Первоначальная настройка

$ git config --list    // вывод списка настроек

$ git config --global user.name %имя%    // настройка информации о пользователе для всех локальных репозиториев

$ git config --global user.email %адрес\_электронной\_почты%    // устанавливает адрес электронной почты, который будет отображаться в информации о выполняемых вами коммитах

Команды линукс

$ ls    // выводит на экран список файлов хранящихся в рабочей директории

$ ls -a    // выводит на экран список всех файлов хранящихся в рабочей директории

$ cd    // навигация по файловой системе

$ cd ~    // переход в домашнюю директорию

$ cd ..    // подняться на уровень выше

$ cd ..    // подняться на уровень выше

$ cd ../..    // подняться на два уровня выше

$ mkdir %имя\_папки%    // создание новых директорий

$ touch %имя\_файла%    // создание пустого файл

$ cp %файл\_источник% %файл\_назначения%    // копирование файлов

$ mv %copiedme% %movedme%    // изменяется только название файла («copiedme» –> «movedme»)

$ echo "firstfile"    // выводит переданный ей аргумент на стандартное устройство вывода (текстовый терминал в нашем случае)

$ echo "firstfile" > %название\_файла%    // записать строку в наш файл (перенаправили вывод с помощью символа больше >)

$ cat    // выводит содержимого файла в терминал

$ rm %название\_файла%    // удаление файла

$ rm -r %имя\_директории%    // удаление директории

$ clear    // очистка консоли

:wq    // w = write (записать), q = quit (выход)

Git работа с репозиторием

$ git init    // создание локального репозитория

$ git status    // статус репозитория

$ git remote add origin %http\_ссылка\_на\_удалённый\_репозиторий%    // добавление удалённого репозитория (для синхронизации), где origin название удалённого репозитория.

$ git remote -v    // список подключенных remote соединений

Определения

коммит - запись текущего состояния в историю изменений

Работа с коммитом

$ git add %название\_файла%    // добавление файла в список отслеживаемых файлов

$ git rm --cached %название\_файла%    // убирание файла из списка отслеживаемых файлов

$ git commit    // создание коммита, потом пишет комментарий, нажали на ESC , :wq

$ git commit -m '%комментарий%'    // создание коммита с комментарием

$ git commit -a    // комбо из двух команд: git add и git commit

$ git commit -am '%комментарий%'    // комбо из двух команд: git add и git commit -m

Системные файлы

.gitkeep - файл, который лежит в пустой директории. Это указывает git, что нужно следить за ней.

.gitignore - файл со списком директорий и файлов, которые git должен игнорировать.

Работа с коммитами

$ git log    // просмотр всех коммитов

$ git log --online    // просмотр всех коммитов в компактном режиме

$ git reset %хэш\_коммита%    // возвращение состояния до коммита, у которого хэш равен %хэш\_коммита%

$ git reflog    // просмотр всех коммитов и операций с ними

$ git revert %хэш\_коммита%    // создаёт новый коммит, отменяющий коммит с хешем %хэш\_коммита%

$ git restore    // сбрасывает состояние файла на состояние последнего коммита

Работа с ветками

$ git branch    // список веток

$ git branch %название\_новой\_ветки%    // создание новой ветки %название\_новой\_ветки% из текущей ветки

$ git branch %название\_новой\_ветки% %название\_ветки\_донора%    // создание новой ветки %название\_новой\_ветки% из ветки %название\_ветки\_донора%

$ git checkout %название\_ветки%    // переключиться на ветку %название\_ветки%

$ git checkout -b %название\_новой\_ветки%    // комбо из двух комманд (создание новой ветки и переключение на неё):   git branch %название\_новой\_ветки%   и   git checkout %название\_новой\_ветки%

Работа с репозиториями

$ ssh-keygen    // генерация SSH-ключа в нашем локальном каталоге

$ git clone %SSH-путь\_до\_репозитория% %имя\_директории\_для\_репозитория%    // клонирование удалённого репозитория в локальную папку

$ git push    // загрузка всех изменений в удалённый репозиторий

$ git push --set-upstream origin dev    // создание ветки dev в удалённом репозитории

   git push -u origin dev

$ git pull    // вытягивание изменений из удалённого репозитория

$ git fetch    // вытягивание информации о репозитории, например, о новых ветках и тегах

Слияние веток

$ git merge %название\_ветки\_донора%    // все изменения из ветки %название\_ветки\_донора% слить в текущую ветку

$ git branch -d %название\_ветки\_1% [, %название\_ветки\_2%]    // удаление веток локально

$ git push --delete origin %название\_ветки\_1% [, %название\_ветки\_2%]    // удаление веток из репозитория origin

$ git rebase %название\_ветки%   // смена старта ветки на указатель ветки %название\_ветки%

$ git push --force    // git push после rebase: перезапись истории коммитов

Версионирование

$ git tag    // список тегов. вариант 1

   git tag -l    // список тегов. вариант 2

   git tag --list    // список тегов. вариант 3

$ git tag %название\_нового\_тега%    // создание тега с именем %название\_нового\_тега%

$ git push --tags    // загрузка всех изменений + тегов

$ git tag -d %название\_старого\_тега%    // удаление тега

$ git push --delete origin %название\_тега\_в\_origin%    // удаление тега из удалённого репозитория

Pull request

Pull request это запрос на слияние. Можно создать либо в GitHub, либо в GitKraken.

Временное хранилище

$ git stash    // сохранить локальные изменения во временном хранилище, для безболезненного переключения на другую ветку

$ git stash list    // вывод списка временных хранилищ

$ git stash push %название файла%    // добавить ещё один файл в текущее временное хранилище

$ git stash pop %название\_сохранённых\_изменений%    // применение изменений из временного хранилища

Объединение нескольких коммитов в один

$ git merge %название\_ветки\_с\_несколькими\_коммитами% --squash    // вариант 1: во время слияния указываем ключик squash

$ git rebase HEAD~3    // rebase в автоматическом режиме с объединением коммитов, где 3 - число последних коммитов, которые нужно объединить.

$ git rebase -i HEAD~3    // rebase в интерактивном режиме.

Взятие одного коммита

$ git log --oneline --all    // получение списка всех коммитов со всех веток

$ git cherry-pick %хеш\_желаемого\_коммита%    // перенос в текущую ветку коммита, у которого хеш равен %хеш\_желаемого\_коммита%

$ git cherry-pick %название\_ветки%    // перенос последнего коммита из ветки %название\_ветки% в текущую

$ git cherry-pick ..%название\_ветки%    // перенести все коммиты из ветки %название\_ветки%

Перезапись последнего коммита

$ git commit --amend    // перезаписать последний коммит

Решение конфликтов

$ git merge dev1 dev2    // слияние двух веток

$ git merge --abort    // отмена слияния