

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

Кафедра инфокоммуникаций

Основы кроссплатформенного программирования

Отчет по лабораторной работе №1.2

Исследование основных возможностей Git для работы с локальными репозиториями.

Выполнил студент группы

ИВТ-б-о-21-1 (2)

Стригалов Д.М. « » _____ 20__ г.

Подпись студента _____

Работа защищена « » _____ 20__ г.

Проверил доцент

Кафедры инфокоммуникаций, старший
преподаватель

Воронкин Р.А.

(подпись)

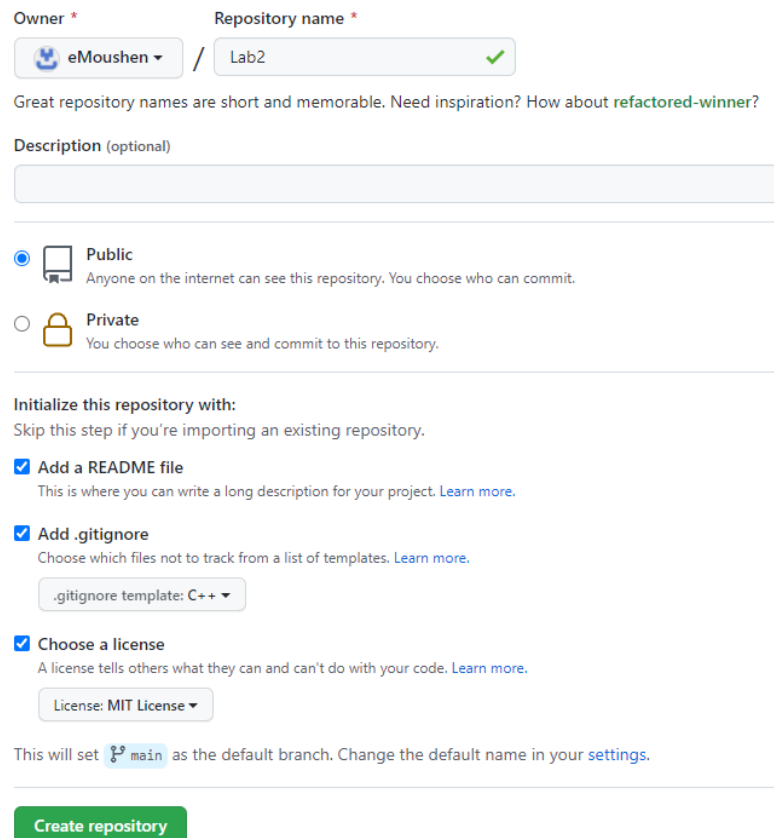
Ставрополь 2022

Тема: Исследование основных возможностей Git для работы с локальными репозиториями.

Цель работы: исследовать базовые возможности системы контроля версий Git для работы с локальными репозиториями.

Порядок выполнения работы:

1. Создание нового репозитория.



Owner * / Repository name *

Great repository names are short and memorable. Need inspiration? How about [refactored-winner?](#)

Description (optional)

☒ Public
Anyone on the internet can see this repository. You choose who can commit.

☐ Private
You choose who can see and commit to this repository.

Initialize this repository with:
Skip this step if you're importing an existing repository.

☒ Add a README file
This is where you can write a long description for your project. [Learn more.](#)

☒ Add .gitignore
Choose which files not to track from a list of templates. [Learn more.](#)

.gitignore template: C++

☒ Choose a license
A license tells others what they can and can't do with your code. [Learn more.](#)

License: MIT License

This will set `main` as the default branch. Change the default name in your [settings](#).

Create repository

Рисунок 1 – Создание репозитория

2. Выполним клонирование созданного репозитория.

```
C:\Users\User>git clone https://github.com/schacon/simplegit-progit
Cloning into 'simplegit-progit'...
remote: Enumerating objects: 13, done.
remote: Total 13 (delta 0), reused 0 (delta 0), pack-reused 13
Receiving objects: 100% (13/13), done.
Resolving deltas: 100% (3/3), done.
```

