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 №2
дисциплины «Программирование на Python»

Выполнил:
Гайчук Дарья Дмитриевна
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: «Исследование возможностей Git для работы с локальными репозиториями».

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

Теоретические сведения

После того, как вы создали несколько коммитов или же клонировали репозиторий с уже существующей историей коммитов, вероятно Вам понадобится возможность посмотреть, что было сделано — историю коммитов. Одним из основных и наиболее мощных инструментов для этого является команда git log.

По умолчанию (без аргументов) git log перечисляет коммиты, сделанные в репозитории в обратном к хронологическому порядке — последние коммиты находятся вверху. Из примера можно увидеть, что данная команда перечисляет коммиты с их SHA-1 контрольными суммами, именем и электронной почтой автора, датой создания и сообщением коммита. Команда git log имеет очень большое количество опций для поиска коммитов по разным критериям. Рассмотрим наиболее популярные из них.

Одним 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.

Следующей действительно полезной опцией является --pretty. Эта опция 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.

Порядок выполнения работы

1. Создать общедоступный репозиторий на GitHub, в котором будет использована лицензия MIT и выбранный Вами язык программирования.

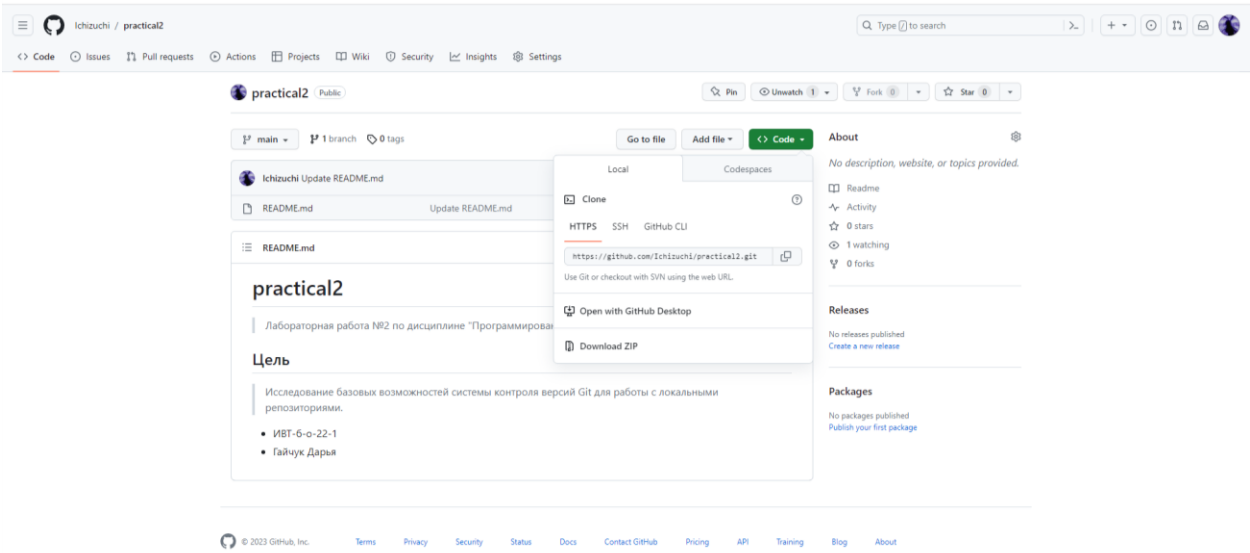

Рисунок 1. Создание общедоступного репозитория

2. Проработайте примеры лабораторной работы. Отразите вывод на консоли при выполнении команд git в отчете для лабораторной работы.

