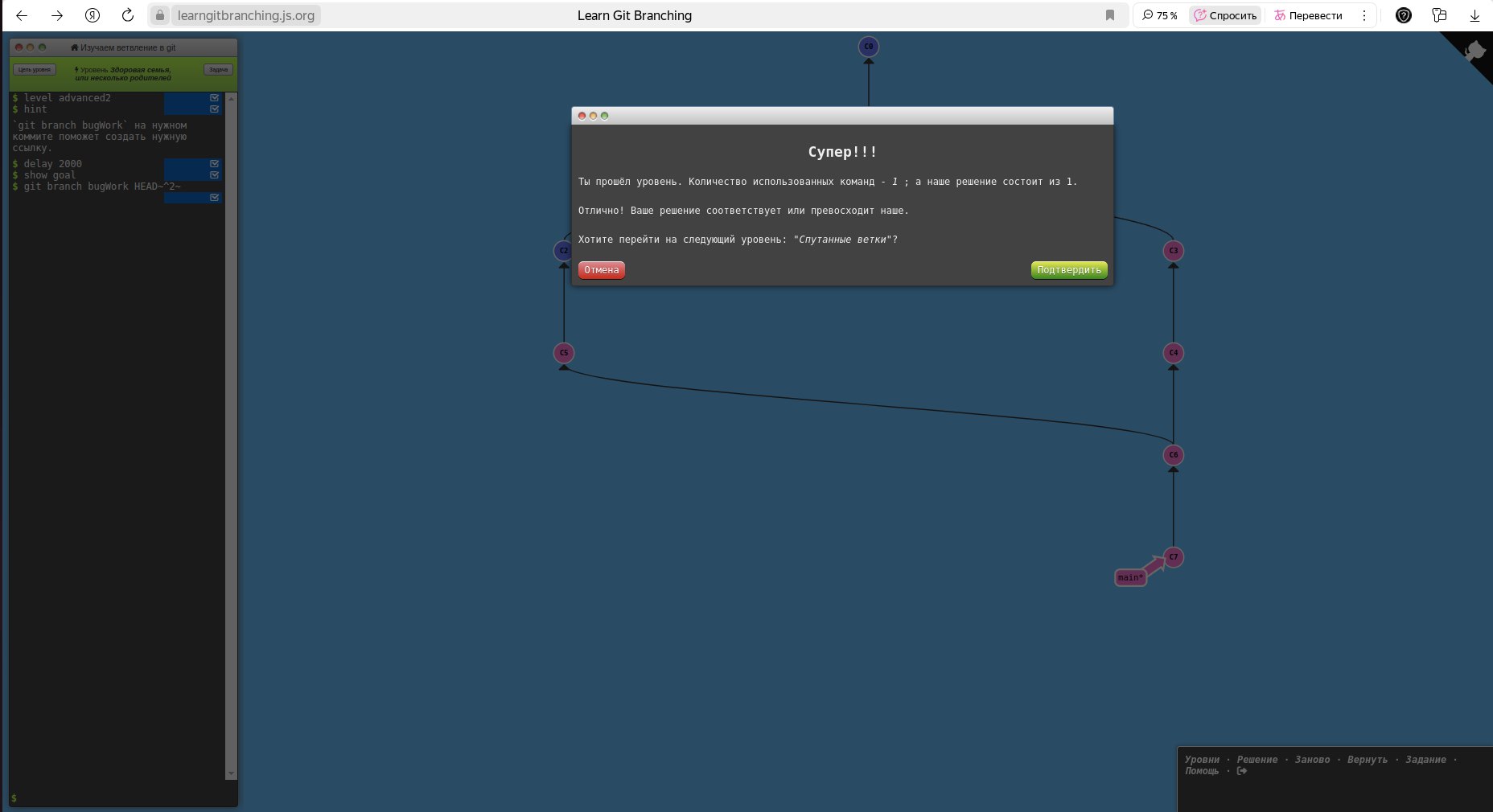
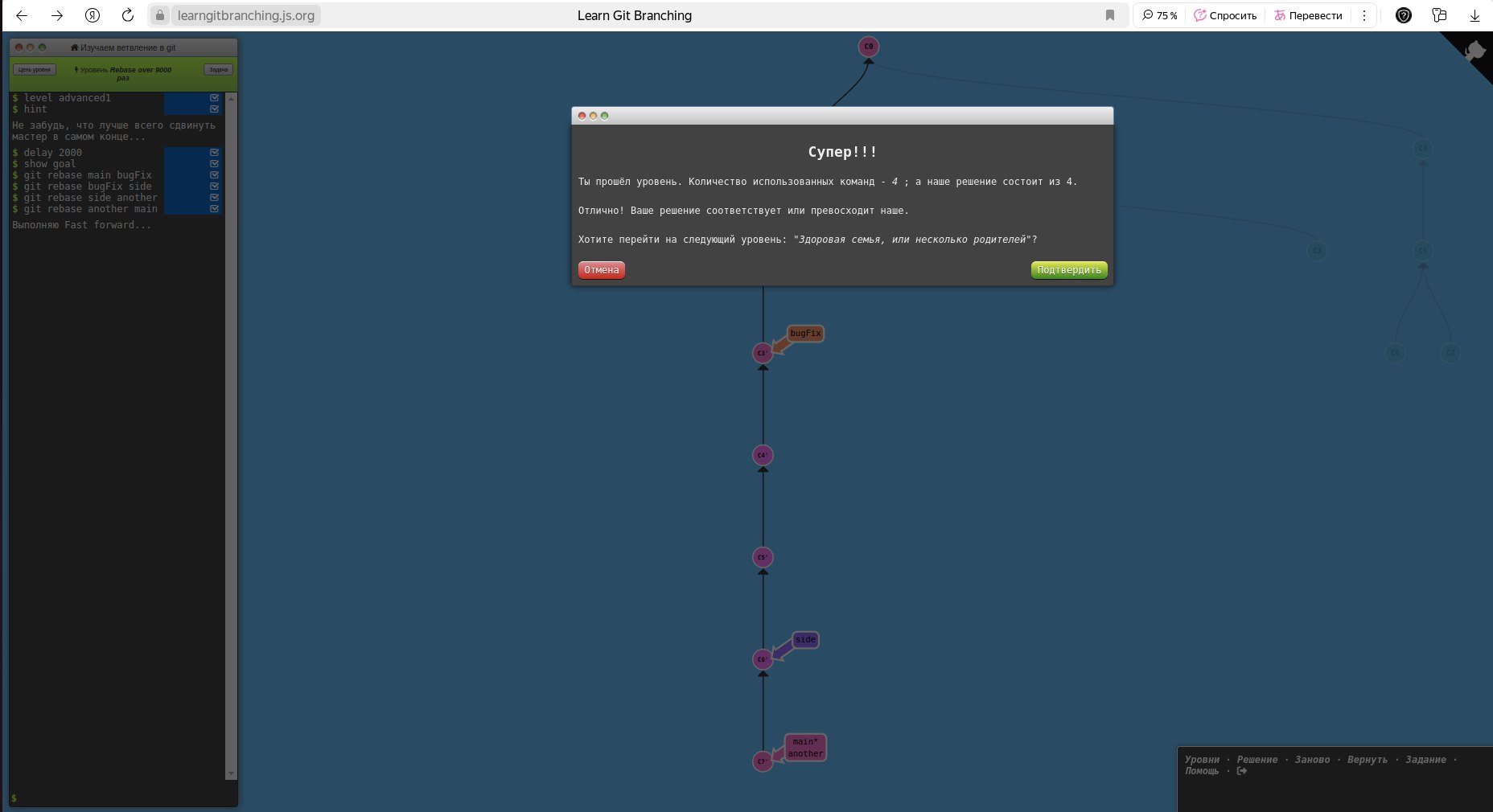
