**Основні команди Git**

**Налаштування параметрів для підпису комітів:**

**git config --global user.name "Andrii Morozov"**

**git config --global user.email morozov@ztu.edu.ua**

**Створення репозиторію у поточному каталозі:**

**git init**

**Додавання до індексу усіх файлів (ті, які змінилися та нові файли):**

**git add .**

**Фіксація останніх змін (коміт):**

**git commit -m "Заголовок коміту" -m "Пояснення до коміту"**

**Перегляд історії комітів:**

**git log**

**Додати до коміту тільки ті файли, які змінилися (нові файли не додаються):**

**git add -u**

**Фіксація останніх змін (коміт):**

**git commit -m "Тут вписати суть змін"**

**Подивитись, які сервери репозиторіїв підключені:**

**git remote**

**Встановити сервер, на який буде завантажуватись репозиторій:**

**git remote add origin** [**https://morozovandriy@bitbucket.org/morozovandriy/firstgitrepo.git**](https://morozovandriy@bitbucket.org/morozovandriy/firstgitrepo.git)

**Завантажити репозиторій на сервер:**

**git push -u origin main**

**Клонування репозиторію (скачування з сервера на локальний диск):**

**git clone** [**https://bitbucket.org/morozovandriy/firstgitrepo.git**](https://bitbucket.org/morozovandriy/firstgitrepo.git)

**Створення нової гілки:**

**git branch НазваГілки**

**Завантажити з сервера усі гілки у локальний репозиторій:**

**git pull --all**

**Переключення на гілку:**

**git checkout НазваГілки**

**Створення нової гілки з одночасним переключенням на неї:**

**git checkout -b НазваГілки**

**Завантаження гілки на сервер:**

**git push origin НазваГілки**

**Переглянути список гілок:**

**git branch**

**Об’єднати дві гілки (приєднати до поточної, вказану у команді гілку):**

**git merge НазваГілки**

**Отримання змін з репозиторію, що зберігається на сервері:**

**git fetch**

**Отримання змін з репозиторію, що зберігається на сервері і злиття їх з поточним репозиторієм:**

**git pull**

**Робота з Git  
(повна документація)**

Створення репозиторію у поточному каталозі:

**git init**

Налаштування параметрів для підпису комітів:

* для поточного репозиторію:

**git config user.name "Andrii Morozov"**

**git config user.email morozov@ztu.edu.ua**

або

**git config –-local user.name "Andrii Morozov"**

**git config –-local user.email morozov@ztu.edu.ua**

* для поточного користувача:

**git config –-global user.name "Andrii Morozov"**

**git config –-global user.email morozov@ztu.edu.ua**

* для усіх користувачів:

**git config –-system user.name "Andrii Morozov"**

**git config –-system user.email morozov@ztu.edu.ua**

Також потрібно відключити врахування прав доступу до файлів при виконанні коммітів:

**git config --global core.fileMode false**

В ОС Linux Git шукає налаштування у:

1) глобальному файлі конфігурації:

**/etc/gitconfig**

2) конфігурації для поточного користувача:

**~/.gitconfig** або **~/.config/git/config**

3) конфігурації поточного репозиторію:

**.git/config**

В ОС Windows Git шукає налаштування у:

1) глобальному файлі конфігурації:

**C:\Program Files\Git\etc\gitconfig**

2) конфігурації для поточного користувача:

**C:\Users\Користувач\.gitconfig**

3) конфігурації поточного репозиторію:

**Проект/.git/config**

Для перегляду налаштувань:

**git config --list**

Для перегляду налаштувань та файлів, у яких вони задані:

**git config --list --show-origin**

Для видалення налаштувань:

**git config --unset user.name**

або для видалення цілої секції налаштувань:

**git config --remove-section user**

Встановлення текстового редактора, який буде використовуватися для набору повідомлень в Git:

- в ОС Linux:

**git config --global core.editor emacs**

- в ОС Windows:

**git config --global core.editor "'C:/Program Files/Notepad++/notepad++.exe' -multiInst -notabbar -nosession -noPlugin"**

Перегляд стану поточного репозиторію:

**git status**

Додавання до індексу файла:

**git add index.html**

Додавання до індексу декількох файлів:

**git add index.html style.css**

Додавання до індексу усіх файлів:

**git add .**

Видалення з індексу файла (-ів):

**git reset HEAD index.html**

Фіксація змін (виконання коміту):

**git commit**

Далі вводиться опис коміту. Він може складатися з заголовку та детального опису (який записується через порожній рядок), наприклад:

**Розроблено головну сторінку сайту**

**\* створено файл index.html**

**\* реалізовано початкові стилі style.css**

Фіксація змін (виконання комміту) з вказуванням опису комміту:

**git commit -m "Тут вписати суть змін"**

Виконання комміту із вказівкою автора та дати комміту:

**git commit –-author="Andrii Morozov <morozov.ztu.edu.ua>" --date="2021-12-31 23:59:59"**

Переглянути інформацію про останній комміт:

**git show**

Переглянути інформацію про конкретний комміт:

**git show ідКомміту**

Переглянути інформацію про попередній комміт:

**git show HEAD~**

Переглянути інформацію про два комміти назад:

**git show HEAD~~**

або

**git show HEAD~2**

Подивитися файл у попередньому комміті:

**git show HEAD~:назваФайла**

Замість **HEAD** можна вказувати назви гілок або ідентифікатори коммітів.

Переглянути розширену інформацію про комміт (інформацію про автора комміту та коммітера):

**git show –-pretty=fuller**

Для того, щоб перерахувати файли та/або каталоги, які не потрібно додавати до індексу, їх потрібно перерахувати у файлі **.gitignore**, який розміщують в корні репозиторію.

