

Необходимые команды для cmd (командной строки)

dir – выводит все файлы, папки которые есть в данном каталоге

cd ./“Название папки” - переход в другой каталог

cd .. - переход на каталог выше


NUL> “Название файла.расширение” – создание файла

del ./“Название файла” – удалить файл

md “Название каталога” – создание каталога


rmdir “Название каталога” - удалить каталог

## Локальный репозиторий

Для создания необходимо запустить  Git Bash Here, выбрать нужную папку где нужно создать репозиторий (дабы не указывать путь вручную), нажать ПКМ и запустить.

Ввести в командную строку «git init»

```
DNS@LAPTOP-T48H98QO MINGW64 ~/Desktop/git_test
$ git init
Initialized empty Git repository in C:/Users/DNS/Desktop/git_test/.git/
```

в выбранной папке появится новая, скрытая папка  .git, это означает что репозиторий создан.

Для просмотра какие файлы не отслеживаются, какие были изменены используется команда «git status»

```
DNS@LAPTOP-T48H98QO MINGW64 ~/Desktop/git_test (master)
$ git status
On branch master

No commits yet

Untracked files:
  (use "git add <file>..." to include in what will be committed)
        file_3.txt
        text_2.txt
        textfile_1.txt

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

так как это новый репозиторий то в нём отображаются только те файлы которые я недавно создал, это 3 текстовых файла.

Далее я добавил первый и второй файл к отслеживанию

```
DNS@LAPTOP-T48H98QO MINGW64 ~/Desktop/git_test (master)
$ git add textfile_1.txt

DNS@LAPTOP-T48H98QO MINGW64 ~/Desktop/git_test (master)
$ git add text_2.txt
```

git status изменился

```
DNS@LAPTOP-T48H98QO MINGW64 ~/Desktop/git_test (master)
$ git status
On branch master

No commits yet

Changes to be committed:
  (use "git rm --cached <file>..." to unstage)
        new file:   text_2.txt
        new file:   textfile_1.txt

Untracked files:
  (use "git add <file>..." to include in what will be committed)
        file_3.txt
```

он показывает что появились новые файлы которые отслеживаются и то что им нужен коммит

Произведём первый коммит

```
DNS@LAPTOP-T48H98QO MINGW64 ~/Desktop/git_test (master)
$ git commit -am 'Первый коммит'
[master (root-commit) 9d183b8] Первый коммит
 2 files changed, 2 insertions(+)
 create mode 100644 text_2.txt
 create mode 100644 textfile_1.txt
```

git status изменился

```
DNS@LAPTOP-T48H98QO MINGW64 ~/Desktop/git_test (master)
$ git status
On branch master
Untracked files:
  (use "git add <file>..." to include in what will be committed)
    file_3.txt

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

теперь он не отображает файлы закоммиченные и без изменений

Отслеживаемые файлы можно посмотреть с помощью команды «git ls-files»

```
DNS@LAPTOP-T48H98QO MINGW64 ~/Desktop/git_test (master)
$ git ls-files
text_2.txt
textfile_1.txt
```

Произведённый коммит можно посмотреть с помощью «git log»

```
DNS@LAPTOP-T48H98QO MINGW64 ~/Desktop/git_test (master)
$ git log
commit 9d183b85e4933d29b95c51aedcf90c935e0df3cc (HEAD -> master)
Author: NikitaMon <Nikon30050@mail.ru>
Date:   Sun Apr 16 06:43:15 2023 +0300

    Первый коммит
```

Далее внесём изменение во второй файл

git status отобразит что файл был изменён

```
DNS@LAPTOP-T48H98QO MINGW64 ~/Desktop/git_test (master)
$ 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:   text_2.txt

Untracked files:
  (use "git add <file>..." to include in what will be committed)
    file_3.txt

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

произведём второй коммит.

Посмотреть изменения файла можно с помощью «git log -p “имя файла”»

```
DNS@LAPTOP-T48H98QO MINGW64 ~/Desktop/git_test (master)
$ git log -p text_2.txt
commit 1481d4d1dbb7bdfb374a565420c47f8880c37593 (HEAD -> master)
Author: NikitaMon <Nikon30050@mail.ru>
Date:   Sun Apr 16 06:51:47 2023 +0300

    Второй коммит

diff --git a/text_2.txt b/text_2.txt
index 9f15d98..6eef9a5 100644
--- a/text_2.txt
+++ b/text_2.txt
@@ -1,2 @@
-второй файл
\ No newline at end of file
+второй файл
+изменение 1
\ No newline at end of file

commit 9d183b85e4933d29b95c51aedcf90c935e0df3cc
Author: NikitaMon <Nikon30050@mail.ru>
Date:   Sun Apr 16 06:43:15 2023 +0300