Просмотр истории коммитов

```
@Ichizuchi →/workspaces/practical2 (main) $  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. Команда «git clone»

```
@Ichizuchi →/workspaces/practical2 (main) $  git log
commit dd143ebe041b3e7024701eb19a9adae908e5704b (HEAD -> main, origin/main, origin/HEAD)
Author: Ichizuchi <144344807+Ichizuchi@users.noreply.github.com>
Date:   Mon Sep 11 11:33:29 2023 +0300

    Update README.md

commit 90f1b0dc46d77b1ec42dec13229fd07b2f9221b6
Author: Ichizuchi <144344807+Ichizuchi@users.noreply.github.com>
Date:   Mon Sep 11 11:28:10 2023 +0300

    Initial commit
@Ichizuchi →/workspaces/practical2 (main) $
```

Рисунок 3. Команда «git log»

```
@Ichizuchi ➜/workspaces/practical2 (main) $  git log -p -2
commit dd143ebe041b3e7024701eb19a9adae908e5704b (HEAD -> main, origin/main, origin/HEAD)
Author: Ichizuchi <144344807+Ichizuchi@users.noreply.github.com>
Date:   Mon Sep 11 11:33:29 2023 +0300

    Update README.md

diff --git a/README.md b/README.md
index ff37ebd..007ff7f 100644
--- a/README.md
+++ b/README.md
@@ -1 +1,6 @@
-# practical2
\ No newline at end of file
+# practical2
+>Лабораторная работа №2 по дисциплине "Программирование на Python"
+## Цель
+> Исследование базовых возможностей системы контроля версий Git для работы с локальными репозиториями.
+- ИВТ-б-о-22-1
+- Гайчук Дарья

commit 90f1b0dc46d77b1ec42dec13229fd07b2f9221b6
Author: Ichizuchi <144344807+Ichizuchi@users.noreply.github.com>
Date:   Mon Sep 11 11:28:10 2023 +0300

:...skipping...
commit dd143ebe041b3e7024701eb19a9adae908e5704b (HEAD -> main, origin/main, origin/HEAD)
Author: Ichizuchi <144344807+Ichizuchi@users.noreply.github.com>
Date:   Mon Sep 11 11:33:29 2023 +0300
```

Рисунок 4. Команда «git log -p -2»

```
commit dd143ebe041b3e7024701eb19a9adae908e5704b (HEAD -> main, origin/main, origin/HEAD)
Author: Ichizuchi <144344807+Ichizuchi@users.noreply.github.com>
Date:   Mon Sep 11 11:33:29 2023 +0300

    Update README.md

diff --git a/README.md b/README.md
index ff37ebd..007ff7f 100644
--- a/README.md
+++ b/README.md
@@ -1 +1,6 @@
-# practical2
\ No newline at end of file
+# practical2
+>Лабораторная работа №2 по дисциплине "Программирование на Python"
+## Цель
+> Исследование базовых возможностей системы контроля версий Git для работы с локальными репозиториями.
+- ИВТ-б-о-22-1
+- Гайчук Дарья

commit 90f1b0dc46d77b1ec42dec13229fd07b2f9221b6
Author: Ichizuchi <144344807+Ichizuchi@users.noreply.github.com>
Date:   Mon Sep 11 11:28:10 2023 +0300

    Initial commit

diff --git a/README.md b/README.md
new file mode 100644
index 0000000..ff37ebd
```

Рисунок 5. Продолжение команды «git log -p -2»

```
@Ichizuchi ➜/workspaces/practical2 (main) $ git log --stat
commit dd143ebe041b3e7024701eb19a9adae908e5704b (HEAD -> main, origin/main, origin/HEAD)
Author: Ichizuchi <144344807+Ichizuchi@users.noreply.github.com>
Date:   Mon Sep 11 11:33:29 2023 +0300

    Update README.md

 README.md | 7 ++++++-
 1 file changed, 6 insertions(+), 1 deletion(-)

commit 90f1b0dc46d77b1ec42dec13229fd07b2f9221b6
Author: Ichizuchi <144344807+Ichizuchi@users.noreply.github.com>
Date:   Mon Sep 11 11:28:10 2023 +0300

    Initial commit

 README.md | 1 +
 1 file changed, 1 insertion(+)
@Ichizuchi ➜/workspaces/practical2 (main) $
```

Рисунок 6. Команда «git log --stat»

