МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ
РОССИЙСКОЙ ФЕДЕРАЦИИ
Федеральное государственное автономное
образовательное учреждение высшего образования
«СЕВЕРО-КАВКАЗСКИЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ»

Кафедра инфокоммуникаций

Отчет по лабораторной работе №1

Исследование основных возможностей Git и GitHub

по дисциплине «Основы кроссплатформенного программирования»

Выполнил студент группы ИВТ-б-о-20-1
Пушкин Н.С. « »_____20__г.
Подпись студента_____
Работа защищена« »_____20__г.

Проверил Воронкин Р.А. _____
(подпись)

Ставрополь 2020

Цель работы: исследовать базовые возможности системы контроля версий Git и веб-сервиса для хостинга IT-проектов GitHub.

Ход работы:

Ссылка на репозиторий:

https://github.com/NiKiN126

1. Первым шагом создал новый репозиторий на github с названием lab 2. После этого клонировал его на локальный сервер и, перейдя в папку репозитория, начал работу.

2. Создал три файла: 1.txt, 2.txt, 3.txt. Сделал их коммиты с комментариями.

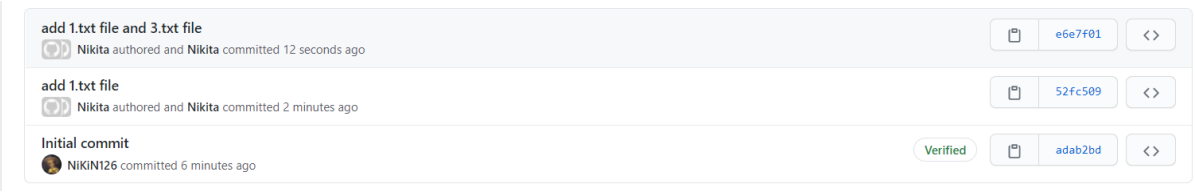

Рисунок 1. Начало работы

3. Создал новую ветку при помощи команды git branch (название ветки) и, перейдя на неё, создал новый файл in_branch.txt, после чего сделал коммит. g

4. Затем создал ещё одну ветку и перешёл на неё, после чего изменил файл 1.txt и закоммитил изменения.

5. Перешёл на ветку main и слил сначала первую ветку, а после и вторую. Удалил слитые ветки.

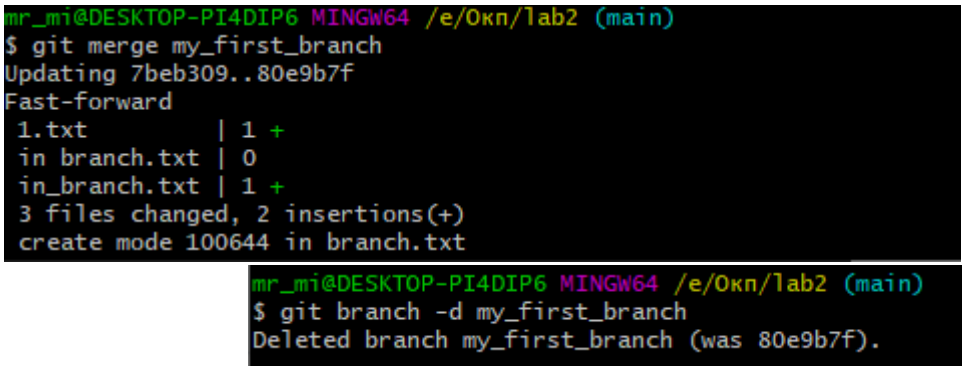

Рисунок 2. Создание веток и их удаление после слияния

6. Убедившись, что прошлые слияния прошли без конфликтов, создал две новые ветки branch1 и branch2. После чего на каждой ветке внёс свои изменения в файлы 1.txt и 3.txt.

7. Начал слияние branch2 в branch1, но из-за образовавшегося конфликта не завершил.

8. Открыл эти файлы 1.txt и 3.txt на ветке branch1 и вручную устранил конфликты. Сделал коммиты этих файлов. Удалил branch2.

