Przewodnik: GitHub i Git dla Windows

Informacje Ogólne

Przeznaczenie: Instrukcja krok po kroku dla studentów rozpoczynających pracę z Git i GitHub
System: Windows 10/11
Poziom: Początkujący
Szacowany czas: 30-45 minut

Część 1: Tworzenie Konta na GitHub.com

Krok 1: Rejestracja

1. **Otwórz przeglądarkę** i przejdź na stronę: **https://github.com**
2. **Kliknij przycisk "Sign up"** (Zarejestruj się) w prawym górnym rogu
3. **Wypełnij formularz rejestracyjny**:
    - **Email**: Podaj swój adres email (najlepiej uczelnianego lub prywatnego, którego regularnie używasz)
    - **Password**: Stwórz silne hasło (minimum 15 znaków lub 8 znaków z cyfrą i małą literą)
    - **Username**: Wybierz nazwę użytkownika
        - **WAŻNE**: Wybierz profesjonalną nazwę - będziesz jej używać w projektach i CV
        - Dobre przykłady: jan-kowalski, jkowalski-dev, kowalskijan
        - Złe przykłady: xXx_destroyer_xXx, kotekmruczek123
    - **Email preferences**: Możesz odznaczyć, jeśli nie chcesz otrzymywać newslettera
4. **Weryfikacja** - rozwiąż puzzle weryfikacyjne (np. dopasuj obrazki)
5. **Potwierdź email**:
    - Sprawdź swoją skrzynkę email
    - Otwórz wiadomość od GitHub
    - Kliknij link aktywacyjny
6. **Ankieta początkowa** (opcjonalnie):
    - GitHub może zapytać o:
        - Liczbę członków zespołu (możesz wybrać "Just me")
        - Czy jesteś studentem/nauczycielem
        - Dla jakich celów będziesz używać GitHub
    - Możesz pominąć, klikając "Skip personalization"

Gratulacje! Masz już konto na GitHub

Część 2: Instalacja Git na Windows

Krok 1: Pobieranie Git

1. Przejdź na stronę: **https://git-scm.com/download/win**
2. Pobieranie powinno rozpocząć się automatycznie
    - Jeśli nie, kliknij odpowiedni link (zazwyczaj "64-bit Git for Windows Setup")

Krok 2: Instalacja

1. **Uruchom pobrany plik** (np. Git-2.43.0-64-bit.exe)
2. **Przejdź przez kroki instalacji**:
    - **License**: Kliknij “Next”
    - **Destination Location**: Zostaw domyślną ścieżkę → “Next”
    - **Select Components**: Zostaw domyślne zaznaczenia → “Next”
    - **Select Start Menu Folder**: Zostaw domyślnie → “Next”
    - **Choosing the default editor**:
        - Jeśli nie wiesz, co wybrać, zostaw “Vim” lub wybierz “Nano”
        - Możesz też wybrać “Visual Studio Code” jeśli go masz
        - Kliknij “Next”
    - **Adjusting the name of the initial branch**:
        - Wybierz “Override the default branch name for new repositories”
        - Wpisz: main
        - Kliknij “Next”
    - **Adjusting your PATH environment**:
        - Wybierz: “Git from the command line and also from 3rd-party software” (domyślne)
        - Kliknij “Next”
    - **Choosing HTTPS transport backend**: Zostaw domyślnie → “Next”
    - **Configuring the line ending conversions**:
        - Zostaw: “Checkout Windows-style, commit Unix-style line endings”
        - Kliknij “Next”
    - **Configuring the terminal emulator**:
        - Zostaw domyślnie: “Use MinTTY”
        - Kliknij “Next”
    - **Choose the default behavior of ‘git pull’**: Zostaw domyślnie → “Next”
    - **Choose a credential helper**: Zostaw domyślnie → “Next”
    - **Configuring extra options**: Zostaw domyślnie → “Next”
    - **Configuring experimental options**: Nic nie zaznaczaj → “Install”
3. **Poczekaj na zakończenie instalacji** → Kliknij “Finish”

Krok 3: Weryfikacja instalacji

1. **Otwórz Command Prompt (CMD)**:
    - Naciśnij Win + R
    - Wpisz: cmd
    - Naciśnij Enter
2. **Sprawdź wersję Git**:

```
git --version
```

Powinno wyświetlić się coś w stylu: git version 2.43.0.windows.1

Git jest zainstalowany!