```
@Ichizuchi →/workspaces/practical2 (main) $ git log --pretty=oneline
dd143ebe041b3e7024701eb19a9adae908e5704b (HEAD -> main, origin/main, origin/HEAD) Update README.md
90f1b0dc46d77b1ec42dec13229fd07b2f9221b6 Initial commit
@Ichizuchi →/workspaces/practical2 (main) $
```

Рисунок 7. Команда «git log --pretty=online»

```
@Ichizuchi →/workspaces/practical2 (main) $ git log --pretty=format:"%h - %an, %ar : %s"
dd143eb - Ichizuchi, 30 minutes ago : Update README.md
90f1b0d - Ichizuchi, 36 minutes ago : Initial commit
@Ichizuchi →/workspaces/practical2 (main) $
```

Рисунок 8. Команда «git log --pretty=format:"%h - %an, %ar : %s"»

```
@Ichizuchi →/workspaces/practical2 (main) $ git log --pretty=format:"%h %s" --graph
* dd143eb Update README.md
* 90f1b0d Initial commit
```

Рисунок 9. Команда «git log --pretty=format:"%h %s" –graph»

Ограничение вывода

```
@Ichizuchi →/workspaces/practical2 (main) $  git log --since=2.weeks
commit dd143ebe041b3e7024701eb19a9adae908e5704b (HEAD -> main, origin/main, origin/HEAD)
Author: Ichizuchi <144344807+Ichizuchi@users.noreply.github.com>
Date:   Mon Sep 11 11:33:29 2023 +0300

    Update README.md

commit 90f1b0dc46d77b1ec42dec13229fd07b2f9221b6
Author: Ichizuchi <144344807+Ichizuchi@users.noreply.github.com>
Date:   Mon Sep 11 11:28:10 2023 +0300

