|  |  |
| --- | --- |
| Git комманды | |
| git --version | смотрим установлен ли гит и какой он версии |
| git update-git-for-windows | обновление гит |
| git status | узнаем активирован ли гит в текущей директории |
| git init | инициализация гит в текущей директории |
| git add index.html | подготовка файла index.html к сохранению |
| git commit -m 'create base page' | сохранение с комментарием |
| git log | узнаем какие есть сохраненные состояния в гите |
| git checkout 8190075149e8b9bcf3cac88677206a0dac095235 | что-бы вернуться к предыдущему состоянию, проще всего скопировать  хэш коммита (8190075149e8b9bcf3cac88677206a0dac095235) и выполнить  комманду git checkout 8190075149e8b9bcf3cac88677206a0dac095235 |
| git checkout 8190075149e8b9bcf3cac88677206a0dac095235 | возврат к указанному коммиту |
| git checkout master | возврат к сохранению с веткой master |
| git branch | просмотр перечня веток |
| git branch develop | добавление ветки develop |
| git config --global alias.ci commit | добавляем быструю комманду ci (это git commit) |
| code ~/.gitconfig | это конфиг гита, где можно прописать алиасы  Это алиас:  [alias]  ci = commit  co = checkout  br = branch  st = status  lg = log --color --graph --pretty=format:'%Cred%h%Creset -%C(yellow)%d%Creset %s %Cgreen(%cr) %C(bold blue)<%an>%Creset' --abbrev-commit --branches |
| git add . | подготовка всех файлов из текущей директории к сохранению |
| git restore --staged .env.git.local | удалить свой файл из подготовленных к сохранению |
| touch .gitignore // | git ignore - записать все файлы и папки которые не нужно подготавливать к сохранению |
| code .gitignore | открыть список гитигнор (сюда добавляем наш файл исключение .env.git.local)  и теперь когда мы будем выполнять команду git add . то наш файл исключение туда уже не попадет |
| git commit -a | все измененные файлы автоматически будут добавлены к сохранению |
| git config --global core.editor nano | настройка гита - добавление редактора внутреннего |
| git commit -am 'add blue color' | сохранение всех изменений без открытия редактора |
| git branch -m main | Основная ветка по умолчанию называется "master" – мы меняем на main, но это можно изменить командой git branch -m main |
|  |  |
| git branch main -m | переименование ветки |
|  |  |
| git branch feat-42/implement-page-footer | создание ветки с кодом задачи "из менеджмент системы Jira" (например) |
| git branch -d feat-42/implement-page-footer | удаление ветки |
| git checkout -b feat-42/implement-page-footer | создание ветки и переключение на неё (это -b) |
| git switch -c develop | также переключение между ветками и коммитами можно сделать с помощью команды git switch. в Данном случае это создание ветки develop и переключение на нее |
|  |  |
|  | /\* Добавление изменений в основную ветку:  - для этого сначала переключаемся на ветку main: git switch main  - выполняем: git merge feat-42/implement-page-footer  \*/  //Но если изменения после слияния веток оказались ошибочными, то:  - можно создать новый коммит (самый безопасный путь)!!!, где будут изменения обратные к тем что мы делали:  -- git revert 2a3ea26 // отмена коммита с указанием хэша коммита (попадаем в редактор и нажимаем Ctrl+X)  - переменстить ветку main на несколько коммитов раньше:  -- git reset 2a3ea26 // возврат к коммиту с указанием хэша коммита. Переставляет ветку куда-то раньше по истории, но не изменяет файлы в рабочей директории  -- git reset HEAD^^ --hard// (перемещаемся на ветку develop) если мы хотим не просто вернуться к какому-то коммиту. а удалить изменения в текущей директории то добавляем флаг '--hard' |
|  |  |
| GitHub | |