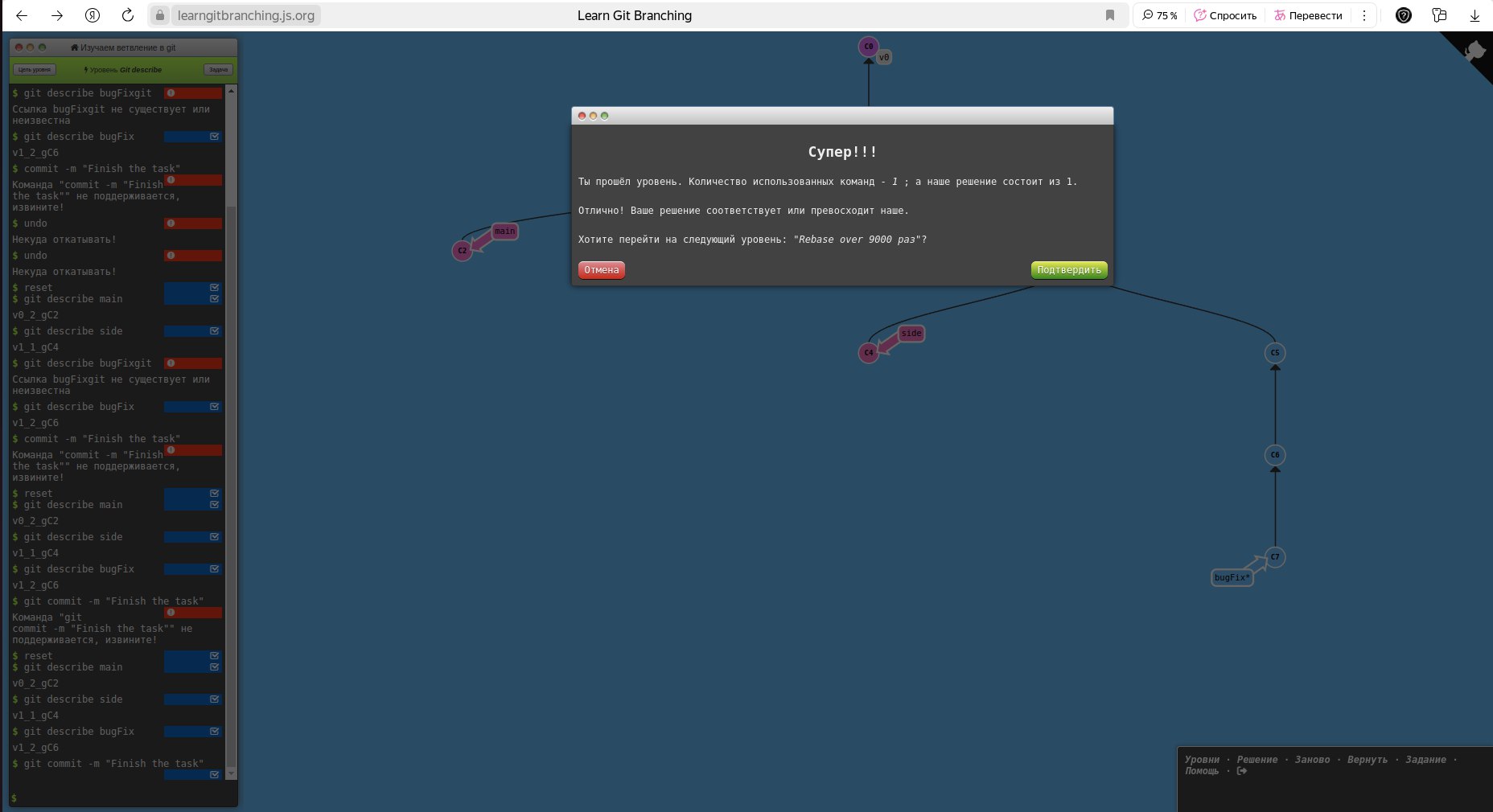
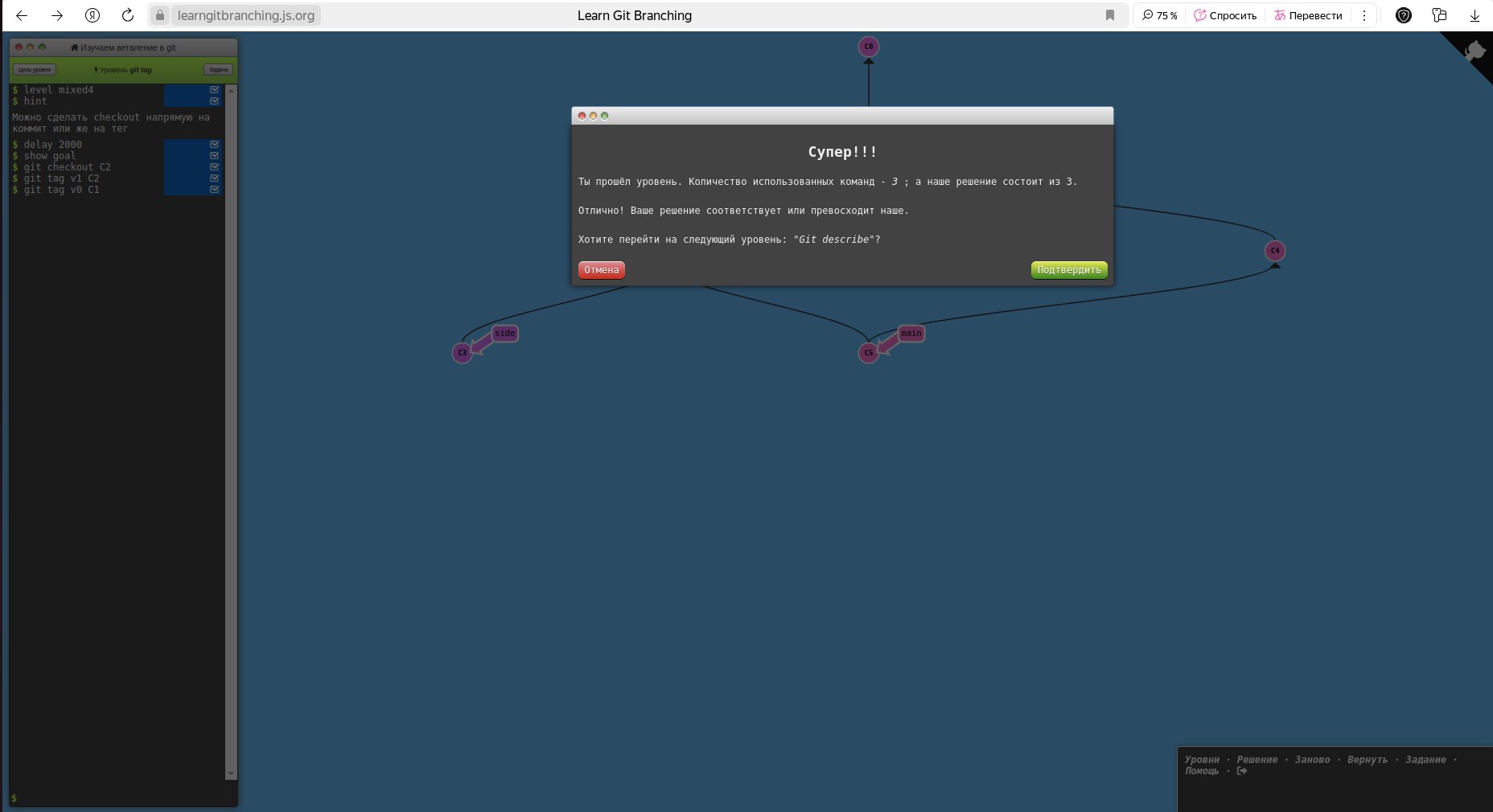
