



МИНОБРНАУКИ РОССИИ

*Федеральное государственное бюджетное образовательное учреждение
высшего образования*

«МИРЭА — Российский технологический университет»

РТУ МИРЭА

Институт Информационных технологий (ИТ)

Кафедра Математического обеспечения и стандартизации
информационных технологий (МОСИТ)

**Отчет по практической работе №1
по дисциплине**

«Технология разработки программных приложений»

Тема: «Основные команды Git»

Выполнил студент группы ИКБО-06-21 Бондарь А.Р.

Принял Туманова М.Б.

Практическая работа выполнена «_»_____2023г. *(подпись студента)*

«Зачтено» «_»_____2023 г. *(подпись руководителя)*

Москва 2023

Содержание

Цель практической работы	4
1 Основные команды Git	5
1.1 Установка и настройка клиент git	5
1.1.1 Установка git	5
1.1.2 Настройка git	5
1.2 Созданиe локальнoй репозиторий и добавление в него несколько файлов	7
1.3 Добавление файлов в репозиторий	7
1.4 Изменение файла	8
1.5 Коммит	8
1.6 Как работает индексация	9
1.7 История коммитов	10
1.8 Отмена изменений	11
1.9 Создание тега	12
1.9.1 Аннотированные теги	12
1.9.2 Легковесный теги	13
1.10 Отмена изменений в индексе	13
1.10.1 Отмена изменений (до индексации)	13
1.10.2 Отмена изменений (после индексации)	14
1.11 Отмена коммита	14

2	Системы управления репозиториями	15
2.1	Создание репозитория на GitHub и на локальной машине . .	15
2.1.1	Создание репозитория на GitHub	15
2.2	Создание репозитория на локальной машине	15
2.3	Создание SSH-ключа	17
2.4	Связывание локального и GitHub репозитория	17
2.5	Создание и слияние новой ветки	19
2.6	Задание варианта	19
3	Работа с ветвлением и оформление кода	23
3.1	Форк репозитория	23
3.2	Клонирование форка на локальную машину	24
3.3	Создание двух веток	24
3.4	Коммиты	24
3.5	Слияние веток	26
3.6	Отправка всех изменений	26

Цель практической работы

Получить навыки по работе с командной строкой и git'ом.

Глава 1

Основные команды Git

1.1 Установка и настройка клиент git

1.1.1 Установка git

Установка в Linux и Unix

- Используйте обычный менеджер пакетов вашего дистрибутива. Откройте терминал и введите подходящие команды.
- Если у вас 21 или более ранняя версия Fedora, используйте `yum install git`.
- Для 22 и последующих версий Fedora вводите `dnf install git`.
- Для дистрибутивов, основанных на Debian, например, Ubuntu, используйте `apt-get`: `sudo apt-get install git`.

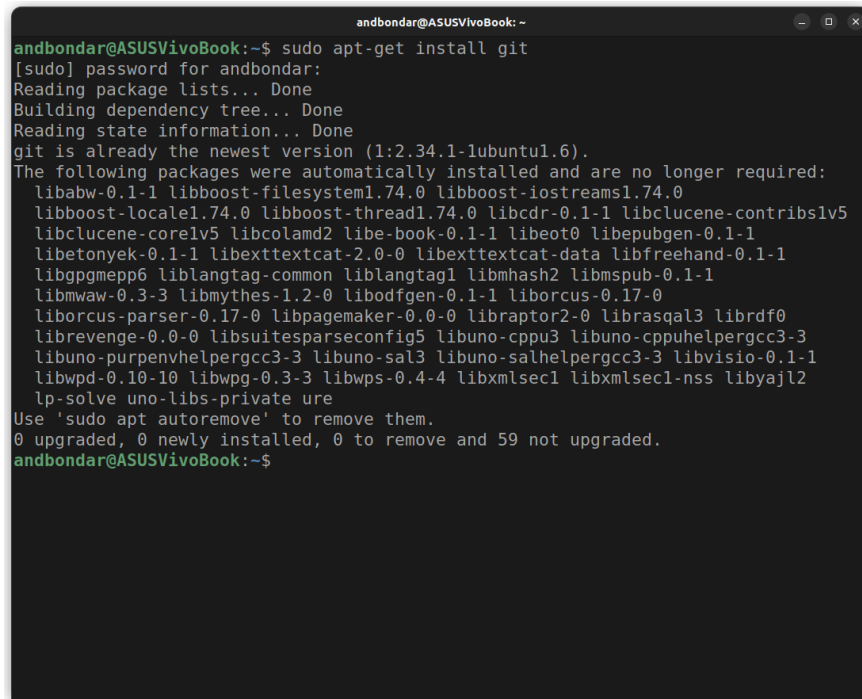
1.1.2 Настройка git

1. Открываем терминал.
2. Необходимо выполнить следующие команды:

```
git config --global user.name "Your Name"
```

```
git config --global user.email "your_email@whatever.com"
```

3. Необходимо выполнить следующие команды:



```
andbondar@ASUSVivoBook: ~  
andbondar@ASUSVivoBook:~$ sudo apt-get install git  
[sudo] password for andbondar:  
Reading package lists... Done  
Building dependency tree... Done  
Reading state information... Done  
git is already the newest version (1:2.34.1-1ubuntu1.6).  
The following packages were automatically installed and are no longer required:  
  libabw-0.1-1 libboost-filesystem1.74.0 libboost-iostreams1.74.0  
  libboost-locale1.74.0 libboost-thread1.74.0 libcdr-0.1-1 libclucene-contribs1v5  
  libclucene-core1v5 libcolamd2 libe-book-0.1-1 libeot0 libepubgen-0.1-1  
  libetonyek-0.1-1 libexttextcat-2.0-0 libexttextcat-data libfreehand-0.1-1  
  libgpgmepp6 liblangtag-common liblangtag1 libmhash2 libmspub-0.1-1  
  libmwaw-0.3-3 libmythes-1.2-0 libodfgen-0.1-1 liborcus-0.17-0  
  liborcus-parser-0.17-0 libpagemaker-0.0-0 libraptor2-0 librasqal3 librdf0  
  librevenge-0.0-0 libsuitesparseconfig5 libuno-cppu3 libuno-cppuhelpergcc3-3  
  libuno-purpenvhelpergcc3-3 libuno-sal3 libuno-salhelpergcc3-3 libvisio-0.1-1  
  libwpd-0.10-10 libwpg-0.3-3 libwps-0.4-4 libxmlsec1 libxmlsec1-nss libyajl2  
  lp-solve uno-libs-private ure  
Use 'sudo apt autoremove' to remove them.  
0 upgraded, 0 newly installed, 0 to remove and 59 not upgraded.  
andbondar@ASUSVivoBook:~$
```

Рисунок 1.1. Установка git

```
git config --global core.autocrlf input
```

```
git config --global core.safecrlf warn
```

4. Необходимо выполнить следующие команды:

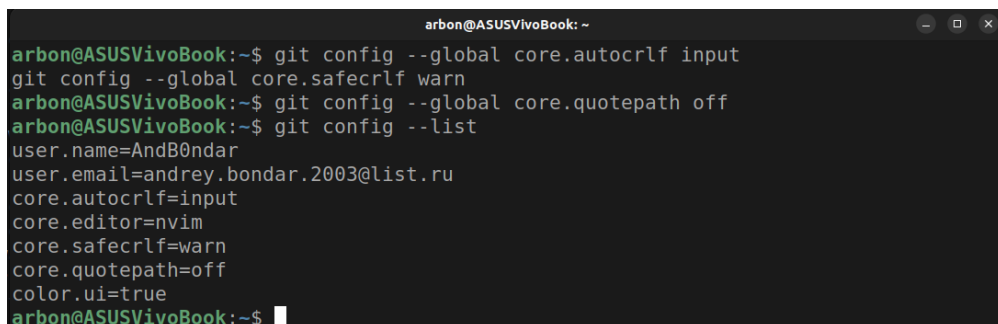
```
git config --global core.quotepath off
```

5. Необходимо выполнить следующие команды:

```
git config --global core.quotepath off
```