|  | Для клонирования репозитория с сайта гитхаб нужно:  - скопировать HTTPS ссылку (например https://github.com/Sergey-Udod/lucesposa.com.git)  - переходим в первоначальную директорию с проэктами в терминале: cd ..  - затем выполняем команду: git clone https://github.com/Sergey-Udod/lucesposa.com.git  - затем заходим в эту папку, например: cd my-project1  - ls  - git status  - git lg (смотрим какие коммиты есть)  git remote // позволяет увидеть все подключенные удаленные репозитории (но тут не видно какой иммено репозиторий подключен)  git remote -v // позволяет увидеть все подключенные удаленные репозитории детально  git remote show origin // позволяет увидеть все ветки подключенных удаленных репозиториев  git remote add origin https://github.com/Sergey-Udod/lucesposa.com.git // добавляем репозиторий на гитхаб (origin - источник)  далее проверяем что репозиторий создался введя команду git remote  затем можно изменить название основной ветки: git branch -M main  отправить изменения с моей ветки на репозиторий гитхаб: git push -u origin main (флаг -u позволяет синхронизировать ветку в локальном репозитории с веткой в отдаленном репозиторие)  git remote remove origin // удаляем связь с удаленным репозиторием  git remote add mirror1 <https://github.com/Sergey-Udod/lucesposa.com.git> //добавляем репозиторий и добавляем новый алиас mirror1  git remote rename mirror1 origin //переименование репозитория  git fetch mirror1 //синхронизация локального репозитория с удаленным репозиторием на гитхаб  git push --set-upstream mirror1 main // когда папка гит с локального репозитория удаляется, то все связи с отдаленным репозиторием разрываются, и поэтому нужно для их восстановления выполнить эту команду  -------  Инструкция по выливке данные на репозиторий ГитХаб:  git status  git init  git add test\_lucesposa.cy.js  git commit -m 'commit\_1'  git remote  git remote add origin https://github.com/Sergey-Udod/lucesposa.com.git  git remote -v  git push --set-upstream origin master (если коннектим удаленный репозиторий впервые)  Инструкция по заливке данных из репозитория ГитХаб(перед эти переходим в терминале в нужную папку):  git pull origin master  -------  Если воспользоваться веб-версией ГитХаб:  Задача сделать пулл-реквест (сравнение) веток develop и main и внести изменения с develop в main  Открываем ветку develop и вносим изменения в код (чтобы ветки develop и main были разные)  Открываем закладку Code, нажимаем кнопку Compare & Pull Request  Слева выбираем ветку main, справа ветку develop (это сравнение веток)  Нажимаю кнопку Create pull request. Мой пулл реквест создан и в нем есть комиты  Адресную ссылку отправляем более опытному разработчику, и если все ОК, то человек тот кто проверил сделает merge этого пул реквеста с основной веткой main (merge – confirm merge) |
|  |  |
|  | 1. Установка git ﻿sudo apt-get install git  2. Первоначальная настройка git git config --global user.name "John Doe" //Ваше имя git config --global user.email "johndoe[**@example**](http://toster.ru/user/example).com" //Ваш e-mail git config --global core.editor nano //Текcтовый редактор для git git config --global merge.tool vimdiff //Утилита сравнения git config --list //Проверка всех настроек 3. Настройка SSH-авторизации ssh-keygen -t rsa -C "your\_email[**@example**](http://toster.ru/user/example).com" //Создаём ssh-ключ на локальной машине, если не был создан ранее xclip -sel clip < ~/.ssh/id\_rsa.pub //Копируем открытый ssh-ключ в буфер обмена Заходим на страницу: SSH Keys и нажимаем кнопку Add SSH Key. Указываем имя и вставляем ключ в поле Key. Нажимаем Add Key. 4. Работа с репозиторием Для начала на сайте github.com создадим новый репозиторий, если это новый проект. Теперь перейдём в папку с проектом на локальной машине и дадим команду: git init //Инициализация git-репозитория git add \* //Добавим существующие файлы для отслеживания изменений в них touch .gitignore //Создадим файл, куда будем заносить файлы, которые не требуется отслеживать git remote add origin git@github.com:yourname/yourproject.git //Слинкуем удалённый репозиторий как origin git pull origin master //Скачиваем последний правки с удалённого репозитория git push //Отправляем новые правки на сервер |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Cypress:

Находиться на ноутбуке по адресу C:\Users\Admin\CypressAutomation

Запускать нужно как раз из-под ЭТОЙ папки командой: npx cypress open