МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ

РОССИЙСКОЙ ФЕДЕРАЦИИ

Федеральное государственное автономное

образовательное учреждение высшего образования

«СЕВЕРО-КАВКАЗСКИЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ»

Кафедра инфокоммуникаций

Отчет по лабораторной работе №2

Основы ветвления Git.

по дисциплине «Основы кроссплатформенного программирования»

Выполнил студент группы ИВТ-б-о-20-1

Павленко М.С. « »_____20__г.

Подпись студента_____

Работа защищена « »_____20__г.

Проверила Воронкин Р.А. _____

(подпись)

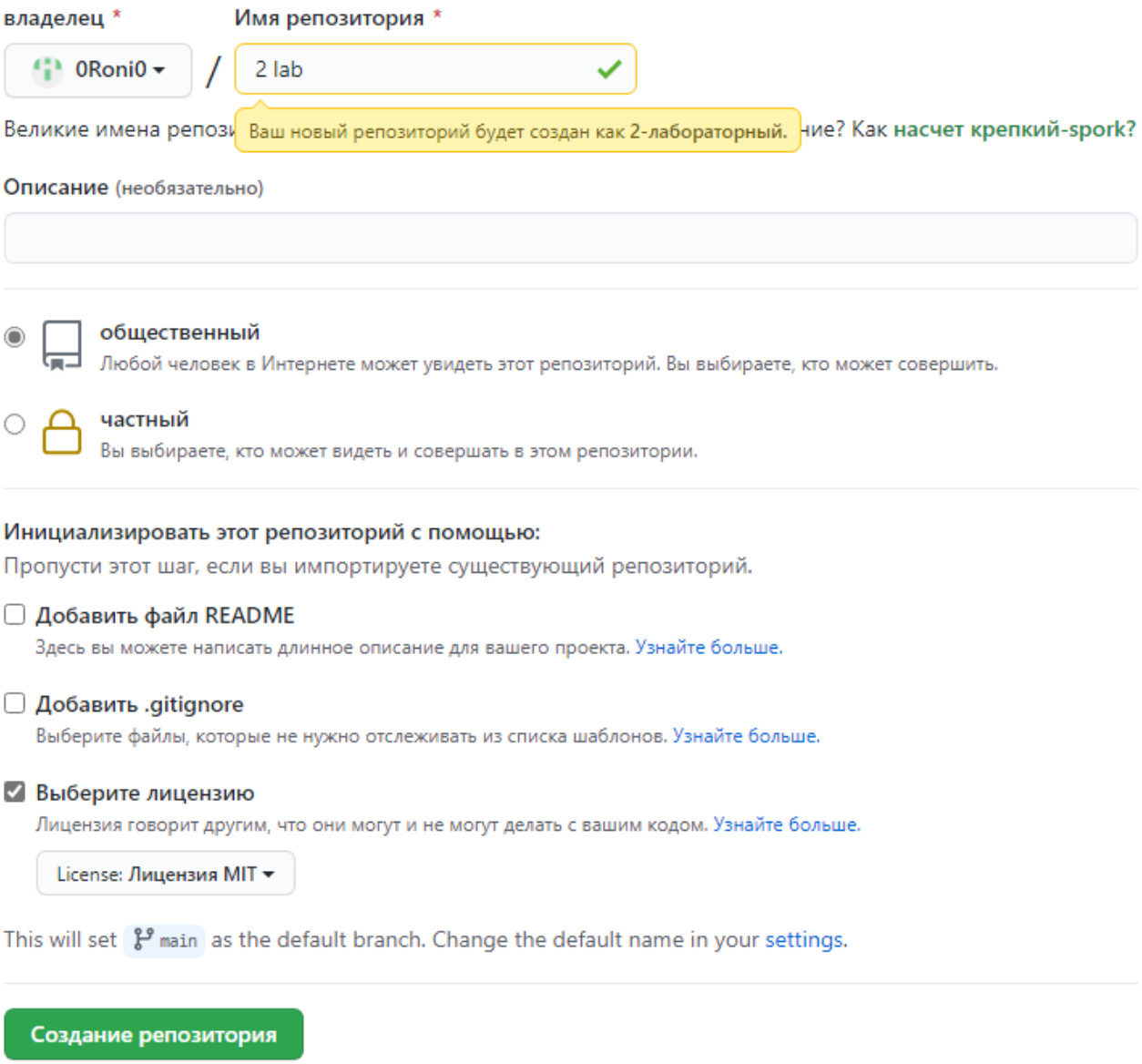
Ставрополь 2021

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

Ссылка на репозиторий:

https://github.com/0Roni0/2-lab.git

Создал репозиторий:



Рисуок 1. Создание репозитория.