Для проверки верности всех введенных команд, введите:

```
git config --list
```

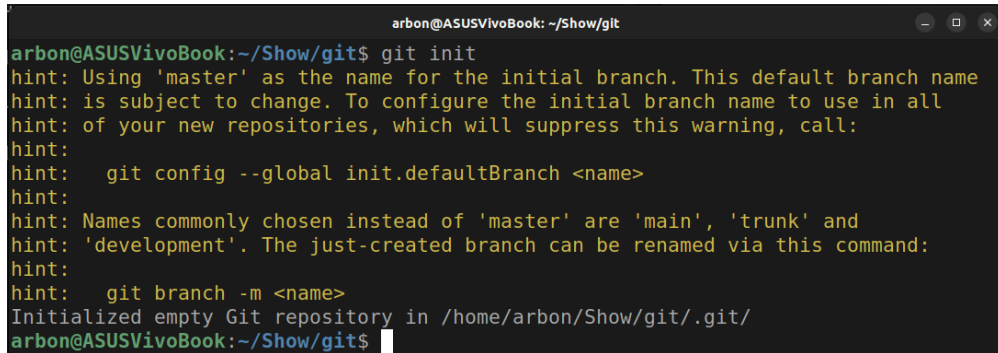


```
arbon@ASUSVivoBook: ~  
arbon@ASUSVivoBook:~$ git config --global core.autocrlf input  
git config --global core.safecrlf warn  
arbon@ASUSVivoBook:~$ git config --global core.quotepath off  
arbon@ASUSVivoBook:~$ git config --list  
user.name=AndB0ndar  
user.email=andrey.bondar.2003@list.ru  
core.autocrlf=input  
core.editor=nvim  
core.safecrlf=warn  
core.quotepath=off  
color.ui=true  
arbon@ASUSVivoBook:~$
```

Рисунок 1.2. Настройка git

1.2 Создарили локальный репозиторий и добавление в него несколько файлов

Выполняем команду `git init`. После выполнения данной команды, должно высветиться данное сообщение, показанное на рис. 1.3.



```
arbon@ASUSVivoBook: ~/Show/git
arbon@ASUSVivoBook:~/Show/git$ git init
hint: Using 'master' as the name for the initial branch. This default branch name
hint: is subject to change. To configure the initial branch name to use in all
hint: of your new repositories, which will suppress this warning, call:
hint:
hint:   git config --global init.defaultBranch <name>
hint:
hint: Names commonly chosen instead of 'master' are 'main', 'trunk' and
hint: 'development'. The just-created branch can be renamed via this command:
hint:
hint:   git branch -m <name>
Initialized empty Git repository in /home/arbon/Show/git/.git/
arbon@ASUSVivoBook:~/Show/git$
```

Рисунок 1.3. Создание репозитория git

1.3 Добавление файлов в репозиторий

Далее, командой `touch file` создадим несколько текстовый файлов. Чтобы добавить файл в репозиторий необходимо выполнить следующее:


1. Вводим команды:

```
git add <Название вашего файла>
```

```
git commit -m "Ваш текст для коммита"
```

2. Чтобы проверить состояние репозитория, выполним команду: `git status`. Команда проверки состояния сообщит, что коммитить нечего. Это означает, что в репозитории хранится текущее состояние рабочего каталога, и нет никаких изменений, ожидающих записи.

Результат этих действий показан на рисунке 1.4.

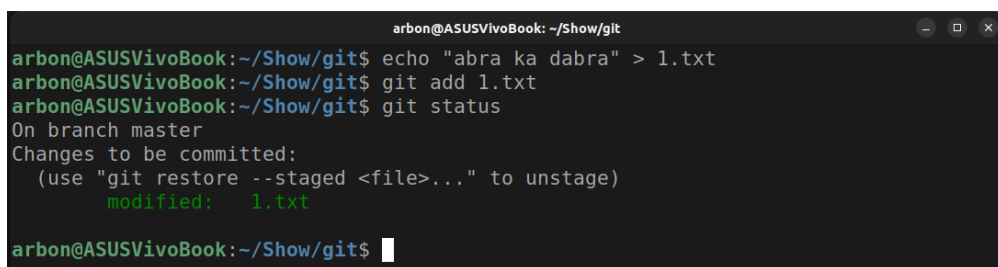


```
arbon@ASUSVivoBook: ~/Show/git
arbon@ASUSVivoBook:~/Show/git$ git add 1.txt 2.txt 3.txt
arbon@ASUSVivoBook:~/Show/git$ git commit -m "first commit"
[master (root-commit) 91e6554] first commit
3 files changed, 0 insertions(+), 0 deletions(-)
create mode 100644 1.txt
create mode 100644 2.txt
create mode 100644 3.txt
arbon@ASUSVivoBook:~/Show/git$
```

Рисунок 1.4. Добавление файлов в репозиторий и первый коммит

1.4 Изменение файла

Далее внесем изменения в один из файлов. Путем перенаправления вывода команды `echo` в файл. Добавим изменения в индекс командой: `git add <имя файла>`. Чтобы проверить внесение команды в индекс, `git` предоставляет команду `git status`. Все эти действия отображены на рисунке 1.5.



```
arbon@ASUSVivoBook: ~/Show/git
arbon@ASUSVivoBook:~/Show/git$ echo "abra ka dabra" > 1.txt
arbon@ASUSVivoBook:~/Show/git$ git add 1.txt
arbon@ASUSVivoBook:~/Show/git$ git status
On branch master
Changes to be committed:
  (use "git restore --staged <file>..." to unstage)
        modified:   1.txt
arbon@ASUSVivoBook:~/Show/git$
```

Рисунок 1.5. Индексирование изменений и их проверка

1.5 Коммит

Теперь, чтобы сохранить изменения в репозитории, необходимо сделать коммит. Для этого существует команда `git commit -m "текст коммита"` (см. рисунок 1.6). Флаг `-m` позволяет сразу добавить комментарий при вводе команды, без него `git`, для комментария, откроет текстовый редактор, установленный по умолчанию, обычно это `vim`.


```
arbon@ASUSVivoBook: ~/Show/git
arbon@ASUSVivoBook:~/Show/git$ git commit -m "Изменение файла 1.txt"
[master 4ae5fa5] Изменение файла 1.txt
1 file changed, 1 insertion(+)
arbon@ASUSVivoBook:~/Show/git$
```

Рисунок 1.6. Коммит

1.6 Как работает индексация

Выполним подряд следующие шаги:

1. Изменим еще один файл.
2. Добавим это изменение в индекс git.
3. Изменим файл еще раз.
4. Проверим состояние и произведем коммит проиндексированного изменения.
5. Добавим второе изменение в индекс, а затем проверим состояние с помощью команды `git status`.
6. Сделаем коммит второго изменения.

Результаты проделанных шагов проиллюстрированы на рисунках 1.7-1.8.

```
arbon@ASUSVivoBook: ~/Show/git
arbon@ASUSVivoBook:~/Show/git$ echo "kada bra bra" > 2.txt
arbon@ASUSVivoBook:~/Show/git$ git add 2.txt
arbon@ASUSVivoBook:~/Show/git$ echo "trulala la lala" >> 2.txt
arbon@ASUSVivoBook:~/Show/git$ git status
On branch master
Changes to be committed:
  (use "git restore --staged <file>..." to unstage)
        modified:   2.txt

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:   2.txt

arbon@ASUSVivoBook:~/Show/git$ git commit "Изменение 2.txt"
error: pathspec 'Изменение 2.txt' did not match any file(s) known to git
arbon@ASUSVivoBook:~/Show/git$ git commit -m "Изменение 2.txt"
[master e159314] Изменение 2.txt
1 file changed, 1 insertion(+)
arbon@ASUSVivoBook:~/Show/git$
```

