

## ОСНОВНЫЕ КОМАНДЫ GIT

`cd <полный путь или продолжение пути из в текущего каталога>`

- перемещение в нужный каталог;

`git init`

- инициализируем git'ом папку, в которой пока не установлено связи с GitHub;

`git remote add origin <address of your repos>`

- подключаем связь этой папки с удалённым репозиторием на GitHub;

`git pull origin main`

- подтягиваем файлы с удалённого репозитория;

`git clone <address of your repos>`

- клонирование уже существующего на гитхаб репозитория с его кодом;

`git branch development`

- создаём новую ветку разработки;

`git checkout development`

- переключаемся в любую существующую ветку;

`git checkout -b development`

- создаём и уходим в отдельную ветку (сжиты две предыдущие команды в одну);

`git status`

- показать состояние всех файлов;

`git add <файл или папка>`

- добавить к будущему коммиту (сохранению) этот файл / папку;

`git add .`

-добавить всё, что выводится как новое / модифицированное через git status;

`git add -u <файл или папка>`

- удалить из репозитория на GitHub файл/папку, которые были удалены локально (например, лишнее попало на GitHub);

`git add -u .`

- удалить из репозитория на GitHub всё, что было удалено локально (перед вызовом в любом случае рекомендуется проверить, что именно будет удалено, с помощью команды git status);

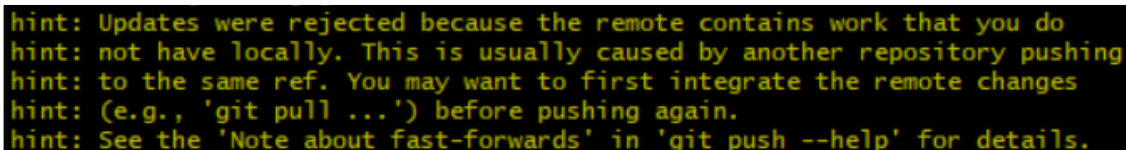
```
git commit -m "very important comment about done work"
```

- формируем коммит / сохранение с осмысленным комментарием;

```
git push origin development
```

- отправляем на GitHub все текущие созданные коммиты (сохранения) в рабочей ветке;

**Примечание 1.** Если git пишет, что в репозитории есть коммиты, отсутствующие в данном локальном репозитории (об этом будет написано в ошибке)



```
hint: Updates were rejected because the remote contains work that you do
hint: not have locally. This is usually caused by another repository pushing
hint: to the same ref. You may want to first integrate the remote changes
hint: (e.g., 'git pull ...') before pushing again.
hint: See the 'Note about fast-forwards' in 'git push --help' for details.
```

следует подтянуть разницу, для этого нужно прописать:

```
git pull origin <имя ветки>
```

- подтянуть новые файлы с удалённого репозитория из указанной ветки;

и после этого попробовать `git push` снова.

**Примечание 2.** Если git не знает origin (об этом будет написано в ошибке), его нужно прописать:

```
git remote add origin <address of your repos>
```

и попробовать команду `git push origin development` снова.

Альтернативный способ:

```
git remote set-url origin <address of your repos>
```

Вообще такая ситуация может возникнуть при работе с чужим репозиторием, когда вместо своей копии вы случайно скачали оригинал, к которому нет доступа.

**Примечание 3.** В качестве логина при авторизации используется логин от GitHub, а в качестве пароля сгенерированный token!

## ПРОЧИЕ ПОЛЕЗНОСТИ

```
git reset badWords.txt
```

- отмена случайно добавленного к коммиту файла, например, файла `badWords.txt`;

```
git branch
```

- показать текущие ветки;

```
git log
```

- показать коммиты;

```
git add .
```

```
git commit -m "неудачно"
```

```
git reset --hard HEAD~1
```

- отмена последнего неудачного коммита, в результате коммит удалён, состояние репозитория откатилось до последнего перед ним коммита;

```
git reset --hard HEAD~3
```

- отмена 3 последовательных последних коммитов;

```
git revert 1af17e
```

- откат до заданного коммита;

```
git diff 09bd8cc..ba25c0ff
```

- посмотреть разницу между двумя коммитами, показывает сколько изменений и где произошло между указанными коммитами, коды коммитов можно взять из информации полученной командой `git log` или с `github`;

```
git branch -D new-branch
```

- удалить ветку принудительно;

```
git branch -m new-name-branch
```

- переименовать ветку;

#### **Case «забыл создать новую ветку и работал в main»**

Если Вы закоммитили (`git commit -m ...`) уже кучу файлов, можно переместить все эти изменения в новую ветку можно с помощью следующих команд (действия выполняются в ветке `main`):

0. Если вы ещё не закоммитили изменения, их нужно закоммитить и затем выполнить указанные действия.

1. Создать рабочую ветку (выполняется, если ветка еще не была создана)

```
git branch название-рабочей-ветки
```

2. Отменить изменения кода (выполнять в `main`-ветке)

```
git reset HEAD~ --hard
```

3. Перейти в рабочую ветку

```
git checkout название-рабочей-ветки
```

#### **Case «не могу залить на GitHub в классе, потому что git привязался к другому пользователю»**

Зайти в Панель управления -> Учётные записи пользователей -> Диспетчер учётных данных -> Учётные данные Windows. Удалить учётные данные для `github`.

**Примечание 1.** Существует также множество других команд, решающих те или иные специфические проблемы. Если вы работаете корректно, то, вообще говоря, их не будет возникать, однако если они всё же возникают - требуется искать решение в `Google`.

*Пример специфической проблемы:* случайно забыл докинуть файлы в коммит и сделал это еще одним коммитом.

*Решение:* Окей Google, как объединить коммиты. Результат первой же ссылки: <https://pingvinus.ru/git/1591>. Исходя из информации в ссылке, нужно прописать `git rebase -i HEAD~2`. Мораль: программист и гугл - лучшие друзья.

**Примечание 2.** А также следует не забывать, что папки Debug, x64, .vs, и прочий ненужный набор файлов мы не заливаем! Потому требуется отредактировать по умолчанию создаваемый файл .gitignore.