Создал три текстовых файла в репозитории:

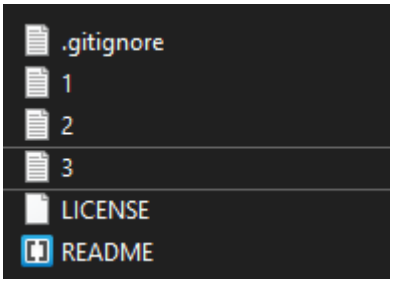


Рисунок 2. Три файла

Проиндексировал 1 файл и прокоммитил, проиндексировал 2 и 3 файлы и прокоммитил:

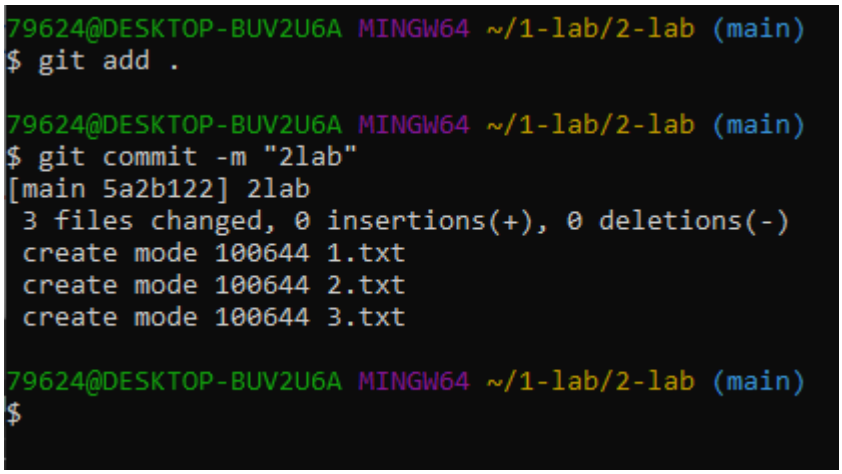


Рисунок 3. Индексация и коммит.

Создал новую ветку, перешёл на неё, закоммитил новый файл и сохранил изменения на Github:

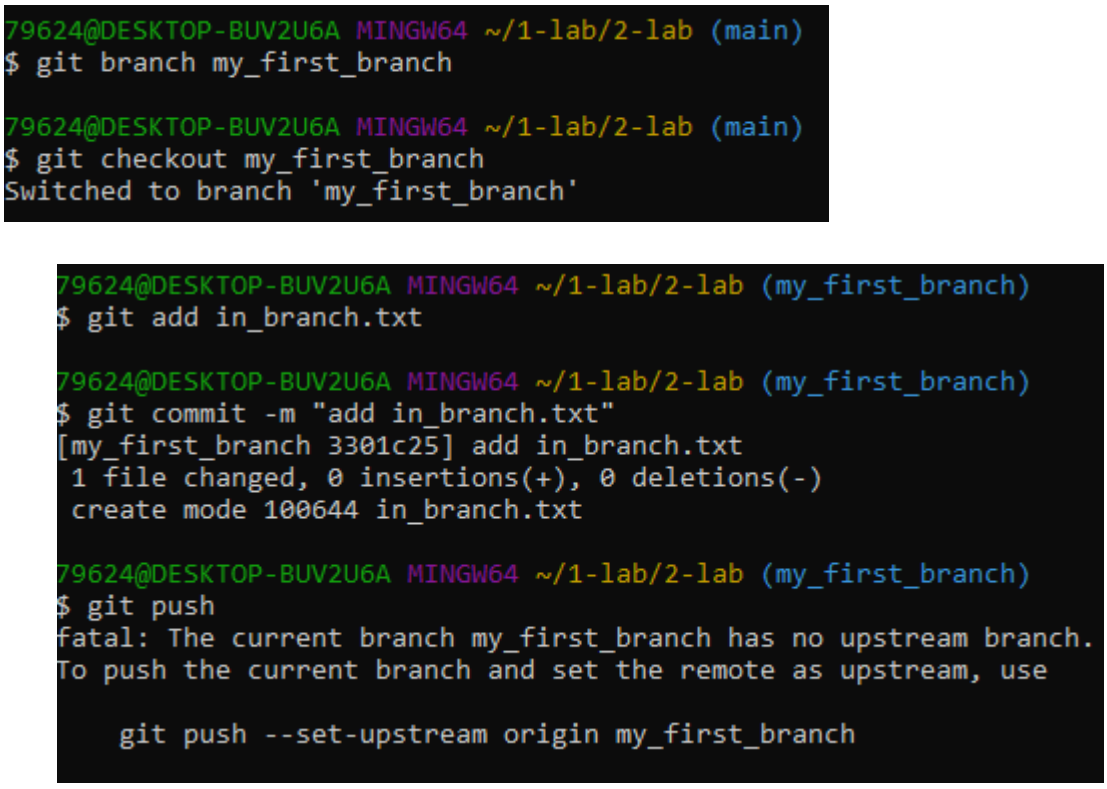


Рисунок 4. Создание новой ветки.