    Initial commit
```

Рисунок 10.

Операции отмены

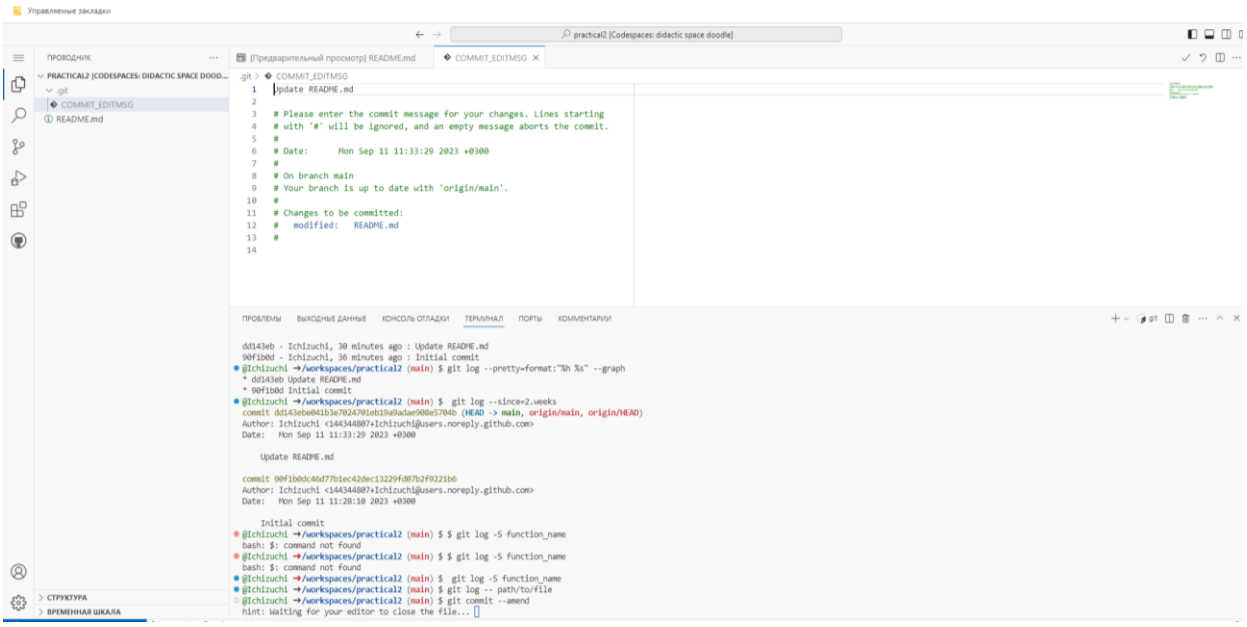

Рисунок 11.

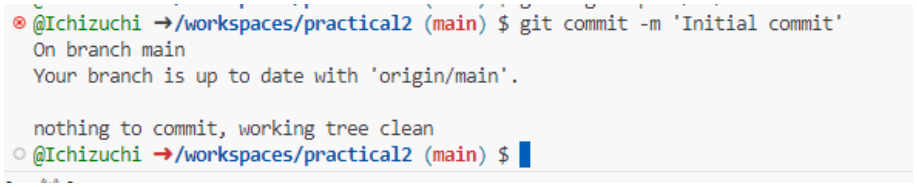

Рисунок 12. Команда «git commit -m "…" –amend»

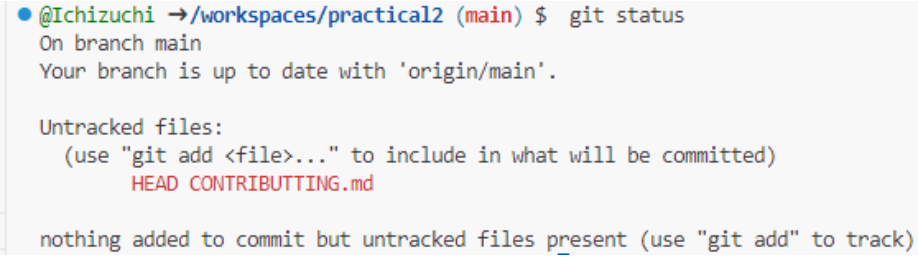

Рисунок 13. Команда «git status»

Просмотр удалённых репозиториев

Рисунок 14. Команда «cd ticgit»

Рисунок 15. Команда «git remote -v»

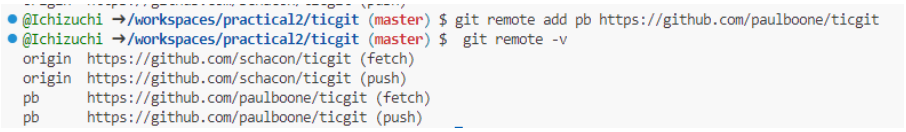

Рисунок 16. Команды «git remote add» и «git remote -v»

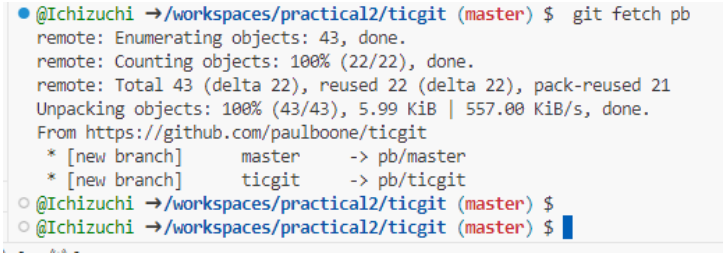

Рисунок 17. Команда «git fetch»

Просмотр удаленного репозитория

```
@Ichizuchi →/workspaces/practical2/ticgit (master) $ git remote show origin
* remote origin
  Fetch URL: https://github.com/schacon/ticgit
  Push  URL: https://github.com/schacon/ticgit
  HEAD branch: master
  Remote branches:
    master tracked
    ticgit tracked
  Local branch configured for 'git pull':
    master merges with remote master
  Local ref configured for 'git push':
    master pushes to master (up to date)
```

Рисунок 18. Команда «git remote show origin»

Удаление и переименование удалённых репозиториев

