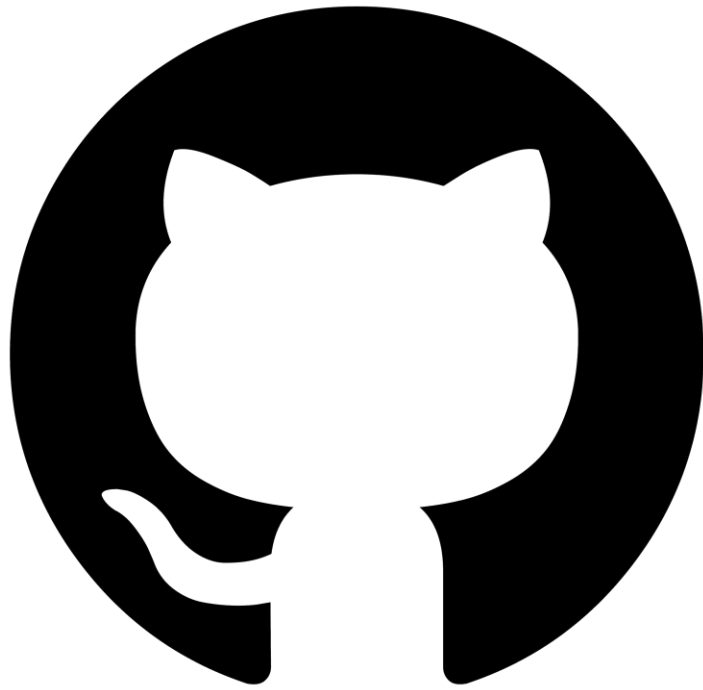


Github & Git Tutorial



git

Spis Treści

Czym jest Github?	3
Jak założyć konto na Githubie?	3
Plany.....	5
Czym jest Git?	6
Jak działa Git?	7
Jak pobrać Git?	7
Jak utworzyć Repozytorium?	8
Sposób 1	8
Sposób 2.....	10
Branche	12
Jak zarządzać Branch'ami?	12
Sposób 1	12
Sposób 2.....	14
Pull requests	14
Jak pobrać repozytorium?	16
Sposób 1	16
Sposób 2.....	16
Readme	18
Licencje	19
.gitignore	20
Codespaces	21
Codespace na podstawie repo.....	22
Jak utworzyć czysty codespace	23
Przydatne komendy Git.....	24
Praca z gałęziami (branches).....	24
Praca z Githubem (zdalne repozytorium).....	25
Cofanie i resetowanie zmian	25

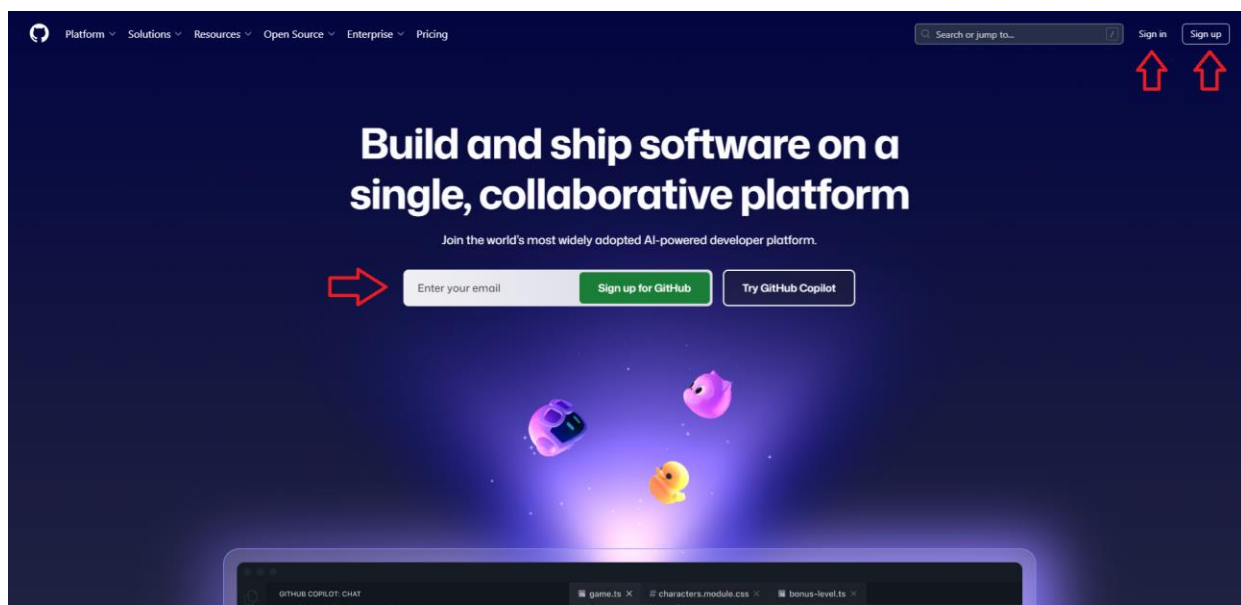
Informacje i diagnostyka.....	26
Inne	26

Czym jest Github?

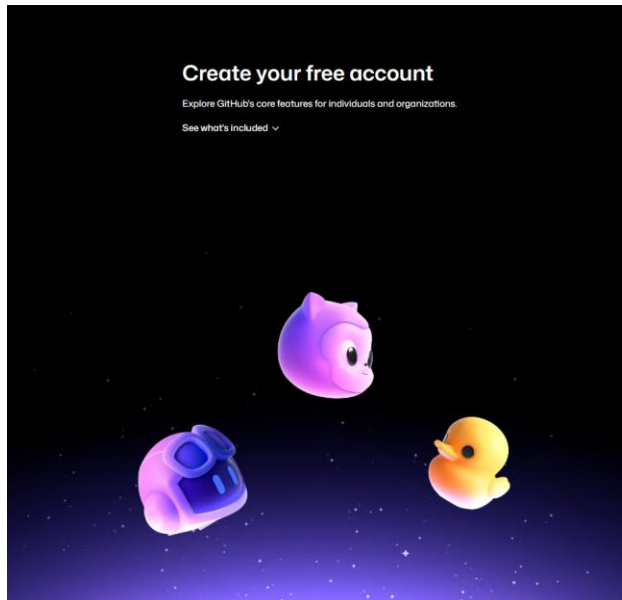
GitHub – internetowa platforma do przechowywania i współpracy nad projektami programistycznymi, działająca w oparciu o system kontroli wersji **Git**. Umożliwia śledzenie zmian w plikach, zarządzanie wersjami, tworzenie gałęzi (*branches*), współpracę wielu osób nad tym samym projektem oraz korzystanie z dodatkowych narzędzi, takich jak *issues*, *pull requests*, *wiki* czy *GitHub Actions*.

Jak założyć konto na Githubie?

1. Przechodzimy na stronę <https://github.com/>
2. Jeżeli chcemy utworzyć konto wpisujemy **e-mail** w **polu na środku strony** lub klikamy **Sign Up**. Jeżeli mamy już konto klikamy **Sign In**.



3. Następnie wypełniamy swoimi danymi
4. Klikamy przycisk Create account



Already have an account? [Sign in](#)

Sign up for GitHub

[Continue with Google](#)

[Continue with Apple](#)

or

Email*

Password*

Password should be at least 15 characters OR at least 8 characters including a number and a lowercase letter.

Username*

Username may only contain alphanumeric characters or single hyphens, and cannot begin or end with a hyphen.

Your Country/Region*

For compliance reasons, we're required to collect country information to send you occasional updates and announcements.

Email preferences

☐ Receive occasional product updates and announcements

[Create account](#)

By creating an account, you agree to the [Terms of Service](#). For more information about GitHub's privacy practices, see the [GitHub Privacy Statement](#). We'll occasionally send you account-related emails.

5. Na maila przyjdzie nam kod weryfikacyjny, który trzeba będzie wpisać na stronie.
6. Klikamy Verify, konto powinno zostać utworzone.