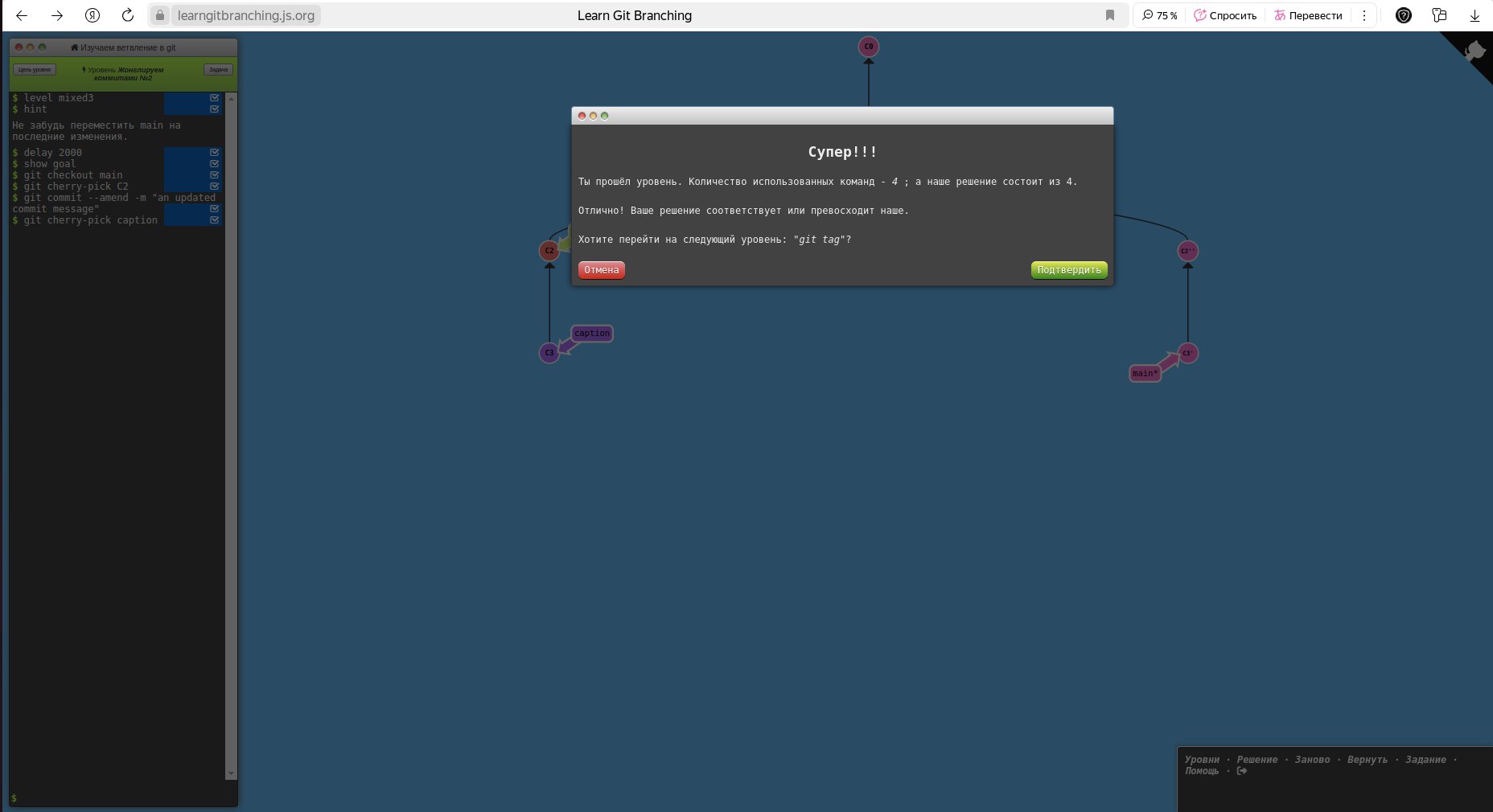
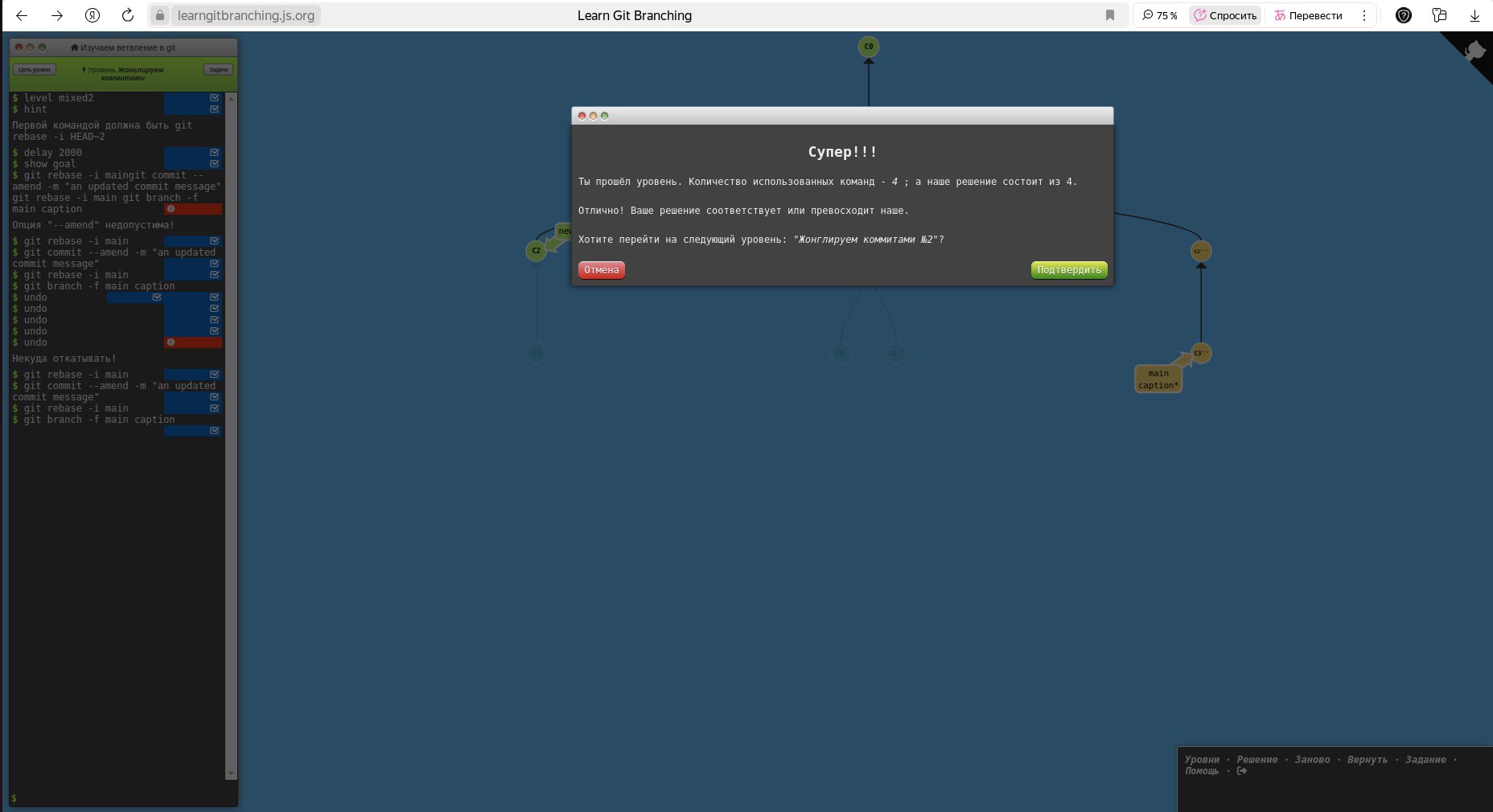
