Министерство науки и высшего образования Российской Федерации
Федеральное государственное автономное образовательное учреждение высшего образования
«СЕВЕРО-КАВКАЗСКИЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ»

Институт цифрового развития
Кафедра инфокоммуникаций

ОТЧЕТ
ПО ЛАБОРАТОРНОЙ РАБОТЕ №1.3
дисциплины «Программирование на Python»

Выполнил:
Магдаев Даламбек Магомедович
2 курс, группа ИВТ-б-о-22-1,
09.03.01 «Информатика и вычислительная техника»,
направленность (профиль) «Программное обеспечение средств вычислительной техники и автоматизированных систем», очная форма обучения

_____
(подпись)

Руководитель практики:
Воронкин Р.А., доцент кафедры инфокоммуникаций

_____
(подпись)

Отчет защищен с оценкой _____ Дата защиты_____

Ставрополь, 2023 г.

Тема: Основы ветвления Git

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

Порядок выполнения работы:

1. Создал новый репозиторий и клонировал его:

```
PS C:\Users\dalam> git clone https://github.com/MagdaevDalambek/Lab1.3.git
Cloning into 'Lab1.3'...
remote: Enumerating objects: 11, done.
remote: Counting objects: 100% (11/11), done.
remote: Compressing objects: 100% (10/10), done.
remote: Total 11 (delta 2), reused 0 (delta 0), pack-reused 0
Receiving objects: 100% (11/11), 4.08 KiB | 522.00 KiB/s, done.
Resolving deltas: 100% (2/2), done.
PS C:\Users\dalam>
```

Рисунок 1. Клонирование репозитория

2. Создал 3 файла: 1.txt, 2.txt и 3.txt. Проиндексировал первый файл и сделал коммит, потом проиндексировал оставшиеся и перезаписал уже сделанный коммит с новым именем:

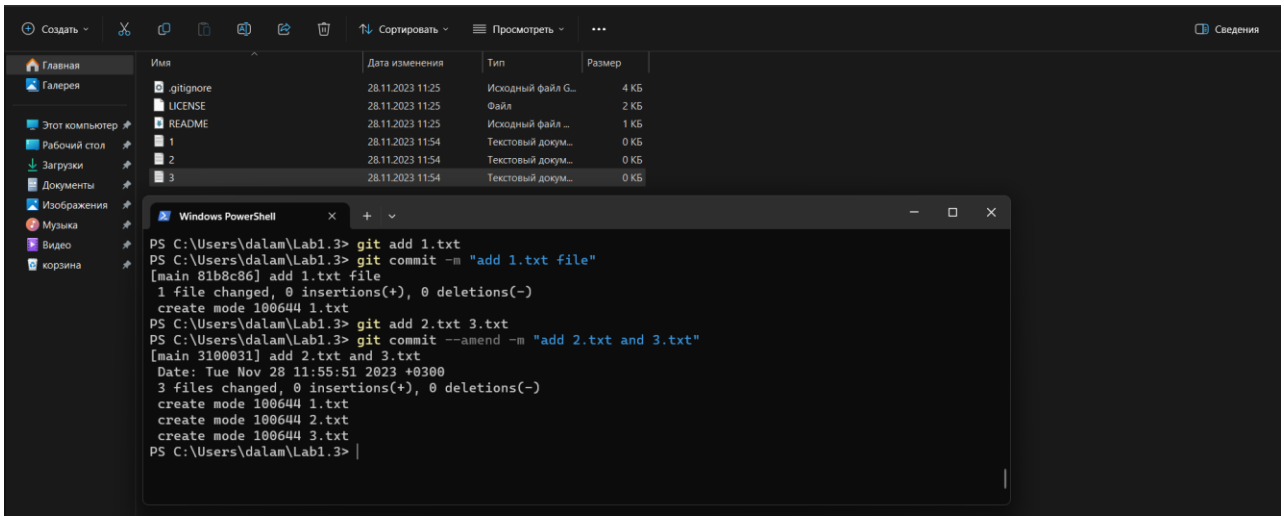

Рисунок 2. Перезапись коммита

3. Создал новую Ветку и перешел на нее, после чего создал новый файл, закоммитил изменения и вернулся на ветку main:

```
PS C:\Users\dalam\Lab1.3> git branch my_first_branch
PS C:\Users\dalam\Lab1.3> git checkout my_first_branch
Switched to branch 'my_first_branch'
PS C:\Users\dalam\Lab1.3> touch in_branch.txt
touch : Имя "touch" не распознано как имя командлета, функции, файла сценария или выполняемой программы.
 Проверьте правильность написания имени, а также наличие и правильность пути, после чего повторите попыт
ку.
строка:1 знак:1
+ touch in_branch.txt
+ ~~~~~
    + CategoryInfo          : ObjectNotFound: (touch:String) [], CommandNotFoundException
    + FullyQualifiedErrorId : CommandNotFoundException

PS C:\Users\dalam\Lab1.3> git add in_branch.txt
PS C:\Users\dalam\Lab1.3> git commit -m "in_branch.txt"
[my_first_branch 032d02d] in_branch.txt
 1 file changed, 0 insertions(+), 0 deletions(-)
 create mode 100644 in_branch.txt
PS C:\Users\dalam\Lab1.3> 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)
PS C:\Users\dalam\Lab1.3>
```

Рисунок 3. Создание и перемещение между ветками

4. Создал и сразу же перешел на новую ветку, добавил строку в файл 1.txt, закоммитил изменения:

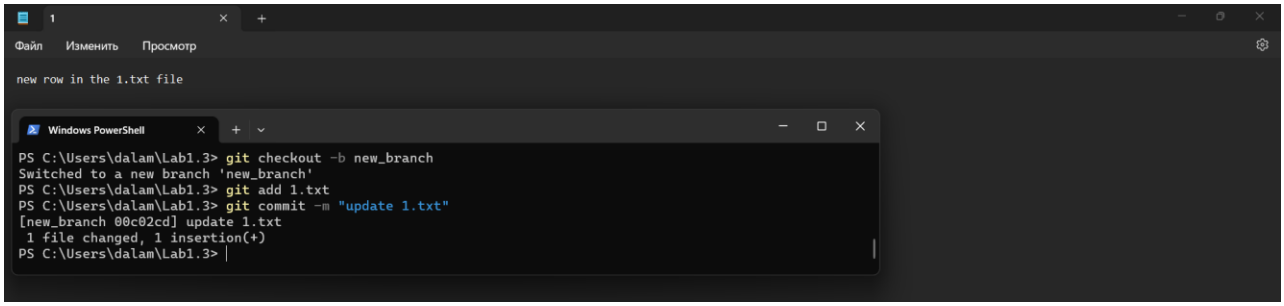

Рисунок 4. Создание новой ветки и коммит обновленного файла

5. Переключился на ветку main, слил ее с my_first_branch и с new_branch:

```
PS C:\Users\dalam\Lab1.3> 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)
PS C:\Users\dalam\Lab1.3> git merge my_first_branch
Updating 3100031..032d02d
Fast-forward
 in_branch.txt | 0
 1 file changed, 0 insertions(+), 0 deletions(-)
 create mode 100644 in_branch.txt
PS C:\Users\dalam\Lab1.3> git merge new_branch
Merge made by the 'ort' strategy.
 1.txt | 1 +
 1 file changed, 1 insertion(+)
PS C:\Users\dalam\Lab1.3>
```

Рисунок 5. Выполнил одно слияние, но не выполнил второе

6. Удалил созданные ветки:

```
PS C:\Users\dalam\Lab1.3> git branch -d my_first_branch
Deleted branch my_first_branch (was 032d02d).
PS C:\Users\dalam\Lab1.3> git branch -d new_branch
Deleted branch new_branch (was 00c02cd).
PS C:\Users\dalam\Lab1.3>
```

Рисунок 6. Удаление веток

7. Создал новые ветки, перешел в ветку branch_1 и изменил файлы 1.txt, 3.txt, закоммитил изменения:

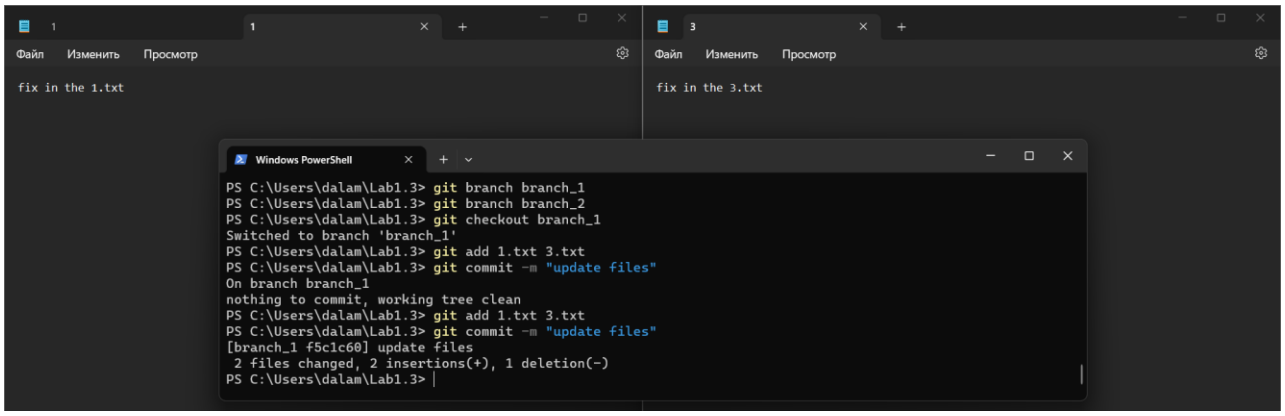

Рисунок 7. Создал новые ветки, закоммитил изменения файлов в первой

8. Перешел во вторую созданную ветку и изменил там те же самые файлы, закоммитил изменения:

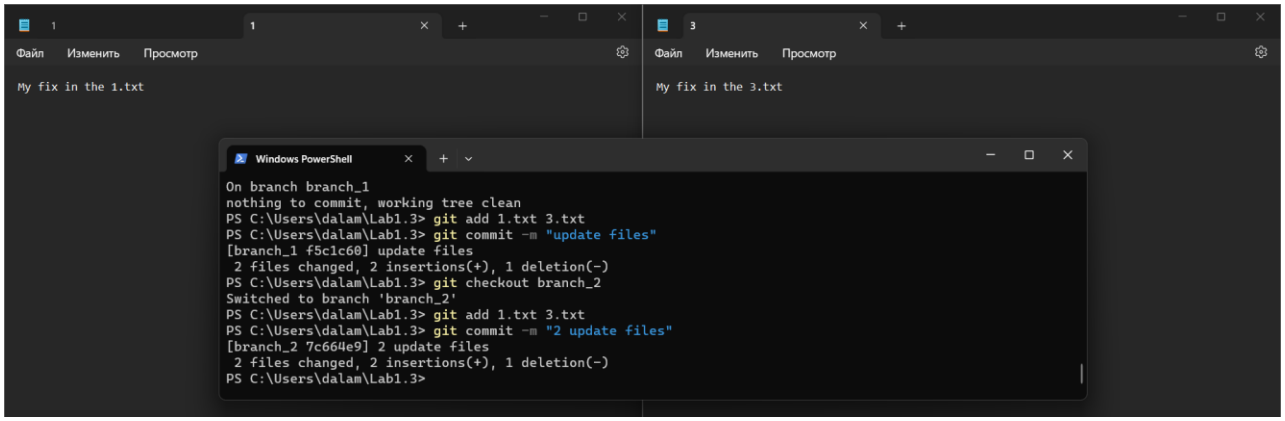

Рисунок 8. Закоммитил новые изменения в файлах в ветке branch_2

9. Перешел в ветку branch_1 и попытался слить в нее branch_2:

```
PS C:\Users\dalam\Lab1.3> git checkout branch_1
Switched to branch 'branch_1'
PS C:\Users\dalam\Lab1.3> git merge branch_2
Auto-merging 1.txt
CONFLICT (content): Merge conflict in 1.txt
Auto-merging 3.txt
CONFLICT (content): Merge conflict in 3.txt
Automatic merge failed; fix conflicts and then commit the result.
PS C:\Users\dalam\Lab1.3>
```

Рисунок 9. Попытка слияния завершилась неудачно

10. Решение первого конфликта вручную:

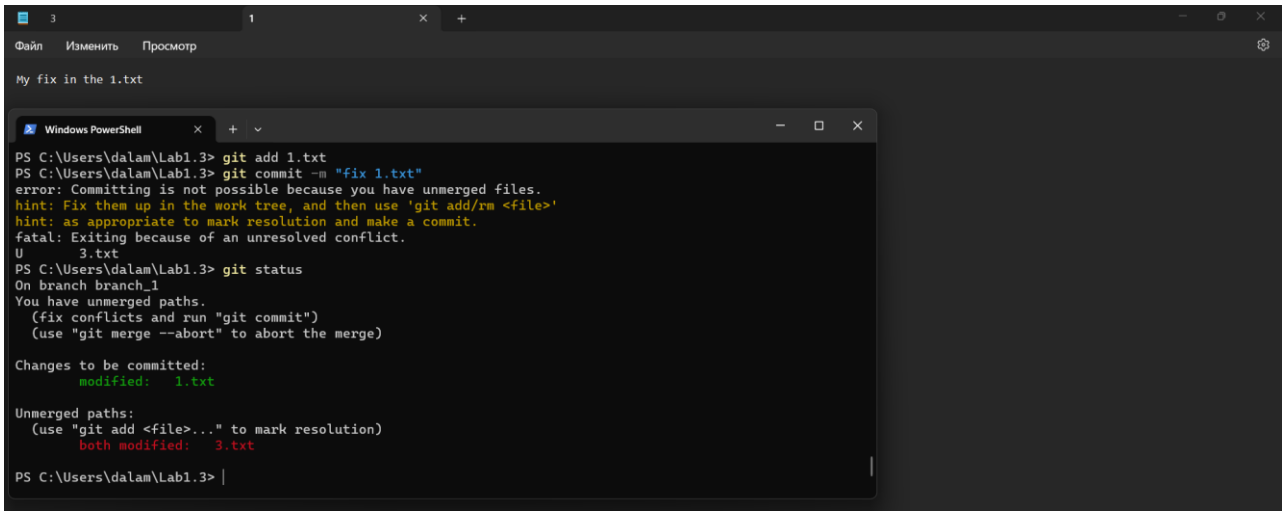

Рисунок 10. Решение первого конфликта вручную

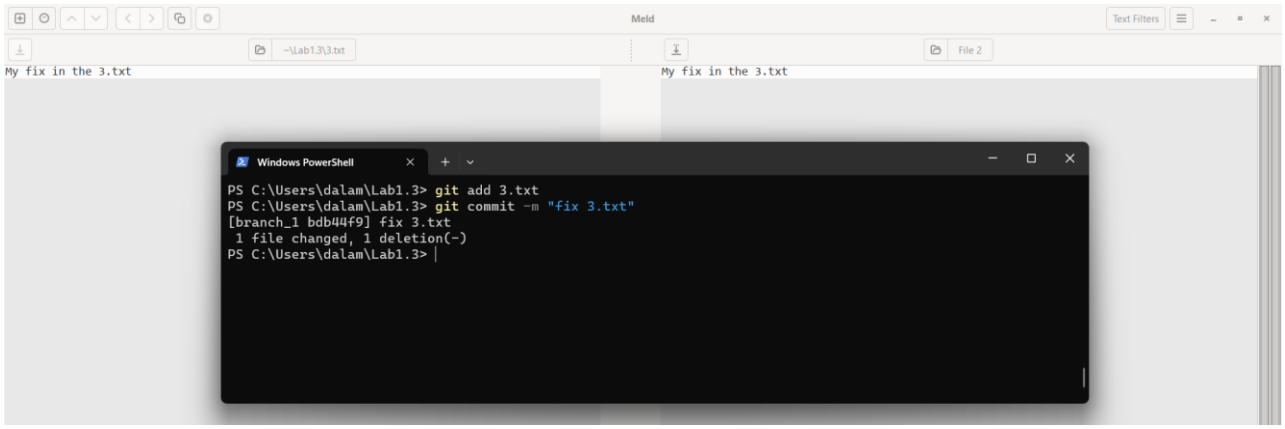

Рисунок 11. Решение второго конфликта с помощью утилиты Meld