Możliwości

Na stronie możemy przeglądać swoje repozytoria, repozytoria innych użytkowników, współpracować przy tworzeniu kodu z innymi, edytować kod, kompilować kod, korzystać z Copilota...

Plany

<h3>Free</h3> <p>A fast way to get started with GitHub Copilot.</p> <p>\$0 <small>USD</small></p> <p>Get started</p> <p>Open in VS Code</p> <p>What's included:</p> <ul style="list-style-type: none">✓ 50 agent mode or chat requests per month✓ 2,000 completions per month✓ Access to Claude Sonnet 3.5, GPT-4.1, and more	<h3>Pro <small>Most popular</small></h3> <p>Unlimited completions and chats with access to more models.</p> <p>\$10 <small>USD</small></p> <p>per month or \$100 per year</p> <p>Try for 30 days free</p> <p>Everything in Free and:</p> <ul style="list-style-type: none">✓ Unlimited agent mode and chats with GPT-5 mini!✓ Unlimited code completions✓ Access to code review, Claude Sonnet 4, GPT-5, Gemini 2.5 Pro, and more✓ 6x more premium requests than Copilot Free to use the latest models, with the option to buy more²✓ Coding agent <p><small>Free for verified students, teachers, and maintainers of popular open source projects. Learn more</small></p>	<h3>Pro+</h3> <p>Maximum flexibility and model choice.</p> <p>\$39 <small>USD</small></p> <p>per month or \$390 per year</p> <p>Get started</p> <p>Everything in Pro and:</p> <ul style="list-style-type: none">✓ Access to all models, including Claude Opus 4.1, o3, and more✓ 30x more premium requests than Copilot Free to use the latest models, with the option to buy more²◆ Access to GitHub Spark
---	---	--

- **Free** – darmowy plan dla indywidualnych programistów, studentów i hobbystów; pozwala na tworzenie publicznych i prywatnych repozytoriów.
- **Pro** – płatny plan dla profesjonalistów indywidualnych; oferuje więcej narzędzi, wsparcie i dodatkowe zasoby w repozytoriach.
- **Pro+** – rozszerzony plan dla profesjonalistów, którzy potrzebują jeszcze większych zasobów i zaawansowanych funkcji zarządzania kodem i integracji.

The screenshot displays the pricing page for GitHub Copilot, comparing two plans: Business and Enterprise. The Business plan is priced at \$19 USD per user per month, while the Enterprise plan is priced at \$39 USD per user per month and is marked as the 'Best value'. Both plans include a 'Get started' button and a 'Contact sales' button. The Business plan includes features like unlimited agent mode, code completions, and access to various models. The Enterprise plan includes everything in the Business plan plus access to all models and GitHub Spark.

Plan	Price (USD)	Per user / month	Key Features
Business	\$19	per user / month	Accelerate workflows with GitHub Copilot. Includes unlimited agent mode, code completions, access to code review, Claude Sonnet 3.5/3.7/4, Gemini 2.5 Pro, and more. 300 premium requests to use latest models per user.
Enterprise	\$39	per user / month	Scale with AI agents and comprehensive model access. Includes everything in Business and access to all models, including Claude Opus 4.1, o3, and more. 3.33x more premium requests than Business. Access to GitHub Spark.

- **Business** – plan dla małych i średnich zespołów programistycznych; oferuje zaawansowane narzędzia współpracy i podstawowe funkcje bezpieczeństwa.
- **Enterprise** – plan dla dużych organizacji i korporacji; zawiera wszystkie funkcje Business oraz dodatkowe opcje bezpieczeństwa, zgodności i wsparcia technicznego 24/7.

Czym jest Git?

Git to rozproszony system kontroli wersji.

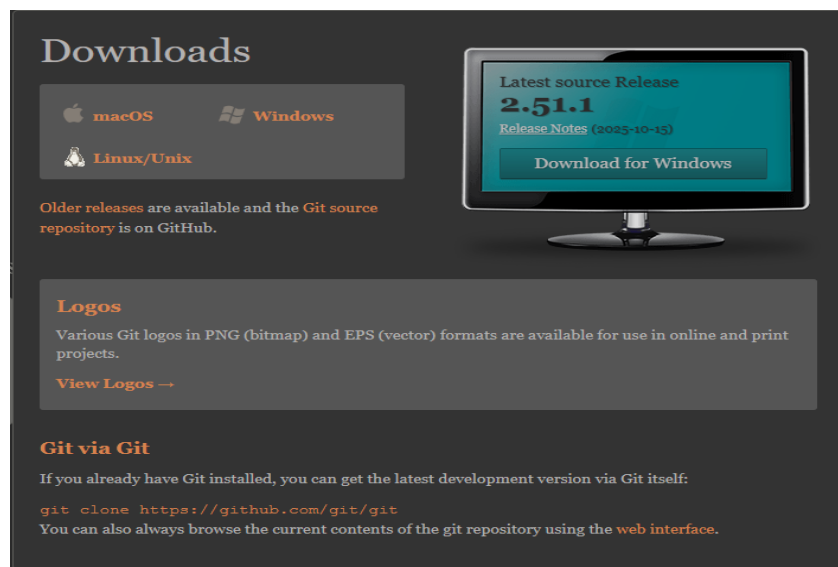
Oznacza to, że służy do **śledzenia zmian w plikach i kodzie źródłowym** oraz **współpracy wielu osób nad jednym projektem**, bez ryzyka utraty danych lub nadpisania cudzej pracy.

Jak działa Git?

Kiedy używasz Git, tworzysz coś, co nazywa się **repozytorium**. W repozytorium znajdują się twoje pliki oraz różne ich wersje i gałęzie. Git zapisuje zmiany między wersjami (tzw. *commity*) i pozwala przechowywać wiele równoległych wersji plików w postaci gałęzi (*branchy*). Można to sobie wyobrazić jak sieć lub drzewo wersji, gdzie każda gałąź reprezentuje inny kierunek rozwoju projektu

Jak pobrać Git?

1. Przechodzimy na stronę: <https://git-scm.com/downloads>
2. Wybieramy nasz system operacyjny i pobieramy



3. Pobierze się instalator, który przeprowadzi Cię przez proces instalacji.
4. Otwieramy Git cmd lub Git bash, piszemy:
 - ❖ `git config --global user.name "[tutaj twoja nazwa użytkownika z github'a]"`
 - ❖ `git config --global user.email "[adres e-mail]"`
 - ❖ `git config --list` (sprawdzamy konfig)

Uwaga!

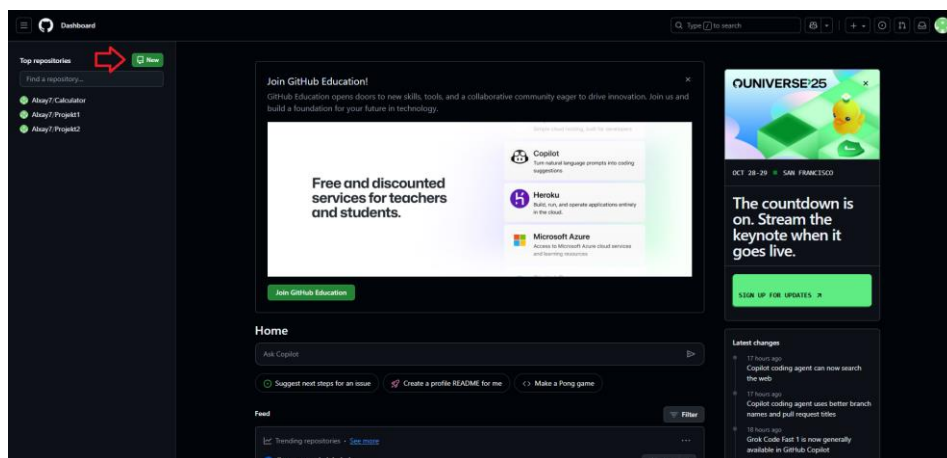
Przy pierwszej próbie wypchnięcia repozytorium do chmury, git zapyta się o nazwę użytkownika oraz hasło, aby wygenerować token.

```
$ git push -u origin main
Username for 'https://github.com': Alxay7
Password for 'https://Alxay7@github.com':
```

Jak utworzyć Repozytorium?

