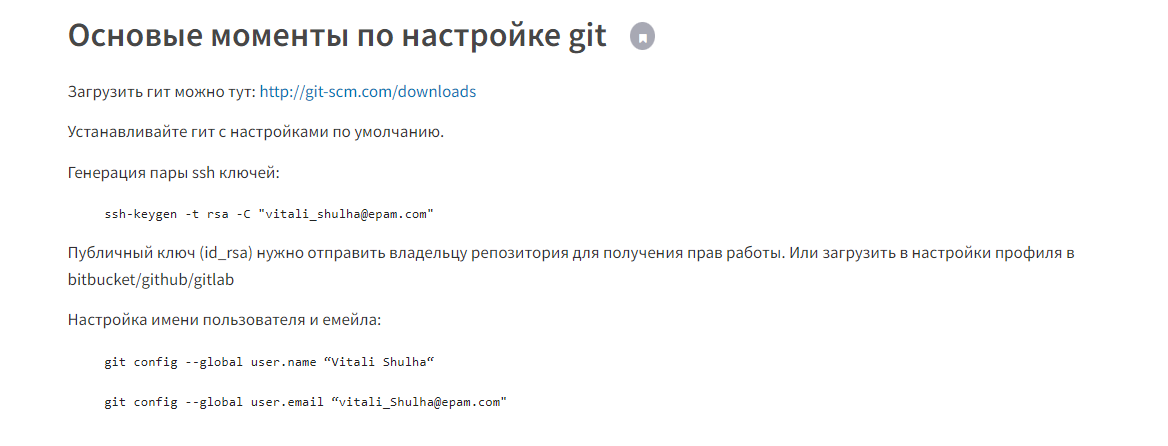
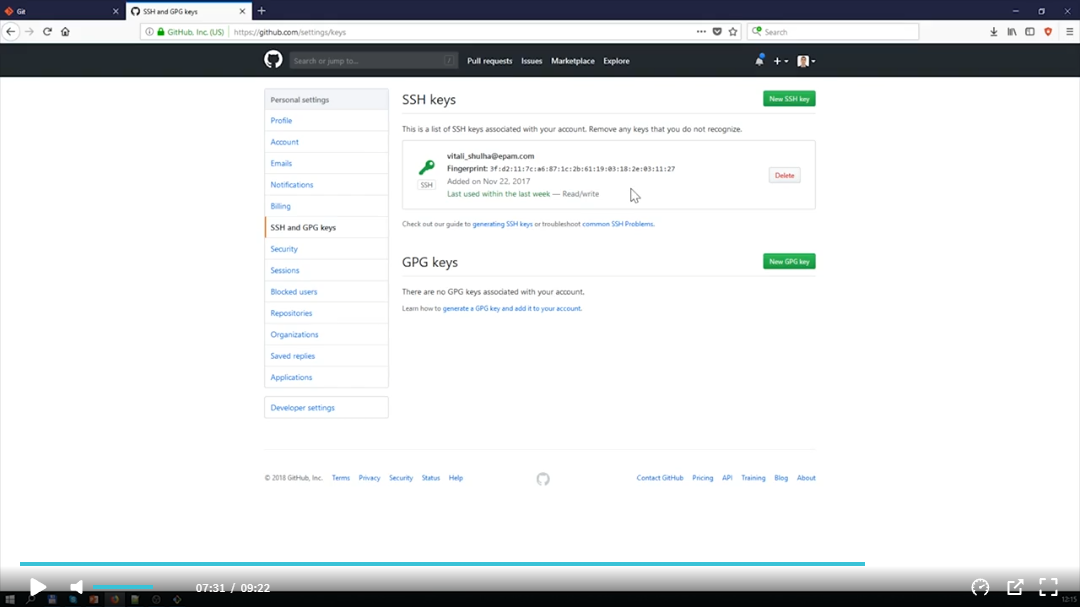
**Подготовка к работе с Git Bash**



Ключи по умолчанию будут помещены в директорию /c/users/user/.ssh/



Затем нужно на веб-хостинге добавить публичный ключ как показано на картинке

**Основные команды Git**

**Установить путь к текущему рабочему каталогу:**



Команда: cd; параметр – путь каталога(cd /… - абсолют. , cd … - относ.), написание ‘/’ конце означает зайти внутрь папки

(cd – сокращение от change directory)

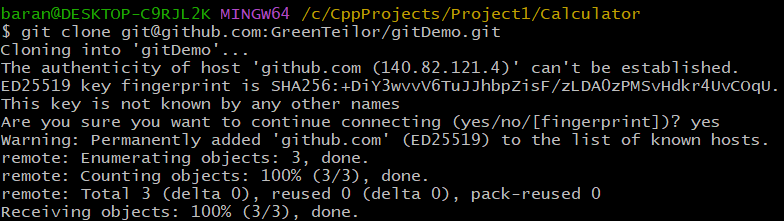
**Вывод пути к текущему рабочему каталогу:**