Создал еще одну ветку, перешёл на неё, изменил файл 1.txt и закоммитил изменения:

```
79624@DESKTOP-BUV2U6A MINGW64 ~/1-lab/2-lab (my_first_branch)
$ git branch new_branch

79624@DESKTOP-BUV2U6A MINGW64 ~/1-lab/2-lab (my_first_branch)
$ git checkout new_branch
Switched to branch 'new_branch'

79624@DESKTOP-BUV2U6A MINGW64 ~/1-lab/2-lab (new_branch)
$ git add 1.txt

79624@DESKTOP-BUV2U6A MINGW64 ~/1-lab/2-lab (new_branch)
$ git commit -m "1.txt"
[new_branch 06c156f] 1.txt
 1 file changed, 1 insertion(+)

79624@DESKTOP-BUV2U6A MINGW64 ~/1-lab/2-lab (new_branch)
$ git checkout main
Switched to branch 'main'
Your branch is ahead of 'origin/main' by 1 commit.
  (use "git push" to publish your local commits)
```

Рисунок 5. Вторая созданная ветка.

Слил первые две ветки:

```
79624@DESKTOP-BUV2U6A MINGW64 ~/1-lab/2-lab (new_branch)
$ git checkout main
Switched to branch 'main'
Your branch is ahead of 'origin/main' by 1 commit.
  (use "git push" to publish your local commits)

79624@DESKTOP-BUV2U6A MINGW64 ~/1-lab/2-lab (main)
$ git merge my_first_branch
Updating 5a2b122..3301c25
Fast-forward
 in_branch.txt | 0
 1 file changed, 0 insertions(+), 0 deletions(-)
 create mode 100644 in_branch.txt

79624@DESKTOP-BUV2U6A MINGW64 ~/1-lab/2-lab (main)
$ git status
On branch main
Your branch is ahead of 'origin/main' by 2 commits.
  (use "git push" to publish your local commits)

nothing to commit, working tree clean

79624@DESKTOP-BUV2U6A MINGW64 ~/1-lab/2-lab (main)
$ git merge new_branch
Updating 3301c25..06c156f
Fast-forward
 1.txt | 1 +
 1 file changed, 1 insertion(+)
```

Рисунок 6. Слияние веток.

Удалил созданные ветки:

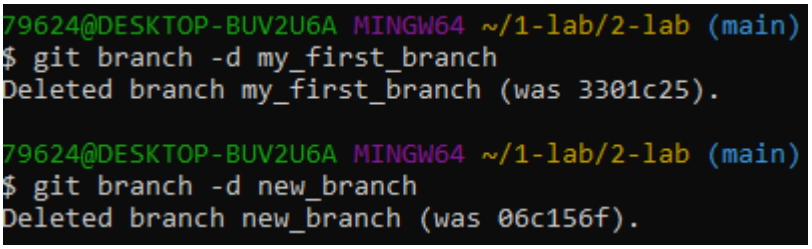


Рисунок 7. Удаление веток.

Создал новые ветки, перешёл на ветку branch_1, изменил нужные файлы и закоммитил изменения:

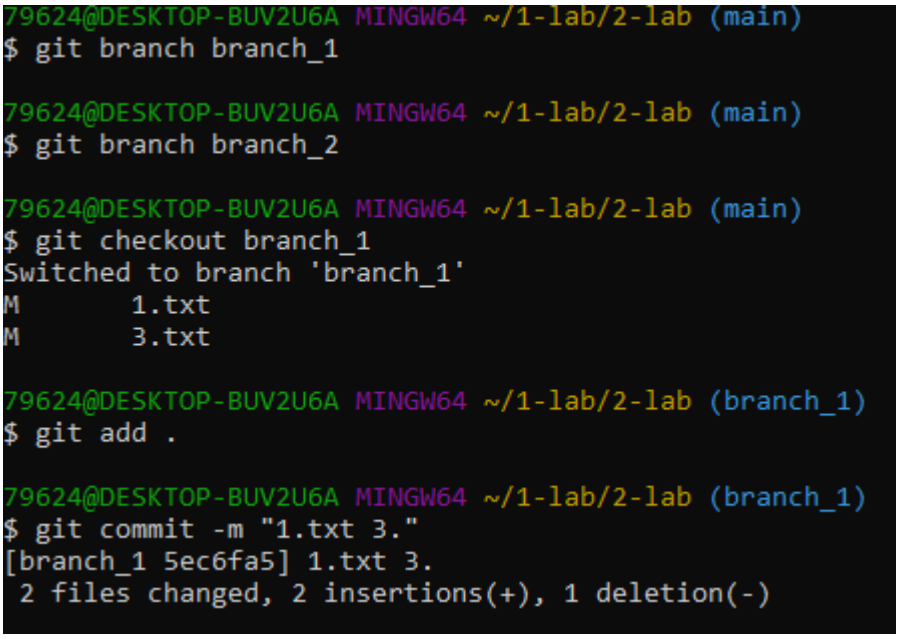


Рисунок 8. Изменение файлов.

Сделал то же самое в ветке branch_2:

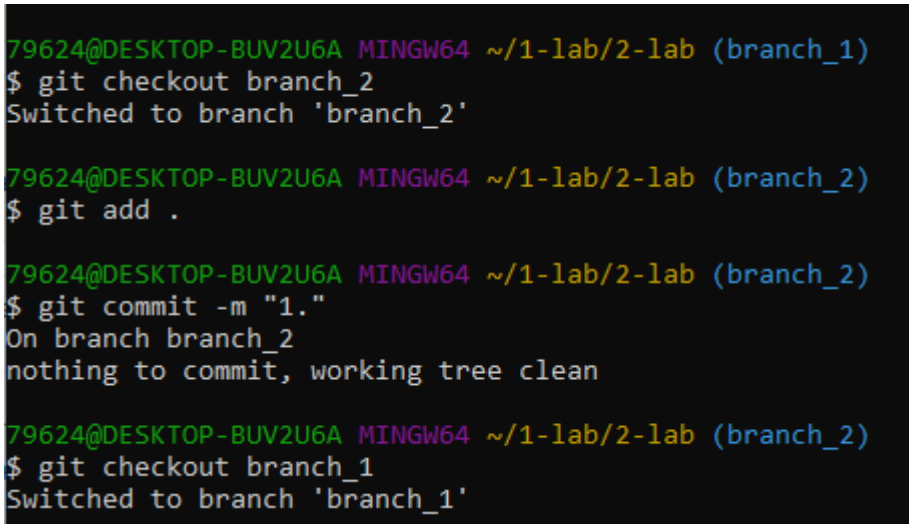


Рисунок 9. Изменения в файлах.

```
79624@DESKTOP-BUV2U6A MINGW64 ~/1-lab/2-lab (branch_1)
$ git status
On branch branch_1
nothing to commit, working tree clean

79624@DESKTOP-BUV2U6A MINGW64 ~/1-lab/2-lab (branch_1)
$ git push origin branch_1
Enumerating objects: 21, done.
Counting objects: 100% (21/21), done.
Delta compression using up to 16 threads
Compressing objects: 100% (14/14), done.
Writing objects: 100% (21/21), 3.40 KiB | 1.70 MiB/s, done.
Total 21 (delta 5), reused 4 (delta 0), pack-reused 0
remote: Resolving deltas: 100% (5/5), done.
remote:
remote: Create a pull request for 'branch_1' on GitHub by visiting:
remote:      https://github.com/0Roni0/2-lab/pull/new/branch_1
remote:
To https://github.com/0Roni0/2-lab.git
 * [new branch]      branch_1 -> branch_1
```

Рисунок 10. Слияние веток.

Удалил ветку branch_2 и создал ветку branch_3 на Github:

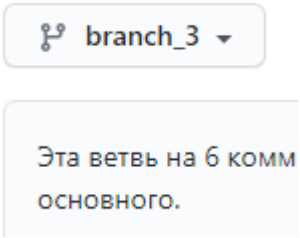


Рисунок 11. Создание ветки на Github.

Получил данные с сервера и создал ветку слежения:

```
79624@DESKTOP-BUV2U6A MINGW64 ~/1-lab/2-lab (branch_1)
$ git fetch origin branch_3
From https://github.com/0Roni0/2-lab
 * branch            branch_3   -> FETCH_HEAD
 * [new branch]      branch_3   -> origin/branch_3
```

Рисунок 13. Ветка слежения.