Sposób 1

1. Wejść na stronę: <https://github.com>

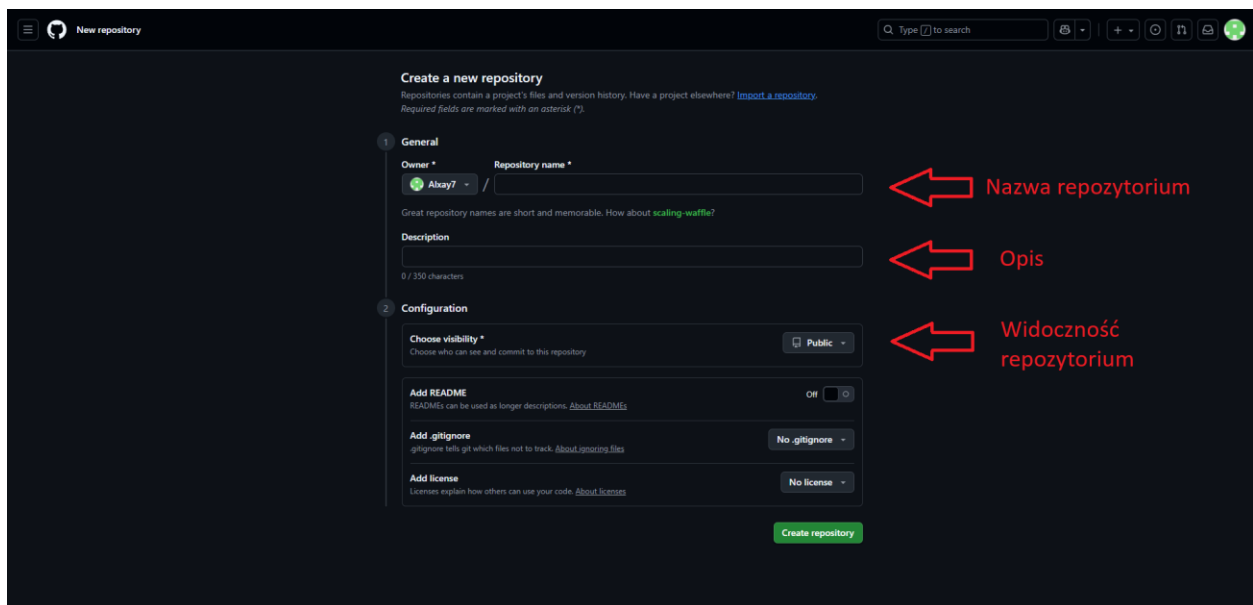


2. Kliknij zielony przycisk **New**, w lewym górnym rogu.

3. Pojawia się okno, w którym możemy zmienić właściwości repozytorium.

Uwaga!

Jeżeli ustawimy widoczność na **public**, każda osoba z dostępem do internetu będzie mogła zobaczyć nasz projekt. Jeżeli widoczność będzie ustawiona na **private**, tylko my będziemy mogli zobaczyć zawartość repozytorium.



The screenshot shows the 'Create a new repository' page on GitHub. It is divided into two sections: 'General' and 'Configuration'. In the 'General' section, there is a 'Repository name' field and a 'Description' field. In the 'Configuration' section, there is a 'Choose visibility' dropdown menu set to 'Public'. Red arrows point from the text labels on the right to these fields: 'Nazwa repozytorium' points to the 'Repository name' field, 'Opis' points to the 'Description' field, and 'Widoczność repozytorium' points to the 'Choose visibility' dropdown. At the bottom of the form is a green 'Create repository' button.

General

Owner * Repository name *

Great repository names are short and memorable. How about [scaling-waffle?](#)

Description

0 / 350 characters

Configuration

Choose visibility *

Choose who can see and commit to this repository

Add README ☐ Off

READMEs can be used as longer descriptions. [About READMEs](#)

Add gitignore

gitignore tells git which files not to track. [About ignoring files](#)

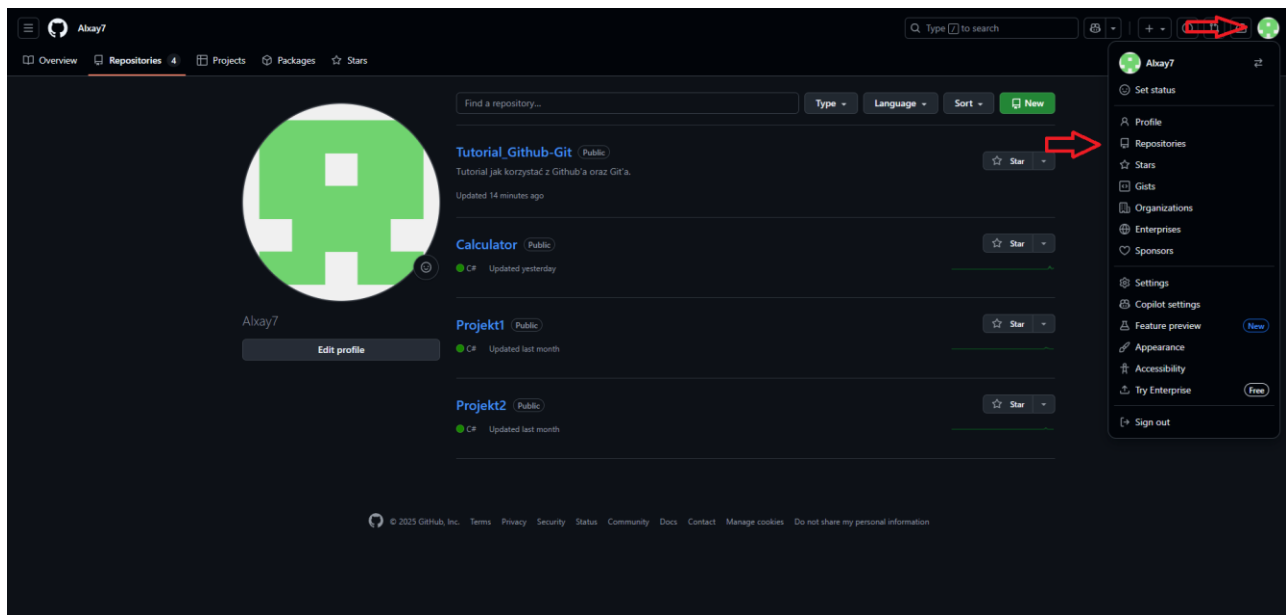
Add license

Licenses explain how others can use your code. [About licenses](#)

Create repository

Dodatkowo poniżej widoczne są opcje [Add README](#), **Add gitignore** oraz [Add license](#). Informacje na temat tych plików znajdziesz w **hiperlinkach**.

4. Kliknij Create repository.



Sposób 2

Uwaga!

Do tego sposobu musisz mieć pobranego git'a

1. Otwieramy cmd.
2. Przechodzimy do katalogu, w którym mamy pliku naszego projektu (cd “[ścieżka do pliku]”)
3. Wpisujemy następujące polecenia:
 - ✓ git init (zainicjowanie repozytorium)
 - ✓ git add . (przygotowanie wszystkich plików z danego katalogu do commita)
 - ✓ git commit -m "Initial commit (zapisuje wcześniej przygotowane pliki w historii repozytorium, w gałęzi Main z opisem “Initial commit”)

```
C:\Users\User\Desktop\Github Tutorial
λ git init
hint: Using 'master' as the name for the initial branch. This default branch name
hint: is subject to change. To configure the initial branch name to use in all
hint: of your new repositories, which will suppress this warning, call:
hint:
hint:   git config --global init.defaultBranch <name>
hint:
hint: Names commonly chosen instead of 'master' are 'main', 'trunk' and
hint: 'development'. The just-created branch can be renamed via this command:
hint:
hint:   git branch -m <name>
Initialized empty Git repository in C:/Users/User/Desktop/Github Tutorial/.git/

C:\Users\User\Desktop\Github Tutorial (master)
λ git add .

C:\Users\User\Desktop\Github Tutorial (master)
λ git commit -m "Initial commit"
[master (root-commit) 2dd23ff] Initial commit
1 file changed, 1 insertion(+)
create mode 100644 Code.py

C:\Users\User\Desktop\Github Tutorial (master)
λ |
```

