**GitHub**

GitHub - VCS - version control system

Після встановлення gitHub

В терміналі --git version

Встановлення імені користувача та пароль:

* git config --global user.name "......"
* git config --global user.email " ....."

Команди в терміналі: https://mate-academy.github.io/fe-program/tools/terminal/useful-commands

* **pwd** - подивитись в якій папці ми зараз знаходимось
* **ls**  - подивитись вміст данної папки або вказати шлях **ls d:/** або **ls ~** корінна папка
* **ls ./documents/project** - вміст папки з данної директорії
* **cd /frontEnd/academy** - перехід в данну папку
* **cd..** - вихід на папку вище
* **cd../../** - вихід 2 папками вище
* **mkdir** -створювання папок
* **mkdir -p** c/images - створення папки c а всередині папки images
* **touch** a/style.css - дозволяє створювати файл або декілька
* **rm** style.css -remove видалення файлу або декількох
* **rm -r** b - видалення папки b
* **rm a/\*** - видалення всіх папок всередині папки a
* **rm -r** a/\* - видалення всіх файлів з папки а
* **rm -r ./\***.md - видалення всіх файлів з включеної папки з закінченням md
* **mv** a aaa - перейменувати папку a в ааа
* **mv a c**(якшо така вже створена) - перемістити папку a в папку с
* **cat** ./document/style.css - відкрити файл style.css
* **head** ./document/style.css - покаже перші 10 строк файлу
* **tail -n 5** ./document/style.css - покаже останні 5 строк файлу style.css
* **tail -f** - дозволяє дивитись все шо додається до файлу в реальному часі
* **grep** margin ~ document/myProject/style.css - буде шукати строку margin в файлі style.css
* **less** ./ document/myProject/style.css - редактор дозволяє відкрити файл і запустити help через h
* **man** (назва команди) - допомагає взнати як використовувати команду
* **history** - показує історію введення команд
* **!3** - включає 3 команду з історії
* **ctrl + r** - вводимо початок останньої команду і натискаючи знову ctrl + r перебираємо команди
* **!!** - включає останню команду cd
* **ctrl + с** - відмінити останню введену команду
* **echo 'Hello world'** - текст виведеться на екран
* **echo 'Hello world' > hello.txt** - зявиться файл hello.txt а в ньому текст 'Hello world'
* **cat .zsh\_history | sort | uniq > sorted\_history** - виведе команди відсортовані по алфавіту без повторів в файл sorted\_history
* **ls grep | 'e'** - виведе папки з данної директорії в котрих в назві є 'e'
* **which ls** - допомагає дізнатись звідки взялась якась команда
* **env** - показує змінні оточення
* **х=123** - встановлення значення змінній х локально
* **export x=456** - встановлення значення змінній х глобально
* **х=122 ./document/print\_x** -використання змінної х при запускові скріпта
* **ls -a -l** - покаже всі файли, навіть скриті в вигляді детального списку
* **alias la='ls -la'** - присвоєння команді la - довшої команди ls -la
* **alias** покаже всі псевдоніми
* **unalias** - видалити псевдонім
* **touch ~/.bashrc** - створить файл
* **open -t ~/.bashrc** - відкриє файл в ньому прописуємо псевдонім і зберігаємо, при наступному запускові терміналу псевдоніми будуть працювати. Так само можна зробити зі змінними котрі треба записати на постійній основі.
* в файлі **.bashrc** - можна робити коментарі, перед текстом ставити #

**git init** - створить git репозиторій

**git status** - стан гіт репозиторію

**git add "файл"** - додати файл до відслідковування

**git commit** - перенесе в текстовий редактор

**git commit -a** - не потрібно буде писати git add

**git commit -a** - зберегти дані потім в редакторі "i" ввести текст збереження ентер, :wq -записати і вийти

**git commit -am 'назва комміту'** - зміни одразу запишуться

**git log** - покаже історію всіх збережень та коментарі

Над кожним коммітом написаний спеціальний номер ХЕШ

**git checkout "номер ХЕШ"** - дозволяє перейти до якогось збереження

**git checkout master** - вернеться на останню версію сторінки

**open -t ~/.gitconfig** - відкриє файл в якому лежать alios

**git config --global alias.co chekout** - створить alias "со" на команду chekout

**git reset index.html** - забирає файл index.html з підготовленних до зберігання

**.gitignore** - в файл можна додати файли за якими гіт не повинен слідкувати і які не будуть показуватись в статусі (якшо працюємо в VSC то треба додати папку .vscode)