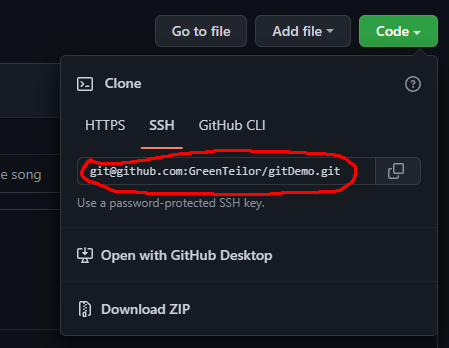
Команда: pwd; параметры – отсутствуют

(pwd – сокращение от present working directory)

**Клонирование репозитория на локальную машину:**



Команда: git clone; параметр – адрес репозитория(можно взять на веб-хостинге, пример на картинке ниже)



(git clone – в расшифровке не нуждается)

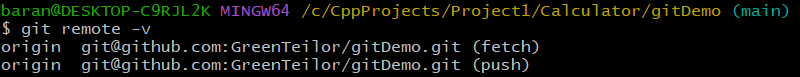
**Команда просмотра содержимого каталога:**



Команда: ls; параметры – отсутствуют

(ls – сокращение неизвестно)

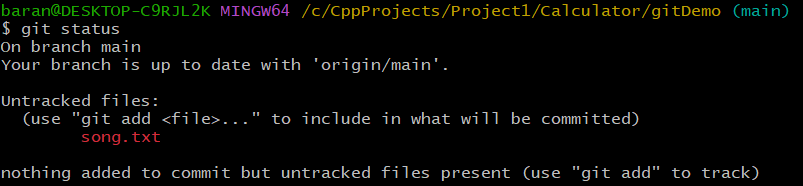
**Проверка нахождения источника на github:**



Команда: git remote -v; параметры неизвестны

(git remote -v – в расшифровке не нуждается)

**Проверка состояния локального репозитория:**



Команда: git status; параметры – отсутствуют, если добавлены файлы, коммит которых проведен не был, то информация будет выведена

(git status – в расшифровке не нуждается)

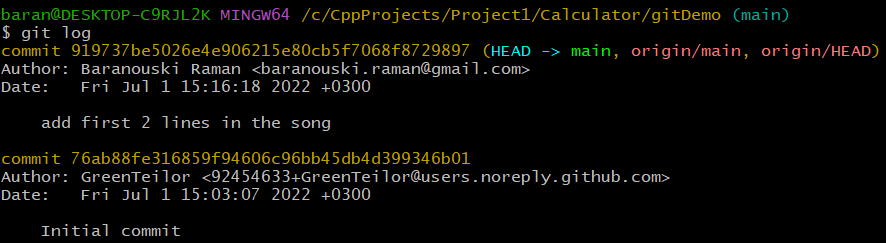
**Добавление файла в коммит:**



Команда git add; параметр – имя файла

(git add – в расшифровке не нуждается)

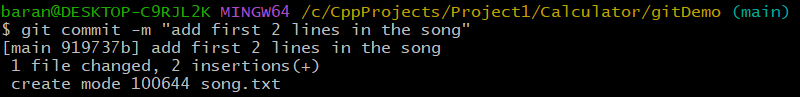
**Просмотреть историю коммитов:**



Команда git log; параметры – отсутствуют

(git log – в расшифровке не нуждается)

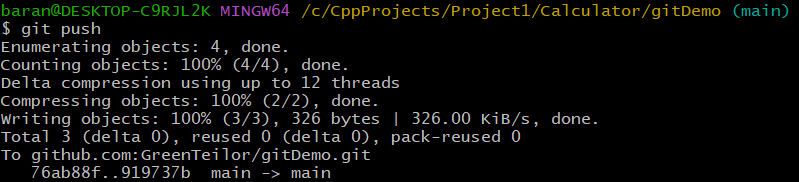
**Сделать коммит:**



Команда git commit; параметры – отсутствуют, -m – флаг для добавления сообщения

(git commit – в расшифровке не нуждается)

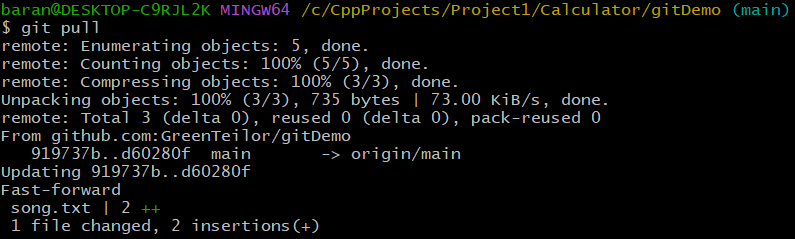
**Сделать пуш:**



Команда git push; параметры отсутствуют

(git push – в расшифровке не нуждается)

**Обновить локальный репозиторий:**



Команда: git pull; параметры – отсутствуют

(git pull – в расшифровке не нуждается)

Небольшое объяснение:

Git clone – клонировать репозиторий на локальную машину

Git add – для добавления файлов в коммит

Git commit – для сохранения изменений в текущем репозитории

Git push – для сохранения в удаленный репозиторий

**Отмена изменений**

**Откат изменений в одном файле:**



Команда: git checkout --; параметр – имя файла

(git checkout – в расшифровке не нуждается)

**Откат изменений всех файлов текущей директории и ниже:**



Команда: git checkout .; параметры – отсутствуют

(git checkout – в расшифровке не нуждается)

**Отмена создания новых файлов:**



Команда: git clean -xdf; параметры – отсутствуют

(git checkout – в расшифровке не нуждается), xdf – флаги

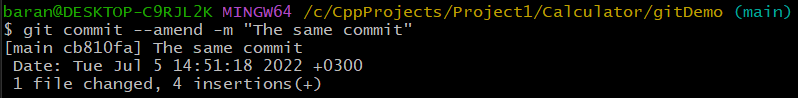
**Откат в unstaged файлы, которые уже были добавлены в коммит:**



Команда: git reset --; параметр – имя файла

(git reset – в расшифровке не нуждается)

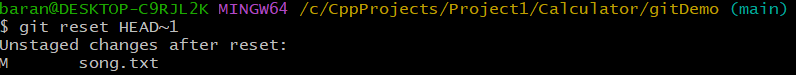
**Дополнить последний коммит:**



Команда: git commit –amend -m “message”

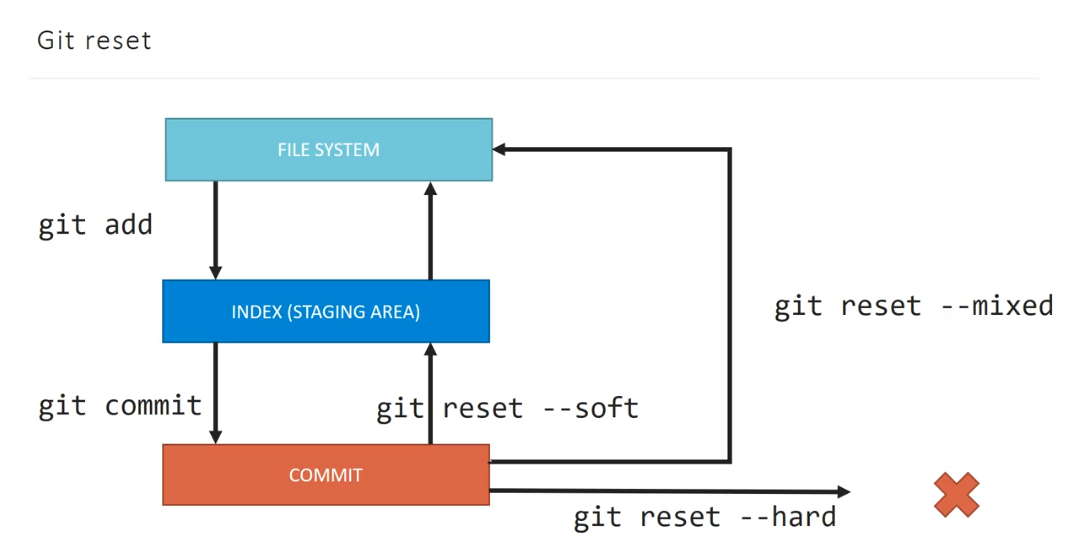
(git commit – в расшифровке не нуждается)

**Отменить последний коммит:**



Команда: git reset HEAD~commits\_number; параметр – количество отменяемых коммитов

(git reset – в расшифровке не нуждается)



Картинка демонстрирует работу с командой reset, используя различные флаги(по умолчанию используется mixed)

**Отмена пуша на удаленный репозиторий:**



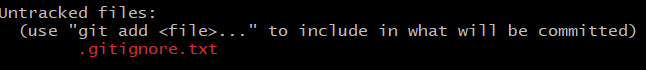
Команда: revert; параметр – номер коммита (делает коммит, противоположный предыдущему)

После использования для сохранения изменений на удаленном репозитории необходимо сделать пуш, для выхода из редактора wq (или q! при нежелании сохранять изменения)

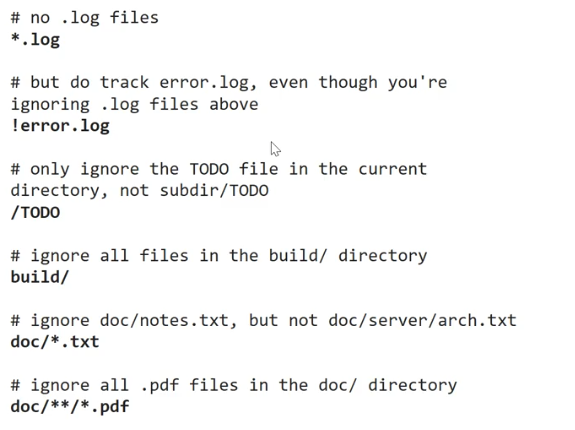
(git revert – в расшифровке не нуждается)

**GitIgnore:**

Необходимо создать файл без расширения с именем gitignore, это позволяет скрывать от гита файлы, которые не нужно коммитить.