    Первый коммит

diff --git a/text_2.txt b/text_2.txt
new file mode 100644
index 0000000..9f15d98
--- /dev/null
+++ b/text_2.txt
@@ -0,0 +1 @@
+второй файл
\ No newline at end of file
```

или все текущие изменения «git diff»

```
DNS@LAPTOP-T48H98QO MINGW64 ~/Desktop/git_test (master)
$ git diff
diff --git a/new_file_4.txt b/new_file_4.txt
index b642f4f..f1729f9 100644
--- a/new_file_4.txt
+++ b/new_file_4.txt
@@ -1,2 @@
-это файл 4
\ No newline at end of file
+это файл 4
+это не файл
\ No newline at end of file
diff --git a/textfile_1.txt b/textfile_1.txt
index 1c118cb..86c1ce0 100644
--- a/textfile_1.txt
+++ b/textfile_1.txt
@@ -1,2 @@
    первый файл
+новая строка
\ No newline at end of file
```

создадим новую ветку с названием new\_branch

```
DNS@LAPTOP-T48H98QO MINGW64 ~/Desktop/git_test (master)
$ git branch new_branch
```

посмотреть существующие ветки можно с помощью git branch

```
DNS@LAPTOP-T48H98QO MINGW64 ~/Desktop/git_test (master)
$ git branch
* master
  new_branch
```

переключение на другую ветку

```
DNS@LAPTOP-T48H98QO MINGW64 ~/Desktop/git_test (master)
$ git checkout new_branch
Switched to branch 'new_branch'
```

(master) поменяется на (new\_branch)

```
DNS@LAPTOP-T48H98QO MINGW64 ~/Desktop/git_test (new_branch)
```

создадим новый файл, добавим к отслеживанию и закоммитим его

```
DNS@LAPTOP-T48H98QO MINGW64 ~/Desktop/git_test (new_branch)
$ git log
commit 798256a21add88d1bd502208573b778b59021143 (HEAD -> new_branch)
Author: NikitaMon <Nikon30050@mail.ru>
Date: Sun Apr 16 10:21:25 2023 +0300
```

первый комит ветки new\_branch






```
commit 1481d4d1dbb7bdfb374a565420c47f8880c37593 (master)
Author: NikitaMon <Nikon30050@mail.ru>
Date: Sun Apr 16 06:51:47 2023 +0300
```

Второй коммит





```
commit 9d183b85e4933d29b95c51aedcf90c935e0df3cc
Author: NikitaMon <Nikon30050@mail.ru>
Date: Sun Apr 16 06:43:15 2023 +0300
```

Первый коммит

содержание репозитория на момент ветки new\_branch

-  .git
-  file\_3.txt
-  text\_2.txt
-  textfile\_1.txt
-  new\_file\_4.txt

переключим на ветку master и содержание репозитория изменится

-  .git
-  file\_3.txt
-  text\_2.txt
-  textfile\_1.txt

объединим ветки master и new\_branch предварительно переключив на ветвь master

```
DNS@LAPTOP-T48H98QO MINGW64 ~/Desktop/git_test (master)
$ git merge new_branch
Updating 1481d4d..798256a
Fast-forward
 new_file_4.txt | 1 +
 1 file changed, 1 insertion(+)
 create mode 100644 new_file_4.txt
```

ветвь master перешла на последний коммит ветви new\_branch

```
commit 798256a21add88d1bd502208573b778b59021143 (HEAD -> master, new_branch)
Author: NikitaMon <Nikon30050@mail.ru>
Date: Sun Apr 16 10:21:25 2023 +0300

    первый комит ветки new_branch
```

удалить ветку new\_branch

```
DNS@LAPTOP-T48H98QO MINGW64 ~/Desktop/git_test (master)
$ git branch -d new_branch
Deleted branch new_branch (was 798256a).
```

посмотреть внутрь прошлого коммита можно с помощью git checkout „хэш коммита“

в таком случае HEAD переходит на коммит хэша.

перейдём на первый коммит

```
DNS@LAPTOP-T48H98QO MINGW64 ~/Desktop/git_test (master)
$ git checkout 9d183b85e4933d29b95c51aedcf90c935e0df3cc
Note: switching to '9d183b85e4933d29b95c51aedcf90c935e0df3cc'.

