**Git**

**Підключення GIT до проекту:**

1. Підключаємо git до робочої консолі, або активуємо git консоль на самому проекті
2. **git init** – ініціалізація git в проекті (якщо все добре має з’явитись прихована папка в проекті - ‘git’)
3. Після **авторизуємося в проекті**:
   1. **git config --local (--global) user.name “…”**
   2. **git config --local (--global) user.email “…”**

**Генерація SSH ключів для швидкого доступу до репозиторію:**

1. **ssh-keygen** – команда для генерація ключа
2. Відаповідаємо на поставлені конфігурації
3. Заходимо в папку **Users/.ssh/id\_rsa** (розширення PUB у даного файлу повинно бути) і копіюємо його вміст
4. Зайти на акаунт віддаленого репозиторію, в налаштування в розділ ssh
5. Добавити новий ssh ключк
6. При наступному git pull – дозволити використання ключа

**Взаємодія з проетом (стани проекту):**

1. **git status**
   1. Файли відмічені червоним – значіть з ними git не працює і вони тільки створені
   2. ***No commits yet*** - немає змін в проекті
2. **git add** -A**. –** додає файли до індексу
   1. **git add** \_filename**\_ -** конкретний файл
   2. **git add –**A – всі непроіндексовані файли
   3. **git add .** – додає в індексацію всі нові файли і ті в яких були зміни
3. **git commit –m ‘comment’ -** Внесення змін із їх описом
   1. **git commit –**am ‘comment’ – одночасно проіндексує і зкомітить усі файли за якими велось спостереження
4. **git push –u origin branchname(**master-default**)** (branchname – назва гілки, -u –команда, що каже гіту, всі наступні push робити саме в цю глку) ***команда пушу*** змін в репозиторій на вказану вище гілку ***всі наступні рази.***

Після задання команди вище, всі подальші пуші на цю гілку можна здійснювати командою git push

1. **git pull –** перенесення змін з віддаленого репозиторію на локальний
   1. **git pull origin branchName** – перенесення змін конкретної гілки

Якщо над проектом працюють багато людей і перш ніж продовдити роботу над проектом – пропишіть команду git pull, так як ви не зможете закомітити даний проект і внести в нього зміни без помилки

При введені команди git pull, зміни з репозиторію перейдуть до нас, а на нашому комп’ютері git попросить зробити опис змін внесених нами, після потрібно ввести CTRL + C :wq! –для підпису коміту! І ваші зміти перейдуть в проект

1. **git diff** [first-branch] [second-branch] – виводить інформацію про різницю двох гілок
   1. **git diff --saged -** різниця між останнім комітом і індексованими файлами
2. **git cat-file –p (code) –** перегляд файла доданого до індексації /commita по його хеш коду (хеш-код можна знайти в папці .git/object – перші 2 символа папки і 3+ перші символи назви файла в цій папці ) Це файл доданий до індексації git в staget area

**Взаємодія із репозиторіями:**

1. **git remote add origin <посилання на репозиторій>** - прив’язка локального репозиторію до віддаленого (origin – ключове слово прив’язане до конкетного віддаленого репозиторію)
2. **git clone <посилання на репозиторій> -** копія віддаленого репозиторію

**Робота з гілками розробки:**

**Виведення існуючих гілок:**

**git branch –a** – виведення всіх гілок в репозиторії

**Подивитись всі локальні гілки:**   
 **git branch**

**Перегляд останнього коміту кожної локальної гілки:**

**git branch –v**

**Подивитись останні коміти на всіх гілках:  
git branch –a -v**

**Ствоення нової гілки:**

**git checkout -b** **branchName** – створення нової гілки

**Видалення гілки:**

**git branch –d** **branchName** – видалення вказаної гілки (коміти даної гілки не видаляються, але і не прив’язані до жодної гілки)