Рисунок 1.7. Коммит первого проиндексированного изменения

```
arbon@ASUSVivoBook: ~/Show/git
arbon@ASUSVivoBook:~/Show/git$ git add 2.txt
arbon@ASUSVivoBook:~/Show/git$ git status
On branch master
Changes to be committed:
  (use "git restore --staged <file>..." to unstage)
        modified:   2.txt

arbon@ASUSVivoBook:~/Show/git$ git commit -m "Второе изменение 2.txt"
[master 1c00310] Второе изменение 2.txt
 1 file changed, 1 insertion(+)
arbon@ASUSVivoBook:~/Show/git$
```

Рисунок 1.8. Коммит второго проиндексированного изменения

1.7 История коммитов

Для просмотра истории коммитов используется команда: `git log` (Рисунок 1.9).

```
arbon@ASUSVivoBook: ~/Show/git
arbon@ASUSVivoBook:~/Show/git$ git log
commit 1c003103c13222dac1bb1bf9dc13f1aab34b27ab (HEAD -> master)
Author: AndB0ndar <andrey.bondar.2003 at list dot ru>
Date:   Sat Feb 18 20:14:25 2023 +0300

    Второе изменение 2.txt

commit e159314d7c64d7a3359dfc76f67fd161d9ab53f5
Author: AndB0ndar <andrey.bondar.2003 at list dot ru>
Date:   Sat Feb 18 20:09:43 2023 +0300

    Изменение 2.txt

commit 4ae5fa55df9588a30ac3140817476d8f3d6dcbc8
Author: AndB0ndar <andrey.bondar.2003 at list dot ru>
Date:   Sat Feb 18 19:51:29 2023 +0300

    Изменение файла 1.txt

commit 9bebabb5ae1ed4f836c5a2ab75ac4fbal1eeb8d0
Author: AndB0ndar <andrey.bondar.2003 at list dot ru>
Date:   Sat Feb 18 19:33:02 2023 +0300

    add proekt.html

commit 91e6554eb129f13245b99ec09405c2e61c0d9e7f
Author: AndB0ndar <andrey.bondar.2003 at list dot ru>
Date:   Tue Feb 14 11:32:16 2023 +0300

    first commit
arbon@ASUSVivoBook:~/Show/git$
```

Рисунок 1.9. История коммитов

Для уточнения формата лога используется флаг `--pretty=format:"..."`.

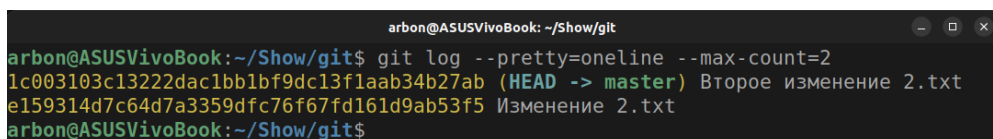
Для примера выполните команду:

```
git log --pretty=format:"%h %ad | %s%d [%an]" --graph --date=short
```

Рассмотрим её в деталях:

- %h — укороченный хэш коммита
- %d — дополнения коммита («ГОЛОВЫ» веток или теги)
- %ad — дата коммита
- %s — комментарий
- %an — имя автора
- --graph — отображает дерево коммитов в виде ASCII-графика
- --date=short — сохраняет формат даты коротким

Приведем пирмер одного из форматирования вывода истории (см. рис. 1.10).



```

arbon@ASUSVivoBook: ~/Show/git
arbon@ASUSVivoBook:~/Show/git$ git log --pretty=oneline --max-count=2
1c003103c13222dac1bb1bf9dc13f1aab34b27ab (HEAD -> master) Второе изменение 2.txt
e159314d7c64d7a3359dfc76f67fd161d9ab53f5 Изменение 2.txt
arbon@ASUSVivoBook:~/Show/git$

```

Рисунок 1.10. Форматирование истории коммитов

1.8 Отмена изменений

Для отмены изменений в репозитории существует команда `git reset`, которая переместит HEAD и текущую ветку обратно туда, куда вы укажете, отказавшись от любых коммитов, которые могут быть оставлены позади. Далее останется очистить индекс командой `git restore`. И так, чтобы вернуть директорию в предыдущее состояние введем слудующие команды:

```

git reser --soft HEAD~
git restore --staged 2.txt

```

Результат этих команд проиллюстрирован на рисунке 1.11.

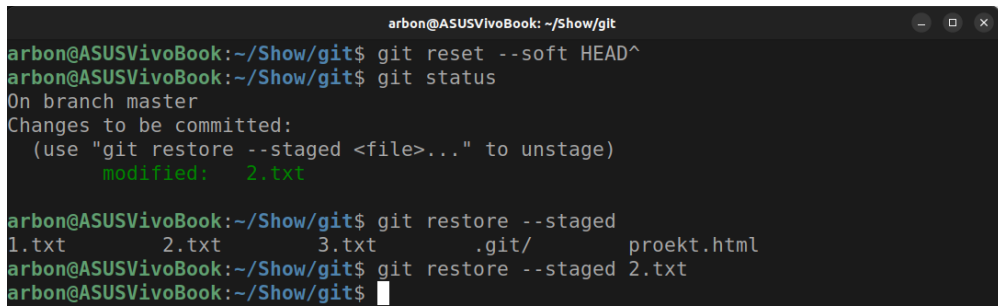
A terminal window titled 'arbon@ASUSVivoBook: ~/Show/git' showing a sequence of Git commands. The user runs 'git reset --soft HEAD^', then 'git status'. The status shows 'On branch master' and 'Changes to be committed: (use "git restore --staged <file>..." to unstage) modified: 2.txt'. Then the user runs 'git restore --staged', which lists '1.txt', '2.txt', '3.txt', '.git/', and 'proekt.html'. Finally, the user runs 'git restore --staged 2.txt'.

Рисунок 1.11. Возвращение рабочего каталога к предыдущему состоянию

1.9 Создание тега

Git использует два основных типа тегов: легковесные и аннотированные.

Легковесный тег — это что-то очень похожее на ветку, которая не изменяется — просто указатель на определённый коммит.

А вот аннотированные теги хранятся в базе данных Git как полноценные объекты. Они имеют контрольную сумму, содержат имя автора, его e-mail и дату создания, имеют комментарий и могут быть подписаны и проверены с помощью GNU Privacy Guard (GPG). Обычно рекомендуется создавать аннотированные теги, чтобы иметь всю перечисленную информацию; но если вы хотите сделать временную метку или по какой-то причине не хотите сохранять остальную информацию, то для этого годятся и легковесные.

1.9.1 Аннотированные теги

Создание аннотированного тега в Git выполняется легко. Самый простой способ — это указать -a при выполнении команды tag:

```
git tag -a <имя тега> -m "my version 1.4"
```

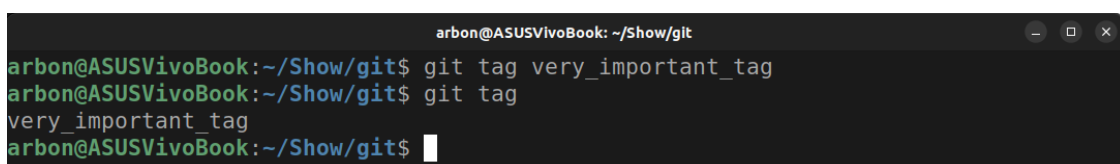
Опция -m задаёт сообщение, которое будет храниться вместе с тегом. Если не указать сообщение, то Git запустит редактор, чтобы вы смогли его ввести.

1.9.2 Легковесный теги

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:

```
git tag <имя тега>
```

Создание легковесного тега показано на рисунке 1.12.

