Практические задания:

1.  Первым делом необходимо завести аккаунт на [https://github.com](https://github.com/) и создать ваш первый репозиторий. При создании вам необходимо выбрать имя репозитория, заполнить описание(опционально) и задать уровень видимости – в данном случае выбираем public, чтобы преподаватели смогли проверить домашнее задание.

2.  Далее необходимо на вашем локальном компьютере создать новую пустую папку. Внутри папки запустить git bash и создать README.md файл (круто, если сделаете это, не выходя из консоли git).

3.  Внутри README-файла заполните свое ФИО и информацию о своем любимом хобби.

4.  Возвращаемся в консоль и инициализируем локальный репозиторий

5.  Теперь нам необходимо сделать первый коммит в репозиторий. Добавляем наш новый README-файл к коммиту и создаем сам коммит. Не забудьте добавить к коммиту осмысленный комментарий.

6.  После создания коммита нам необходимо переключиться на ветку main, которая является веткой по умолчанию на github.com

7.  Далее нам необходимо подключиться к ранее созданному удаленному репозиторию на github.com из консоли и запушить в него наш первый коммит. В итоге на вашем репозитории в github.com должен появится README-файл.

8.  Теперь нам важно создать новую ветку, в которой мы будем вести разработку – создайте новую ветку develop и переключитесь на нее.

9.  Попробуйте вписать в ваш README-файл информацию о вашем образовании и закоммитить изменения в новую ветку.

10.  Откройте в консоли список последних коммитов, запишите номер и время последнего коммита в ваш README-файл и запуште изменения в ветку develop.

11.  Напишите в чат вашей группы сообщение с просьбой, чтобы кто-нибудь из коллег поделился ссылкой на свой репозиторий с вами. Склонируйте репозиторий вашего коллеги на ваш локальный компьютер и добавьте в README-файл вашего репозитория ссылку на репозиторий коллеги и номер его последнего коммита.

mkdir test-project # создаем новую папку для проекта  
cd test-project # переходим в папку проекта  
git init # инициализируем локальный репозиторий  
touch README # создаем новый файл README  
git add README # добавляем его к версионированию  
git commit -m 'first commit' # делаем commit с коментарием о первом комите  
git remote add origin http://gitlab.@example.com/Admin/test-project.git # добавляем адрес удаленного репозитория в HTTP формате  
git push -u origin master # отправляем изменения на сервер

Без добавления файла в Индекс у нас не получится создать коммит, проверьте это сами с помощью комманды:

Для добавления файлов в Индекс используется следующая команда:

git add README

echo "67890" >> README

git status

git log

git commit -m "Второй коммит"

git log -p -2

git checkout b66f9

Представим, что нам надо посмотреть, как выглядел наш репозиторий после второго коммита. Для этого используем команду checkout и хэш второго коммита. Кстати, можно не указывать его целиком, достаточно первых 5 символов.

git switch -c Two\_new-branch-name

git commit -m "Three коммит"

git switch -c Three \_new-branch-name

**git merge --no-ff** Three\_new-branch-name

git checkout -b new\_branch

git commit -m "some commits"

git push origin new\_branch

git checkout -b my\_new\_branch\_4

git pull origin new\_branch

git push origin my\_new\_branch\_4

git checkout master

git merge my\_new\_branch\_4

git add --all

git checkout master

git push origin master

Бикмуллина Ильсияр Ильдаровна

робототехника, АИС

Какую специальность получил в результате окончания учебного заведения:

1) КНИТУ-КАИ, Автоматизированные системы обработки информации и управления (инженер)

2)КНИТУ-КАИ, «Информатика и вычислительная техника» (магистр техники и технологии)

3)к.т.н.

23.10.2267890

Другие пользователи не увидят вашу ветку, пока она не будет отправлена на удаленный репозиторий. Поэтому, после того как вы слили все изменения в master , нужно отправить их в GitHub. Для этого **обязательно** нужно находиться в ветке master :

