[Git (git-scm.com)](https://git-scm.com/) — официальный сайт децентрализованная системы контроля версий git. Год рождения git 2005. Создан git разработчиками ядра Linux. Сайт содержит клиент гита и документацию на русском языке.

Понятия: fork, индекс, коммит, origin, master или main, Pull Request

git init

git clone <url удалённого репозитория>

git remote -v

git remote add <имя для удалённого репозитория> <url удалённого репозитория>

git status

git status -uall

git config --list

git add . или <имена файлов>

git commit -m “[ibusko] комментарий по-английски”

git push

git push --set-upstream <имя для удалённого репозитория> <имя ветки>

git push -f

git pull <имя удалённого репозитория> <имя ветки>

.gitignore

git branch

Переключение между ветками (git checkout <имя ветки>).

Отмена локальных изменений до индексации (git checkout . или <имена файлов>).

Запись текущих изменений в последний коммит, т. е. как не создавать коммит для изменений, которые являются продолжением последнего коммита.

Untracked files можно удалить командой git clean -f

Если есть untracked каталог, то git clean -fd

**Fork** — ответвление (англ.)

**Fork** — это копия стороннего репозитория.

Когда есть общий репозиторий или сторонний проект, в котором надо или хочется реализовать функциональность и нет прав на внесение изменений путём отправки push, то надо сделать fork этого репозитория. Эта копия репозитория будет находится в вашем пространстве имён github и можно будет для него легко делать изменения путём отправки push. И тогда весь рабочий процесс выглядит:

1. Создайте форк проекта.
2. Создайте тематическую ветку на основании ветки master.
3. Создайте один или несколько коммитов с изменениями, улучшающих проект.
4. Отправьте эту ветку в ваш проект на github (git push).
5. Откройте запрос на слияние на github (pull request).
6. Обсуждайте его, вносите изменения, если нужно.
7. Владелец проекта принимает решение о принятии изменений, либо об их отклонении.
8. Получите обновлённую ветку master (git pull) и отправьте её в свой форк.

-------------------------------------------------------------------------------------------------------

**git init**

Создаёт гит репозиторий в текущем каталоге.

-------------------------------------------------------------------------------------------------------

**git clone <url репозитория>**

Клонирует удалённый репозиторий на локальную машину

-------------------------------------------------------------------------------------------------------

**git remote -v**

Показывает список удалённых репозиториев, о которых знает гит:

<имя> <url>

По правилам гита, тот удалённый репозиторий с которого сделан git clone и в который будут идти push команды, называется origin.

**origin** — имя по умолчанию для удалённого репозитория, с которого сделан fork

**master или main** — имя по умолчанию для ветки при создании репозитория локально или на удалённом сервере.

**git remote add <имя для удалённого репозитория> <url удалённого репозитория>**

Если надо добавить удалённый репозиторий для отправки ваших изменений (url вашего fork-а) или для скачивания изменений кода другими разработчиками из главного репозитория (от которого сделан fork), то надо выполнить:

git remote add origin <origin url>

git remote add ibuko <url of kmms\_2023\_SmartSort>

-------------------------------------------------------------------------------------------------------

**.gitignore**

Файл, имя которого пусто, а расширение gitignore. Этот файл надо создать самостоятельно и положить в корень репозитория. Он содержит список регулярных выражений файлов и каталогов, за изменениями которых гит не должен следить.

Пример для Visual Studio:

.\*

Solution/.vs/

Solution/x64/

Solution/Имя проекта/x64/

\*.sln

\*.vcxproj

\*.vcxproj.\*

-------------------------------------------------------------------------------------------------------

**git status**

**git status -uall**

Если git status среди изменений выводит имя каталога, но не отображает все файлы этого каталога с их полными путями, то надо применить команду git status -uall

Показывает текущее состояние рабочего каталога:

1. Changes to be committed: (выводятся зеленым) — это разница между последним коммитом и индексом. Т.е., то, что попадёт в коммит.

2. Changes not staged for commit: (выводятся красным) — это разница между индексом и рабочей копией. Т.е., то, что в коммит не попадет.

-------------------------------------------------------------------------------------------------------

**git config --list**

Показывает настройки гита. Для того чтобы сделать коммит надо, чтобы были установлены user.name и user.email:

git config --global user.name “Имя Фамилия”

git config --global user.email “действующая почта”

-------------------------------------------------------------------------------------------------------

**git add . или <имена файлов>**

**Индекс** — это временный коммит, это черновик коммита, это промежуточное состояние изменений.

**Индексация** — это процесс добавления информации об изменённых файлах в базу данных гита.

*Что происходит с технической точки зрения?*

— Берётся файл, к нему дописывается служебная информация, высчитывает хэш, затем всё сжимается и сохраняется в .git/objects. Затем в файл .git/index добавляется запись о добавленном файле, вычисленный хэш и служебная информация.

-------------------------------------------------------------------------------------------------------

**git commit -m “[ibusko] комментарий по-английски”**

**Commit** — фиксировать (англ.)

**Коммит** — это основной элемент временной шкалы проекта Git. Коммит можно рассматривать как снимок состояния проекта.

При создании коммита вся информация о том, какие изменения должны быть зафиксированы, берётся из .git/index.

При выполнении команды git commit именно индекс, а не текущее состояние рабочей копии становится новым коммитом. Сразу после выполнения commit индекс совпадает с последним коммитом текущей ветки.

-------------------------------------------------------------------------------------------------------

**git push**

Отправляет новые коммиты на сервер в репозиторий, от которого сделан git clone (на тот url, на который указывает имя origin).

**git push --set-upstream <имя для удалённого репозитория> <имя ветки>**

При первом push-е ветки будет выдана ошибка ошибка, о том, что гит не знает какую ветку удалённого репозитория надо отслеживать для вышей локальной ветки. И надо выполнить предложенную команду. Затем push этой ветки можно делать просто: git push

**git push -f**

Иногда надо записать изменения в удалённый репозиторий, которые не содержат новых коммитов, например, вы эти изменения внесли командой git commit –amend. git отклоняет попытку push их с ошибкой, т. к. один и тот же коммит в локальном репозитории и удалённом имеет разный хэш:

error: failed to push some refs to 'https://github.com/BuskoIlya/kmms\_2023\_SmartSort.git'

В таком случае можно применить git push -f



-------------------------------------------------------------------------------------------------------

**git pull <имя удалённого репозитория> <имя ветки>**

git pull загружает содержимое из указанного удалённого репозитория и его ветки и соединяет их с текущей веткой:

git pull ibusko master

-------------------------------------------------------------------------------------------------------

**git branch**

Выводит список существующих веток

-------------------------------------------------------------------------------------------------------

Переключение между ветками **— git checkout <имя ветки>**

-------------------------------------------------------------------------------------------------------

Отмена локальных изменений до индексации —

**git checkout . или <имена файлов>**

**-------------------------------------------------------------------------------------------------------**

**Записать текущие изменения в последний коммит — git commit --amend**

Если сделанные изменения надо поместить в последний коммит, то их надо вначале проиндексировать. А затем выполнить команду git commit --amend.

**-------------------------------------------------------------------------------------------------------**

**Pull Request** — это запрос в системе github к автору оригинального проекта на принятие изменений от другого разработчика.

\*Не путайте pull request с командой git pull