A screenshot of a terminal window with a dark background. The title bar reads 'arbron@ASUSVivoBook: ~/Show/git'. The terminal shows the following commands and output: 'arbron@ASUSVivoBook:~/Show/git\$ git tag very_important_tag', 'arbron@ASUSVivoBook:~/Show/git\$ git tag', 'very_important_tag', and 'arbron@ASUSVivoBook:~/Show/git\$' followed by a cursor.

```
arbron@ASUSVivoBook: ~/Show/git
arbron@ASUSVivoBook:~/Show/git$ git tag very_important_tag
arbron@ASUSVivoBook:~/Show/git$ git tag
very_important_tag
arbron@ASUSVivoBook:~/Show/git$
```

Рисунок 1.12. Создание тега

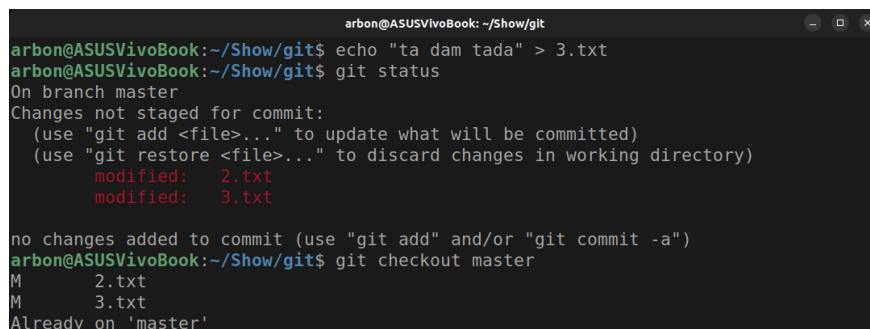
1.10 Отмена изменений в индексе

1.10.1 Отмена изменений (до индексации)

Для отмены изменений необходимо выполнить команду:

```
git checkout master
```

Результат выполнения этой команды показан на рисунке 1.13.

A screenshot of a terminal window with a dark background. The title bar reads 'arbron@ASUSVivoBook: ~/Show/git'. The terminal shows the following commands and output: 'arbron@ASUSVivoBook:~/Show/git\$ echo "ta dam tada" > 3.txt', 'arbron@ASUSVivoBook:~/Show/git\$ git status', 'On branch master', '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: 2.txt', 'modified: 3.txt', 'no changes added to commit (use "git add" and/or "git commit -a")', 'arbron@ASUSVivoBook:~/Show/git\$ git checkout master', 'M 2.txt', 'M 3.txt', and 'Already on 'master''.

```
arbron@ASUSVivoBook: ~/Show/git
arbron@ASUSVivoBook:~/Show/git$ echo "ta dam tada" > 3.txt
arbron@ASUSVivoBook:~/Show/git$ git status
On branch master
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:   2.txt
        modified:   3.txt
no changes added to commit (use "git add" and/or "git commit -a")
arbron@ASUSVivoBook:~/Show/git$ git checkout master
M      2.txt
M      3.txt
Already on 'master'
```

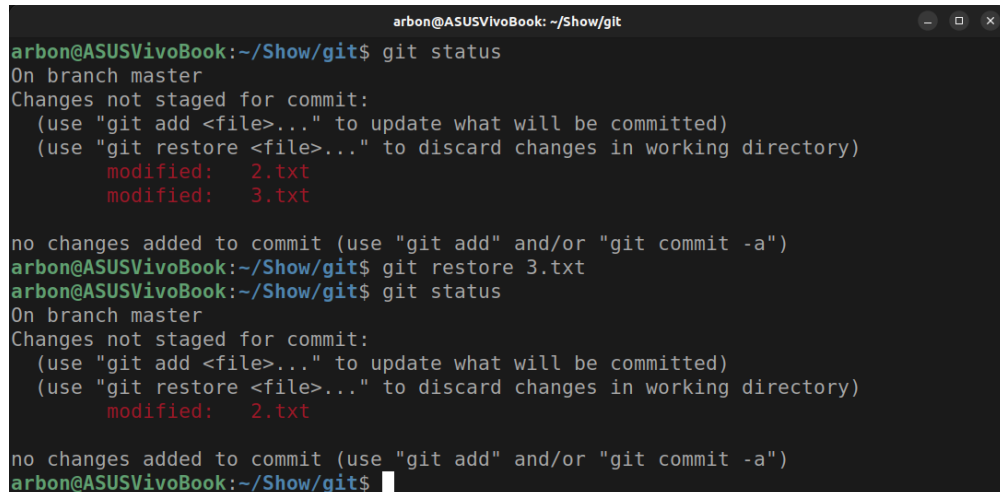
Рисунок 1.13. Отмена изменений (до индексации)

1.10.2 Отмена изменений (после индексации)

Для отмены изменений, уже проиндексированных, необходимо выполнить команду:

```
git reset HEAD имя_файла
```

Результат выполнения этой команды показан на рисунке 1.14.



```
arbron@ASUSVivoBook: ~/Show/git
arbron@ASUSVivoBook:~/Show/git$ git status
On branch master
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:   2.txt
        modified:   3.txt

no changes added to commit (use "git add" and/or "git commit -a")
arbron@ASUSVivoBook:~/Show/git$ git restore 3.txt
arbron@ASUSVivoBook:~/Show/git$ git status
On branch master
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:   2.txt

no changes added to commit (use "git add" and/or "git commit -a")
arbron@ASUSVivoBook:~/Show/git$
```

Рисунок 1.14. Отмена изменений (после индексации)

1.11 Отмена коммита

Для отмены изменений в репозитории существует команда: `git reset`, которая переместит HEAD и текущую ветку обратно туда, куда вы укажете, отказавшись от любых коммитов, которые могут быть оставлены позади. Данное действие уже показано на рисунке 1.11.

Глава 2

Системы управления репозиториями

2.1 Создание репозитория на GitHub и на локальной машине

2.1.1 Создание репозитория на GitHub

Чтобы создать репозиторий в GitHub, нужно:

1. Перейти во вкладку «repositories».
2. Нажать на кнопку «New».
3. Ввести название для репозитория и нажать кнопку.

Окно создания репозитория изображено на рисунке 2.1.

2.2 Создание репозитория на локальной машине

Создание репозитория рассматривалось в прошлой части. Так что опишем вкратце:

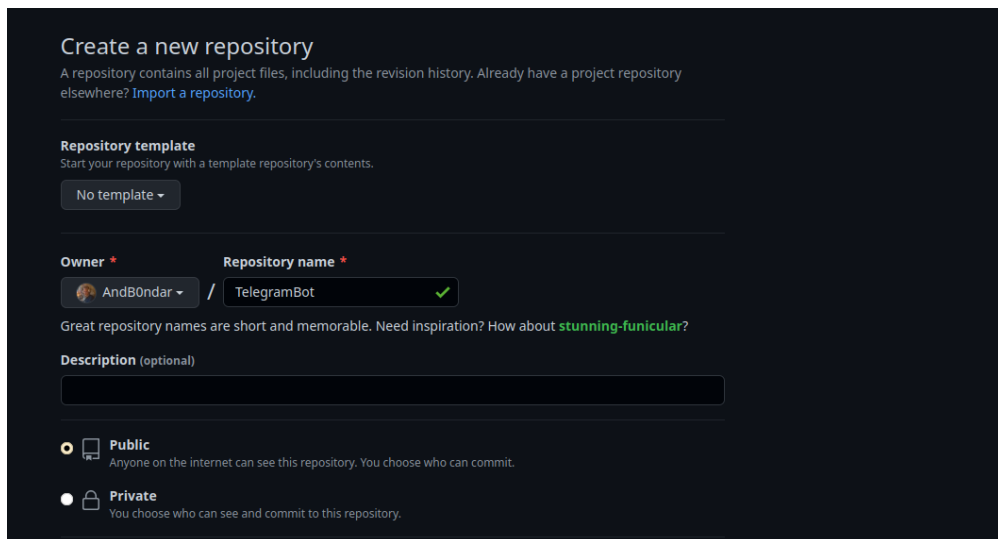


Рисунок 2.1. Создание репозитория в GitHub

1. Создадим каталог, где будет размещаться репозиторий.
2. Добавим файлы в каталог.
3. Создадим локальный репозиторий, командой `git init`.
4. Добавим файлы в индекс, командой `git add <файлы>`.
5. Сделаем коммит, чтобы сохранить изменения в репозиторий, командой `git commit -m "текст коммита"`.