```
PS C:\Users\dalam\Lab1.3> git merge branch_2
Already up to date.
PS C:\Users\dalam\Lab1.3>
```

Рисунок 12. Завершил слияние

11. Отправил ветку branch_1 на удаленный репозиторий:

```
PS C:\Users\dalam\Lab1.3> git push origin branch_1
Enumerating objects: 27, done.
Counting objects: 100% (27/27), done.
Delta compression using up to 12 threads
Compressing objects: 100% (17/17), done.
Writing objects: 100% (26/26), 2.10 KiB | 537.00 KiB/s, done.
Total 26 (delta 10), reused 0 (delta 0), pack-reused 0
remote: Resolving deltas: 100% (10/10), done.
remote:
remote: Create a pull request for 'branch_1' on GitHub by visiting:
remote:      https://github.com/MagdaevDalambek/Lab1.3/pull/new/branch_1
remote:
To https://github.com/MagdaevDalambek/Lab1.3.git
 * [new branch]      branch_1 -> branch_1
PS C:\Users\dalam\Lab1.3>
```

Рисунок 13. Отправка ветки

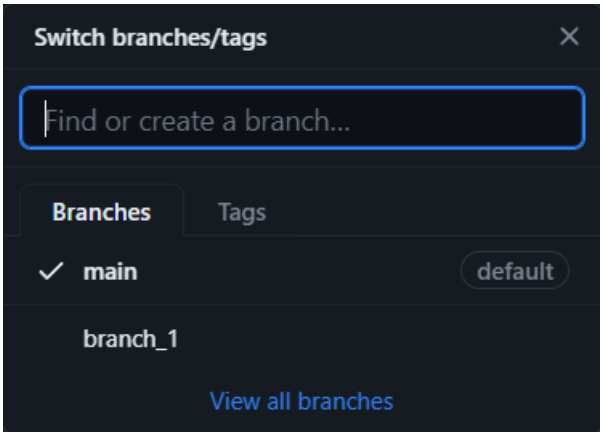

Рисунок 14. Ветки в GitHub

12. Создал ветку средствами GitHub и перенес ее в локальный репозиторий:

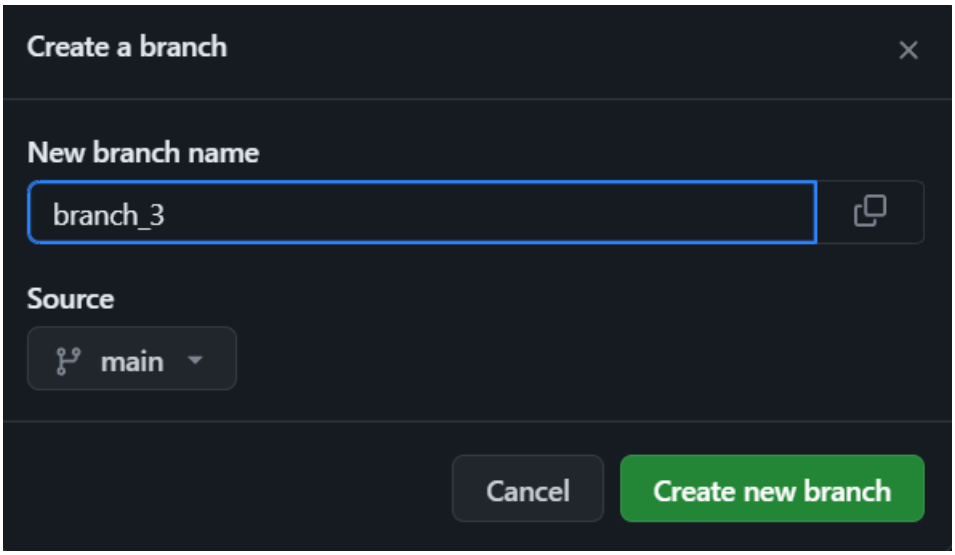

Рисунок 15. Создание ветки