4. Utworzyliśmy **lokalne repozytorium** aby je “wypchnąć na githuba” należy [stworzyć repozytorium w chmurze](#).

5. Następnie piszemy:

- ❖ git remote add origin https://github.com/[nazwa użytkownika]/[nazwa repo].git (łączy z repo w chmurze)
- ❖ git push -u origin master (wysyłamy repo do chmury)

```
C:\Users\User\Desktop\Github Tutorial (master)
λ git remote add origin https://github.com/Alxay7/Tutorial_Github-Git

C:\Users\User\Desktop\Github Tutorial (master)
λ git push -u origin master
Enumerating objects: 3, done.
Counting objects: 100% (3/3), done.
Writing objects: 100% (3/3), 231 bytes | 231.00 KiB/s, done.
Total 3 (delta 0), reused 0 (delta 0), pack-reused 0 (from 0)
To https://github.com/Alxay7/Tutorial_Github-Git
 * [new branch]      master -> master
branch 'master' set up to track 'origin/master'.

C:\Users\User\Desktop\Github Tutorial (master -> origin)
λ |
```

Gdzie znaleźć repozytoria?

Repozytorium możesz znaleźć pod adresem:

[https://github.com/\[nazwa uzytkownika\]/\[nazwa repozytorium\]](https://github.com/[nazwa uzytkownika]/[nazwa repozytorium]) lub klikając Profil/Repositories.

Branche

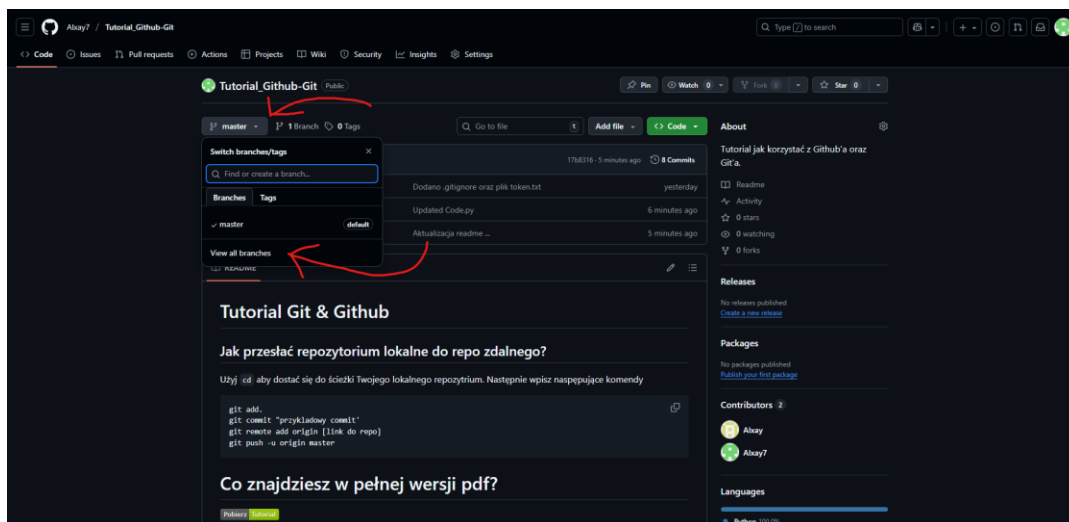
Branche (gałęzie) to jedna z najważniejszych funkcji w systemie Git. Pozwalają one na pracę nad różnymi wersjami projektu **równolegle**, bez wpływu na główną wersję kodu.

Każda gałąź to osobna linia rozwoju – możesz w niej wprowadzać zmiany, testować nowe funkcje lub naprawiać błędy, a dopiero później połączyć je z główną gałęzią (**main**).

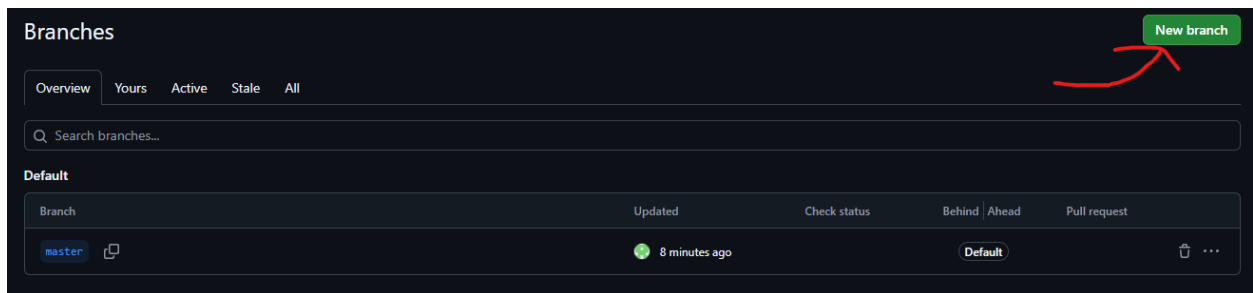
Jak zarządzać Branch'ami?

Sposób 1

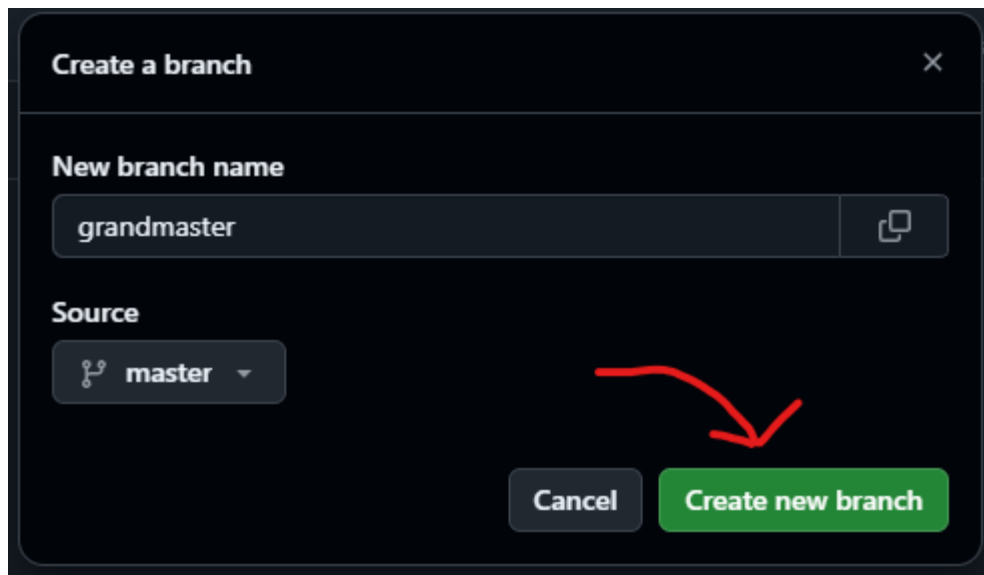
1. Przechodzimy na stronę naszego repozytorium
2. Klikamy w aktualnego brancha, a następnie View all branches



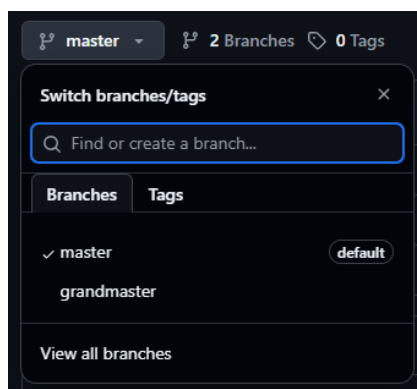
3. Na stronie widzimy branche w naszym repozytorium, aby utworzyć nowy klikamy New branch



4. Następnie podajemy nazwę i klikamy Create branch



5. Możemy wrócić na stronę repo i wybrać branch, który chcemy zobaczyć, edytować



Sposób 2

1. Łączymy się z naszym [repozytorium w terminalu](#) (git add remote origin [link do repo])
2. Wpisujemy git branch [nazwa_brancha]

```
C:\Users\User\Desktop\Github Tutorial (master -> origin)  
λ git branch World_Champion
```