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 9d183b8 Первый коммит
DNS@LAPTOP-T48H98QO MINGW64 ~/Desktop/git_test ((9d183b8...))
$ git log
commit 9d183b85e4933d29b95c51aedcf90c935e0df3cc (HEAD)
Author: NikitaMon <Nikon30050@mail.ru>
Date: Sun Apr 16 06:43:15 2023 +0300

    Первый коммит
```

если нужно перевести ветку на ЭТОТ КОММИТ то  
git revert 'хэш'

переключим на второй коммит

```
DNS@LAPTOP-T48H98QO MINGW64 ~/Desktop/git_test (master)
$ git log
commit 1481d4d1dbb7bdfb374a565420c47f8880c37593 (HEAD -> master)
Author: NikitaMon <Nikon30050@mail.ru>
Date: Sun Apr 16 06:51:47 2023 +0300

    Второй коммит

commit 9d183b85e4933d29b95c51aedcf90c935e0df3cc
Author: NikitaMon <Nikon30050@mail.ru>
Date: Sun Apr 16 06:43:15 2023 +0300

    Первый коммит
```

изменим комментарий текущего коммита

```
DNS@LAPTOP-T48H98QO MINGW64 ~/Desktop/git_test (master)
$ git commit --amend
```

в этом окне вносится комментарий

комментарием является всё где отсутствует знак # в начале

```
первый комит ветки new_branch

# Please enter the commit message for your changes. Lines starting
# with '#' will be ignored, and an empty message aborts the commit.
#
# Date: Sun Apr 16 10:21:25 2023 +0300
#
# On branch master
# Changes to be committed:
#   new file:   new_file_4.txt
#
# Untracked files:
#   file_3.txt
#
[master a8f3632] не первый комит ветки master
Date: Sun Apr 16 10:21:25 2023 +0300
```

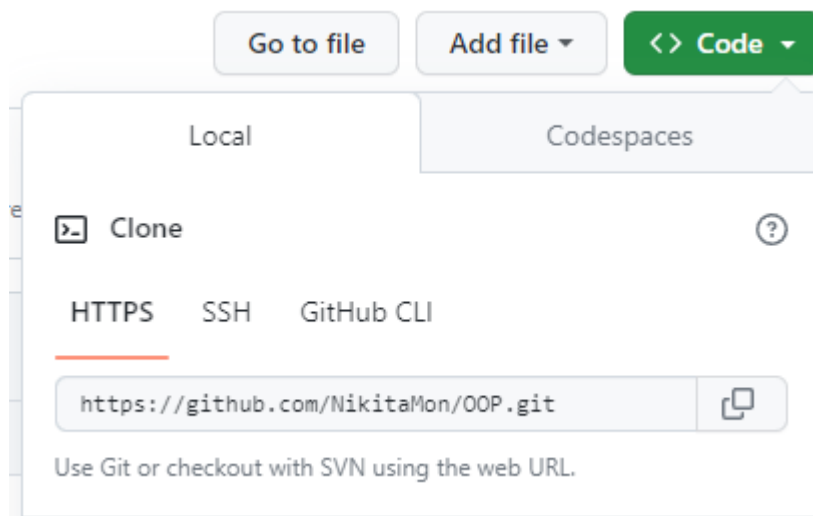
у коммита изменился хэш и комментарий

```
DNS@LAPTOP-T48H98QO MINGW64 ~/Desktop/git_test (master)
$ git log
commit a8f36321434bc5278f7c209cc4a93317ab59eac9 (HEAD -> master)
Author: NikitaMon <Nikon30050@mail.ru>
Date: Sun Apr 16 10:21:25 2023 +0300

    не первый комит ветки master
```

Удалённый репозиторий

Для начала надо создать свой репозиторий в github  
и получить ссылку



и с помощью этой ссылки создать копию

```
DNS@LAPTOP-T48H98QO MINGW64 ~/Desktop/git_test (master)
$ git clone https://github.com/NikitaMon/OOP.git
Cloning into 'OOP'...
remote: Enumerating objects: 17, done.
remote: Counting objects: 100% (17/17), done.
remote: Compressing objects: 100% (14/14), done.
remote: Total 17 (delta 0), reused 11 (delta 0), pack-reused 0
Receiving objects: 100% (17/17), 8.84 KiB | 8.84 MiB/s, done.
```

перейдём в папку репозитория

```
DNS@LAPTOP-T48H98QO MINGW64 ~/Desktop/git_test (master)
$ cd OOP/
```

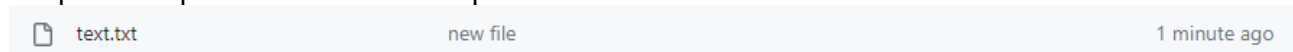
создадим в папке текстовый файл и добавим к отслеживанию  
приведём коммит

произведём пуш ветви