У файлі **.gitignore**:

* кожний рядок задає правило;
* назви каталогів мають закінчуватися символом **/**, інакше правило задаватиме файл(и);
* порожні рядки ігноруються;
* рядки, які починаються з **#** є коментарями;
* символ **/** на початку рядка означає поточна папка, де лежить **.gitignore**;
* символ **\*** означає - будь-яка послідовність символів;
* символи **\*\*** задають усі підкаталоги;
* символ **!** на початку рядка інвертує шаблон, тобто вказує, що дане правило не потрібно ігнорувати.

Для того, щоб задати глобальний файл .gitignore, який буде застосовуватися до усіх репозиторіїв поточного користувача:

**git config –global core.excludesFile ~/.gitignore**

Для ОС Windows потрібно створити даний файл у папці:

**C:\Users\Користувач\.gitconfig**

Для ОС Linux файл створюється у папці:

**~/.gitconfig**

Ігноровані файли неможливо додати до індексу за допомогою

**git add файл**

Для примусового додавання до індексу ігнорованого файла:

**git add -f файл**

Для видалення файла з диску та індексу:

**git rm файл**

Для видалення файла тільки з індексу (на диску файл залишиться):

**git rm --cached** **файл**

Для видалення папки з диску та індексу:

**git rm -r папка**

Для видалення папки тільки з індексу (на диску папка залишиться):

**git rm -r --cached** **папка**

Для перейменування файла:

**git mv стараНазва новаНазва**

Перегляд списку гілок:

**git branch**

Перегляд списку гілок з останніми комітами:

**git branch -v**

Створення нової гілки:

**git branch НазваГілки**

Створення нової гілки:

**git branch НазваГілки**

Перемикання на іншу гілку:

**git checkout НазваГілки**

Перемикання на попередню гілку:

**git checkout -**

Створення нової гілки з автоматичним перемиканням на неї:

**git checkout -b НазваГілки**

Відмінити усі зроблені зміни і повернутися до стану файлів останнього комміту:

**git checkout -f**

або

**git reset –-hard**

Вімінити зміни конкретного файла і повернутися до стану цього файла з останнього комміту:

**git checkout файл**

Зберегти незакоммічені зміни у спеціальному архіві і видалити зміни з файлів:

**git stash**

Щоб повернутися до стану файлів, які збережено в архіві:

**git stash pop**

Створення нової гілки із заданого комміту:

**git branch НазваГілки ідКоміту**

Перенесення існуючої гілки на заданий комміт (перед цим потрібно перейти на іншу гілку, оскільки Git не дозволить змінити поточну гілку):

**git branch -f НазваГілки ідКоміту**

Перенесення існуючої гілки на заданий комміт і перехід на цю гілку:

**git checkout -B НазваГілки ідКоміту**

Перехід на конкретний комміт (стан відділенного HEAD):

**git checkout** **ідКоміту**

Відновити стан конкретного файла/каталога з вказаного комміту:

**git checkout ідКоміту шляхДоФайла**

Переглянути список коммітів:

**git log гілка**

Переглянути список коммітів у певній гілці:

**git log**

Переглянути список коммітів у скороченому вигляді:

**git log --oneline**

Просте об’єднання гілок:

1. перемикаємося на гілку (наприклад, **master**), в яку потрібно влити іншу гілку (наприклад, **fix**):

**git checkout master**

1. вказуємо з якої гілки хочемо влити зміни (наприклад, **fix**):

**git merge fix**

1. видалити гілку **fix**:

**git branch -D fix**

Подивитися, які файли відрізняються у двох гілках:

**git diff --name-only гілка1 гілка2**

Злиття гілок, які містять конфлікти:

1. перемикаємося на гілку (наприклад, **master**), в яку потрібно влити іншу гілку (наприклад, **fix**):

**git checkout master**

1. вказуємо з якої гілки хочемо влити зміни (наприклад, **fix**):

**git merge fix --log**

Ключ **--log** є необов’язковим тут використовується для того, щоб опис комміту злиття сформувати з усіх коммітів гілки **fix**

1. відкрити файли, які містять конфлікти і виправити фрагменти

**<<<<<<< HEAD**

**…**

**=======**

**…**

**>>>>>>> fix**

1. якщо треба відмінити злиття гілок:

**git reset --merge**

1. якщо всі конфлікти виправлено, то додаємо до індексу змінені файли:

**git add файл**

1. виконуємо:

**git merge --continue**

Інколи при злитті виникають семантичні конфлікти, які Git не помічає при автоматичному вирішення конфліктів.

У такому разі, якщо в результаті злиття двох гілок виявлено семантичний конфлікт, потрібно:

1) відмінити останній комміт:

**git reset --hard HEAD~**

2) виконати злиття з ключем **--no-commit**

**git merge fix --no-commit**

3) внести зміни у файли, щою вирішити конфлікти

4) додати в індекс змінені файли:

**git add файл**

5) виконаємо:

**git merge --continue**

Якщо в гілці, в яку відбувається влиття (наприклад, **master**) іншої гілки (наприклад, **feature**) не було коммітів, то за замовчуванням при злитті гілок відбувається “злиття перемоткою”, що зводиться до переміщення вказівника гілки **master** на останній комміт гілки **feature**. Це призводить до того, що у гілки **master** виходить рівна історія коммітів, з якої не можна визначити, де саме відбулося розгалуження та злиття гілок.

Для того, щоб відбувалося злиття гілок зі збереженням історії злиття потрібно використовувати ключ **--no-ff:**

**git merge fix –-no-ff --log**

або задати відповідний параметр у конфігурації:

**git config merge.ff false**

Комміт злиття є особливим, у нього два батьківських комміти.