Синтаксис .gitignore:



Пример содержимого .gitignore :

 или 

**Ветки**

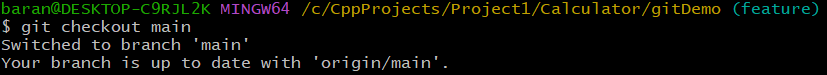
**Создание новой ветки и переключение на нее:**



Команда: git checkout -b <name>; параметр – имя ветки(branch)

(git checkout – в расшифровке не нуждается)

**Переход между ветками:**



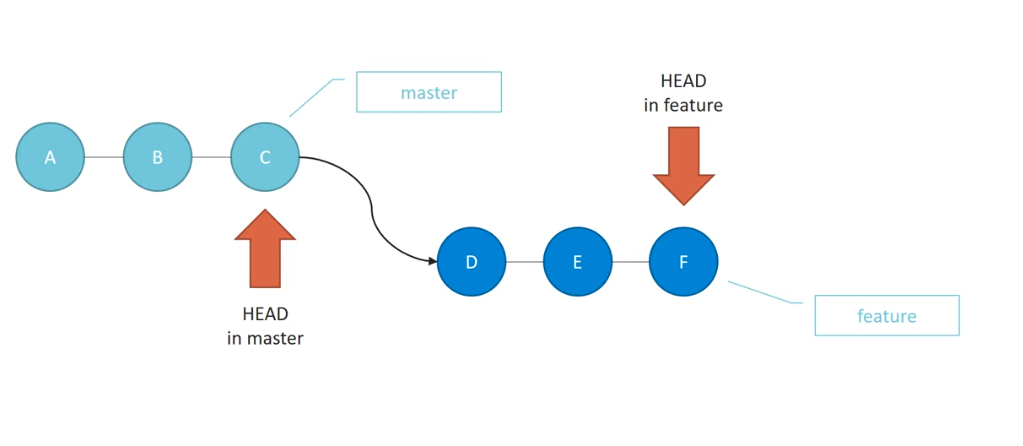
Команда: git checkout <name>; параметр – имя ветки(branch)

(git checkout – в расшифровке не нуждается)

**Синхронизация веток путем fast-forward:**



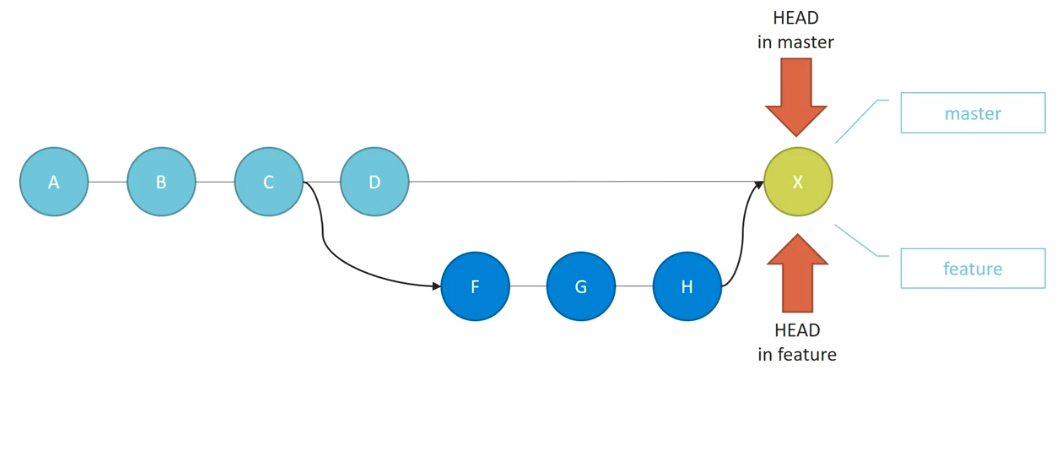
Команда: git merge <name>; параметр – имя ветки, с которой нужно синхронизировать, синхронизация происходит путем перекидывания головы с главной на голову ветки, таким образом применять git merge нужно из ветки, которую нужно синхронизировать:



(git merge – в расшифровке не нуждается)

**Синхронизация веток путем non fast-forward:**

Иллюстрация данного метода приведена ниже:



Делается так же, как и fast-forward, но затем вылезает редактор, в котором можно что-то сделать, но еще можно просто выйти и гит сам сделает merge по одной из стратегий

**Руководство к разрешению конфликтов:**

Для разрешений конфликтов необходимо зайти в файлы с конфликтами и отредактировать так, как считаешь правильным.