3. Jeżeli chcemy teraz pracować na tym branchu piszemy: git switch [nazwa_brancha]

```
C:\Users\User\Desktop\Github Tutorial (master -> origin)  
λ git switch World_Champion  
Switched to branch 'World_Champion'
```

4. Po wypchnięciu do zdalnego repo, branch powinien być widoczny online

```
C:\Users\User\Desktop\Github Tutorial (World_Champion)  
λ git push -u origin World_Champion
```

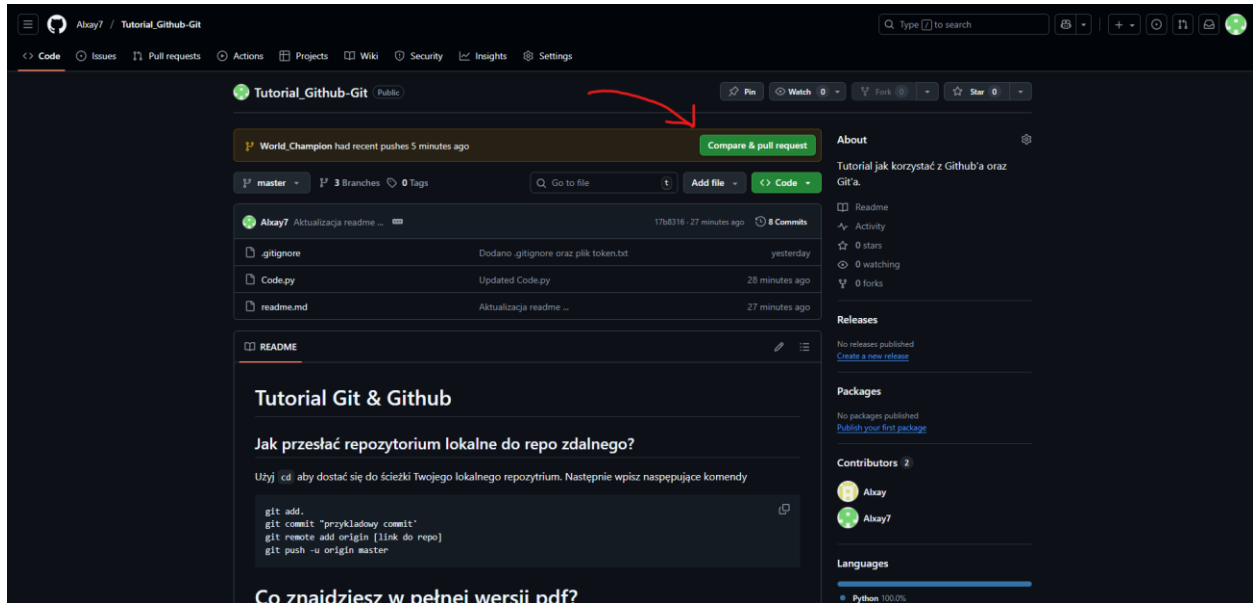
Pull requests

Pull Request (w skrócie **PR**) to **prośba o włączenie zmian z jednej gałęzi (branch) do innej** — najczęściej z gałęzi roboczej (**feature**) do głównej (**main**).

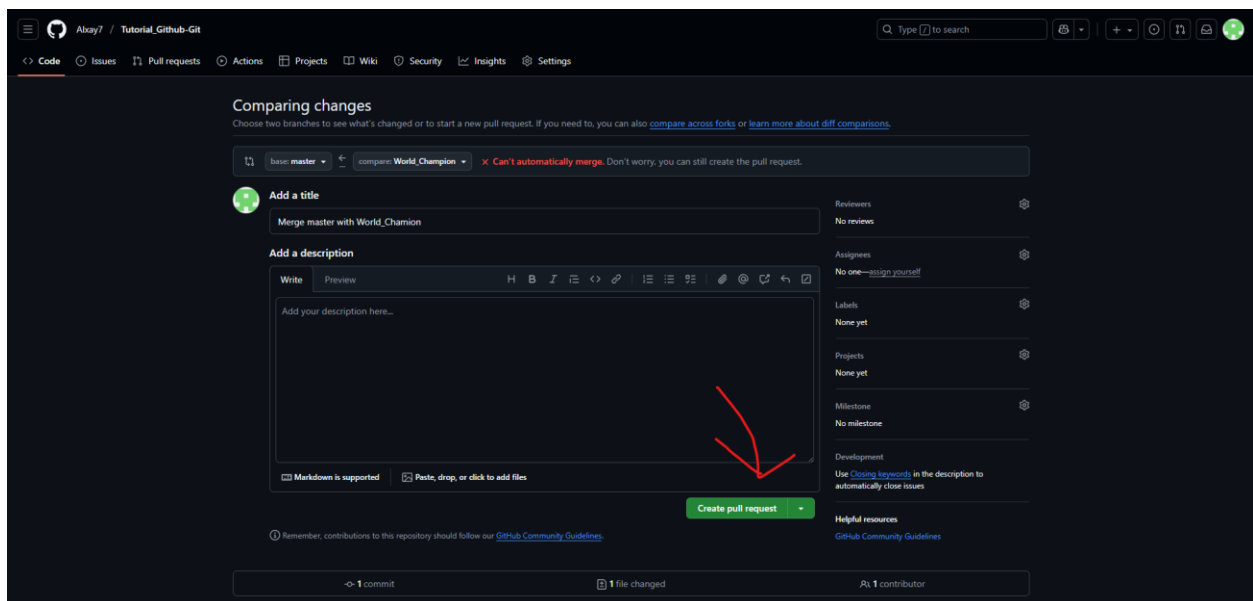
Dzięki temu można **przeglądać, komentować i zatwierdzać zmiany**, zanim zostaną połączone z głównym kodem.

Po przestaniu plików do nowej gałęzi możemy wykonać pull request.

1. Klikamy Compare & pull request na stronie naszego repo (powinno być widoczne po co najmniej 2 branchy z różnymi plikami).



2. Na następnej stronie klikamy Create pull request

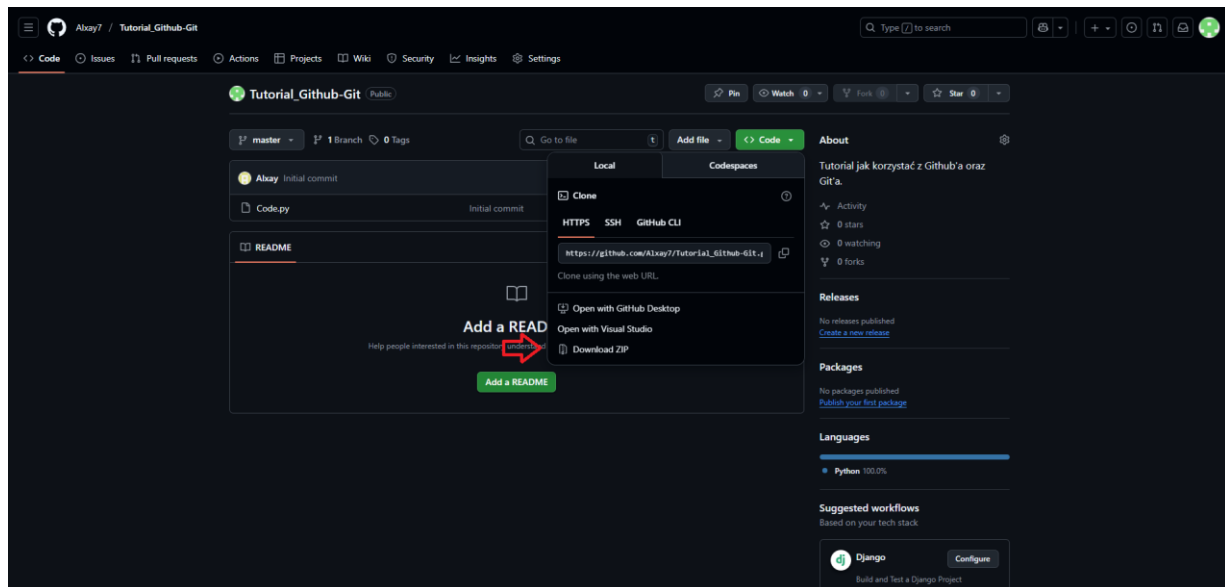


3. Pliki z wybranego brancha powinny pojawić się w branchu master

Jak pobrać repozytorium?

