# Руководство по работе с Lab5 Git ## Часть

1: Основные операции 1. \*\*Создайте репозиторий GitHub\*\* - Войдите в GitHub, нажмите ➕ > "новый репозиторий" в правом верхнем углу, введите имя хранилища lab5 и скопируйте HTTPS/SSH-адрес хранилища 2. \*\*Клонируйте репозиторий на локальный компьютер\*\* ``Bash start git https://github.com/Jyhqq/lab5 start lab5 `` 3. \*\*Добавьте файлы\*\* - Создайте example.txt и добавьте содержимое: - Зафиксируйте изменения ``Баш мерзавец добавить example.txt мерзавец сделал -м "Add example file" git start commit основных ``

## Часть 2: Управление ветками 4. \*\*Создание ветки функций\*\* `` bash git- git checkout- git checkout "`

5. \*\*\*\*\* - Редактировать example.txt и добавить новый контент - Зафиксировать изменения: `добавляем git в bash . git commit -m "Feature branch modification" отправляем исходную функцию git-ветвь "`

6. \*\*Объедините ветку\*\* ``из исходного источника git bash git checkout-ветвь git push origin main ``.

## Практика разработки ветки Часть 3: 7. \*\*Создание ветки функции входа\*\* ``функция-вход в Git Баш кассе-Б `` 8. \*\*Добавление новой главы\*\* - Изменение example.txt: - Зафиксировать изменения: ``Баш мерзавец добавить example.txt мерзавец сделал -м "Add login chapter" git толчок часть-логин происхождения ``

## Часть 4: Разрешение конфликтов 9. \*\*Имитация конфликтов\*\* - Изменение веток ``Создание новой ветки в bash git-build - инициирование конфликта "'Создание новой ветки в bash git - git pull origin main "`

10. \*\*Разрешите конфликты\*\* - Маркеры конфликтов появляются при слиянии: - Зафиксируйте окончательную версию: ``Баш мерзавец добавить example.txt мерзавец сделал -м "Resolve conflicts in Chapter 2" git толчок происхождение основных ``

## Часть 5: Автоматическая проверка формата 11. \*\*Создание хука pre-commit\*\* - Создание .txt в корневом каталоге проекта Git/крючки/пре-коммит - Баш: Добавление скрипта проверки ``#! /Бен/Баш # Проверка всех .txt файлов для файла в $(мерзавец дифф ----кэширования имен-только | команда grep "\.файл txt$") # все Пример проверки: Необходимо включить заголовок раздела если ! grep -q "## Раздел" "$file"; затем echo "Ошибка: $file отсутствует заголовок раздела" выход 1 завершен `` - Добавление разрешений на выполнение: ``bash chmod +x .git/перехватчики/предварительная фиксация "`

## Git Flow workflow practice 12. \*\*\*\*\* "'bash sudo apt`получить установку git-flow # Установить зависимости инициализация git flow # Инициализировать workflow ``

13. \*\*Разработка функций\*\* ``Купить ветку функции управления задачами # Создать ветку функции управления задачами `` - Реализовать функцию (пример кода): `` # task\_manager.py с Def create\_task(название, описание): """Новая логика задачи""" печати(Ф"[System] Создать новую задачу: {название}") `` - Отправить изменения: ``Купить ветку task\_manager.py мерзавец сделал -м "Реализовать базовую функцию создания задач" `` 13. \*\*Завершить разработку функций\*\* ``Новая ветка в 'новом' каталоге # Автоматически объединить в 'новую' ветку ``

14. \*\*\*\*\*\*\* "`bash git checkout разработка git flow запуск выпуска версии 1.0.0 # Создать ветку релиза echo "версия 1.0.0" > version.txt # Обновить номер версии добавить git version.txt git commit -m "Version upgrade v1.0.0" завершение выпуска git flow v1.0.0 # Слить в главная/разработка "`

15. \*\*\*\*\* ``git commit git branch git bash git-1.0.1 # Создать ветку исправления # Исправить код проблемы... git add file\_with\_error.py git commit -m "Исправить уязвимость проверки прав" git flow branch git-1.0.1 # Слить исправление с основной веткой ``

15. \*\*Синхронизировать удаленный репозиторий\*\* "'bash git push origin # Отправить ветку процесса git push origin главная # Отправить основную ветку "`