В противном случае можно воспользоваться одним из следующих вариантов:



Правильная полная запись 2 команды: git checkout --ours/theirs <path>

<path> - путь к файлу, изменения в котором нужно принять

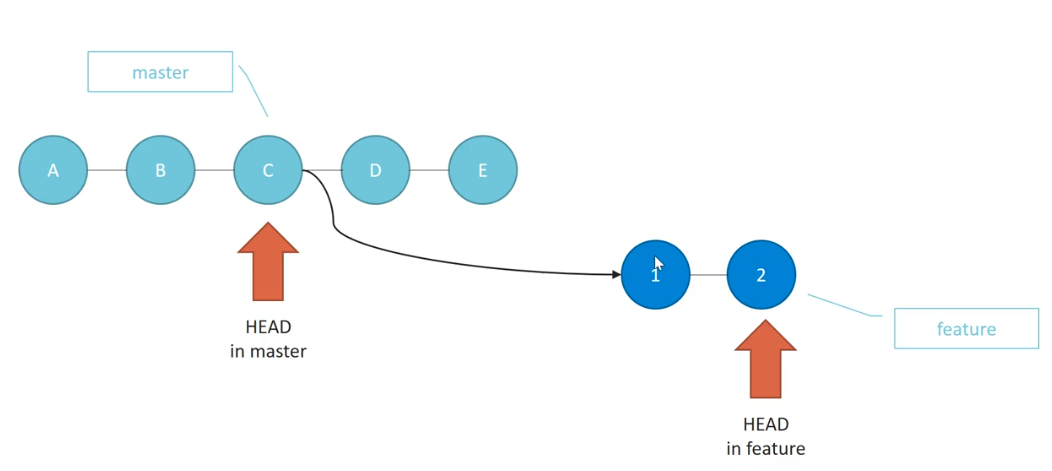
**Команда rebase:**



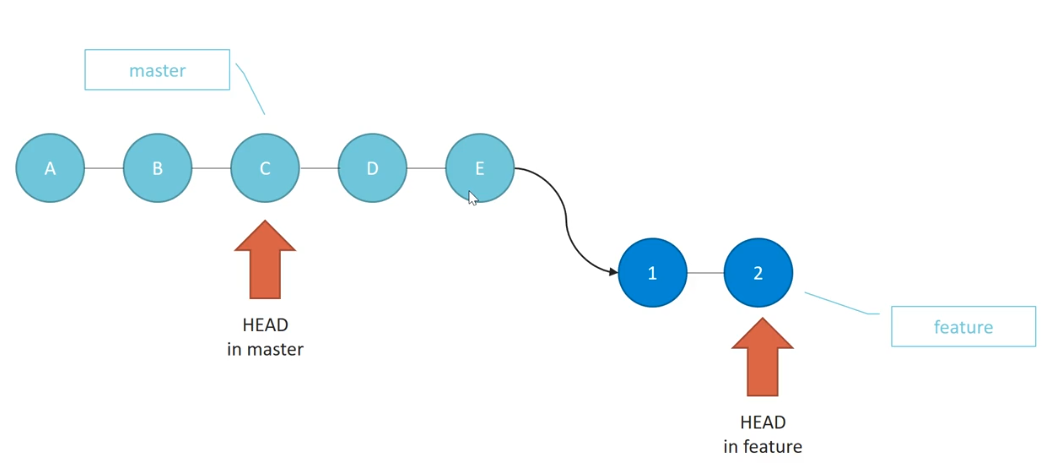
Команда: git rebase; параметр – имя ветки

Данная команда служит для синхронизации дополнительной ветки с основной. Иллюстрация:

Было до rebase:



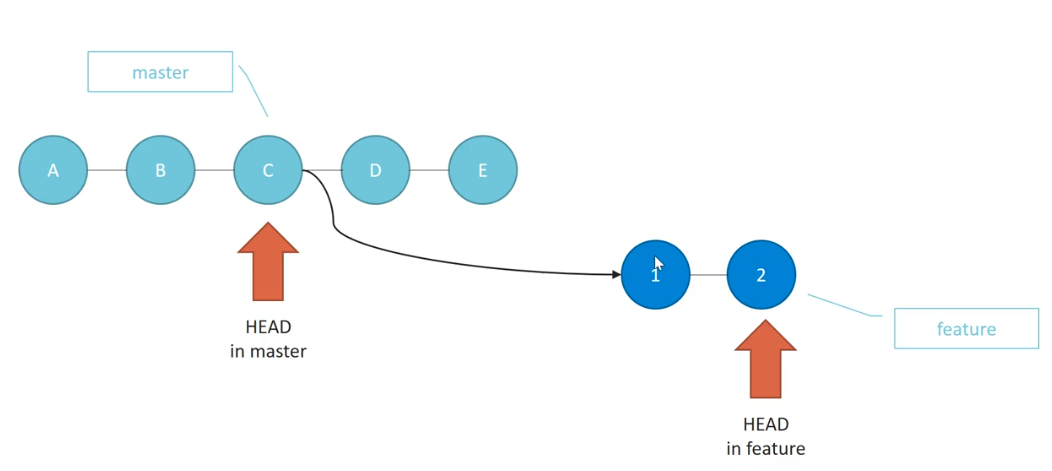
Стало после rebase:



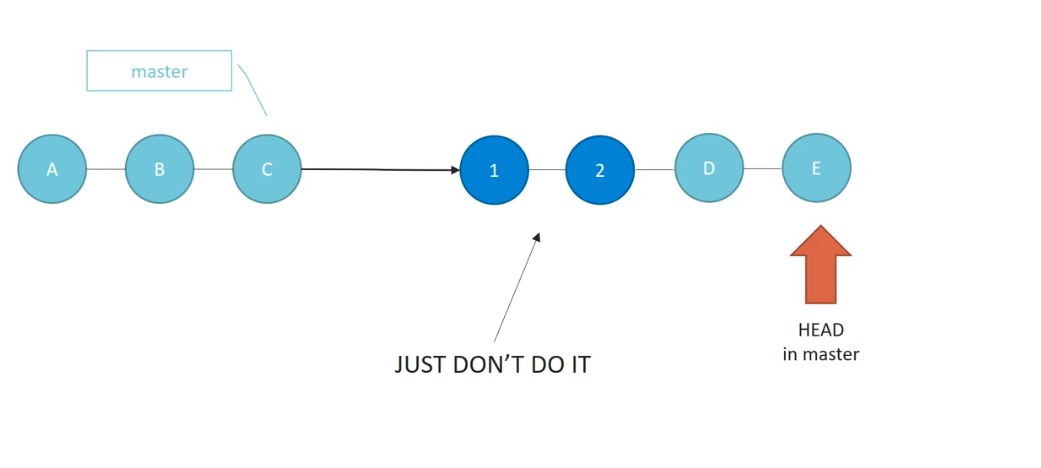
ВАЖНО!:

Нельзя использовать rebase в основных ветках, т.к. это нарушает последовательность коммитов и путает разработчиков. Иллюстрация:

До применения rebase из main:

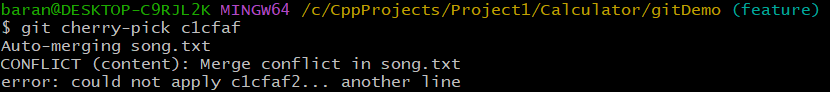


После применения rebase из main:



(rebase – в расшифровке не нуждается)

**Команда cherry-pick:**



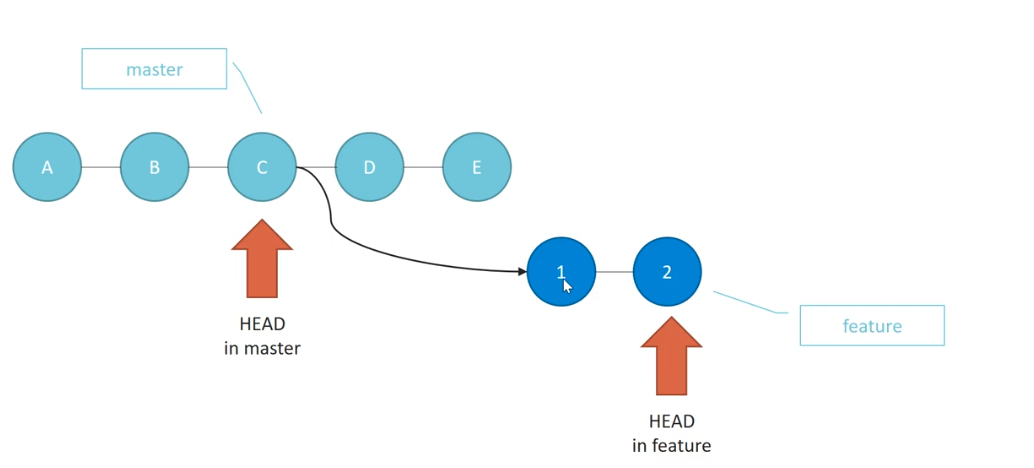
Команда: cherry-pick; параметр – номер коммита

Используется для внесения изменения в ветку только из одного коммита, т.е. cherry-pick – команды, из которых состоит rebase