9. На удалённом сервере создал новую ветку branch3 и создал ветку для отслеживания branch3.

10. После изменения файла 2.txt на branch3 я слил ветку branch3 в main, убедившись, что никаких конфликтов не обнаружено.

11. Отправил изменения на удалённый сервер.

Контрольные вопросы:

1. Что такое ветка?

Ветка в Git — это простой перемещаемый указатель на один из коммитов. По умолчанию, имя основной ветки в Git — master. Как только вы начнёте создавать коммиты, ветка master будет всегда указывать на последний коммит. Каждый раз при создании коммита указатель ветки master будет передвигаться на следующий коммит автоматически.

2. Что такое HEAD?

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

3. Способы создания веток.

Ветки можно создать с помощью команды “git branch имя_ветки”, и чтобы перейти на неё необходимо использовать команду “git checkout имя_ветки”. Можно же создать ветку и сразу перейти на неё с помощью “git checkout –b имя_ветки”.

4. Как узнать текущую ветку?

С помощью команды git branch высветятся все ветки, текущая будет подкрашена и/или будет со знаком *.

5. Как переключаться между ветками?

Чтобы переходить по веткам необходимо использовать команду git checkout имя_ветки

6. Что такое удаленная ветка?

Это ветка, находящаяся на удалённом сервере GitHub.

7. Что такое ветка отслеживания?

Ветки слежения — это ссылки на определённое состояние удалённых веток. Это локальные ветки, которые напрямую связаны с удалённой веткой. Если, находясь на ветке слежения, выполнить git pull, то Git уже будет знать с какого сервера получать данные и какую ветку использовать для слияния. При клонировании репозитория, как правило, автоматически создаётся ветка master, которая следит за origin/master.

8. Как создать ветку отслеживания?

С помощью команды git branch --track имя_ветки origin/имя_ветки.

9. Как отправить изменения из локальной ветки в удаленную ветку? С помощью команды git push имя_ветки.

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

Git fetch лишь показывает изменения веток на сервере, но не копирует их на локальный репозиторий, в отличие от команды Git pull.

11. Как удалить локальную и удаленную ветки?

Локальную ветку можно удалить с помощью команды git branch –d имя_ветки.

12. Изучить модель ветвления git-flow (использовать материалы статей https://www.atlassian.com/ru/git/tutorials/comparing-workflows/gitflowworkflow, 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 описывает несколько веток для разработки, релизов и взаимодействия между ними.

Репозиторий содержит 2 главные ветки:

– master;
– develop;

master — дефолтная ветка знакомая каждому. Параллельно в этой концепции существует еще одна ветка develop.

Master в этой концепции всегда содержит стабильный код, а develop существует для того чтобы от нее ответвляться и сливать туда уже готовые фичи для последующего сливания в master. Как следствие master выступает релизной веткой в этой концепции. В Git Flow мы можем использовать следующие типы веток:

– Feature branches;
– Release branches;
– Hotfix branches; Минусы git-flow:
– git flow может замедлять работу;
– релизы сложно делать чаще, чем раз в неделю;
– большие функции могут потратить дни на конфликты и форсировать несколько циклов тестирования;
– история проекта в гите имеет кучу merge commits и затрудняет просмотр реальной работы;
– может быть проблематичным в CI/CD сценариях;

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.

Sourcetree позволяет работать с теми же инструментами, которые представление в строке CMD, но при этом управление здесь намного интуитивно понятнее и проще:

– клонирование: копирование удаленного репозитория из Bitbucket Cloud в локальную систему.

– добавление или индексирование: принятие внесенных изменений и их подготовка к добавлению в историю Git.

– коммит: добавление новых или измененных файлов в историю Git для репозитория.

– pull (извлечение): добавление в локальный репозиторий новых изменений, внесенных в репозиторий другими разработчиками.

– push: отправка изменений из локальной системы в удаленный репозиторий.

Вывод: на лабораторной работе изучил основы по созданию, редактированию и удалению веток, а также научился решать конфликты слияния.