МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ
РОССИЙСКОЙ ФЕДЕРАЦИИ
ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ
ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ
ВЫСШЕГО ОБРАЗОВАНИЯ
«СЕВЕРО-КАВКАЗСКИЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ»
ИНСТИТУТ ЦИФРОВОГО РАЗВИТИЯ

Отчет о лабораторной работе №3 по дисциплине «Основы программной инженерии»

Выполнил:
Турклиев Владимир
Назирович,
2 курс, группа ПИЖ-б-о-20-1,
Проверил:
Доцент кафедры инфокоммуникаций,
Воронкин Р.А.

Ставрополь, 2021 г

Порядок выполнения работы:

1. Добавление файлов с помощью перезаписи коммитов.

```
C:\Users\Vova\Desktop\учеба\основы ии\lab3\lab3>git commit -m "add 2.txt and 3.txt"
[main 765a45e] add 2.txt and 3.txt
 2 files changed, 0 insertions(+), 0 deletions(-)
 create mode 100644 2.txt
 create mode 100644 3.txt

C:\Users\Vova\Desktop\учеба\основы ии\lab3\lab3>git commit --amend -m "add 2.txt and 3.txt"
[main 56558a0] add 2.txt and 3.txt
 Date: Wed Jan 26 21:38:08 2022 +0300
 2 files changed, 0 insertions(+), 0 deletions(-)
 create mode 100644 2.txt
 create mode 100644 3.txt

C:\Users\Vova\Desktop\учеба\основы ии\lab3\lab3>
```

Рисунок 1 – Перезапись коммитов с помощью команды –amend.

2. Создание новой ветки и добавление файла.

```
C:\Users\Vova\Desktop\учеба\основы ии\lab3\lab3>git branch ttt

C:\Users\Vova\Desktop\учеба\основы ии\lab3\lab3>git checkout ttt
Switched to branch 'ttt'

C:\Users\Vova\Desktop\учеба\основы ии\lab3\lab3>
C:\Users\Vova\Desktop\учеба\основы ии\lab3\lab3>git add TTT.txt
```

Рисунок 2 – Коммит файла в новой ветке.

3. Создание новой ветки и изменение файла в ней.

```
C:\Users\Vova\Desktop\учеба\основы ии\lab3\lab3>git checkout -b YYYY
Switched to a new branch 'YYYY'

C:\Users\Vova\Desktop\учеба\основы ии\lab3\lab3>git status
On branch YYYY
Untracked files:
  (use "git add <file>..." to include in what will be committed)
        YYYY.txt

nothing added to commit but untracked files present (use "git add" to track)

C:\Users\Vova\Desktop\учеба\основы ии\lab3\lab3>git add YYYY.txt

C:\Users\Vova\Desktop\учеба\основы ии\lab3\lab3>git commit -m "new row in the YYYY.txt"
[YYYY 200356f] new row in the YYYY.txt
 1 file changed, 0 insertions(+), 0 deletions(-)
 create mode 100644 YYYY.txt

C:\Users\Vova\Desktop\учеба\основы ии\lab3\lab3>
```

Рисунок 3 – Создание ветки и мгновенное переключение на нее с помощьюкоманды “git checkout -b”, добавление файла в ветку.

4. Слияние веток.

```
C:\Users\Vova\Desktop\учеба\основы ии\lab3\lab3>git checkout main
Switched to branch 'main'
Your branch is ahead of 'origin/main' by 2 commits.
  (use "git push" to publish your local commits)

C:\Users\Vova\Desktop\учеба\основы ии\lab3\lab3>git merge TTT
Updating 56558a0..86e9f98
Fast-forward
 TTT.txt | 0
 1 file changed, 0 insertions(+), 0 deletions(-)
 create mode 100644 TTT.txt

C:\Users\Vova\Desktop\учеба\основы ии\lab3\lab3>git merge YYYY
Updating 86e9f98..200356f
Fast-forward
 YYYY.txt | 0
 1 file changed, 0 insertions(+), 0 deletions(-)
```

Рисунок 4 – Слияние побочных веток с главной с помощью “git merge”.