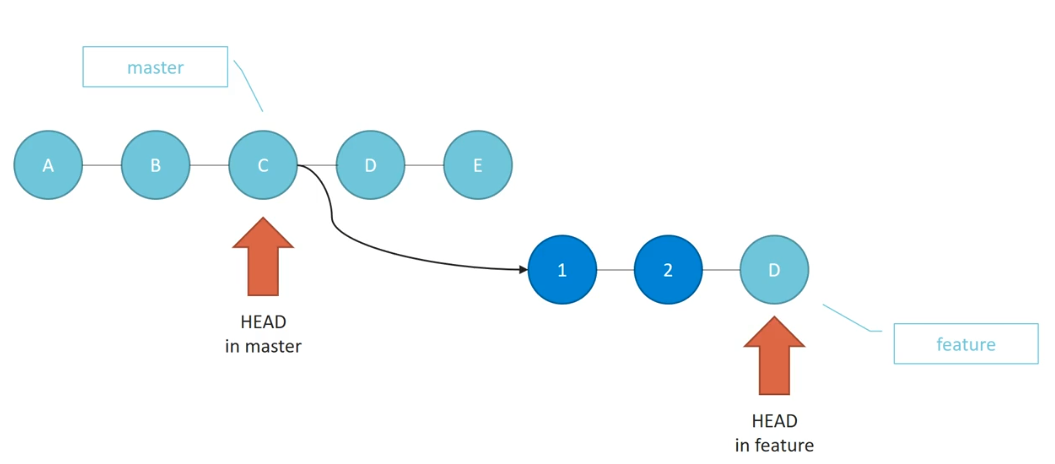
(cherry-pick – в расшифровке не нуждается)

Иллюстрация работы cherry-pick:

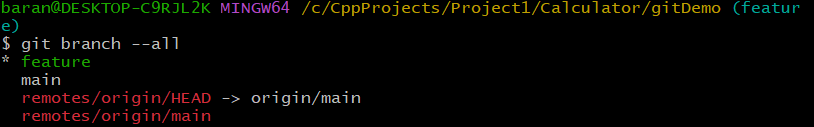
До использования:



После использования:



**Просмотр всех веток:**



Команда: git branch --all; параметры – отсутствуют

(git branch – в расшифровке не нуждается)

**Дополнительные фишки**

**Теги:**



Команда: git tag <name>; параметр – название тега

(git tag – в расшифровке не нуждается)

**Просмотр всех тегов:**



Команда: git tag --list; параметры – отсутствуют

(git tag – в расшифровке не нуждается)

Для передания тегов на удаленный репозиторий используется:



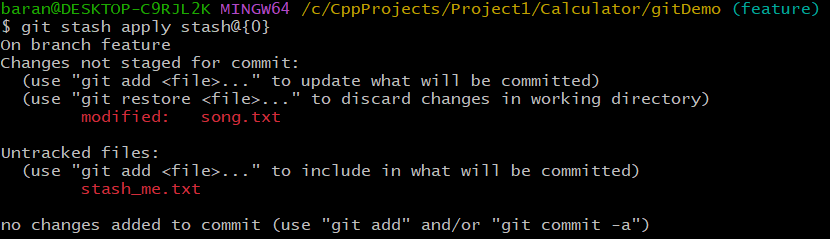
**Сохранение в stash изменений:**



Команда: git stash save “<description>”’; параметр – описание стэша

Используется для сохранения модифицированных файлов(не добавленных), которые пока не хочется откатывать или коммитить.

**Извлечение из stash:**



Команда: git stash apply <name> или git stash pop <name>; параметр – имя стэша

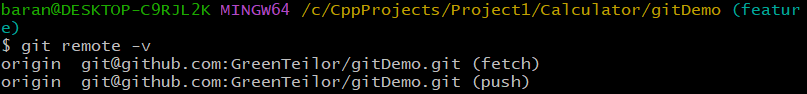
В первой команде изменения переносятся в файлы, но не пропадают из стэша, во второй команде изменения переносятся в файлы и пропадают из стэша.