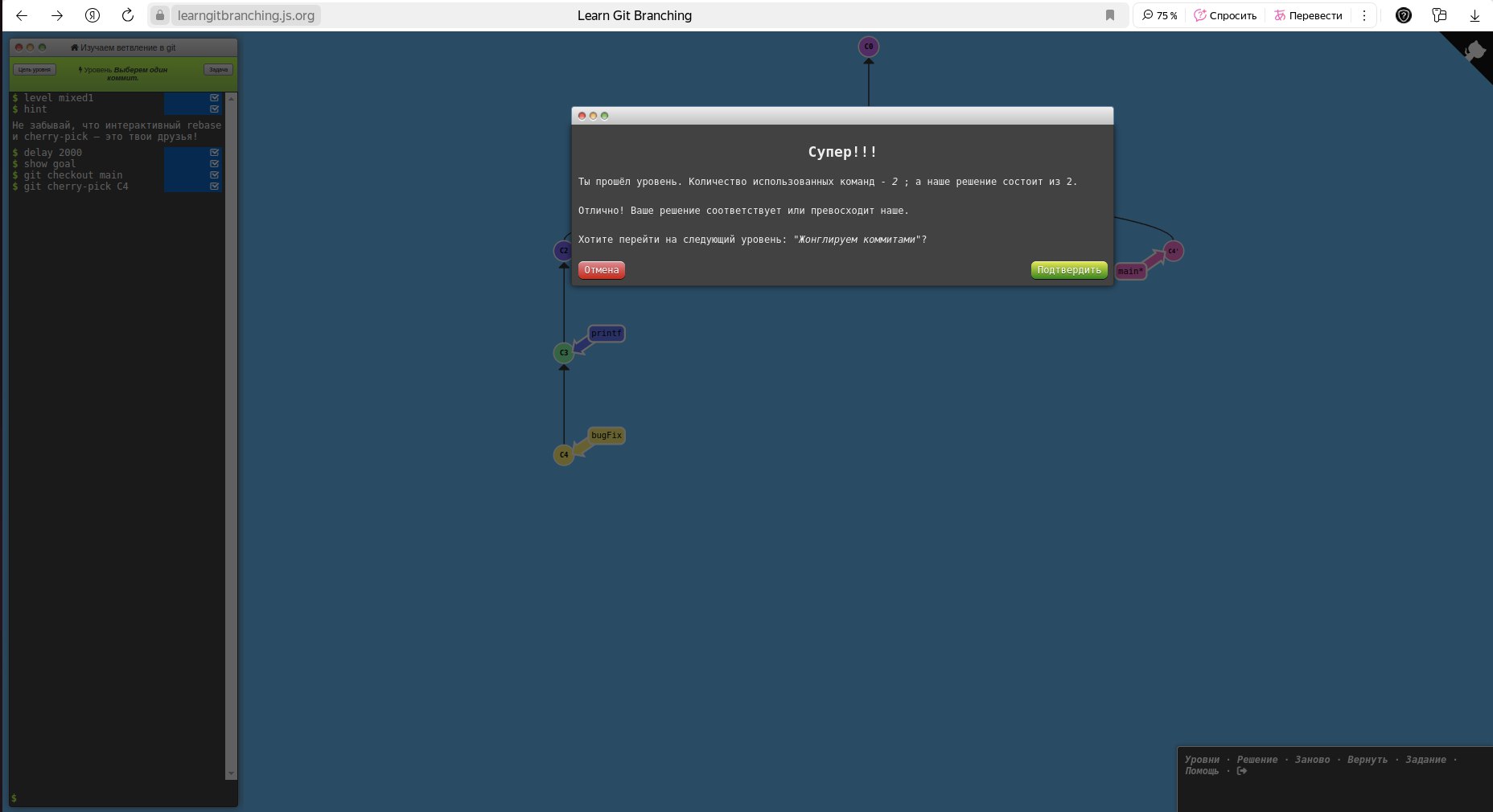
**Результаты:**

* Все 56 упражнений курса выполнены
* Ключевые концепции Git освоены на практике
* Сформировано понимание workflow реальных проектов

**Приложения:**

1. Скриншоты выполненных заданий (с индивидуальными пометками)
2. Лог выполненных команд

**Скрины:**

****