5. Удаление веток.

```
C:\Users\Vova\Desktop\учеба\основы ии\lab3\lab3> git branch -d TTT
Deleted branch TTT (was 86e9f98).

C:\Users\Vova\Desktop\учеба\основы ии\lab3\lab3> git branch -d YYYY
Deleted branch YYYY (was 200356f).

C:\Users\Vova\Desktop\учеба\основы ии\lab3\lab3>
```

Рисунок 5 – Удаление слитых веток с помощью “git branch -d”.

6. слияния веток.

```
C:\Users\Vova\Desktop\учеба\основы ии\lab3\lab3>git branch RRR

C:\Users\Vova\Desktop\учеба\основы ии\lab3\lab3>git branch FFF
```

Рисунок 6.1 – создание двух веток

```
C:\Users\Vova\Desktop\учеба\основы ии\lab3\lab3>git checkout RRR
Already on 'RRR'
M       1.txt
M       3.txt

C:\Users\Vova\Desktop\учеба\основы ии\lab3\lab3>git status
On branch RRR
Changes not staged for commit:
  (use "git add <file>..." to update what will be committed)
  (use "git restore <file>..." to discard changes in working directory)
        modified:   1.txt
        modified:   3.txt

no changes added to commit (use "git add" and/or "git commit -a")

C:\Users\Vova\Desktop\учеба\основы ии\lab3\lab3>git add .

C:\Users\Vova\Desktop\учеба\основы ии\lab3\lab3>git comit -m "changed 1 and 2"
git: 'comit' is not a git command. See 'git --help'.

The most similar command is
        commit

C:\Users\Vova\Desktop\учеба\основы ии\lab3\lab3>git commit -m "changed 1 and 2"
[RRR c8ee8c2] changed 1 and 2
 2 files changed, 2 insertions(+)
```

Рисунок 6.2 – Изменения файлов в ветке RRR.

```
C:\Users\Vova\Desktop\учеба\основы ии\lab3\lab3>git merge YYY
Already up to date.

C:\Users\Vova\Desktop\учеба\основы ии\lab3\lab3>git merge FFF
Already up to date.

C:\Users\Vova\Desktop\учеба\основы ии\lab3\lab3>git merge RRR
Updating 200356f..b34b6cb
Fast-forward
 1.txt | 1 +
 3.txt | 1 +
 2 files changed, 2 insertions(+)

C:\Users\Vova\Desktop\учеба\основы ии\lab3\lab3>
```

Рисунок 6.3 – Слияние веток.

7. Конфликт слияния

```
C:\Users\Vova\Desktop\учеба\основы ии\lab3\lab3>git merge FFF
Auto-merging 3.txt
CONFLICT (content): Merge conflict in 3.txt
Automatic merge failed; fix conflicts and then commit the result.
```

Рисунок 7 – Конфликт слияния

8. Создание удаленной ветки

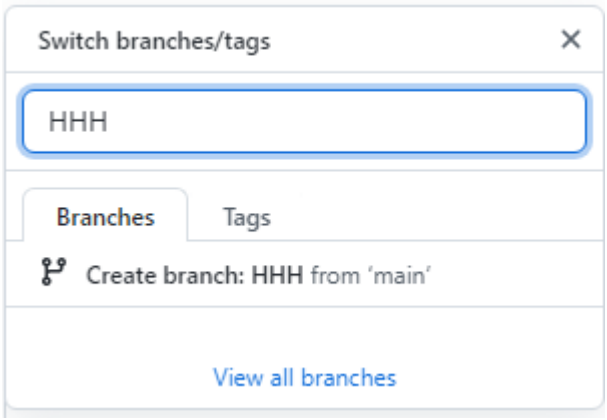

Рисунок 8 – Окно создания удаленной ветки

Ссылка: https://github.com/vegas007gof/lab3

Ответы на вопросы:

1. Что такое ветка?

Ветка в Git — это простой перемещаемый указатель на один из коммитов. По умолчанию, имя основной ветки в Git — master.

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.

С помощью команды «git branch» или же «git checkout –b» - в этом случае создается новая ветка и указатель сразу перемещается на неё.

4. Как узнать текущую ветку?

Если ввести команду git branch без параметров, то она выведет список всех веток и символом «*» пометит ветку, на которой вы находитесь.

5. Как переключаться между ветками?

С помощью команды «git checkout».

6. Что такое удаленная ветка?

Удалённые ссылки — это ссылки (указатели) в ваших удалённых репозиториях, включая ветки, теги и так далее.

7. Что такое ветка отслеживания?