**Удалить определенный stash:**



Команда: git stash drop <name>; параметр – имя стэша

**Связь с удаленным репозиторием**

**Проверка удаленного репозитория:**

Команда: git remote -v; параметры – отсутствуют

**Удаление связи с удаленным репозиторием:**



Команда: git remote remove origin; параметры – отсутствуют

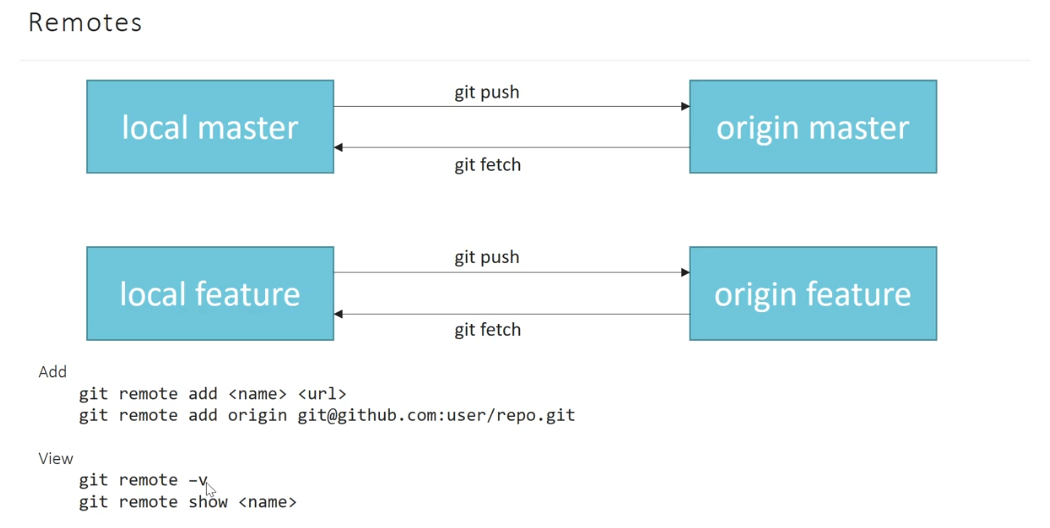
**Добавить удаленный репозиторий:**



Команда: git remote add origin <name>; параметр – адрес репозитория

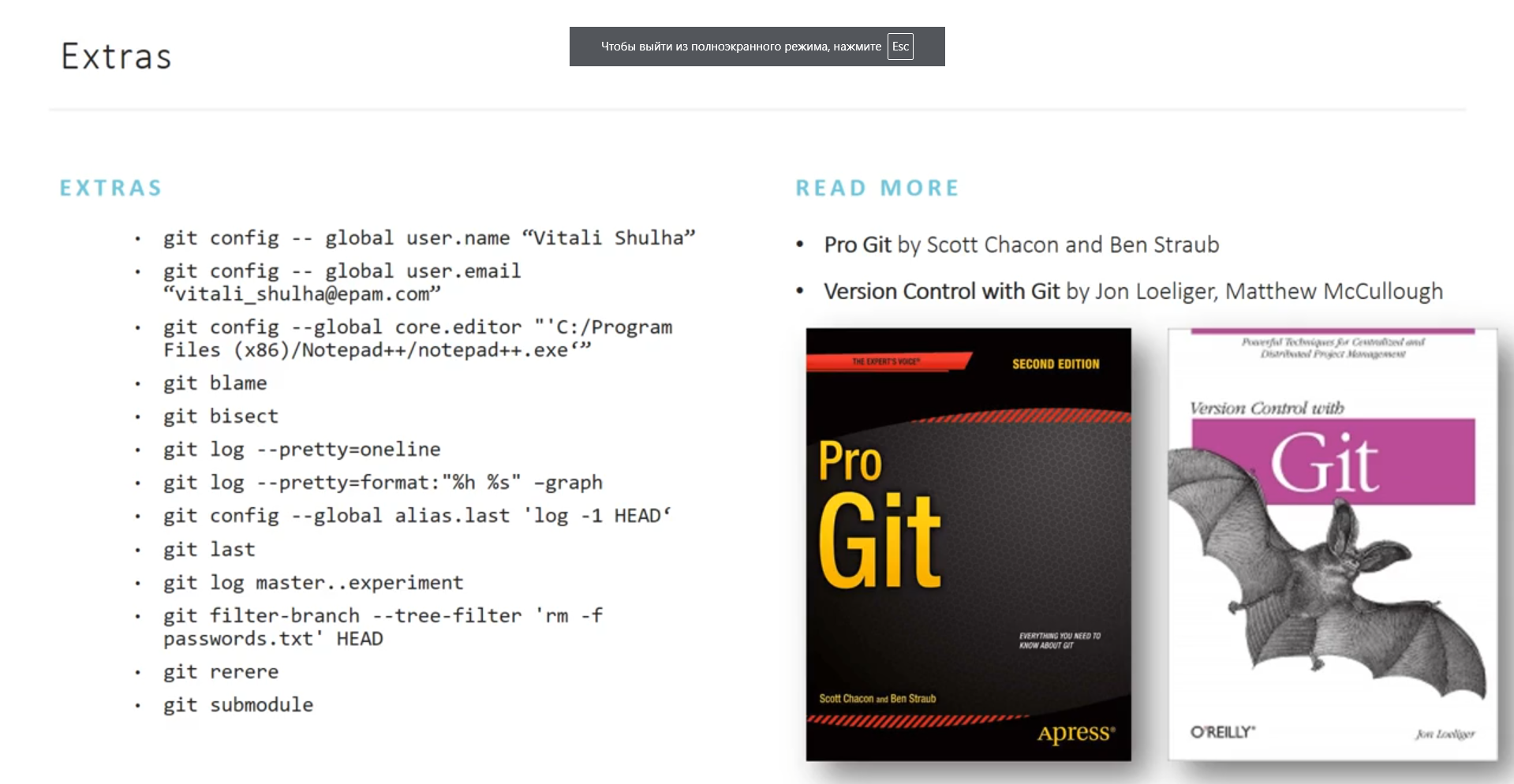
Затем для синхронизации необходимо произвести обычный пуш

Памятка по git remote:



**Полезные книги**

Две книги: первая – бесплатная, на сайте git.



**Про команды:**

1. Настройка имени
2. Настройка email
3. Настройка желаемого редактора, вместо базового
4. Параметр – файл. Позволяет узнать, кто редактировал файл
5. Позволяет найти коммит с помощью бинарного поиска
6. Позволяет посмотреть все коммиты в одной строке
7. Красивый график с ветками, коммитами
8. Остальные: почитать