Консольный вывод показан на рисунке 2.2.

```
arbon@ASUSVivoBook: ~/Programming/TelegramBot
arbon@ASUSVivoBook:~/Programming/TelegramBot$ git init
hint: Using 'master' as the name for the initial branch. This default branch name
hint: is subject to change. To configure the initial branch name to use in all
hint: of your new repositories, which will suppress this warning, call:
hint:
hint:   git config --global init.defaultBranch <name>
hint:
hint: Names commonly chosen instead of 'master' are 'main', 'trunk' and
hint: 'development'. The just-created branch can be renamed via this command:
hint:
hint:   git branch -m <name>
Initialised empty Git repository in /home/arbon/Programming/TelegramBot/.git/
arbon@ASUSVivoBook:~/Programming/TelegramBot$ git add *
arbon@ASUSVivoBook:~/Programming/TelegramBot$ git commit -m "first commit"
[master (root-commit) 31744cc] first commit
 2 files changed, 2 insertions(+)
 create mode 100644 main.py
 create mode 100644 tmp.txt
arbon@ASUSVivoBook:~/Programming/TelegramBot$
```

Рисунок 2.2. Создание репозитория в GitHub

2.3 Создание SSH-ключа

Чтобы работать со своего компьютера с GitHub, иметь доступ к проектам, хранящимся на сервисе, выполнять команды в консоли без постоянного подтверждения пароля, нужно пройти авторизацию у сервера. В этом помогают SSH-ключи.

Для создания ключа нужно воспользоваться командой:

```
ssh-keygen -t rsa -b 4096 -C "your_email@example.com"
```

Во время работы команды потребуется ввести: путь, по которому будет располагаться созданный ключ, и пароль к создаваемому ключу. Обычно ключи сохраняются в каталоге `.ssh` домашней директории. Его содержимое отображено на рисунке 2.3.

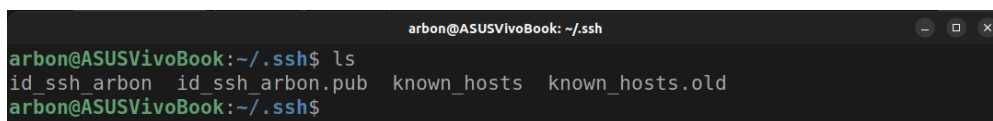


Рисунок 2.3. Каталог с созданными ssh ключами

2.4 Связывание локального и GitHub репозитория

Для того, чтобы связать репозиторий на локальной машине и созданным выше удаленный репозиториум, вначале необходимо зарегистрировать наш ssh ключ в GitHub. Мы должны его скопировать из консоли и перейти на страницу для работы с ключами в профиле на GitHub. Выбираем кнопку “New SSH key”, открывается окно с вводом данных, в поле “key” вставляем скопированный ключ, в “Title” вводим любое имя ключа и нажимаем “Add SSH key”. Добавленный ключ показан на рисунке 2.4

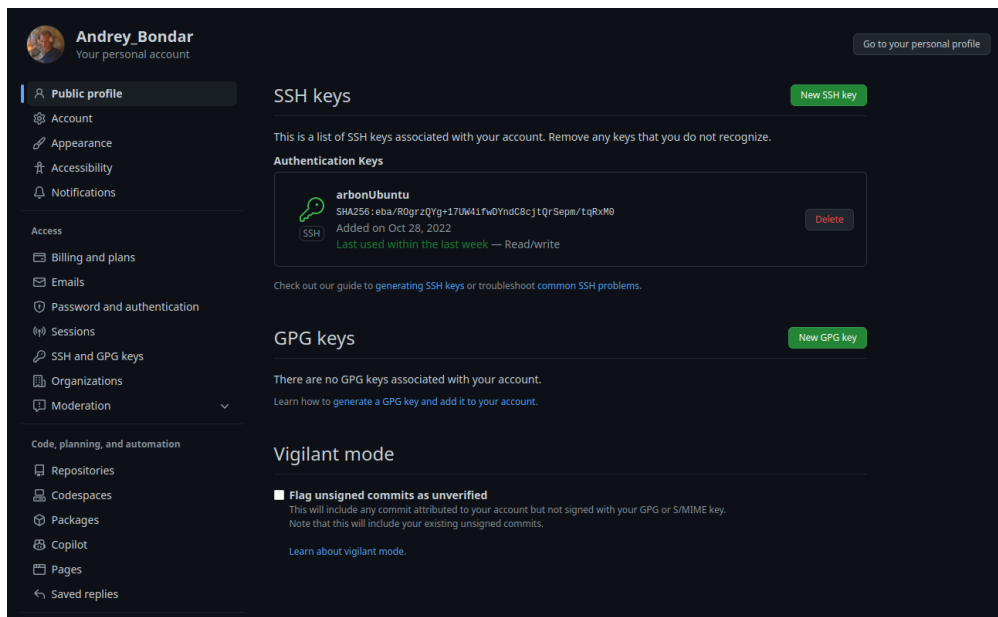


Рисунок 2.4. Добавленный ssh-ключ в GitHub

Чтобы связать локальный и удаленный репозитории друг с другом необходимо ввести в консоль следующую команду:

```
git remote add project \
git@github.com:<ваши имя и название репозитория>.git
git branch -M main
git push -u origin main
```

Вывод команд показан на рисунке 2.5.

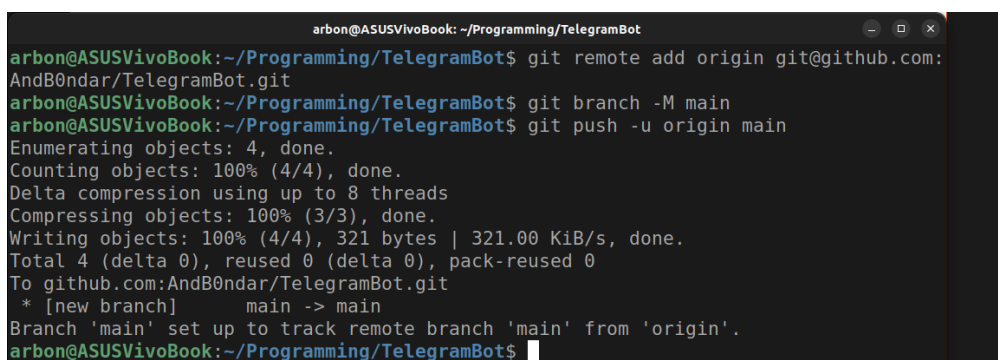


Рисунок 2.5. Связывание репозитория

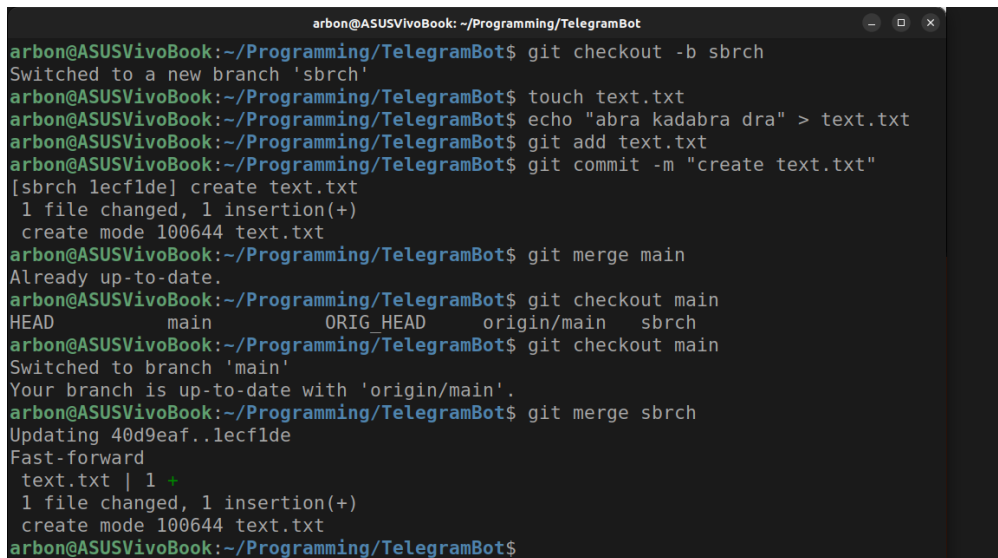
2.5 Создание и слияние новой ветки

Создание новой ветки в git выполняется командой:
`git branch <имя ветки>`.