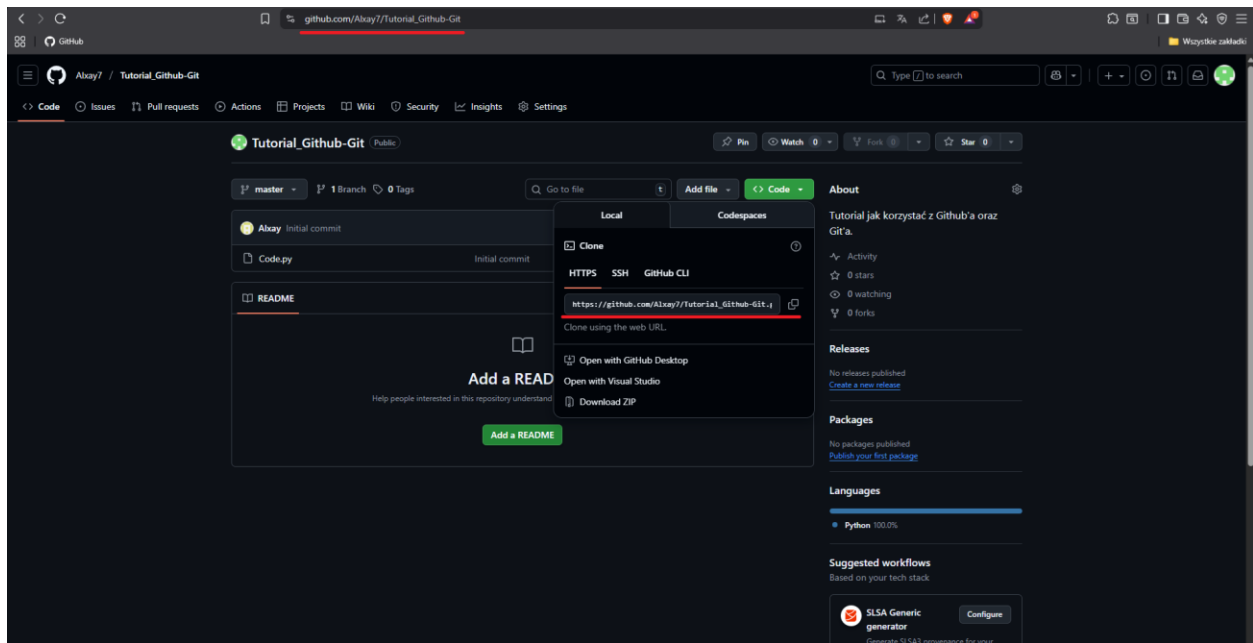
Sposób 1

1. Wchodzimy na stronę repozytorium.
2. Klikamy przycisk Code, następnie wybieramy “Dwload ZIP”



Sposób 2

1. Wchodzimy na stronę repozytorium.
2. Kopiujemy link do repozytorium (można go znaleźć po kliknięciu przycisku Code, lub w pasku adresu przeglądarki).



3. Otwieramy cmd przechodzimy do katalogu, do którego chcemy pobrać nasz plik.
4. Wpisujemy: git clone [adres repozytorium].

Uwaga!

Jeżeli kopiujemy link z paska adresu przeglądarki należy podmienić # na końcu linku na .git

```
Clink v1.8.6 is available.
- To apply the update, run 'clink update'.
- To stop checking for updates, run 'clink set clink.autoupdate off'.
- To view the release notes, visit the Releases page:
  https://github.com/chrisant996/clink/releases

C:\Program Files\Cmder
λ cd ../

C:\Program Files
λ cd ../

C:\
λ cd Users\User\Desktop

C:\Users\User\Desktop
λ git clone https://github.com/Alxay7/Tutorial_Github-Git.git
Cloning into 'Tutorial_Github-Git'...
remote: Enumerating objects: 3, done.
remote: Counting objects: 100% (3/3), done.
remote: Total 3 (delta 0), reused 3 (delta 0), pack-reused 0 (from 0)
Receiving objects: 100% (3/3), done.

C:\Users\User\Desktop
λ
```

Readme

Plik **README.md** to podstawowy plik opisowy każdego repozytorium na GitHubie. Zawiera informacje o projekcie, jego celu, funkcjach, sposobie instalacji, użycia oraz autorach. Jest zapisany w języku **Markdown (.md)**, który umożliwia łatwe formatowanie tekstu, takie jak nagłówki, listy, linki i obrazy. Po umieszczeniu pliku w repozytorium jego zawartość jest automatycznie wyświetlana na stronie głównej projektu. Więcej informacji na temat formatowania tekstu w plikach README znajdziesz [tutaj](#).

Przykład

```
# Tutorial Git & Github
## Jak przesłać repozytorium lokalne do repo zdalnego?
Użyj `cd` aby dostać się do ścieżki Twojego lokalnego repozytrium.
Następnie wpisz następujące komendy
...

git add.
git commit "przykładowy commit"
git remote add origin [link do repo]
git push -u origin master
...

# Co znajdziesz w pełnej wersji pdf?
[[Pobierz](https://img.shields.io/badge/Pobierz-Repozytorium-blue)](https://github.com/Alxay7/Tutorial-Git/blob/master/tutorial.pdf)
Pełna wersja *pdf* zawiera podstawowe informacje na temat korzystania oraz możliwości
zarówno *githuba* jak i *gita*. Po samouczku dowiesz się, jak utworzyć własne repozytoria te lokalne
jak i te zdalne, jak dodawać commity, jak pobrać twoje zdalne repozytorium i wiele więcej!

> "I think it's possible for ordinary people to choose to be extraordinary." ~ Elon Musk
```

Po wrzuceniu na githuba readme wygląda tak:

README

Tutorial Git & Github

Jak przesłać repozytorium lokalne do repo zdalnego?

Użyj `cd` aby dostać się do ścieżki Twojego lokalnego repozytrium. Następnie wpisz następujące komendy

```
git add.  
git commit "przykładowy commit"  
git remote add origin [link do repo]  
git push -u origin master
```

Co znajdziesz w pełnej wersji pdf?

Pobierz Repozytorium

Pełna wersja *pdf* zawiera podstawowe informacje na temat korzystania oraz możliwości zarówno *githuba* jak i *gita*. Po samouczku dowiesz się, jak utworzyć własne repozytoria te lokalne jak i te zdalne, jak dodawać commity, jak pobrać twoje zdalne repozytorium i wiele więcej!

"I think it's possible for ordinary people to choose to be extraordinary." ~ Elon Musk

Licencje

Każde publiczne repozytorium na GitHubie powinno mieć plik **LICENSE** – określa on, **na jakich zasadach inni mogą korzystać z Twojego kodu**. Bez licencji Twój projekt jest **domyślnie chroniony prawem autorskim**, co oznacza, że inni **nie mogą legalnie kopiować, modyfikować ani wykorzystywać** Twojego kodu.

GitHub oferuje kilka popularnych rodzajów licencji, które możesz łatwo dodać podczas [tworzenia repozytorium](#) lub ręcznie, tworząc plik LICENSE. Więcej informacji na temat licencji znajdziesz [tutaj](#).