Ветки слежения — 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.

8. Как создать ветку отслеживания?

С помощью команды «git checkout --track».

9. Как отправить изменения из локальной ветки в удаленную?

С помощью команды «git push <remote> <branch>»

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

Команда «git fetch» получает с сервера все изменения, которых у вас ещё нет, но не будет изменять состояние вашей рабочей директории. Команда «git pull», которая в большинстве случаев является командой «git fetch», за которой непосредственно следует команда «git merge», 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 удаленные ветки?

Удаленную ветку можно удалить с помощью команды «git push --delete», локальная ветка удаляется с помощью команды «git branch -d».

12. Изучить модель ветвления git-flow. Какие основные типы веток присуствуют в модели git-flow? Как организована работа с ветками в модели git-flow? В чем недостатки git-flow?

git-flow — это набор расширений git предоставляющий высокоуровневые операции над репозиторием для поддержки модели ветвления Vincent Driessen. В нём присутствуют такие ветки как «feature», «release» и «hotfix». Gitflow автоматизирует процессы слияния веток. Для начала использования необходимо установить gitflow и прописать команду «git flow init». Разработка новых фич начинается из ветки "develop". Для начала разработки фичи выполните: git flow feature start MYFEATURE.

Это действие создаёт новую ветку фичи, основанную на ветке "develop", и переключается на неё. Окончание разработки фичи. Это действие выполняется так:

1) Слияние ветки MYFEATURE в "develop"
2) Удаление ветки фичи
3) Переключение обратно на ветку "develop"
4) git flow feature finish MYFEATURE

Для начала работы над релизом используйте команду git flow release. Она создаёт ветку релиза, ответляя от ветки "develop": git flow release start RELEASE [BASE]

При желании вы можете указать [BASE]-коммит в виде его хеша sha-1, чтобы начать релиз с него. Этот коммит должен принадлежать ветке "develop". Желательно сразу публиковать ветку релиза после создания, чтобы позволить другим разработчиками выполнять коммиты в ветку релиза. Это делается так же, как и при публикации фичи, с помощью команды: git flow release publish RELEASE

Вы также можете отслеживать удалённый релиз с помощью команды git flow release track RELEASE

Завершение релиза — один из самых больших шагов в git-ветвлени. При этомпроисходит несколько действий:

1) Ветка релиза сливается в ветку "master"

2) Релиз помечается тегом равным его имени

3) Ветка релиза сливается обратно в ветку "develop"

4) Ветка релиза удаляется git flow release finish RELEASE

Не забудьте отправить изменения в тегах с помощью команды git push–tags.

Исправления нужны в том случае, когда нужно незамедлительно устранить нежелательное состояние продакшн-версии продукта. Она может ответвляться отсоответствующего тега на ветке "master", который отмечает выпуск продакшн- версии. Как и в случае с другими командами git flow, работа над исправлением начинается так:

git flow hotfix start VERSION [BASENAME]

Аргумент VERSION определяет имя нового, исправленного релиза. Прижелании можно указать BASENAME-коммит, от которого произойдёт
ответвление. Когда исправление готово, оно сливается обратно в ветки "develop"и "master". Кроме того, коммит в ветке "master" помечается тегом с версией исправления.

git flow hotfix finish

VERSIONНедостатки

gitflow:

1) Git Flow может замедлять работу, когда приходится ревьювить большиепулл реквесты, когда вы пытаетесь выполнить итерацию быстро.

2) Релизы сложно делать чаще, чем раз в неделю.

3) Большие функции могут потратить дни на мерж и резолв конфликтов ифорсировать несколько циклов тестирования.

4) История проекта в гите имеет кучу merge commits и затрудняет просмотрреальной работы.

5) Может быть проблематичным в CI/CD сценариях.

13. На прошлой лабораторной работе было задание выбрать одно изпрограммных средств с GUI для работы с Git. Необходимо в рамках этого вопроса привести описание инструментов для работы с ветками Git, предоставляемых этим средством.

Пример с GItKraken: для подключения удаленного репозитория в стартовом окне GitKraken выбираем последовательно Open Repo, Init, Local Only. В открывшемся окне нужно указать ссылку на удаленный репозиторий (из адреснойстроки браузера) и папку на компьютере, куда сохранятся файлы проекта. Если все сделано верно, содержимое репозитория отобразится на клиенте.