Чтобы изменить текущую ветку есть команда:
`git checkout <имя ветки>`.

Эти две команды можно объединить, чтобы создать новую ветку и сразу на нее переключиться: `git checkout -b <имя ветки>`.

Чтобы слить две ветки, нужно перейти в ту ветку в которую нужно объединить изменения и ввести команду: `git merge <название ветки>`, как показано на рисунке 2.6.



```
arbon@ASUSVivoBook: ~/Programming/TelegramBot
arbon@ASUSVivoBook:~/Programming/TelegramBot$ git checkout -b sbrch
Switched to a new branch 'sbrch'
arbon@ASUSVivoBook:~/Programming/TelegramBot$ touch text.txt
arbon@ASUSVivoBook:~/Programming/TelegramBot$ echo "abra kadabra dra" > text.txt
arbon@ASUSVivoBook:~/Programming/TelegramBot$ git add text.txt
arbon@ASUSVivoBook:~/Programming/TelegramBot$ git commit -m "create text.txt"
[sbrch lecflde] create text.txt
1 file changed, 1 insertion(+)
create mode 100644 text.txt
arbon@ASUSVivoBook:~/Programming/TelegramBot$ git merge main
Already up-to-date.
arbon@ASUSVivoBook:~/Programming/TelegramBot$ git checkout main
HEAD      main      ORIG HEAD  origin/main  sbrch
arbon@ASUSVivoBook:~/Programming/TelegramBot$ git checkout main
Switched to branch 'main'
Your branch is up-to-date with 'origin/main'.
arbon@ASUSVivoBook:~/Programming/TelegramBot$ git merge sbrch
Updating 40d9eaf..lecflde
Fast-forward
 text.txt | 1 +
1 file changed, 1 insertion(+)
create mode 100644 text.txt
arbon@ASUSVivoBook:~/Programming/TelegramBot$
```

Рисунок 2.6. Слияние веток в git

2.6 Задание варианта

Выполним цепочку действий в репозитории, согласно 3-ему варианту:

- Клонировем непустой удаленный репозиторий на локальную машину (Рисунок 2.7).

```
arbron@ASUSVivoBook: ~/Show/CloneRepo
arbron@ASUSVivoBook:~/Show/CloneRepo$ git clone git@github.com:AndB0ndar/TelegramBot.git
Cloning into 'TelegramBot'...
remote: Enumerating objects: 7, done.
remote: Counting objects: 100% (7/7), done.
remote: Compressing objects: 100% (5/5), done.
remote: Total 7 (delta 0), reused 7 (delta 0), pack-reused 0
Receiving objects: 100% (7/7), done.
arbron@ASUSVivoBook:~/Show/CloneRepo$
```

Рисунок 2.7. Клонирование удаленного репозитория

- Создадим тег указывающий на последний коммит в ветке master(Рисунок 2.8).
- Создадим новую ветку и выведем список всех веток (Рисунок 2.8).

```
arbron@ASUSVivoBook: ~/Show/CloneRepo/TelegramBot
arbron@ASUSVivoBook:~/Show/CloneRepo/TelegramBot$ git tag tgl
arbron@ASUSVivoBook:~/Show/CloneRepo/TelegramBot$ git checkout -b nw_branch
Switched to a new branch 'nw_branch'
arbron@ASUSVivoBook:~/Show/CloneRepo/TelegramBot$ git branch
  main
* nw_branch
arbron@ASUSVivoBook:~/Show/CloneRepo/TelegramBot$
```

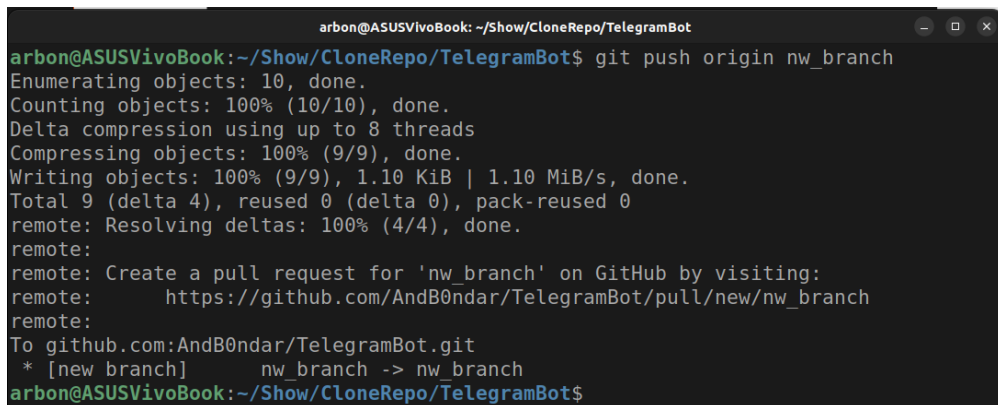
Рисунок 2.8. Создание тега и новой ветки

- Произведем 3 коммита в новой ветке (Рисунок 2.9).

```
arbron@ASUSVivoBook: ~/Show/CloneRepo/TelegramBot
arbron@ASUSVivoBook:~/Show/CloneRepo/TelegramBot$ touch poems.txt
arbron@ASUSVivoBook:~/Show/CloneRepo/TelegramBot$ echo "Есенин С. А. - Пускай я порою от спирта вымок...\n" > poems.txt
arbron@ASUSVivoBook:~/Show/CloneRepo/TelegramBot$ git add poems.txt
arbron@ASUSVivoBook:~/Show/CloneRepo/TelegramBot$ git commit -m "first commit in new branch"
[nw_branch 2a886ed] first commit in new branch
1 file changed, 1 insertion(+)
create mode 100644 poems.txt
arbron@ASUSVivoBook:~/Show/CloneRepo/TelegramBot$ echo "Пускай я порою от спирта вымок,
Пусть сердце слабеет, тускнеют очи,
Но, Гурвич! взглянувши на этот снимок,
Ты вспомни меня и "Бакинский рабочий".\n" >> poems.txt
arbron@ASUSVivoBook:~/Show/CloneRepo/TelegramBot$ git add poems.txt
arbron@ASUSVivoBook:~/Show/CloneRepo/TelegramBot$ git commit -m "second commit in new branch"
[nw_branch 7891f9c] second commit in new branch
1 file changed, 4 insertions(+)
arbron@ASUSVivoBook:~/Show/CloneRepo/TelegramBot$ echo "Не знаю, мой праздник иль худший день их,
Мы часто друг друга по-сучьи лаем,
Но если бы Фришберг давал всем денег,
Тогда бы газета была нам раем." >> poems.txt
arbron@ASUSVivoBook:~/Show/CloneRepo/TelegramBot$ git add poems.txt
arbron@ASUSVivoBook:~/Show/CloneRepo/TelegramBot$ git commit -m "third commit in new branch"
[nw_branch 6da105d] third commit in new branch
1 file changed, 4 insertions(+)
arbron@ASUSVivoBook:~/Show/CloneRepo/TelegramBot$
```

Рисунок 2.9. Создание трех коммитов

- Выгрузим все изменения в удаленный репозиторий (Рисунок 2.10).