Рисунок 2 – Клонирование проекта

3. Просмотрим историю хранилища.

```
C:\Users\User\Lab2>git log
commit 113a55f148f81ab4915300a69dc96d0ba95bb9d0 (HEAD -> main,
Author: Dmitry <strigalov1991@mail.ru>
Date:   Wed Mar 16 23:04:47 2022 +0300

    Update file.

commit d9108501c786e8814134ba90f942874e283a0500
Author: eMoushen <99791335+eMoushen@users.noreply.github.com>
Date:   Wed Mar 16 22:48:33 2022 +0300

    Update README.md

commit 1745d402a91c0d73ebdcc2e972576fa56bb4ec30
Author: eMoushen <99791335+eMoushen@users.noreply.github.com>
Date:   Wed Mar 16 22:47:52 2022 +0300

    Initial commit
```

Рисунок 3 – История хранилища

4. Ограничиваем количество записей в выводе команды, используя параметр -2 для вывода только двух записей.

```
C:\Users\User\Lab2>git log -p -2
commit 113a55f148f81ab4915300a69dc96d0ba95bb9d0 (HEAD -> main,
Author: Dmitry <strigalov1991@mail.ru>
Date:   Wed Mar 16 23:04:47 2022 +0300

    Update file.

diff --git a/test.cpp b/test.cpp
new file mode 100644
index 0000000..4ed60b2
--- /dev/null
+++ b/test.cpp
@@ -0,0 +1,13 @@
+#include <iostream>
+using namespace std;
+
+int main()
+{
+    cout << "Hello, world!" << endl;
+    cout << "My name is Dmitry" << endl;
+    cout << "I am 20 years old" << endl;
+    cout << "I live in Stavropol" << endl;
+    cout << "Thank,you" << endl;
+    cout << "Bye" << endl;
+    return 0;
+}
```

Рисунок 4 – Команда git log -p -2

5. Для просмотра сокращенной статистики каждого коммита, используем опцию --stat.

```
C:\Users\User\Lab2>git log --stat
commit 113a55f148f81ab4915300a69dc96d0ba95bb9d0 (HEAD -> main,
Author: Dmitry <strigalov1991@mail.ru>
Date:   Wed Mar 16 23:04:47 2022 +0300

    Update file.

test.cpp | 13 ++++++++
1 file changed, 13 insertions(+)

commit d9108501c786e8814134ba90f942874e283a0500
Author: eMoushen <99791335+eMoushen@users.noreply.github.com>
Date:   Wed Mar 16 22:48:33 2022 +0300

    Update README.md

README.md | 2 +-
1 file changed, 1 insertion(+), 1 deletion(-)

commit 1745d402a91c0d73ebdcc2e972576fa56bb4ec30
Author: eMoushen <99791335+eMoushen@users.noreply.github.com>
Date:   Wed Mar 16 22:47:52 2022 +0300

    Initial commit

.gitignore | 32 ++++++++
LICENSE    | 21 ++++++++
README.md  | 1 +
3 files changed, 54 insertions(+)
```

Рисунок 5 – Команда git log --stat

6. Поменяем формат вывода с помощью опции --pretty. Далее с помощью опции oneline выводим каждым коммит в одну строку.

```
c:\Users\User\Lab2>git log --pretty=oneline
113a55f148f81ab4915300a69dc96d0ba95bb9d0 (HEAD -> main, origin/main, origin/HEAD) Update file.
d9108501c786e8814134ba90f942874e283a0500 Update README.md
1745d402a91c0d73ebdcc2e972576fa56bb4ec30 Initial commit
```

Рисунок 6 – Просмотр коммитов

7. Укажем формат для вывода информации с помощью опции format.

```
c:\Users\User\Lab2>git log --pretty=format:"%h - %an, %ar : %s"
113a55f - Dmitry, 10 minutes ago : Update file.
d910850 - eMoushen, 26 minutes ago : Update README.md
1745d40 - eMoushen, 27 minutes ago : Initial commit
```

Рисунок 7 – Опция format

8. Просмотр истории хранилища.

```
C:\Users\User\Lab2>git log --graph --pretty=oneline --abbrev-commit
* 3bee952 (HEAD, tag: v1.3, main) fix bag
* a8f7bfa (tag: v0.1) fix bags
* 7f6d1ba Commit message
* 113a55f (origin/main, origin/HEAD) Update file.
* d910850 Update README.md
* 1745d40 Initial commit
```