Najpopularniejsze licencje

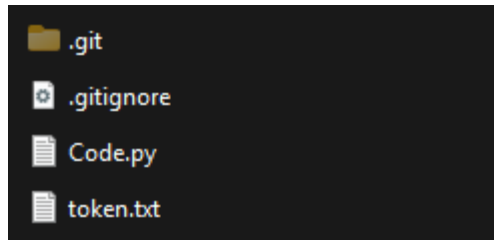
Licencja	Dla kogo	Co pozwala	Co wymaga
----------	----------	------------	-----------

MIT License	Najczęściej używana	Dowolne użycie, modyfikacja, dystrybucja	Zachowanie informacji o autorze
Apache 2.0	Dla firm i projektów komercyjnych	Dowolne użycie + ochrona patentowa	Zachowanie informacji o licencji i zmianach
GNU GPL v3	Dla projektów open-source	Wymaga udostępnienia kodu źródłowego po modyfikacji	Utrzymanie tej samej licencji
BSD 2/3-Clause	Dla projektów naukowych / edukacyjnych	Swobodne użycie i modyfikacja	Uznanie autorstwa
Creative Commons (CC0)	Dla dokumentacji, grafik itp.	Pełne udostępnienie (public domain)	Brak wymagań

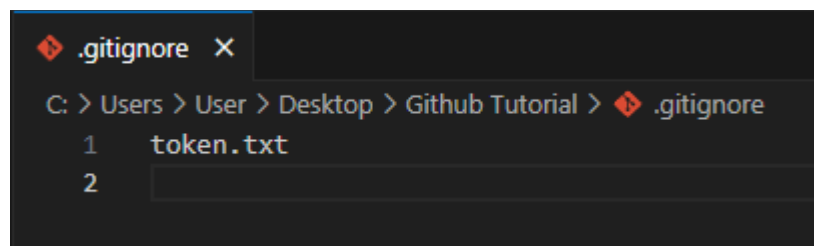
.gitignore

Plik **.gitignore** służy do określania, które pliki i foldery mają być pomijane przez system Git podczas śledzenia zmian w repozytorium. Dzięki niemu można uniknąć przypadkowego dodania do repozytorium plików tymczasowych, konfiguracyjnych lub prywatnych, takich jak dane logowania, pliki kompilacji czy ustawienia środowiska. Plik **.gitignore** zawiera listę wzorców nazw plików lub katalogów, które Git ma ignorować. Jest szczególnie przydatny w projektach programistycznych, gdzie generowane są pliki binarne, zależności lub dane lokalne, które nie powinny być przechowywane w repozytorium.

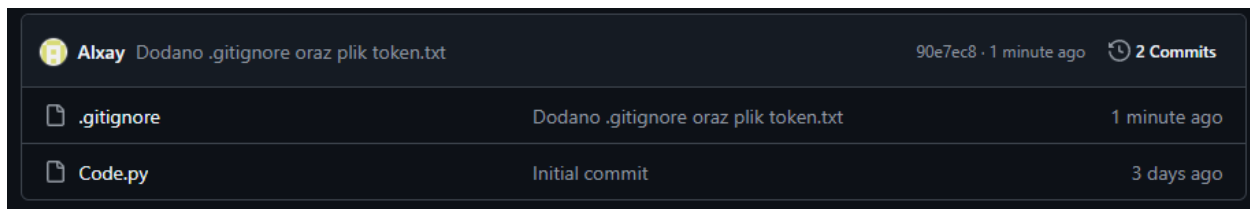
Przykład użycia:



Mam takie pliki (patrz zdjęcia powyżej), w pliku token.txt przechowuje poufne informacje, a więc nie chcę aby w moim publicznym repozytorium po przesłaniu plików ktoś je zobaczył. Utworzyłem więc plik .gitignore i zapisałem, w nim "token.txt".



Teraz przesyłam [repozytorium do chmury](#) . Po przesłaniu możemy zaobserwować, że mamy plik **.gitignore**, natomiast plik **token.txt** nie został dodany do zdalnego repozytorium



Codespaces

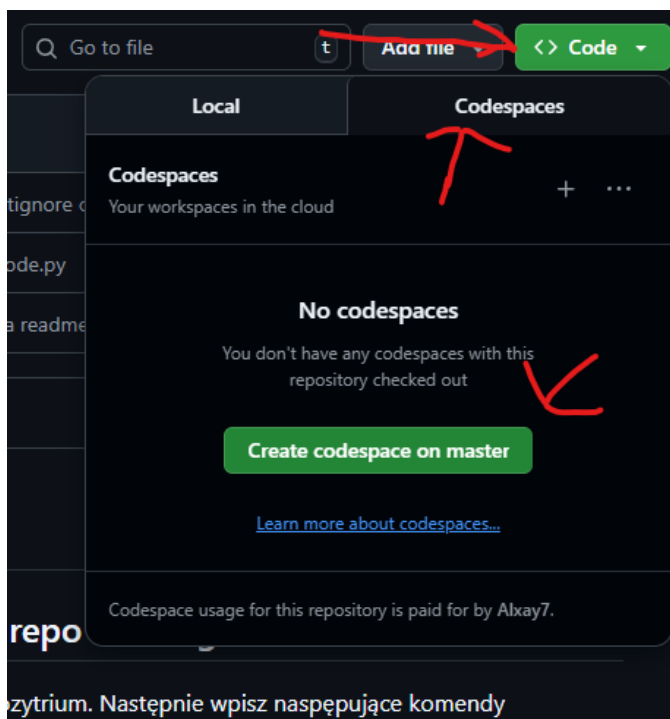
GitHub Codespaces to **wirtualne środowisko programistyczne w chmurze**, które pozwala pisać, testować i uruchamiać kod **bezpośrednio w przeglądarce**.

To tak, jakbyś miał zainstalowanego **Visual Studio Code** online – dostępnego z każdego miejsca, bez potrzeby konfiguracji komputera.

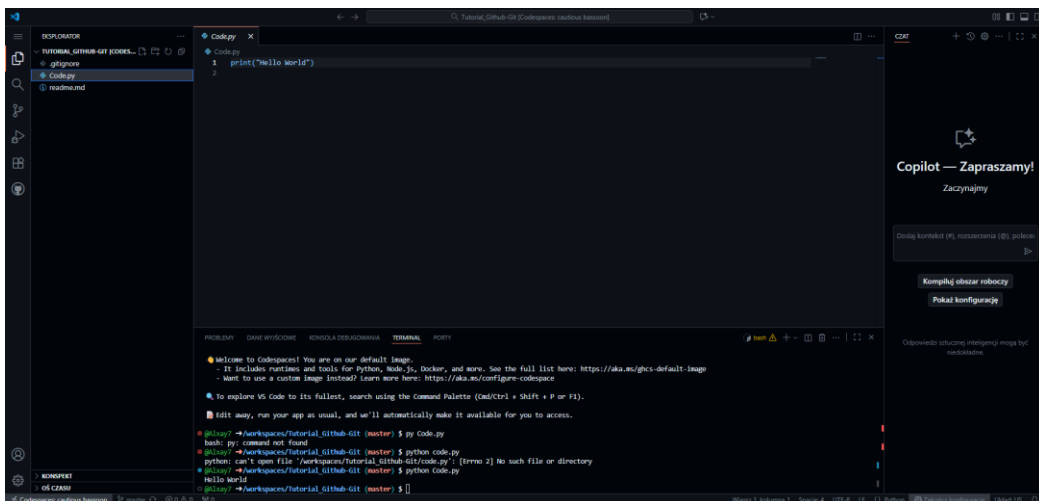
Codespaces mogą być tworzone na podstawie repozytorium jak i bez żadnych plików na start.