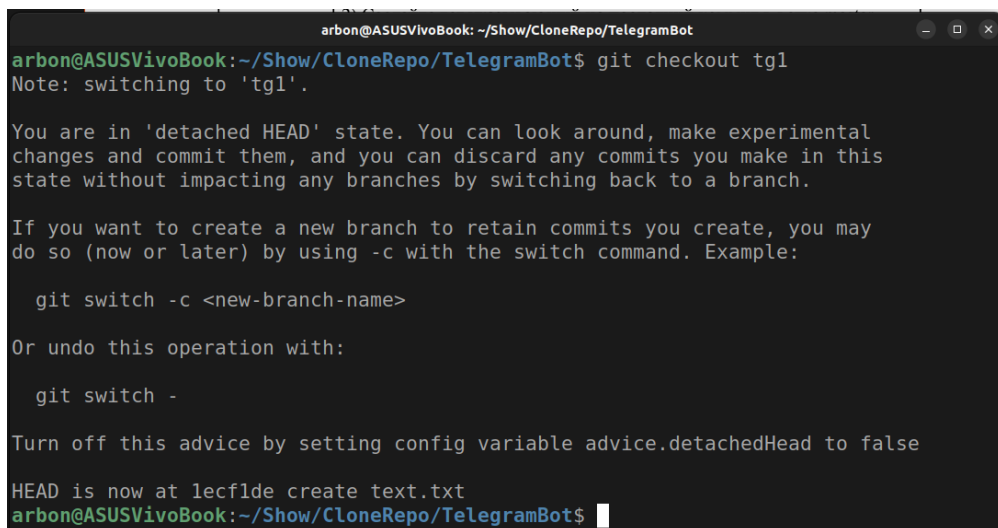
```

arbon@ASUSVivoBook: ~/Show/CloneRepo/TelegramBot
arbon@ASUSVivoBook:~/Show/CloneRepo/TelegramBot$ git push origin nw_branch
Enumerating objects: 10, done.
Counting objects: 100% (10/10), done.
Delta compression using up to 8 threads
Compressing objects: 100% (9/9), done.
Writing objects: 100% (9/9), 1.10 KiB | 1.10 MiB/s, done.
Total 9 (delta 4), reused 0 (delta 0), pack-reused 0
remote: Resolving deltas: 100% (4/4), done.
remote:
remote: Create a pull request for 'nw_branch' on GitHub by visiting:
remote:   https://github.com/AndB0ndar/TelegramBot/pull/new/nw_branch
remote:
To github.com:AndB0ndar/TelegramBot.git
 * [new branch]      nw_branch -> nw_branch
arbon@ASUSVivoBook:~/Show/CloneRepo/TelegramBot$

```

Рисунок 2.10. Отправка изменений в удаленный репозиторий

- Откатим ветку к созданному тегу (в том числе в удаленном репозитории) (Рисунок 2.11).



```

arbon@ASUSVivoBook: ~/Show/CloneRepo/TelegramBot
arbon@ASUSVivoBook:~/Show/CloneRepo/TelegramBot$ git checkout tgl
Note: switching to 'tgl'.

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 lecflde create text.txt
arbon@ASUSVivoBook:~/Show/CloneRepo/TelegramBot$

```

Рисунок 2.11. Откатка ветки к созданному тегу

- Выведем в консоли различия между веткой master и новой веткой (Рисунок 2.12).

```
arbron@ASUSVivoBook: ~/Show/CloneRepo/TelegramBot
arbron@ASUSVivoBook:~/Show/CloneRepo/TelegramBot$ git diff main nw_branch
diff --git a/poems.txt b/poems.txt
new file mode 100644
index 0000000..f0dcc89
--- /dev/null
+++ b/poems.txt
@@ -0,0 +1,9 @@
+Есенин С. А. - Пускай я порою от спирта вымок...\n
+Пускай я порою от спирта вымок,
+Пусть сердце слабеет, тускнеют очи,
+Но, Гурвич! взглянувши на этот снимок,
+Ты вспомни меня и "Бакинский рабочий".\n
+Не знаю, мой праздник иль худший день их,
+Мы часто друг друга по-сучьи лаем,
+Но если бы Фришберг давал всем денег,
+Тогда бы газета была нам раем.
arbron@ASUSVivoBook:~/Show/CloneRepo/TelegramBot$
```

Рисунок 2.12. Различия между ветками

Глава 3

Работа с ветвлением и оформление кода

3.1 Форк репозитория

Чтобы сделать форк перейдем к нужному репозитории в GitHub и в верхнем левом углу нажмем на кнопку «fork». На появившейся странице потребуется ввести имя форка (Рисунок 3.1).

Create a new fork

A fork is a copy of a repository. Forking a repository allows you to freely experiment with changes without affecting the original project. [View existing forks.](#)

Owner * Repository name *

AndB0ndar / python-cheatsheet ✓

By default, forks are named the same as their upstream repository. You can customize the name to distinguish it further.

Description (optional)

Comprehensive Python Cheatsheet

☒ Copy the `main` branch only

Contribute back to gto76/python-cheatsheet by adding your own branch. [Learn more.](#)

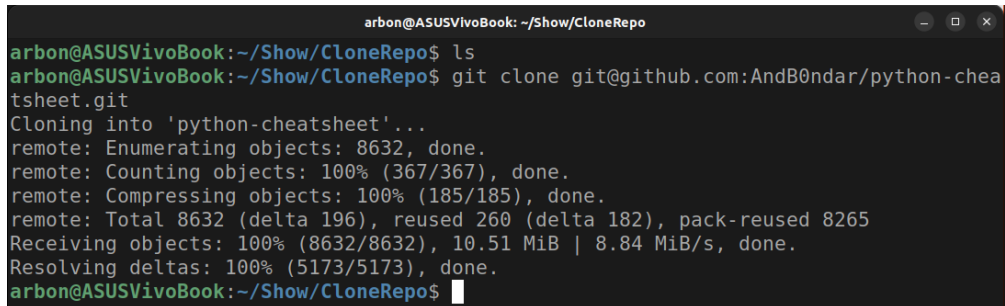
i You are creating a fork in your personal account.

Create fork

Рисунок 3.1. Создание форка в GitHub

3.2 Клонирование форка на локальную машину

Клонирование репозитория производится командой:
`git clone <адрес форка>` (Рисунок 3.2).

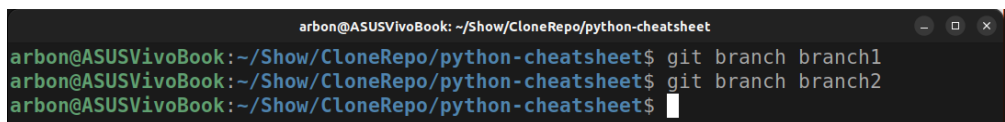


```
arbon@ASUSVivoBook: ~/Show/CloneRepo
arbon@ASUSVivoBook:~/Show/CloneRepo$ ls
arbon@ASUSVivoBook:~/Show/CloneRepo$ git clone git@github.com:AndB0ndar/python-cheatsheet.git
Cloning into 'python-cheatsheet'...
remote: Enumerating objects: 8632, done.
remote: Counting objects: 100% (367/367), done.
remote: Compressing objects: 100% (185/185), done.
remote: Total 8632 (delta 196), reused 260 (delta 182), pack-reused 8265
Receiving objects: 100% (8632/8632), 10.51 MiB | 8.84 MiB/s, done.
Resolving deltas: 100% (5173/5173), done.
arbon@ASUSVivoBook:~/Show/CloneRepo$
```

Рисунок 3.2. Клонирование форка

3.3 Создание двух веток

Создание новой ветки производится командой:
`git branch <имя ветки>` (Рисунок 3.3).



```
arbon@ASUSVivoBook: ~/Show/CloneRepo/python-cheatsheet
arbon@ASUSVivoBook:~/Show/CloneRepo/python-cheatsheet$ git branch branch1
arbon@ASUSVivoBook:~/Show/CloneRepo/python-cheatsheet$ git branch branch2
arbon@ASUSVivoBook:~/Show/CloneRepo/python-cheatsheet$
```