Рисунок 8 – История хранилища

9. С помощью опции --graph команды log, мы сможем увидеть небольшой граф в формате ASCII, который показывает текущую ветку и историю слияний.

```
c:\Users\User\Lab2>git log --pretty=format:"%h %s" --graph
* 113a55f Update file.
* d910850 Update README.md
* 1745d40 Initial commit
```

Рисунок 9 – Ветка и история слияния

10. Просмотрим список коммитов, сделанных за последние две недели.

```
c:\Users\User\Lab2>git log --since=2.weeks
commit 113a55f148f81ab4915300a69dc96d0ba95bb9d0 (HEAD -> main,
Author: Dmitry <strigalov1991@mail.ru>
Date:   Wed Mar 16 23:04:47 2022 +0300

    Update file.

commit d9108501c786e8814134ba90f942874e283a0500
Author: eMoushen <99791335+eMoushen@users.noreply.github.com>
Date:   Wed Mar 16 22:48:33 2022 +0300

    Update README.md

commit 1745d402a91c0d73ebdcc2e972576fa56bb4ec30
Author: eMoushen <99791335+eMoushen@users.noreply.github.com>
Date:   Wed Mar 16 22:47:52 2022 +0300
```

Рисунок 10 – Список коммитов

11. Переделаем коммит, внесём необходимые изменения, указав параметр --amend.

```

Found a swap file by the name "C:/Users/User/Lab2/.git/COMMIT_EDITMSG.swp"
  owned by: User   dated: Wed Mar 16 23:56:55 2022
  file name: ~User/Lab2/.git/COMMIT_EDITMSG
  modified: YES
  user name: User   host name: WIN-N0MCRQHPPTA
  process ID: 649
While opening file "C:/Users/User/Lab2/.git/COMMIT_EDITMSG"
  dated: Thu Mar 17 00:12:42 2022
  NEWER than swap file!

(1) Another program may be editing the same file.  If this is the case,
    be careful not to end up with two different instances of the same
    file when making changes.  Quit, or continue with caution.
(2) An edit session for this file crashed.
    If this is the case, use ":recover" or "vim -r C:/Users/User/Lab2/.git/COMMIT_EDITMSG"
    to recover the changes (see ":help recovery").
    If you did this already, delete the swap file "C:/Users/User/Lab2/.git/COMMIT_EDITMSG.swp"
    to avoid this message.

```

Рисунок 11 – Исправление коммита

12. Отменим индексирование файла test.cpp.

```

C:\Users\User\Lab2>git reset HEAD test.cpp
Unstaged changes after reset:
M   test.cpp

C:\Users\User\Lab2>git status
On branch main
Your branch is up to date with 'origin/main'.

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:   test.cpp

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

```

Рисунок 12 – Отмена индексации

13. Отменим существующие изменения.

```

C:\Users\User\Lab2>git checkout -- test.cpp

C:\Users\User\Lab2>git status
On branch main
Your branch is ahead of 'origin/main' by 1 commit.
  (use "git push" to publish your local commits)

Changes to be committed:
  (use "git restore --staged <file>..." to unstage)
        modified:   test.cpp

```

Рисунок 13 – Откат изменений

14. Просмотрим список настроенных удалённых репозиторий.

```

C:\Users\User\Lab2>git remote
origin

C:\Users\User\Lab2>git remote -v
origin https://github.com/eMoushen/Lab2.git (fetch)
origin https://github.com/eMoushen/Lab2.git (push)

```

Рисунок 14 – Просмотр удалённых репозиторий

15. Просмотр адреса для чтения и записи, привязанные к репозиторию.

```

C:\Users\User\Lab2>git remote add lab2 https://github.com/eMoushen/Lab2

C:\Users\User\Lab2>git remote -v
lab2 https://github.com/eMoushen/Lab2 (fetch)
lab2 https://github.com/eMoushen/Lab2 (push)
origin https://github.com/eMoushen/Lab2.git (fetch)
origin https://github.com/eMoushen/Lab2.git (push)

```

Рисунок 14 – Просмотр адреса

16. Получим изменения, которые есть у другого разработчика.

```
C:\Users\User\Lab2>git fetch lab2
remote: Enumerating objects: 13, done.
remote: Counting objects: 100% (13/13), done.
remote: Compressing objects: 100% (11/11), done.
remote: Total 11 (delta 6), reused 0 (delta 0), pack-reused 0
Unpacking objects: 100% (11/11), 2.98 KiB | 101.00 KiB/s, done.
From https://github.com/eMoushen/Lab2
* [new branch]      main      -> lab2/main
```