```
@Ichizuchi →/workspaces/practical2/ticgit (master) $ git remote rename pb paul
Renaming remote references: 100% (2/2), done.
@Ichizuchi →/workspaces/practical2/ticgit (master) $ git remote
origin
paul
```

Рисунок 19. Команда «git remote rename»

Работа с тегами

```
@Ichizuchi →/workspaces/practical2 (main) $ git tag v1.4
@Ichizuchi →/workspaces/practical2 (main) $ git show v1.4
commit dd143ebe041b3e7024701eb19a9adae908e5704b (HEAD -> main, tag: v1.4, origin/main, origin/HEAD)
Author: Ichizuchi <144344807+Ichizuchi@users.noreply.github.com>
Date:   Mon Sep 11 11:33:29 2023 +0300

    Update README.md

diff --git a/README.md b/README.md
index ff37ebd..007ff7f 100644
--- a/README.md
+++ b/README.md
@@ -1 +1,6 @@
-# practical2
\ No newline at end of file
+# practical2
+>Лабораторная работа №2 по дисциплине "Программирование на Python"
+## Цель
+> Исследование базовых возможностей системы контроля версий Git для работы с локальными репозиториями.
+- ИВТ-б-о-22-1
+- Гайчук Дарья
```

Рисунок 20. Команды «git tag» и «git show»

```
@Ichizuchi →/workspaces/practical2 (main) $ git push origin v1.4
Total 0 (delta 0), reused 0 (delta 0), pack-reused 0
To https://github.com/Ichizuchi/practical2
 * [new tag]         v1.4 -> v1.4
```

Рисунок 21. Команда «git push origin»

```
@Ichizuchi →/workspaces/practical2 (main) $ git tag -d v1.4
Deleted tag 'v1.4' (was dd143eb)
```

Рисунок 22. Команда «git tag -d».

```
@Ichizuchi →/workspaces/practical2 (main) $ git push origin :refs/tags/v1.4
To https://github.com/Ichizuchi/practical2
 - [deleted]         v1.4
```

Рисунок 23. Команда «git push origin –tags».

```
@Ichizuchi →/workspaces/practical2 (main) $ git checkout v2.0.0
Note: switching to 'v2.0.0'.

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 dd143eb Update README.md
```

Рисунок 24. Команда «git checkout»