**Перейменування гілок:  
git branch -m <oldname> <newname>**

**Злиття гілок (знаходимось в master і заливаємо hotfix в master)  
git checkout master  
git merge hotfix**

**Переключення по гілкам:**

**git checkout benchName** **–** перехід на вказану гілку

**git checkout** **(hesh) –** перехід між комітами (при переході із неприкріпленого до гілки коміта в будь-яку нову гілку, цей коміт автоматично прив’язується до цієї нової гілки)

**Виведення нових комітів із видалених гілок, але без злиття з локальними:**

**git fetch –** виведення комітів видалених гілок

**Робота з редагуванням файлів/commits:**

**Видалення:**

1. **git rm <filename>**– видалення файлів і з відстежуваних
2. **git rm -f <file\_name>** - видалення проіндексованого зміненого файла
3. **git rm --cached filename** – вдалення файла з індексації (наприклад файл, який ми забули додати в .gitignore), але який ми не видаляєм з каталога
4. **git rm** log/\\*.log – видалення всіх файлів даного типу в репозиторії log

**Перейменування:**

**git mv <**old\_file\_name> <new\_file\_name**>**- перейменування файлів

**Переміщення по версіям:**

1. **git checkout** <hash> - **переміщення по версіям –** хеш можна взати з перегляду комітів в будь-якої версії
2. **git checkout** master **– остання версія в вказаній гілці**
3. **git checkout** tagName – перескакуємо по назвам версій

**Відміни команд:**

1. **git reset HEAD** **filename** – відміняє індексацію файлу
2. **git checkout -**- **filename** – скасовує останній коміт(останні зміни відкатує назад)
3. **git commit --ammed –m ‘’** – редагування остннього коміту
4. **git merge --abort** – відміняє злиття гілок

**Взаємодія з комітами:**

**Оформлення інформації про коміти**

1. **git log** – ***список commit***, за весь час роботи в даній гілці
2. **git log --all** – виведення усі коміти в усіх гілках
   1. **git log** –p –number – вказує на змінив в вказаній кількості комітів
   2. **git log** –pretty= oneline/short/full/fuller – вивід інформації про коміт в одну строку
   3. git log --pretty=oneline --max-count=2
   4. git log --pretty=oneline --since='5 minutes ago'
   5. git log --pretty=oneline --until='5 minutes ago'
   6. git log --pretty=oneline --author=<your name>
   7. git log --pretty=oneline --all
   8. **git log** --pretty=format: виввід інформації в вказаному форматі
3. **git log $(git rev-list –g –all )** – виводить усі коміти (включно із неприкріпленими до конкретної гілки)

**Шаблонний формат:**

git log --pretty=format:"%h %ad | %s%d [%an]" --graph --date=short

* --pretty="..." — определяет формат вывода.
* %h — укороченный хэш коммита
* %d — дополнения коммита («головы» веток или теги)
* %ad — дата коммита
* %s — комментарий
* %an — имя автора
* --graph — отображает дерево коммитов в виде ASCII-графика
* --date=short — сохраняет формат даты коротким и симпатичным
* --max-count=1 – к-сть виведених коміті

**Створення readme file**

<https://www.markdownguide.org/basic-syntax/>

**Плагін в VS Code:** markdown prewiev enhanced

**Створюємо файл: README.md**

**Синтаксис:**

1. # - вказує розмір заголовка, як в html тегах h1 = #, h3 = ###
2. \*\*text\*\* - виділений шрифт
3. \_italic\_ - похилий шрифт
4. ~~text~~ - перекреслений текст
5. [text](url) – даний текст стане посиланням
6. ![img](url) – картинка
7. [symbol]: url – шорткат, завдяки якому ми через символ присвоюватимемо url, який до нього відноситься
8. – text - елемент списка
9. – text

-text - вложений список

1.text – нумерований список

10. --- - слабковиділена лінія

11. > text – виділений фон для певного тексту

12. ’’’js/html/css

Все тут вписане виведеться, як код даного файлу

- виділено червоним(немає)

+ виділено зеленим(додано)

```

13.Таблиця

| text | text |

|: ----| ----- |

| text | text |

14. – [\*] – чекбокс виконаний