Рисунок 15 – Получение изменений

16. Получим побольше информации об одном из удалённых репозиториях.

```
C:\Users\User\Lab2>git remote show origin
* remote origin
Fetch URL: https://github.com/eMoushen/Lab2.git
Push URL: https://github.com/eMoushen/Lab2.git
HEAD branch: main
Remote branch:
  main tracked
Local branch configured for 'git pull':
  main merges with remote main
Local ref configured for 'git push':
  main pushes to main (local out of date)
```

Рисунок 16 – Получении информации

17. Переименуем удалённый репозиторий.

```
C:\Users\User\Lab2>git remote rename lab2 lb2

C:\Users\User\Lab2>git remote
lb2
origin
```

Рисунок 17 – Переименование репозитория

18. Создадим аннотированный тег в Git.

```
C:\Users\User\Lab2>git tag -a v1.4 -m "my version 1.4"

C:\Users\User\Lab2>git tag
v1.4
```

Рисунок 18 – Создание аннотированного тега

19. Посмотрим данные тега вместе с коммитом.

```
C:\Users\User\Lab2>git show v1.4
tag v1.4
Tagger: Dmitry <strigalov1991@mail.ru>
Date: Thu Mar 17 00:40:25 2022 +0300

my version 1.4

commit 7f6d1ba9239392653b072e217ee6b8d4df144fdb (HEAD -> main, tag: v1.4)
Author: Dmitry <strigalov1991@mail.ru>
Date: Thu Mar 17 00:23:20 2022 +0300

    Commit message

diff --git a/test.cpp b/test.cpp
index 4ed60b2..36c52db 100644
--- a/test.cpp
+++ b/test.cpp
@@ -9,5 +9,6 @@ int main()
     cout << "I live in Stavropol" << endl;
     cout << "Thank,you" << endl;
     cout << "Bye" << endl;
+    cout << "Bye" << endl;
     return 0;
 }
```

Рисунок 19 – Просмотр данных тега

20. Отправим тег на удалённый сервер.

```
C:\Users\User\Lab2>git push origin v1.4
Enumerating objects: 6, done.
Counting objects: 100% (6/6), done.
Delta compression using up to 12 threads
Compressing objects: 100% (4/4), done.
Writing objects: 100% (4/4), 419 bytes | 419.00 KiB/s, done.
Total 4 (delta 2), reused 0 (delta 0), pack-reused 0
remote: Resolving deltas: 100% (2/2), completed with 2 local objects.
To https://github.com/eMoushen/Lab2.git
* [new tag]          v1.4 -> v1.4
```

Рисунок 20 – Отправка тега

21. Удалим тег в локальном репозитории.

```
C:\Users\User\Lab2>git tag -d v1.4
Deleted tag 'v1.4' (was 669f6fd)
```

Рисунок 21 – Удаление тега

22. Уберем тег из внешнего репозитория.

```
C:\Users\User\Lab2>git push origin :refs/tags/v1.4
To https://github.com/eMoushen/Lab2.git
- [deleted]          v1.4
```

Рисунок 22 – Изъятие тега

23. Получим версии файлов, на которые указывает тег.

```
C:\Users\User\Lab2>git checkout v1.3
Note: switching to 'v1.3'.

You are in 'detached HEAD' state. You can look around, make experimental
changes and commit them, and you can discard any commits you make in this
state without impacting any branches by switching back to a branch.

If you want to create a new branch to retain commits you create, you may
do so (now or later) by using -c with the switch command. Example:

    git switch -c <new-branch-name>

Or undo this operation with:

    git switch -

Turn off this advice by setting config variable advice.detachedHead to false

HEAD is now at 3bee952 fix bag
M   test.cpp
```

Рисунок 23 – Получение версий файлов

24. Изучим возможность отката к заданной версии.

```
C:\Users\User\Lab2>git commit -m "Otkat"
[detached HEAD 3b91242] Otkat
1 file changed, 1 deletion(-)

C:\Users\User\Lab2>git reset --hard Head~1
HEAD is now at 3bee952 fix bag
```