```
@Ichizuchi →/workspaces/practical2 (dd143eb) $ git checkout -b version2 v2.0.0
Switched to a new branch 'version2'
```

Рисунок 25. Команда «git checkout -b»

Контрольные вопросы

1. За просмотр истории коммитов отвечает команда git log. В сочетании с различными параметрами эта команда выводит историю по-разному. git log -p, расширенный вывод истории, git log --oneline, короткая запись, git log --stat --graph, история в виде дерева

2. Кроме опций для форматирования вывода, git log имеет ряд полезных ограничительных параметров, то есть параметров, которые дают возможность отобразить часть коммитов. Параметр -2, который отображает только два последних коммита. На самом деле, вы можете задать -, где n это количество отображаемых коммитов. А вот параметры, ограничивающие по времени, такие как --since и --until, весьма полезны. Например, следующая команда выдаёт список коммитов, сделанных за последние две недели: $ git log --since=2.weeks

3. Если вы хотите изменить только сообщение вашего последнего коммита, это очень просто: $ git commit --amend Эта команда откроет в вашем текстовом редакторе сообщение вашего последнего коммита, для того, чтобы вы могли его исправить. Когда вы сохраните его и закроете редактор, будет

создан новый коммит, содержащий это сообщение, который теперь и будет вашим последним коммитом. Если вы создали коммит и затем хотите изменить зафиксированный снимок, добавив или изменив файлы (возможно, вы забыли добавить вновь созданный файл, когда совершали изначальный коммит), то процесс выглядит в основном так же. Вы добавляете в индекс необходимые изменения, редактируя файл и выполняя для него git add или git rm для отслеживаемого файла, а последующая команда git commit --amend берет вашу текущую область подготовленных изменений и делает её снимок для нового коммита.

4. Отмена может потребоваться, если вы сделали коммит слишком рано, например, забыв добавить какие-то файлы или комментарий к коммиту. Если вы хотите переделать коммит — внесите необходимые изменения, добавьте их в индекс и сделайте коммит ещё раз, указав параметр --amend: $ git commit --amend

5. Прямо под текстом «Changes to be committed» говорится: используйте git reset HEAD … для исключения из индекса. $ git reset HEAD CONTRIBUTING.md Unstaged changes after reset: M CONTRIBUTING.md $ git status On branch master Changes to be committed: (use "git reset HEAD ..." to unstage) renamed: README.md -> README Changes not staged for commit: (use "git add ..." to update what will be committed) (use "git checkout -- ..." to discard changes in working directory) modified: CONTRIBUTING.md

6. Распределённый репозиторий—это, фактически, удалённая копия вашей программы, которую вы делаете с помощью git (или другой системы контроля версий) на удалённом сервере, доступном через интернет. После этого все, кому вы дадите доступ (в т.ч. и вы сами), смогут скачивать вашу программу, вносить в неё изменения и загружать их на удалённый сервер с помощью всё того же git.

7. Для того, чтобы просмотреть список настроенных удалённых репозиториев, вы можете запустить команду git remote. Она выведет названия доступных удалённых репозиториев. Если вы клонировали репозиторий, то

увидите, как минимум origin — имя по умолчанию, которое Git даёт серверу, с которого производилось клонирование 8. Чтобы добавить новый удаленный репозиторий, выполните команду git remote add в терминале в каталоге, в котором хранится репозиторий. Команда git remote add принимает два аргумента: имя удаленного репозитория, например, origin; URL-адрес удаленного репозитория,

9. $ git fetch [remote-name] Данная команда связывается с указанным удалённым проектом и забирает все те данные проекта, которых у вас ещё нет. После того как вы выполнили команду, у вас должны появиться ссылки на все ветки из этого удалённого проекта, которые вы можете просмотреть или слить в любой момент. Когда вы хотите поделиться своими наработками, вам необходимо отправить их в удалённый репозиторий. Команда для этого действия простая: git push . Чтобы отправить вашу ветку master на сервер origin (повторимся, что клонирование обычно настраивает оба этих имени автоматически), вы можете выполнить следующую команду для отправки ваших коммитов: $ git push origin master

10. Для того, чтобы просмотреть список настроенных удалённых репозиториев, вы можете запустить команду git remote. Она выведет названия доступных удалённых репозиториев. Если вы клонировали репозиторий, то увидите, как минимум origin — имя по умолчанию, которое Git даёт серверу, с которого производилось клонирование:

11. «Git» теги — это метки, которыми можно помечать коммиты в истории коммитов, хранящихся в Git-репозитории. Часто тегами помечают коммиты, после которых выходит очередной релиз компьютерной программы, если речь идет о проекте написания некой компьютерной программы.

12. Git использует два основных типа тегов: легковесные и аннотированные. Создание аннотированного тега в Git выполняется легко. Самый простой способ — это указать -a при выполнении команды tag: Легковесный тег — это ещё один способ пометить коммит. По сути, это контрольная сумма коммита, сохранённая в файл — больше никакой

информации не хранится. Для создания легковесного тега не передавайте опций -a, -s и -m, укажите только название:

13. Это запускает git fsck --unreachable, используя все ссылки, доступные brefs/, необязательно с дополнительным набором объектов, указанных в командной строке, и удаляет все распакованные объекты, недоступные для любого из этих головных объектов, из базы данных объектов. Кроме того, он удаляет распакованные объекты, которые также находятся в пакетах, запустив git prune-packed . Он также удаляет записи из .git/shallow, которые недоступны ни по одной ссылке.

Вывод: исследовала базовые возможности системы контроля версий Git для работы с локальными репозиториями.