Перешёл на новую ветку, изменил 2.txt и закоммитил:

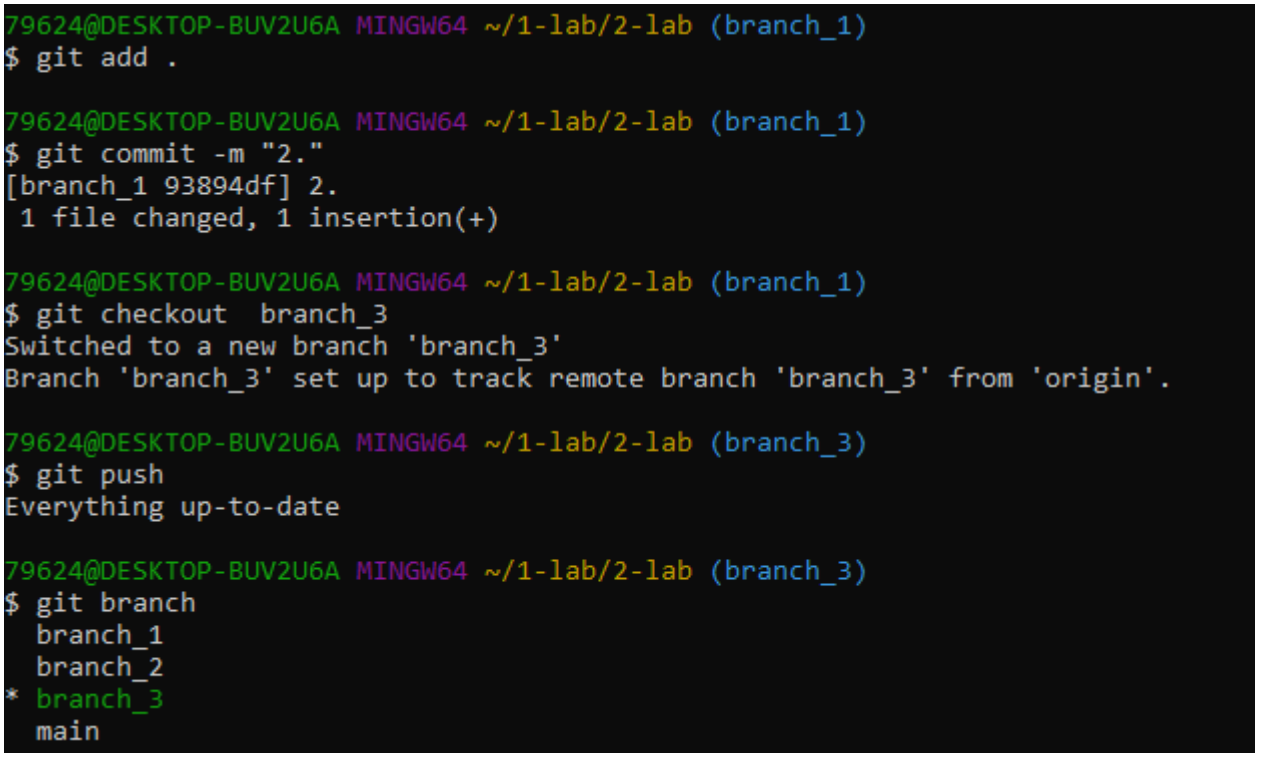


Рисунок 14. Коммит файла.

Отправил всё на Github.

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 “git 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

checkout –b имя_ветки”.

4. Как узнать текущую ветку?

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 в

етка

master, которая следит за origin/master.

5. Как создать ветку отслеживания?

С помощью команды git checkout -b имя_ветки origin/имя_ветки.

6. Как отправить изменения из локальной ветки в удаленную ветку?

С помощью команды git push имя_ветки.

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

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

8. Как удалить локальную и удаленную ветки?

Локальную ветку можно удалить с помощью команды git branch –dимя_ветки.

9. 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 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, a 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;

Этого большие функции могут потратить дни на конфликты и форсироватьнесколько циклов тестирования;

– история проекта в гите имеет кучу merge commits и затрудняетпросмотр реальной работы;

– может быть проблематичным в CI/CD сценариях;

10. На прошлой лабораторной работе было задание выбрать одно из программных средств с GUI для работы с Git. Необходимо в рамках вопроса привести описание инструментов для работы с ветками Git, предоставляемых этим средством. Sourcetree позволяет работать с

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: при выполнении заданий были исследованы базовые возможности работы с локальными и удаленными ветками Git.