Рисунок 24 – Создание коммита и откат

25. Выполним команду для получения всех изменений с репозитория GitHub, после чего отправим содержимое локального репозитория в репозиторий BitBucket с помощью команды.

```
C:\Users\User\Lab2>git push --prune https://eMoushen@bitbucket.org/emoushen/lab2.git
Enumerating objects: 30, done.
Counting objects: 100% (30/30), done.
Delta compression using up to 12 threads
Compressing objects: 100% (28/28), done.
Writing objects: 100% (30/30), 6.00 KiB | 3.00 MiB/s, done.
Total 30 (delta 14), reused 7 (delta 1), pack-reused 0
To https://bitbucket.org/emoushen/lab2.git
* [new branch]      main -> main
```

Рисунок 25 – Отправка локального репозитория

26. Зеркало успешно создано на BitBucket.

Dmitry / Lab2

Lab2

Here's where you'll find this repository's source files. To give your users an idea of what they'll find here, [add a description to your repository](#).

main Files Filter files


/			
Name	Size	Last commit	Message
 .gitignore	270 B	3 hours ago	Initial commit
 LICENSE	1.04 KB	3 hours ago	Initial commit
 README.md	77 B	3 hours ago	Update README.md
 test.cpp	320 B	1 hour ago	fix bag

Рисунок 26 – Зеркало на BitBucket

Контрольные вопросы:

1. Как выполнить историю коммитов в Git? Какие существуют дополнительные опции для просмотра истории коммитов?

Наиболее простой и в то же время мощный инструмент для этого — команда `git log`. По умолчанию, без аргументов, `git log` выводит список коммитов созданных в данном репозитории в обратном хронологическом порядке. То есть самые последние коммиты показываются первыми. Одна из опций, когда вы хотите увидеть сокращенную статистику для каждого коммита, вы можете использовать опцию `–stat`. Вторая опция (одна из самых полезных аргументов) является `-p` или `-- patch`, который показывает разницу (выводит патч), внесенную в каждый коммит. Так же вы можете ограничить количество записей в выводе команды; используйте параметр `-2` для вывода только двух записей (пример команды `git log –p -2`). Третья действительно полезная опция это `--pretty`. Она меняет формат вывода. Существует несколько встроенных вариантов отображения. Опция `oneline` выводит каждый коммит в одну строку, что может быть очень удобным если вы просматриваете большое количество коммитов. К тому же, опции `short`, `full` и `fuller` делают вывод приблизительно в том же формате, но с меньшим или большим количеством информации соответственно. Наиболее интересной опцией является `format`, которая позволяет указать формат для вывода информации. Особенно это может быть полезным, когда вы хотите

сгенерировать вывод для автоматического анализа — так как вы указываете формат явно, он не будет изменен даже после обновления Git. Для опции `git log -pretty=format` существуют различного рода опции для изменения формата отображения.

2. Как ограничить вывод при просмотре истории коммитов?

Для ограничения может использоваться функция `git log`, где `n` число записей. Также, существуют опции для ограничения вывода по времени, такие как `--since` и `--until`, они являются очень удобными. Например, следующая команда покажет список коммитов, сделанных за последние две недели: `git log --since=2.weeks` Это команда работает с большим количеством форматов — вы можете указать определенную дату вида `2008-01-15` или же относительную дату, например `2 years 1 day 3 minutes ago`. Также вы можете фильтровать список коммитов по заданным параметрам. Опция `-author` дает возможность фильтровать по автору коммита, а опция `-grep` (показывает только коммиты, сообщение которых содержит указанную строку) искать по ключевым словам в сообщении коммита.⁹ Функция `-S` показывает только коммиты, в которых изменение в коде повлекло за собой добавление или удаление указанной строки.

3. Как внести изменения в уже сделанный коммит?

Внести изменения можно с помощью команды `git commit -amend` Эта команда берёт индекс и применяет его к последнему коммиту. Если после последнего коммита не было никаких проиндексированных изменений (например, вы запустили приведённую команду сразу после предыдущего коммита), то состояние проекта будет абсолютно таким же и всё, что мы изменим, это комментарий к коммиту. Для того, чтобы внести необходимые изменения - нам нужно проиндексировать их и выполнить команду `git commit --amend`. `git commit -m 'initial commit' git add forgotten_file git commit --amend` Эффект от выполнения этой команды такой, как будто мы не выполнили предыдущий коммит, а еще раз выполнили команду `git add` и выполнили коммит.