```
PS C:\Users\dalam\Lab1.3> git pull
From https://github.com/MagdaevDalambek/Lab1.3
 * [new branch]      branch_3   -> origin/branch_3
There is no tracking information for the current branch.
Please specify which branch you want to merge with.
See git-pull(1) for details.

    git pull <remote> <branch>

If you wish to set tracking information for this branch you can do so with:

    git branch --set-upstream-to=origin/<branch> branch_1

PS C:\Users\dalam\Lab1.3> git checkout branch_3
Switched to a new branch 'branch_3'
branch 'branch_3' set up to track 'origin/branch_3'.
PS C:\Users\dalam\Lab1.3>
```

Рисунок 16. Успешный перенос ветки

13. Добавил в файл 2.txt новую строку и закоммитил:

```
PS C:\Users\dalam\Lab1.3> git add 2.txt
PS C:\Users\dalam\Lab1.3> git commit -m "update 2.txt"
[branch_3 b7ad045] update 2.txt
 1 file changed, 1 insertion(+)
 create mode 100644 2.txt
```

Рисунок 17. Коммит файла 2.txt

14. Выполнил перемещение ветки main на ветку banch_2 и отправил изменения обоих веток на GitHub:

```
PS C:\Users\dalam\Lab1.3> git checkout branch_2
Switched to branch 'branch_2'
PS C:\Users\dalam\Lab1.3> git rebase main
Current branch branch_2 is up to date.
PS C:\Users\dalam\Lab1.3> git checkout main
Switched to branch 'main'
Your branch is ahead of 'origin/main' by 4 commits.
  (use "git push" to publish your local commits)
PS C:\Users\dalam\Lab1.3> git push origin main
Total 0 (delta 0), reused 0 (delta 0), pack-reused 0
To https://github.com/MagdaevDalambek/Lab1.3.git
   2e8c5c5..0ae6345  main -> main
PS C:\Users\dalam\Lab1.3> git push origin branch_2
Total 0 (delta 0), reused 0 (delta 0), pack-reused 0
remote:
remote: Create a pull request for 'branch_2' on GitHub by visiting:
remote:      https://github.com/MagdaevDalambek/Lab1.3/pull/new/branch_2
remote:
To https://github.com/MagdaevDalambek/Lab1.3.git
 * [new branch]      branch_2 -> branch_2
PS C:\Users\dalam\Lab1.3>
```

Рисунок 18. Перенос ветки и отправка изменений

Ответы на контрольные вопросы:

1. Что такое ветка?

Ветка в Git – это простой перемещаемый указатель на один из коммитов.

2. Что такое HEAD?

HEAD – это указатель, задача которого ссылаться на определенный коммит в репозитории. Суть данного указателя можно попытаться объяснить с разных сторон.

Во-первых, HEAD – это указатель на коммит в вашем репозитории, который станет родителем следующего коммита.

Во-вторых, HEAD указывает на коммит, относительного которого будет создана рабочая копия во время операции checkout.

3. Способы создания веток.

Чтобы создать новую ветку, необходимо использовать команду git branch.

Чтобы создать ветку и сразу переключиться на нее, можно использовать команду git checkout -b.

4. Как узнать текущую ветку?

Увидеть текущую ветку можно при помощи простой команды git log, которая покажет куда указывают указатели веток. Эта опция называется --decorate. HEAD будет стоять рядом с текущей веткой. Также можно использовать git branch -v, рядом с текущей веткой будет значек *.

5. Как переключаться между ветками?

Для переключения на существующую ветку выполните команду git checkout.

6. Что такое удаленная ветка?

Удалённые ссылки – это ссылки (указатели) в ваших удалённых репозиториях, включая ветки, теги и так далее. Полный список удалённых ссылок можно получить с помощью команды git ls-remote <remote> или команды git remote show <remote> для получения удалённых веток и

дополнительной информации.

7. Что такое ветка отслеживания?

Ветки слежения – это ссылки на определённое состояние удалённых веток. Это локальные ветки, которые нельзя перемещать; Git перемещает их автоматически при любой коммуникации с удаленным репозиторием, чтобы гарантировать точное соответствие с ним.

8. Как создать ветку отслеживания?

При клонировании репозитория, как правило, автоматически создаётся ветка master, которая следит за origin/master. Однако, при желании можно настроить отслеживание и других веток — следить за ветками на других серверах или отключить слежение за веткой master. Сделать это можно с помощью команды git checkout -b <branch> <remote>/<branch>. Это часто используемая команда, поэтому Git предоставляет сокращённую форму записи в виде флага --track.

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