Codespace na podstawie repo

1. Przechodzimy na stronę repozytorium
2. Klikamy Code>Codespaces>Create codespace

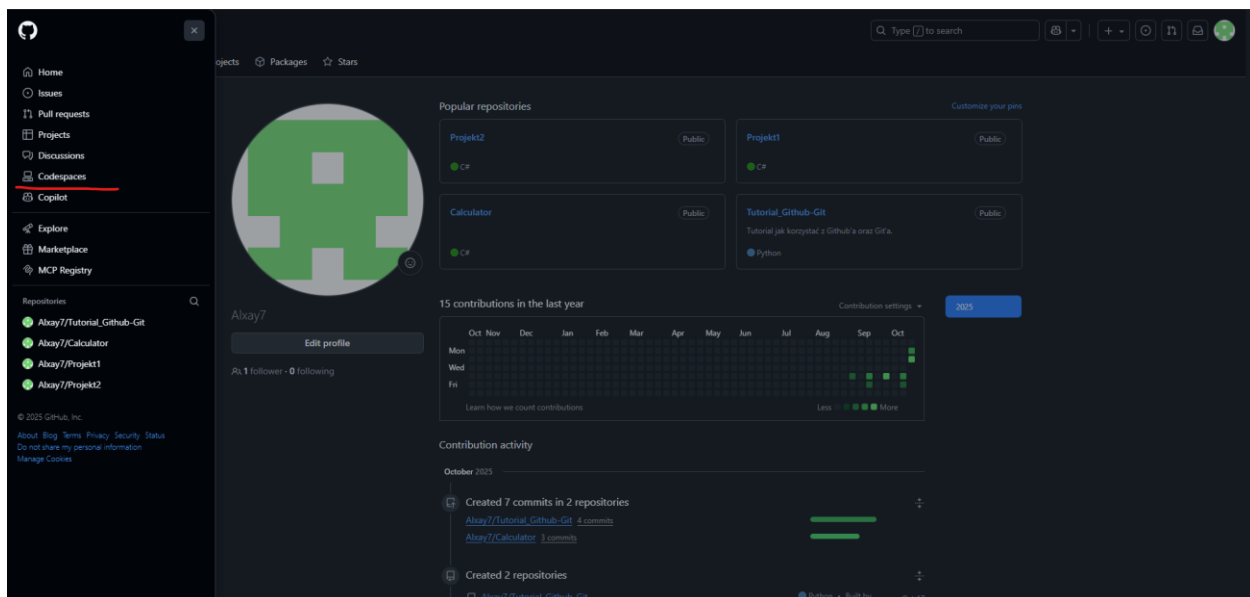


3. Powinniśmy mieć włączonego VS code w przeglądarce z plikami z naszego repozytorium

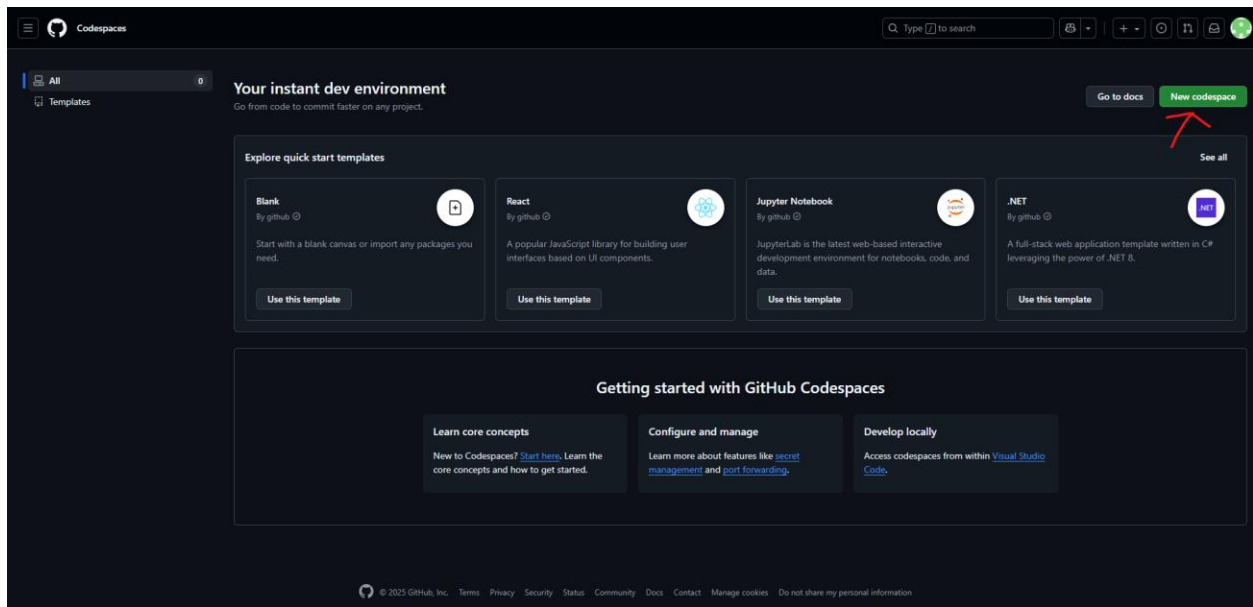


Jak utworzyć czysty codespace

1. Przechodzimy na stronę naszego profilu na githubie
2. Klikamy ikonkę, w lewym górnym rogu > Codespaces



3. Klikamy przycisk new codespace



Przydatne komendy Git

Komenda	Opis
<code>git init</code>	Tworzy nowe lokalne repozytorium Git
<code>git clone <url></code>	Pobiera (klonuje) istniejące repozytorium z GitHuba
<code>git status</code>	Pokazuje status plików (zmienione, nowe, nieśledzone)
<code>git add <plik></code>	Dodaje plik do obszaru przygotowania (<i>staging area</i>)
<code>git add .</code>	Dodaje wszystkie zmienione pliki
<code>git commit -m "opis"</code>	Zapisuje zmiany z komentarzem
<code>git log</code>	Pokazuje historię commitów
<code>git diff</code>	Pokazuje różnice między wersjami plików
<code>git show <commit></code>	Pokazuje szczegóły konkretnego commitu

Praca z gałęziami (branches)

Komenda	Opis
git branch	Wyświetla listę gałęzi
git branch <nazwa>	Tworzy nową gałąź
git checkout <nazwa>	Przełącza się na inną gałąź
git switch <nazwa>	Alternatywa dla checkout
git merge <nazwa>	Łączy inną gałąź z bieżącą
git branch -d <nazwa>	Usuwa lokalną gałąź
git branch -m <stara> <nowa>	Zmienia nazwę gałęzi

Praca z **Githubem** (zdalne repozytorium)

Komenda	Opis
git remote -v	Pokazuje powiązane repozytoria zdalne
git remote add origin <url>	Dodaje repozytorium zdalne
git push -u origin main	Wysyła (pushuje) lokalne zmiany do GitHuba
git pull	Pobiera najnowsze zmiany z GitHuba
git fetch	Pobiera zmiany zdalne, ale nie łączy ich od razu
git clone <url>	Kopiuje całe repozytorium z GitHuba lokalnie

Cofanie i resetowanie zmian

Komenda	Opis
git restore <plik>	Przywraca plik do wersji z ostatniego commitu
git reset <plik>	Usuwa plik z <i>staging area</i>
git reset --hard <commit>	Przywraca repozytorium do konkretnego commitu (usuwa zmiany!)
git revert <commit>	Tworzy nowy commit cofający zmiany z wybranego commitu
git checkout -- <plik>	Przywraca plik do poprzedniego stanu

Informacje i diagnostyka

Komenda	Opis
git config --list	Pokazuje wszystkie ustawienia Git
git config --global user.name "Twoje Imię"	Ustawia globalną nazwę użytkownika
git config --global user.email " email@example.com "	Ustawia globalny adres e-mail
git help <komenda>	Pokazuje pomoc dotyczącą danej komendy
git log --oneline --graph --all	Skrócona historia commitów w formie drzewa

Inne

Komenda	Opis
git tag <nazwa>	Tworzy tag wersji dla konkretnego commitu
git push origin <tag>	Wysyła tag do repozytorium zdalnego
git clean -f	Usuwa nieśledzone pliki z katalogu projektu
git gc	Optymalizuje repozytorium, czyści stare dane
git credential-manager-core list	Wyświetla zapisane dane logowania do repozytoriów
git credential-manager-core erase	Usuwa zapisane dane logowania
git remote set-url origin <nowy_url>	Zmienia adres repozytorium zdalnego
git add . && git commit -m "msg" && git push	Szybkie wykonanie dodania, commitu i wysłania zmian
git branch --show-current	Pokazuje nazwę bieżącej gałęzi
git reset --soft HEAD~1	Cofnięcie ostatniego commitu bez usuwania zmian