4. Как отменить индексацию файла в Git?

Например, вы изменили два файла и хотите добавить их в разные коммиты, но случайно выполнили команду `git add *` и добавили в индекс оба. Как исключить из индекса один из них? Команда `git status` напомним вам: Прямо под текстом «Changes to be committed» говорится: используйте `git reset HEAD` для исключения из индекса.

5. Как отменить изменения в файле?

С помощью команды `git checkout --`.

6. Что такое удаленный репозиторий Git?

Удалённый репозиторий - это своего рода наше облако, в которое мы сохраняем те или иные изменения в нашей программе/коде/файлах.

7. Как выполнить просмотр удаленных репозиториях данного локального репозитория?

Для того, чтобы просмотреть список настроенных удалённых репозиториях, необходимо запустить команду `git remote`. Также можно указать ключ `-v`, чтобы просмотреть адреса для чтения и записи, привязанные к репозиторию. Пример: `git remote -v`

8. Как добавить удаленный репозиторий для данного локального репозитория?

Для того, чтобы добавить удалённый репозиторий и присвоить ему имя (shortname), просто выполните команду `git remote add`.

9. Как выполнить отправку/получение изменений с удаленного репозитория?

Если необходимо получить изменения, которые есть у Пола, но нету у вас, вы можете выполнить команду `git fetch`. Важно отметить, что команда `git fetch` забирает данные в ваш локальный репозиторий, но не сливает их с какими-либо вашими наработками и не модифицирует то, над чем вы работаете в данный момент. Вам необходимо вручную слить эти данные с вашими, когда вы будете готовы. Если ветка настроена на отслеживание

удалённой ветки, то вы можете использовать команду `git pull` чтобы автоматически получить изменения из удалённой ветки и слить их со своей текущей. Выполнение `git pull`, как правило, извлекает (`fetch`) данные с сервера, с которого вы изначально¹¹ клонировали, и автоматически пытается слить (`merge`) их с кодом, над которым вы в данный момент работаете. Чтобы отправить изменения на удалённый репозиторий необходимо отправить их в удалённый репозиторий. Команда для этого действия простая: `git push` .

10. Как выполнить просмотр удаленного репозитория?

Для просмотра удалённого репозитория, можно использовать команду `git remote show` .

11. Каково назначение тэгов Git?

Теги - это ссылки указывающие на определённые версии кода/написанной программы. Они удобно чтобы в случае чего вернуться к нужному моменту. Также при помощи тегов можно помечать важные моменты.

12. Как осуществляется работа с тэгами Git?

Просмотреть наличие тегов можно с помощью команды: `git tag`. А назначить (указать, добавить тег) можно с помощью команды `git tag -a v1.4(версия изначальная) -m "Название"`. С помощью команды `git show` вы можете посмотреть данные тега вместе с коммитом: `git show v1.4`. Отправка тегов, по умолчанию, команда `git push` не отправляет теги на удалённые сервера. После создания теги нужно отправлять явно на удалённый сервер. Процесс аналогичен отправке веток — достаточно выполнить команду `git push origin` . Для отправки всех тегов можно использовать команду `git push origin tags`. Для удаления тега в локальном репозитории достаточно выполнить команду `git tag -d` . Например, удалить созданный ранее легковесный тег можно следующим образом: `git tag -d v1.4-lw` Для удаления тега из внешнего репозитория используется команда `git push origin -delete` . Если вы хотите получить версии файлов, на которые указывает тег, то вы можете сделать `git checkout` для тега пример: `git checkout -b version2 v2.0.0`.

13. Самостоятельно изучите назначение флага `--prune` в командах `git fetch` и `git push`. Каково назначение этого флага?

`Git fetch --prune` команда получения всех изменений с репозитория GitHub. В команде `git push --prune` удаляет удаленные ветки, у которых нет локального аналога. Вывод: исследовал базовые возможности системы контроля версий `git` для работы с локальными репозиториями. Также, благодаря созданию тегов и пункту 7 лабораторной работы после изменения файлов освоил возможность отката к заданной версии.

Вывод: исследовал базовые возможности системы контроля версий `git` для работы с локальными репозиториями. Освоил возможность отката измененных файлов.