**git show** для такого комміту показує інформацію про вирішені конфлікти.

Якщо треба подивитися відмінності коду комміту злиття від першого батьківського комміту, то виконується команда:

**git diff HEAD~1**

Для перегляду відмінності комміту злиття від другого батьківського комміту:

**git diff HEAD~2**

Git зберігає історію (**reflog**) переходу між гілками та коммітами у файлі **.git/logs/HEAD**

Можна переглянути **reflog** історію для конкретної гілки:

**git reflog гілка**

Вивести **reflog** для **HEAD**:

**git reflog**

Можна виводити дату для записів **reflog**:

**git reflog –-date=iso**

Повернути файли до стану, заданого останнім коммітом (нові файли, які не відслідковуються залишаться без змін):

**git reset --hard**

Видалити з каталогу усі файли, які не відслідковуються та усі файли, що ігноруються

**git clean -dxf**

Відмінити останній комміт (повернути файли до стану, заданого останнім коммітом):

**git reset –-hard HEAD**

Відкотити стан файлів до конкретного комміту:

**git reset --hard ідКомміту**

Відмінити комміт, залишивши файли як вони є на поточний момпент (перемістити вказівник гілки на інший комміт, залишивши файли без змін):

**git reset --soft ідКомміту**

Якщо коміти вже були відправлені іншим розробникам, то просто відмінити такий коміт не можна, тоді треба виконувати коміт з протилежними змінами, тобто такими, які повертають код до стану попереднього комміту:

**git revert HEAD**

Зробити комміт з описом, таким самим як у попередньо зробленого комміту:

**git commit -C ORIG\_HEAD --reset-author**

Замінити останній комміт:

**git commit –-amend --reset-author**

Замінити останній комміт зі зміною опису комміту:

**git commit --amend -m "Опис"**

Різні види скидання коммітів:

|  | **Зміна файлів** | **Зміна індексу** | **Переміщення гілки** |
| --- | --- | --- | --- |
| **git reset –-soft** | ні | ні | **так** |
| **git reset --mixed** | ні | **так** | **так** |
| **git reset --hard** | **так** | **так** | **так** |

Повернення індексу до стану попереднього комміту:

**git reset**

Порівняння стану проекту у різних гілках або коммітах:

**git diff гілка1 гілка2**

або

**git diff ідКомміту1 ідКомміту2**

Що саме змінилося у **гілці2** з моменту (комміту) її відходження від **гілки1**:

**git diff гілка1...гілка2**

Порівняти поточний стан файлів з попереднім коммітом (ігноруються файли, які не відслідковуються):

**git diff HEAD**

Порівняти зміни, які проіндексовані, але ще не закоммічені з попереднім коммітом (так, можна дивитися, що ми збираємось закоммітити):

**git diff --cached**

Порівняти зміни у конкретному файлі:

**git diff ідКомміту1 ідКомміту2 –- файл**

або

**git diff гілка1 гілка2 — файл**

Різні варіанти перегляду переліку коммітів:

**git log --oneline**

**git log --oneline --no-decorate**

**git log --pretty=format:'%h %cr | %s%d [%an]'**

**git log --pretty=format:'%C(yellow)%h %C(dim green)%cr %C(reset)| %C(cyan)%s%d %C(#667788)[%an]'**

Позначення специфікаторів форматування можна подивитися в документації:

**git help log**

Подивитися список коммітів і що в них зроблено:

**git log -p**

Подивитися список коммітів у вигляді дерева:

**git log --all --graph**

Подивитися список коммітів **гілки2** після її відходження від **гілки1**:

**git log гілка2 ^гілка1**

або

**git log гілка1..гілка2**

Подивитися комміти, в яких змінювався конкретний файл:

**git log файл**

Подивитися комміти, в яких змінювався конкретний файл і які саме зміни було зроблено:

**git log -p файл**

Подивитися хто писав рядки конкретного файла:

**git blame файл**

Копіювання комміту із заданим ідентифікатором у поточний комміт:

**git cherry-pick ідКомміту**

Перебазування поточної гілки, де ми знаходимось у вказану гілку:

**git rebase master**

Якщо в процесі виникнуть конфлікти, то їх треба вирішити і виконати команду:

**git rebase --continue**

Якщо в процесі вирішення конфліктів потрібно відмінити процедуру перебазування гілки:

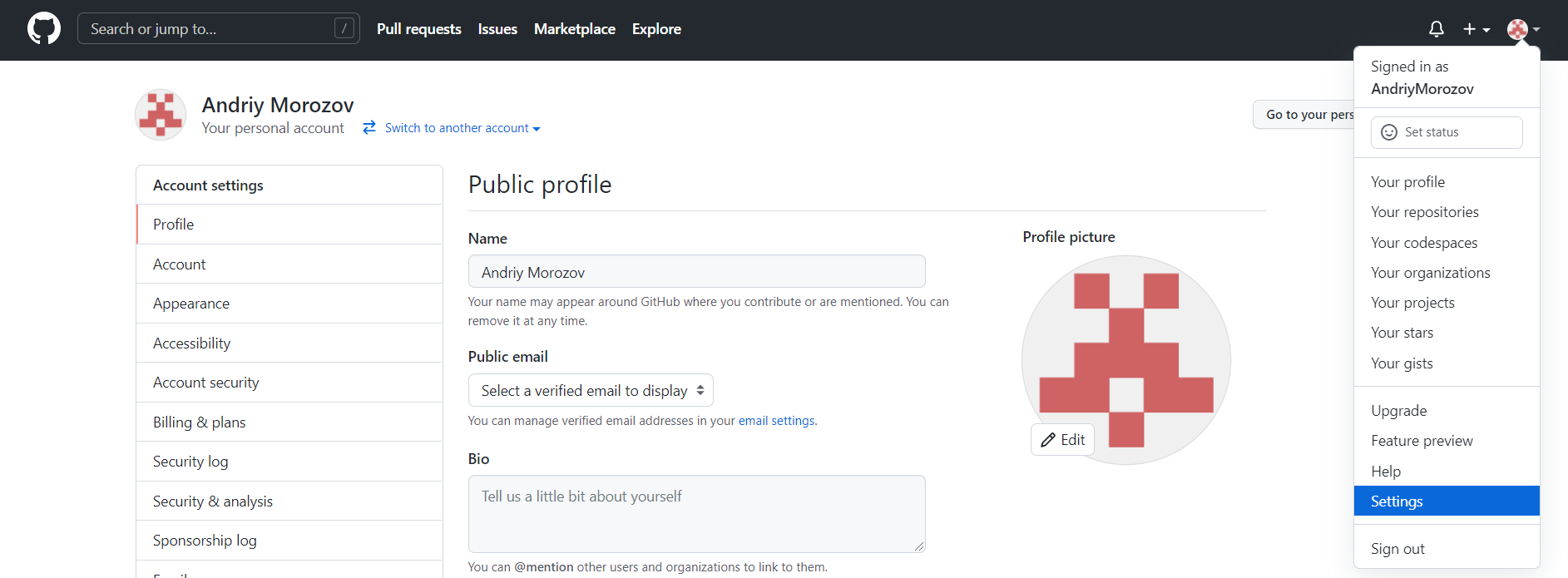
**git rebase --abort**

Можна вказувати яку гілку у яку потрібно перебазувати:

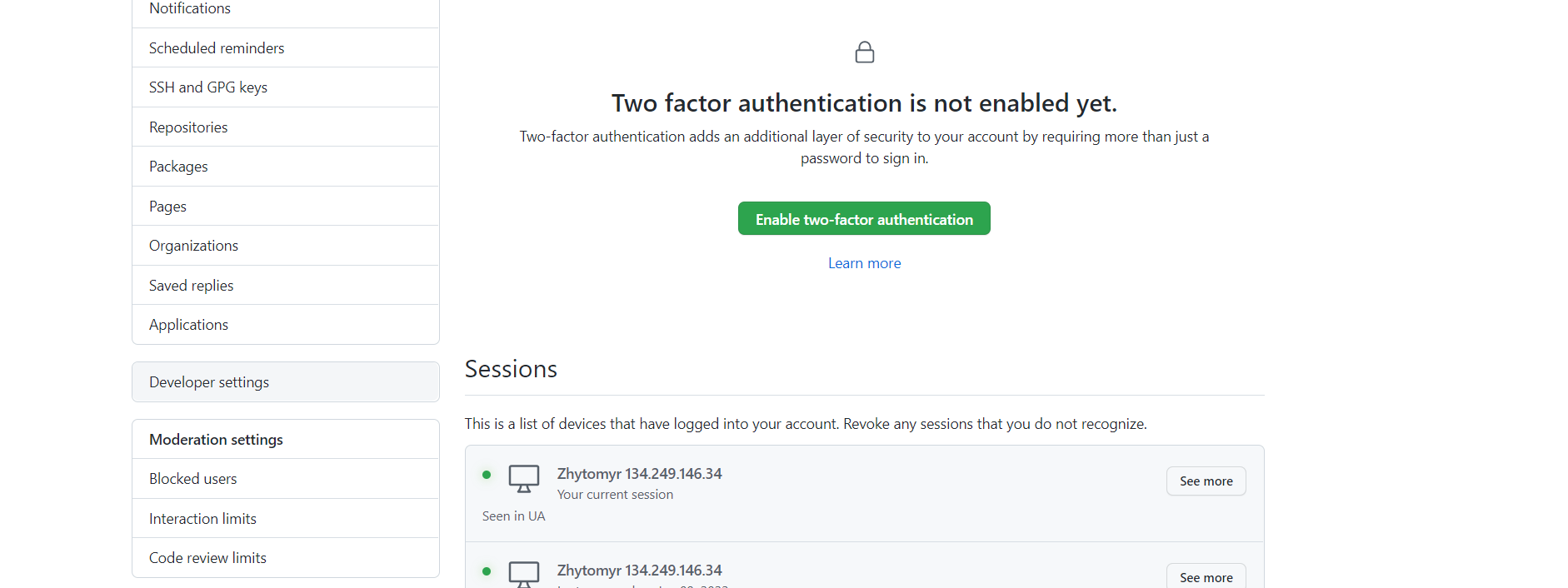
**git rebase кудаПеребазовуємо якуГілкуПеребазовуємо**

**Аутентифікація GitHub за допомогою токена (для роботи через HTTP):**

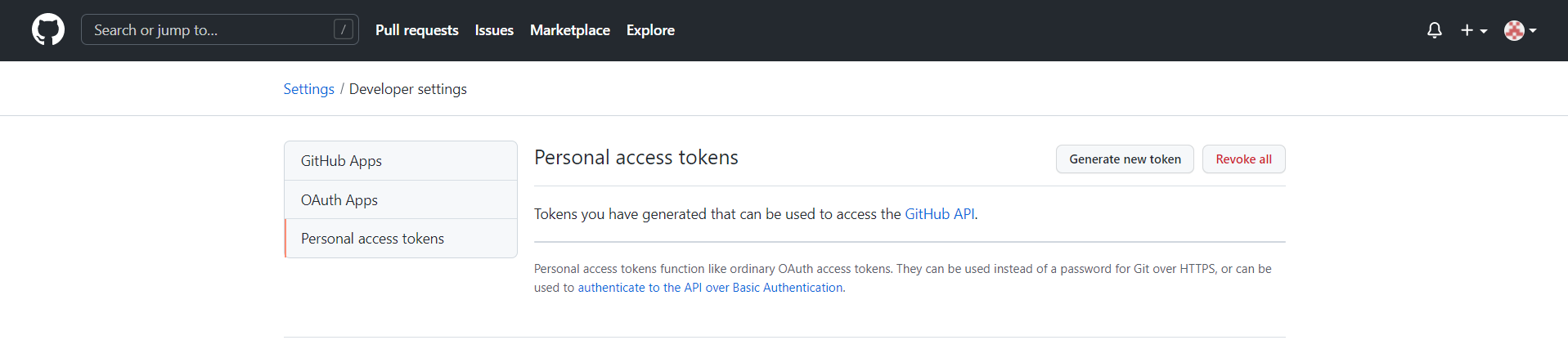
1. переходимо до “Settings”:



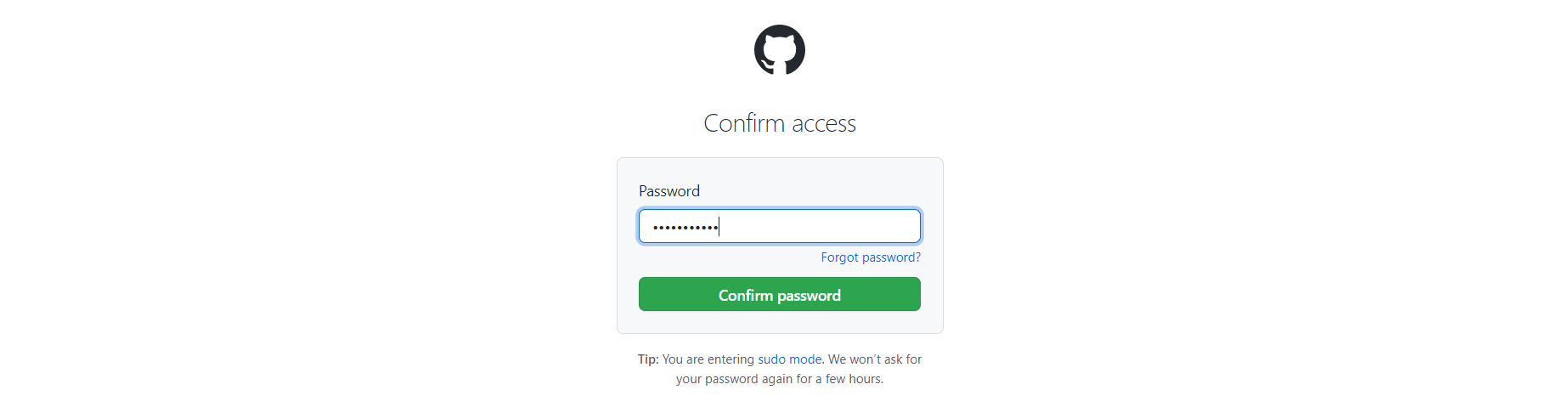
1. вибираємо “Developer settings”:



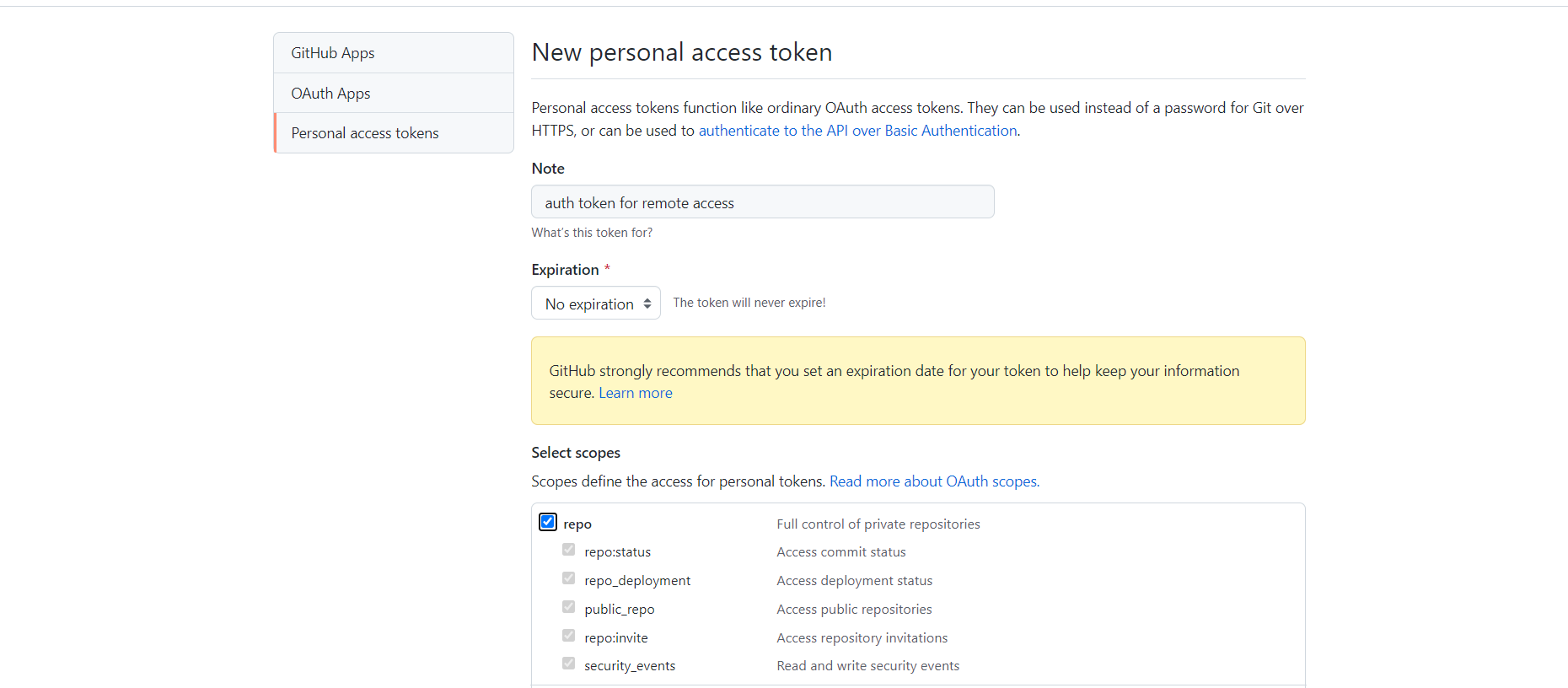
1. далі - “Personal access tokens” і “Generate new token”:



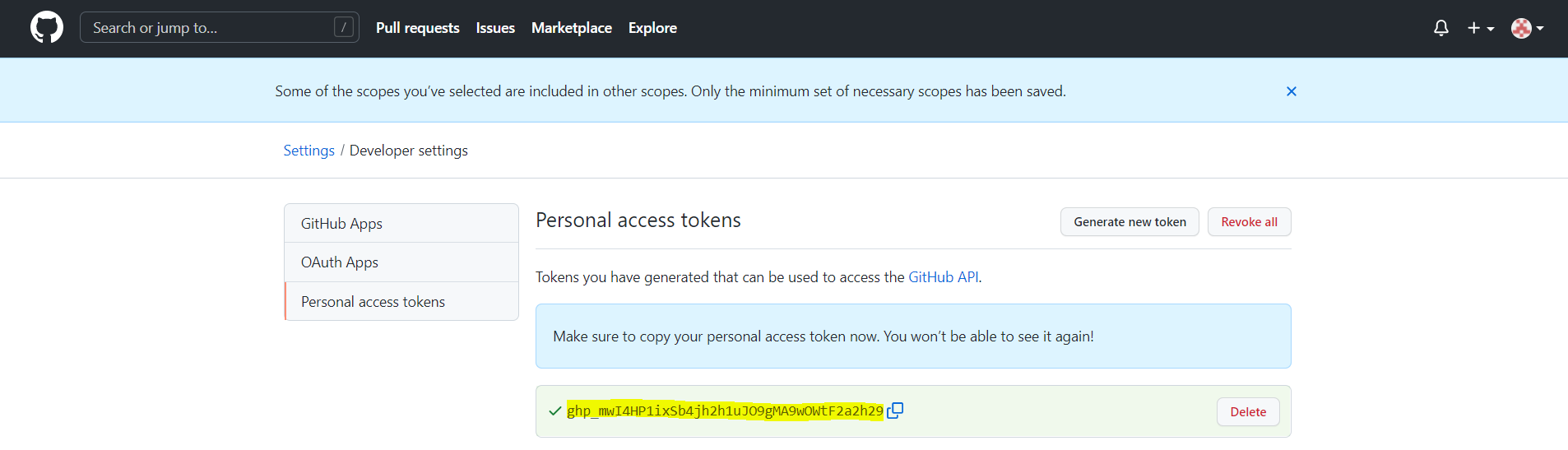
1. вводимо пароль від аккаунта:



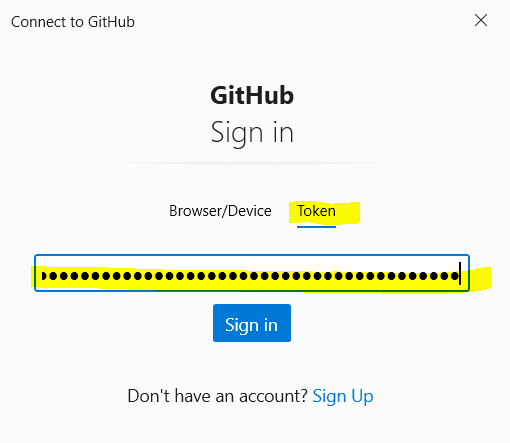
1. вводимо інформацію у полі “Note”, вибираємо термін дії токена та вибираємо права доступу, підтверджуємо генерацію токена:



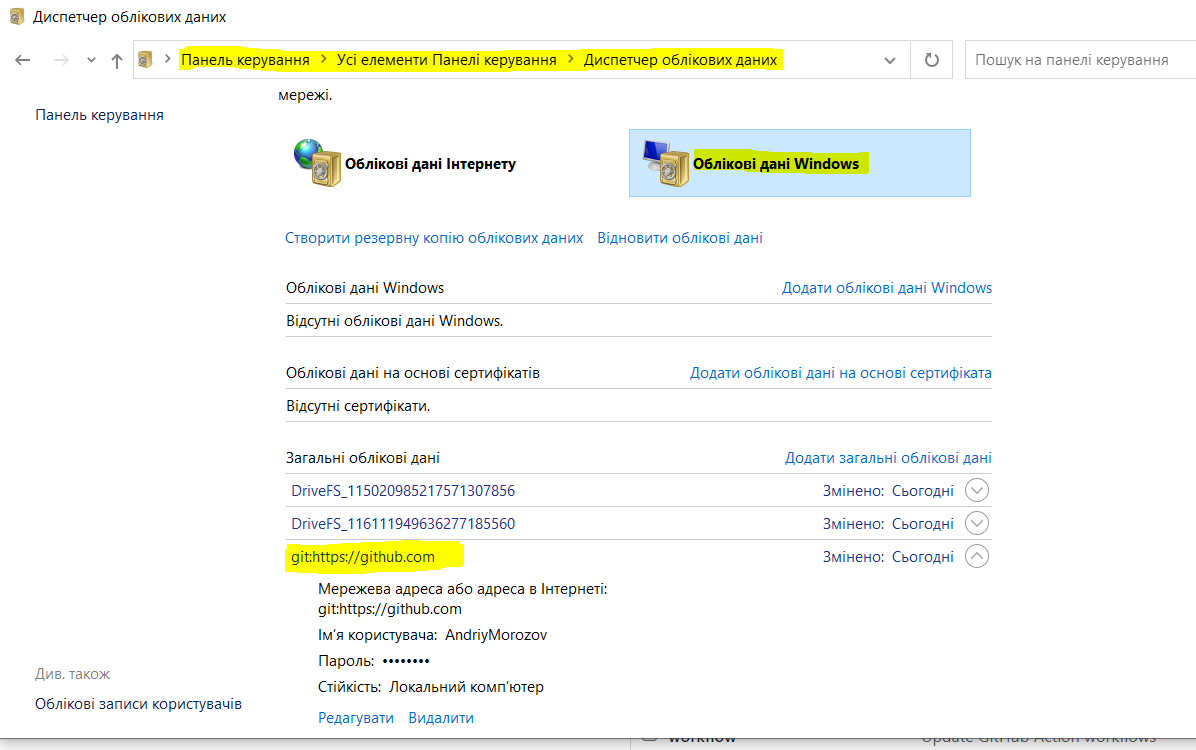
1. копіюємо токен:



1. при виконанні операції над віддаленим репозиторієм чи клонуванні репозиторію в ОС Windows буде виведено вікно аутентифікації. У ньому вибираємо “Token” і копіюємо токен у відповідне поле:

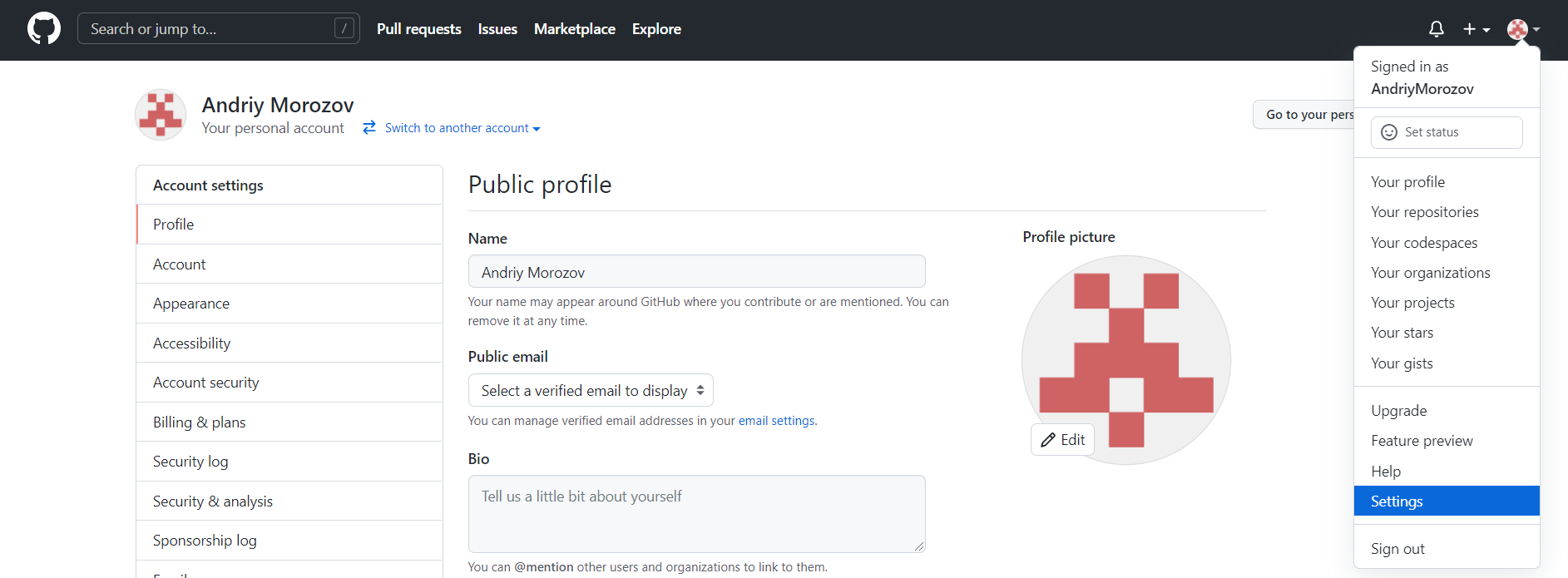


При такому способі аутентифікації інформація про користувача в ОС Windows запам’ятовується у “Credential Manager Control panel”:

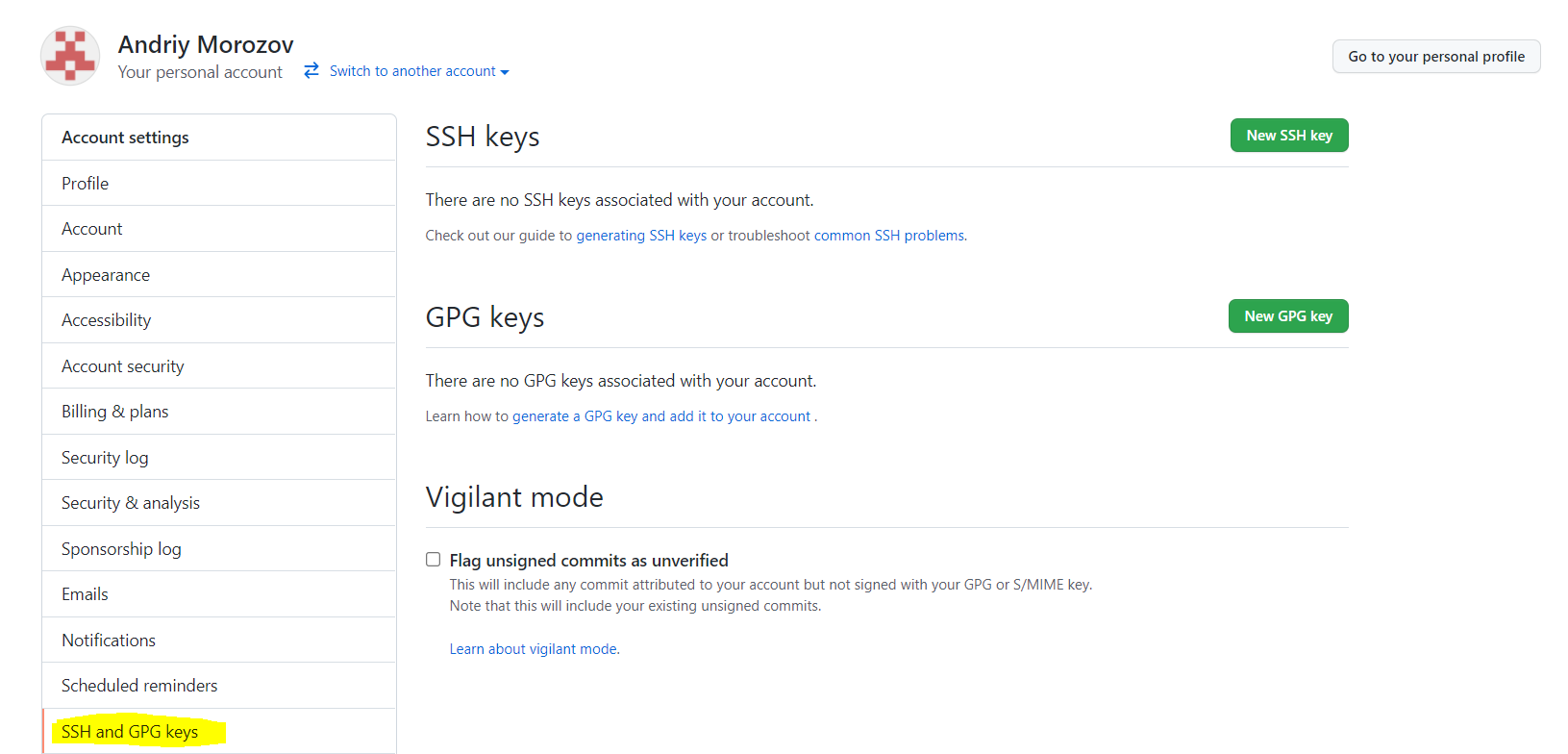


**Аутентифікація GitHub за допомогою SSH (для роботи через SSH):**

1. переходимо до “Settings”:



1. переходимо до пункту “SSH and GPG keys”, натискаємо на кнопку “New SSH key”:



1. відкриваємо термінал та виконуємо генерування ключа за допомогою команди:

**ssh-keygen -t ed25519 -C "morozov@ztu.edu.ua"**

Буде запропоновано вибрати місце збереження файлів ключа, погоджуємось:

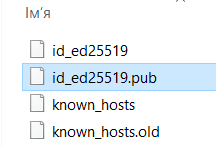
* **Linux:** Enter a file in which to save the key (/home/**you**/.ssh/**algorithm**): [Press enter]
* **MacOS:** Enter a file in which to save the key (/Users/**you**/.ssh/id\_algorithm): [Press enter]
* **Windows:** Enter file in which to save the key (C:\Users\**Admin**/.ssh/id\_ed25519): [Press enter]

Далі буде запропоновано два рази ввести пароль для ключа (це не пароль від аккаунта). Звертаємо увагу, що при введенні пароль не відображатиметься на екрані:

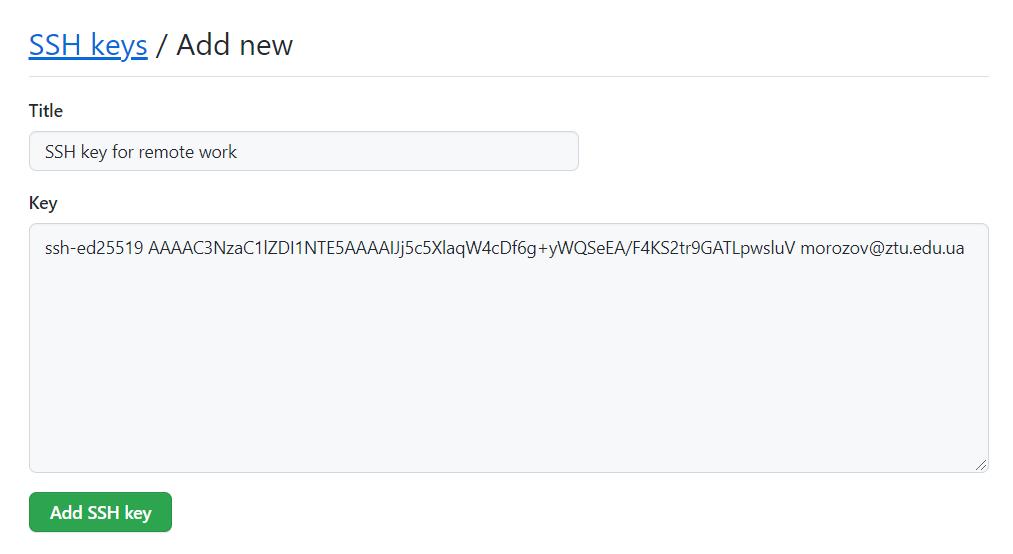
Enter passphrase (empty for no passphrase):

Enter same passphrase again:

1. знаходимо pub-файл ключа та копіюємо його вміст у буфер обміну:



1. вставляємо вміст файла у поле “Key”, пишемо “Title” і натискаємо на кнопку “Add SSH key”:



В залежності від налаштованого режиму аутентифікації можна обрати

**Файл README.md**

[Форматування тексту](https://docs.github.com/en/get-started/writing-on-github/getting-started-with-writing-and-formatting-on-github/basic-writing-and-formatting-syntax)

Додати інформацію про:

**git push --set-upstream <remote> <branch>**