Когда вы хотите поделиться веткой, вам необходимо отправить её на удалённый сервер, где у вас есть права на запись. Ваши локальные ветки автоматически не синхронизируются с удалёнными при отправке — вам нужно явно указать те ветки, которые вы хотите отправить.

Таким образом, вы можете использовать свои личные ветки для работы, которую не хотите показывать, а отправлять только те тематические ветки, над которыми вы хотите работать с кем-то совместно. Отправка осуществляется командой git push <remote> <branch>.

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

Команда git fetch получает с сервера все изменения, которых у вас ещё нет, но не будет изменять состояние вашей рабочей директории. Эта команда

просто получает данные и позволяет вам самостоятельно сделать слияние. Тем не менее, существует команда git pull, которая в большинстве случаев является командой git fetch , за которой непосредственно следует команда git merge .

Если у вас настроена ветка слежения, или она явно установлена, или она была создана автоматически командами clone или checkout, git pull определит сервер и ветку, за которыми следит ваша текущая ветка, получит данные с этого сервера и затем попытается слить удалённую ветку.

11. Как удалить локальную и удаленную ветки?

Для удаления локальной ветки выполните команду git branch с параметром -d.

Вы можете удалить ветку на удалённом сервере используя параметр --delete для команды git push .

12. 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 присуствуют в модели git-flow? Как организована работа с ветками в модели git-flow? В чем недостатки git-flow?

В модели ветвления git-flow, основные типы веток это:

master - главная ветка проекта, которая содержит только стабильный код и используется для создания релизов.

develop - ветка, в которой ведется основная разработка проекта. Она содержит все изменения, которые были сделаны разработчиками.

feature - ветки для добавления новой функциональности в проект. Они ветвятся от ветки develop, и после завершения работы ветки объединяются с develop веткой.

release - ветки для подготовки новой версии проекта. Они ветвятся от ветки develop, содержат минимальный набор изменений и используются для подготовки релиза. После тестирования и отладки, ветка объединяется с

веткой master и develop.

hotfix - ветки для исправления критических ошибок на производственной версии проекта. Они ветвятся от ветки master, и после исправления ошибок объединяются с master и develop ветками.

Работа с ветками в модели git-flow организована следующим образом:

Начало разработки новой функциональности начинается с ветвления от ветки develop ветки feature, на которой работает разработчик.

После завершения работы, все изменения ветки feature тестируются, затем вливаются обратно в ветку develop.

В момент подготовки новой версии программного продукта ветка release создается из develop, и она используется для проведения основных тестов наиболее важных функций.

Ветка hotfix создается из ветки master, если в производственной версии обнаружена критическая ошибка, и на этой ветке выполняется исправление.

Недостатки git-flow включают в себя:

Сложность и необходимость управления множеством веток, что может быть трудным для маленьких команд.

Не слишком хорошо подходит для быстрой разработки и быстрой реализации необходимых исправлений или функций в связи с наличием большого количества ветвлений.

Накладывает значительный набор процедур и правил для разработки и управления релизами.

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.

В Git клиенте codeberg.org присутствует возможность работы с ветками Git. Инструменты для работы с ветками в codeberg.org соответствуют стандартным инструментам Git и включают в себя:

Создание новой ветки из текущей ветки.Переключение между ветками.

Объединение веток (с помощью команды merge), которые могут быть выполнены из интерфейса Git клиента.

Возможность создавать теги и выполнять релизы (включая создание новой ветки для релиза и склеивание веток для внесения исправлений в производственную версию приложения).

Вывод: в результате выполнения работы были исследованы базовые возможности по работе с локальными и удаленными ветками Git.