git checkout master

# Отправляем наши изменения в GitHub

git push origin master

**git merge --no-ff some-branch-name**

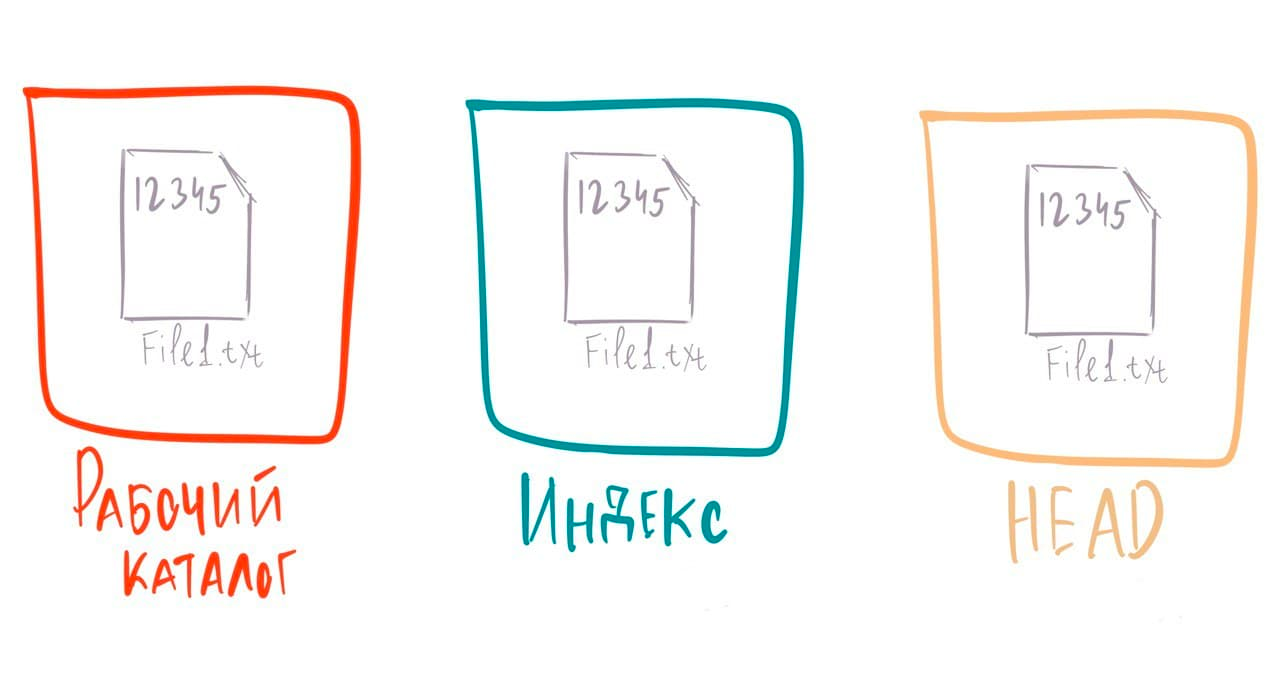
git switch -c <new-branch-name>

Когда у вас много файлов, вы можете добавить их все разом

git add --all.

Все, коммит готов. И файл попал в область HEAD.

Если сейчас выполнить git status, то мы не увидим никаких изменений, так как все три области одинаковые.



Теперь мы хотим внести изменения в файл и сделать новый коммит. Мы пройдём через ту же процедуру: сначала отредактируем файл в нашем рабочем каталоге. Давайте называть эту версию файла v2 и обозначать зеленым цветом. Допишем в файл цифры 67890.

Предположим, было так:

A - B - C master

\

D new\_branch

Что делает git checkout -b

Эта команда создаёт новую ветку, указывающую на тот же коммит, на который указывает текущая ветка:

git checkout -b my\_new\_branch\_2

A - B - C master

\

D new\_branch, my\_new\_branch\_2

То есть после выполнения этой команды ветки new\_branch и my\_new\_branch\_2 идентичны.

Что делает git pull origin имяветки

Команда git pull производит мерж ветки из удаленного репозитория в текущую ветку. Если замержить коммит, который уже принадлежит текущей ветке (т.е. доступен по цепочке предков), то слияние не произведёт изменений.

Если никто кроме вас не пушит в ветку new\_branch, то она указывает на тот же коммит D, что и ветка my\_new\_branch\_2, так что эта команда ничего не делает:

git pull origin new\_branch

Мерж предка

Если после этого сделать ещё коммиты в my\_new\_branch\_2, но не менять new\_branch, то вторая фактически будет «частью» первой. Ветка new\_branch указывает на коммит D, который является предком E, а следовательно принадлежит ветке my\_new\_branch\_2:

A - B - C master

\

D new\_branch

\

E my\_new\_branch\_2

Когда вы мержите my\_new\_branch\_2 в master, в составе этой ветки приходят и коммиты ветки new\_branch. Мерж скорее всего у вас происходит по модели "fast-forward" – когда указатель master просто перемещается вперёд на указатель my\_new\_branch\_2:

A - B - C - D new\_branch

\

E my\_new\_branch\_2, master

Поэтому попытка повторно замержить new\_branch в master не приносит результата.