Рисунок 3.3. Создание двух веток

3.4 Коммиты

Теперь создадим по три коммита в каждой ветке. Для этого воспользуемся командой: `git commit -am "текст коммита"`, которая добавит в индекс все изменения в индекс и сделает коммит. Изменения проиллюстрированы на рисунках 3.4-3.5.


```
arbron@ASUSVivoBook: ~/Show/CloneRepo/python-cheatsheet
arbron@ASUSVivoBook:~/Show/CloneRepo/python-cheatsheet$ echo "Владимир Маяковский –
Ничего не понимают" > VeryImportantInfo.txt
arbron@ASUSVivoBook:~/Show/CloneRepo/python-cheatsheet$ git commit -am "first change
file in br1"
[branch1 0a2b93d] first change file in br1
1 file changed, 1 insertion(+)
arbron@ASUSVivoBook:~/Show/CloneRepo/python-cheatsheet$ echo "Вошел к парикмахеру, с
казал – спокойный:
«Будьте добры, причешите мне уши».
Гладкий парикмахер сразу стал хвойный,
лицо вытянулось, как у груши.
«Сумасшедший!
Рыжий!»-
запрыгали слова." >> VeryImportantInfo.txt
bash: !>: event not found
arbron@ASUSVivoBook:~/Show/CloneRepo/python-cheatsheet$ git commit -am "second chang
e file in br1"
[branch1 d1dc061] second change file in br1
1 file changed, 6 insertions(+)
arbron@ASUSVivoBook:~/Show/CloneRepo/python-cheatsheet$ echo "Ругань металась от пис
ка до писка,
и до-о-о-о-лго
хихикала чья-то голова,
выдерживаясь из толпы, как старая редиска." >> VeryImportantInfo.txt
arbron@ASUSVivoBook:~/Show/CloneRepo/python-cheatsheet$ git commit -am "third change
file in br1"
[branch1 b15040f] third change file in br1
1 file changed, 4 insertions(+)
arbron@ASUSVivoBook:~/Show/CloneRepo/python-cheatsheet$
```

Рисунок 3.4. Создание трех коммитов в ветке branch1

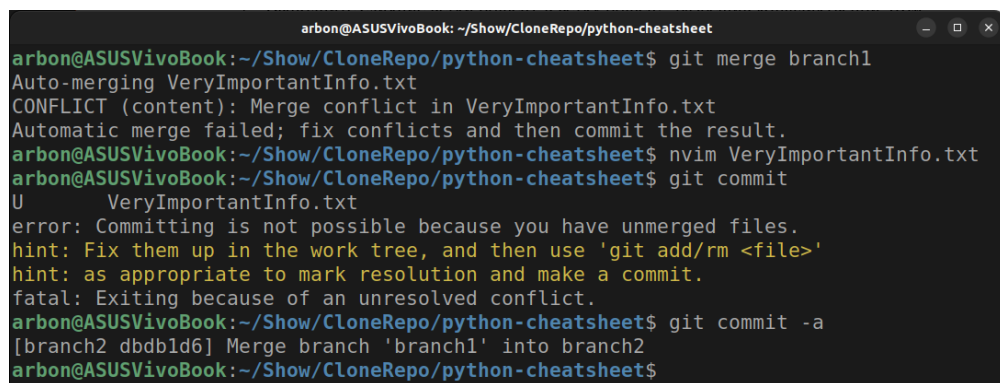
```
arbron@ASUSVivoBook: ~/Show/CloneRepo/python-cheatsheet
arbron@ASUSVivoBook:~/Show/CloneRepo/python-cheatsheet$ echo "Блок А. А. - Кошмар" >
VeryImportantInfo.txt
arbron@ASUSVivoBook:~/Show/CloneRepo/python-cheatsheet$ git commit -am "1 change in
br2"
[branch2 8d9b30f] 1 change in br2
1 file changed, 1 insertion(+)
arbron@ASUSVivoBook:~/Show/CloneRepo/python-cheatsheet$ echo "Я проснулся внезапно в
ночной тишине,
И душа испугалась молчания ночи.
Я увидел на темной стене
Чьи-то скорбные очи." >> VeryImportantInfo.txt
arbron@ASUSVivoBook:~/Show/CloneRepo/python-cheatsheet$ git commit -am "2 change in
br2"
[branch2 c261012] 2 change in br2
1 file changed, 4 insertions(+)
arbron@ASUSVivoBook:~/Show/CloneRepo/python-cheatsheet$ echo "Без конца на пустой и
безмолвной стене
Эти полные скорби и ужаса очи
Всё мерещатся мне в тишине
Леденеющей ночи." >> VeryImportantInfo.txt
arbron@ASUSVivoBook:~/Show/CloneRepo/python-cheatsheet$ git commit -am "3 change in
br2"
[branch2 dclad7b] 3 change in br2
1 file changed, 4 insertions(+)
arbron@ASUSVivoBook:~/Show/CloneRepo/python-cheatsheet$
```

Рисунок 3.5. Создание трех коммитов в ветке branch2

3.5 Слияние веток

Чтобы слить ветку А в ветку Б используется команда: `git merge Б`. Если при слиянии появился конфликт и `git` не смог автоматически разрешить его, то необходимо открыть файл с конфликтом в любом из редакторов и исправить выделенный участок. Затем останется выполнить команду `git commit` для создания коммита слияния.

Эти действия проиллюстрированы на рисунке 3.6.



```
arbon@ASUSVivoBook: ~/Show/CloneRepo/python-cheatsheet
arbon@ASUSVivoBook:~/Show/CloneRepo/python-cheatsheet$ git merge branch1
Auto-merging VeryImportantInfo.txt
CONFLICT (content): Merge conflict in VeryImportantInfo.txt
Automatic merge failed; fix conflicts and then commit the result.
arbon@ASUSVivoBook:~/Show/CloneRepo/python-cheatsheet$ nvim VeryImportantInfo.txt
arbon@ASUSVivoBook:~/Show/CloneRepo/python-cheatsheet$ git commit
U      VeryImportantInfo.txt
error: Committing is not possible because you have unmerged files.
hint: Fix them up in the work tree, and then use 'git add/rm <file>'
hint: as appropriate to mark resolution and make a commit.
fatal: Exiting because of an unresolved conflict.
arbon@ASUSVivoBook:~/Show/CloneRepo/python-cheatsheet$ git commit -a
[branch2 dbdb1d6] Merge branch 'branch1' into branch2
arbon@ASUSVivoBook:~/Show/CloneRepo/python-cheatsheet$
```

Рисунок 3.6. Слияние ветки branch1 в ветку branch2

3.6 Отправка всех изменений

Чтобы отправить все изменения введем команду `git push origin <название ветки>` для каждой ветки (Рисунок 3.7).

```
arbon@ASUSVivoBook: ~/Show/CloneRepo/python-cheatsheet
arbon@ASUSVivoBook:~/Show/CloneRepo/python-cheatsheet$ git push origin
branch1      HEAD      ORIG HEAD    origin/main
branch2      main      origin/HEAD
arbon@ASUSVivoBook:~/Show/CloneRepo/python-cheatsheet$ git push origin main
Enumerating objects: 4, done.
Counting objects: 100% (4/4), done.
Delta compression using up to 8 threads
Compressing objects: 100% (2/2), done.
Writing objects: 100% (3/3), 294 bytes | 294.00 KiB/s, done.
Total 3 (delta 1), reused 0 (delta 0), pack-reused 0
remote: Resolving deltas: 100% (1/1), completed with 1 local object.
To github.com:AndB0ndar/python-cheatsheet.git
    4e3a328..8964933  main -> main
arbon@ASUSVivoBook:~/Show/CloneRepo/python-cheatsheet$ git push origin branch1
Enumerating objects: 11, done.
Counting objects: 100% (11/11), done.
Delta compression using up to 8 threads
Compressing objects: 100% (9/9), done.
Writing objects: 100% (9/9), 1.07 KiB | 1.07 MiB/s, done.
Total 9 (delta 5), reused 0 (delta 0), pack-reused 0
remote: Resolving deltas: 100% (5/5), completed with 1 local object.
remote:
remote: Create a pull request for 'branch1' on GitHub by visiting:
remote:   https://github.com/AndB0ndar/python-cheatsheet/pull/new/branch1
remote:
To github.com:AndB0ndar/python-cheatsheet.git
 * [new branch]      branch1 -> branch1
arbon@ASUSVivoBook:~/Show/CloneRepo/python-cheatsheet$ git push origin branch2
Enumerating objects: 16, done.
Counting objects: 100% (16/16), done.
Delta compression using up to 8 threads
Compressing objects: 100% (11/11), done.
```

Рисунок 3.7. Отправка всех изменений