```
DNS@LAPTOP-T48H98QO MINGW64 ~/Desktop/git_test/OOP (main)
$ git push origin main
Enumerating objects: 4, done.
Counting objects: 100% (4/4), done.
Delta compression using up to 16 threads
Compressing objects: 100% (2/2), done.
Writing objects: 100% (3/3), 269 bytes | 269.00 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 https://github.com/NikitaMon/OOP.git
2a08242..6c0ee8a main -> main
```

все изменения отправились на github

и в репозитории появился новый файл



запрос изменений с github  
удалим файл из репозитория github  
и запросим изменения

```
DNS@LAPTOP-T48H98QO MINGW64 ~/Desktop/git_test/OOP (main)
$ git pull origin main
remote: Enumerating objects: 3, done.
remote: Counting objects: 100% (3/3), done.
remote: Compressing objects: 100% (2/2), done.
remote: Total 2 (delta 1), reused 0 (delta 0), pack-reused 0
Unpacking objects: 100% (2/2), 597 bytes | 199.00 KiB/s, done.
From https://github.com/NikitaMon/OOP
* branch          main          -> FETCH_HEAD
  6c0ee8a..bf80728 main          -> origin/main
Updating 6c0ee8a..bf80728
Fast-forward
 text.txt | 1 -
 1 file changed, 1 deletion(-)
 delete mode 100644 text.txt
```

в папке локальной копии удалится файл который мы создавали так как его уже нету на github

чтобы с локального репозитория запустить в удалённый надо указать адрес

```
DNS@LAPTOP-T48H98QO MINGW64 ~/Desktop/git_test/test (branch_new)
$ git remote add origin https://github.com/NikitaMon/OOP.git
```

и произвести пуш

```
DNS@LAPTOP-T48H98QO MINGW64 ~/Desktop/git_test/test (branch_new)
$ git push origin branch_new
Enumerating objects: 3, done.
Counting objects: 100% (3/3), done.
Writing objects: 100% (3/3), 207 bytes | 207.00 KiB/s, done.
Total 3 (delta 0), reused 0 (delta 0), pack-reused 0
remote:
remote: Create a pull request for 'branch_new' on GitHub by visiting:
remote:   https://github.com/NikitaMon/OOP/pull/new/branch_new
remote:
To https://github.com/NikitaMon/OOP.git
 * [new branch]      branch_new -> branch_new
```

но github не сможет объединить эту ветвь с main так как они не связаны между собой



**There isn't anything to compare.**

main and branch\_new are entirely different commit histories.



## Создание нового локального и пуш с гитхаба

```
DNS@LAPTOP-T48H98QO MINGW64 ~/Desktop/git_test/test2 (master)
$ git init
Initialized empty Git repository in C:/Users/DNS/Desktop/git_test/test2/.git/

DNS@LAPTOP-T48H98QO MINGW64 ~/Desktop/git_test/test2 (master)
$ git remote add origin https://github.com/NikitaMon/OOP.git

DNS@LAPTOP-T48H98QO MINGW64 ~/Desktop/git_test/test2 (master)
$ git branch

DNS@LAPTOP-T48H98QO MINGW64 ~/Desktop/git_test/test2 (master)
$ git pull origin main
remote: Enumerating objects: 21, done.
remote: Counting objects: 100% (21/21), done.
remote: Compressing objects: 100% (16/16), done.
remote: Total 21 (delta 1), reused 14 (delta 1), pack-reused 0
Unpacking objects: 100% (21/21), 9.57 KiB | 279.00 KiB/s, done.
From https://github.com/NikitaMon/OOP
 * branch            main       -> FETCH_HEAD
 * [new branch]      main       -> origin/main

DNS@LAPTOP-T48H98QO MINGW64 ~/Desktop/git_test/test2 (master)
$ git branch
* master
```

но если это не новый локальный репозиторий и там были коммиты то выдаст ошибку

```
fatal: refusing to merge unrelated histories
```

### git clone

На самом деле git clone работает как обёртка над некоторыми другими командами. Она создаёт новый каталог, переходит внутрь и выполняет git init для создания пустого репозитория, затем она добавляет новый удалённый репозиторий (git remote add) для указанного URL (по умолчанию он получит имя origin), выполняет git fetch для этого репозитория и, наконец, извлекает последний коммит в ваш рабочий каталог, используя git checkout.

### git fetch

Команда git fetch связывается с удалённым репозиторием и забирает из него все изменения, которых у вас пока нет и сохраняет их локально.

### git pull

Команда git pull работает как комбинация команд git fetch и git merge, т. е. Git вначале забирает изменения из указанного удалённого репозитория, а затем пытается слить их с текущей веткой.