|  |  |
| --- | --- |
| **git config --global user.email "git-user@selectel.ru"**  **git config --global user.name "Selectel Git User"** | имя и e-mail для записи в коммиты |
| **mkdir first-repo && cd first-repo** | **создать каталог, перейти в каталог** |
| **git init** | Сделать каталог репозиторием |
| copy NUL .gitconfig | инфа . убедиться что встал |
| git config --list –global | проверяем инфу в файле .gitconfig |
| **touch** README.md | Создать файл |
| **git add** README.md | **добавляем файлы в гит чтобы они отслеживались** |
| **git add .** | Относительный путь. будет обработано содержимое только текущей директории |
| **git commit** | Описание и сохранение изменений (контрольные точки) |
| **git commit -m "create and modified file File1.txt"** | Фиксируем изменения с подробным описанием что сделали |
| **git commit –m** и **Git push** | Синхронизируем файлы локальный и удаленный |
| **git commit -a -m "new commit today 11.32"** | Добавляем все файлы (a) и сообщение для коммита (m) |
| **git reset Head~1** | Откат изменений последнего коммита |
| Git status | Текущее состояние данных и веток |
| **git log** | информация по всем сделанным коммитам |
| **git log -1** | инфа по последнему коммиту |
| **git log - -oneline - -all - -graph** | Подробно инфа по коммитам и номер хэш выдан сокрашенно |
| **git log --author rimma** | См изменения определенных авторов |
| **git show** номер хэша | См содержание определенного комита |
| **git show** | инфа по самому последнему комиту |
| **Ctrl + C** или **Q и кнопка Enter** | выйти из просмотра коммитов |
| **git blame File1.txt** | инфа по автору комита, какие изменения внес |
| **git blame File1.txt|grep 123** | Кто вносил изменения по конкретной строке в файле |
| **git blame File1.txt|grep rimma** | какие строки вносил конкретный автор в этот файл. |
| **echo "add line" >> File1.txt** | добавить строку в файл |
| **git diff** | см разницу между версиями файла после изменений |
| **cat** File1.txt | смотрим содержимое файла |
| **git checkout** file.txt | Откатываем изменение для одного файла |
| **git checkout** | Откатываем изменение для нескольких файлов |
| git stash | помещаем во временное хранилище  строки больше не видно в файле |
| git stash pop | Вернуться к изменениям |
| Git stash clear | Удалить строку навсегда из временного хранилища |
| **сd** имя папки | Сделать корневым каталог |
| **mv** file-name new-directory-name**/** | переместить существующий файл в папку |
| git **add** new-directory-name/ | проиндексировать изменения |
| git **rm -r** directory-name/ | удаляет папку с содержимым |
| git **mv** testmain.c main.c | Переименуем файл testmain.c на main.c |
|  |  |
| **git branch** first\_branch | Создать ветку |
| **git branch** | Проверить что она есть (основная ветка помечена \*) |
| **git checkout** first\_branch | Переключиться на новую ветку |
| **git switch -c** <new-branch-name> | Создать новую ветку и переключиться на нее |
| git push –u origin first\_branch | Сохраняем изменения в гит хаб  Связываем локальную ветку с удаленной  если ее еще нет удаленно |
| **git checkout - или git checkout** main  **git merge** first\_branch | Закончили вносить изменения в новую ветку и хотим залить изменения в основную ветку main.  Для этого Вернуться в основную ветку.  Заливаем новую ветку в основную. |
| **git branch** –m first\_branch frbranch | Переименовать ветку first\_branch в frbranch (если пустая) |
| **git branch** –m first\_branch frbranch  git push origin : first\_branch  git push –u origin frbranch | 1Переименовать ветку (если с файлами)  2На гит хабе не переименовалась Поэтому **удаляем ветку** first\_branch  Обращаемся к ней.  3 Добавляем новое название |
| **git branch** –d first\_branch | **Удалить** ветку из **локального** репозитория. |
| git fetch origin first\_branch | Вернуть ветку если удалена |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
| GIT HUB |  |
| **gitk&** | запуск инструмента для отслеживания ветвей |
| **git clone https://github.com/rimma-er/Rm\_storage.git** | Копируем репозиторий в локальную папку |
|  |  |
| **git push** | отправить файл на удаленный репозиторий |
| **git pull** | вносит изменения из удаленного репозитория в локальный |
| **git fetch** | показывает что есть изменения в вебе, ничего не меняет |
| git merge --abort | Отменить слияние файлов/веток |
|  |  |
| **git push origin --delete** namebranch  или  **git push origin** :<branchName> | **Удалить** ветку из **удаленного (github)** репозитория: |
| **git branch –r** | показывает список удаленных веток |
| **git branch –a** | показывает как локальные, так и удаленные ветки |
| **git fetch --prune** | подчищает локальные ссылки, после удаления веток, которым больше нет соответствий в удаленном репозитории. |
|  |  |
| **git cherry-pick** | выборочно применить коммиты Git к текущей рабочей ветке |
| **git cherry-pick** *хэш коммитаF* | Переключиться на ветку, в которую вставляем коммит |
| **git cherry-pick** *хэш коммитаF* **- -edit** | git запросит новый коммит меседж для комита который переносим в новую ветку. |
| **git cherry-pick** *хэш коммитаF --***no-commit** или  **git cherry-pick** *–***n** *хэш коммитаF* | вместо создания нового коммита в ветке master, помещает изменения коммита в рабочую директорию. Тогда в комите можно что-то доработать/изменить. |
| **git cherry-pick –x** | добавить в описание коммита строку с информацией откуда был скопирован |
| *хэш коммитаFкопия* | Посмотерть разницу комита и его копии |
|  | У комиита есть **имя и дата автора** кто его придумал, имя и дата автора кто закомитил его в репозиторий. Посмотреть и того и другого |
|  | Скопировать все изменения из ветки F в мастер с перечислением хэшей комитов |
|  | Скопировать все изменения из ветки F в мастер (**все комиты F которых нет в мастер**) указав диапазон аналогичный команде git log |
|  | Отменить уже скопированые комиты. Вернуть все как было до команды |
|  | Остановиться там, где мы сейчас и сбросить(удалить) запомненное состояние |
| **Git reset - -hard @~2** | Отменить уже завершенный cherry-pick  *cherry-pick - -abort не сработает т.к. прекращает только не оконченное копирование*  *Если скопирован 1 коммит , то жестко один шаг назад. Если* ***2****, то два шага.* |
| Если не понятно что произошло: |  |
|  |  |
|  |  |