|  |  |
| --- | --- |
| git config --global |  |
| git clone repo\_url | clone an existing repo |
| git init | initialize new git repo in current folder |
| git status | show info about the repo |
| git add ./path/to/files | prepare changes for saving |
| git commit | save changes and open text editor |
| git commit -m 'a commit message' | save changes with a message |
| git commit -am 'a commit message' | prepare all changed files and save with a message |
| git commit -am --amend 'a commit message' | replace the last commit |
| git log | show the commits history |
| git checkout commit\_hash | switch to the specified commit |
| git tag | list all existing tags |
| git tag tag-name | create a tag tag-name for current commit |
| git tag -d tag-name | delete a tag tag-name |
| git checkout tag-name | switch to the commit specified by the tag-name |
| git branch | show a list of all the local branches |
| git branch branch\_name | create a new branch\_name branch |
| git branch -D branch\_name | delete a branch branch\_name |
| git switch branch\_name | --- the same as previous |
| git checkout -b branch\_name | create a branch at current commit and activate it |
| git switch -c branch\_name | --- the same as previous |
| git merge branch\_name | merge branch\_name into current branch |
| git rebase branch\_name | rebuild current branch on top of branch\_name |
| git rebase -i branch\_name | run rebase with ability to choose commits and squash them |
| git restore file\_name | reset changes not prepared for commit in file\_name |
| git restore file\_name | reset changes not prepared for commit in file\_name |
| git restore --staged file\_name | remove changes in file\_name from prepared for commit |
| git restore --staged --worktree file\_name | reset all the changes in file\_name |
| git reset commit\_hash | Move current branch back to a specified commit keeping changes in files |
| git reset commit\_hash --hard | Move current branch back to a specified commit deleting changes in files |
| git reset HEAD^^ | Move current branch 2 commits back |
| git revert commit\_hash | Create a new commit with changes reverting commit\_hash commit |
| git remote | show a list of remote servers |
| git remote show server\_alias | shows the detailed info about a server |
| git remote add server\_alias repo\_url | adds a new alias for a server with given repo\_url |
| git remote remove server\_alias | disconnects from a server |
| git push server\_alias branch\_name | send the branch\_name to server\_alias |
| git fetch server\_alias branch\_name | downloads the changes without merging them to local branch |
| git pull server\_alias branch\_name | downloads and merges changes |

**Переміщення по збереженим версіям git checkout:**

* номер **хеш**
* використовують гілки (команда **git branch** 'name of branch') активна гілка під час збереження переміщується на далі, а не активні гілки branch залишаються на тому ж рівні
* кожній зміні присвоювати **git tag** ' назва тегу' і переміщуватись між версіями за допомогою тегів
* HEAD^ - відштовхування від версії на якій ми зараз знаходимось на 1 (^) 2 (^^) або n(^^..) - кроків

Для того щоб не змінювати корінний файл **master** ми створюємо **branch** цього файлу вносимо зміни і якшо все гуд то ми можемо додати зміни до корінного файлу **master** для цього потрібно **git checkout master,** потім **git merge** 'назва гілки з якої додати зміни'

**Merge Conflict** - коли 2 розробники в 1 і тій же строці роблять зміни і гіт не знає які зміни прийняти, тоді потрібно або вибрати якийсь варіант або внести зміни власноруч і зберегти їх.

**git rebase master**- повторне збереження якоїсь частини історії, щоб історія завжди була лінійна тобто 1 commit за іншим і не була розділена.

**rebase conflict** - зміни в тії самій строці, вибрати вірний варіант або поправити зміни ввести **git add** "назва файлу" потім **git rebase --continue** і ввести повідомлення про зміни

**git rabase -i master** - інтерактивний ребейс, в текстовому редакторі pick змінити на squash а в повідомленнях закоментувати через # те що не треба

**git clone "ссилка на репозиторій"** - клонування репозиторію скачування

**git pull "назва псевдоніму" "назва гілки"** - скачати собі оновлення

**git push "назва псевдоніму" "назва гілки"** - додати зміни на віддаленний репозиторій