Część 3: Konfiguracja Git

Podstawowa konfiguracja (wymagana)

Otwórz **Command Prompt (CMD)** lub **Git Bash** i wykonaj poniższe komendy:

```
# Ustaw swoją nazwę (będzie widoczna w commitach)
git config --global user.name "Jan Kowalski"

# Ustaw swój email (ten sam co na GitHub)
git config --global user.email "jan.kowalski@example.com"
```

Sprawdzenie konfiguracji

```
# Wyświetl całą konfigurację
git config --list

# Lub sprawdź konkretne ustawienia
git config user.name
git config user.email
```

Opcjonalne ustawienia (zalecane)

```
# Ustaw domyślną nazwę głównej gałęzi na "main"
git config --global init.defaultBranch main

# Włącz kolorowanie w terminalu
git config --global color.ui auto

# Ustaw domyślny edytor (np. Notepad)
git config --global core.editor notepad
```

Część 4: Tworzenie Pierwszego Repozytorium na GitHub

Metoda 1: Przez Stronę Internetową

1. **Zaloguj się na GitHub** (https://github.com)
2. **Kliknij zielony przycisk “New”** (lub ikonę “+” w prawym górnym rogu → “New repository”)
3. **Wypełnij formularz**:
    - **Repository name**: Wpisz nazwę, np. moj-pierwszy-projekt
        - Używaj małych liter, myślników zamiast spacji
        - Nazwa powinna być opisowa
    - **Description** (opcjonalnie): Krótki opis, np. “Mój pierwszy projekt na GitHub”
    - **Public / Private**:

        - **Public**: Każdy może zobaczyć (zalecane dla nauki)
        - **Private**: Tylko Ty i osoby, które zaprosisz

    - **Initialize this repository with**:

        - Zaznacz “Add a README file” (zalecane dla początkujących)
        - Możesz też wybrać “.gitignore” template (np. dla Python, Node, itp.)
        - Możesz wybrać licencję (np. MIT License)

4. **Kliknij "Create repository"**

Repozytorium utworzone!

Część 5: Praca z Repozytorium - Podstawy

Scenariusz 1: Klonowanie Istniejącego Repozytorium

1. **Na stronie swojego repozytorium na GitHub**:

    - Kliknij zielony przycisk **"Code"**

    - Skopiuj URL (zakładka HTTPS), np.:

      https://github.com/twoja-nazwa/moj-pierwszy-projekt.git

2. **Otwórz Command Prompt**:

   ```
   # Przejdź do folderu, gdzie chcesz mieć projekt (np. Dokumenty)
   cd C:\Users\TwojaNazwa\Documents

   # Sklonuj repozytorium
   git clone https://github.com/twoja-nazwa/moj-pierwszy-projekt.git

   # Wejdź do folderu projektu
   cd moj-pierwszy-projekt
   ```

Scenariusz 2: Tworzenie Lokalnego Repozytorium i Wysyłanie na GitHub

1. **Stwórz folder projektu**:

   ```
   # Utwórz folder
   mkdir C:\Users\TwojaNazwa\Documents\nowy-projekt

   # Wejdź do niego
   cd C:\Users\TwojaNazwa\Documents\nowy-projekt
   ```

2. **Zainicjuj repozytorium Git**:

   ```
   git init
   ```

3. **Stwórz plik README**:

   ```
   echo "# Mój Nowy Projekt" > README.md
   ```

4. **Dodaj plik do śledzenia**:

   ```
   git add README.md
   ```

5. **Wykonaj pierwszy commit**:

   ```
   git commit -m "Pierwszy commit: dodanie README"
   ```

6. **Połącz z repozytorium na GitHub**:

   Najpierw **utwórz puste repozytorium na GitHub** (bez README), a następnie:

   ```
   # Dodaj zdalne repozytorium
   git remote add origin https://github.com/twoja-nazwa/nowy-projekt.git

   # Wyślij zmiany na GitHub
   git push -u origin main
   ```

Część 6: Podstawowe Komendy Git

Status i Informacje

```
# Sprawdź status repozytorium (które pliki zostały zmienione)
git status

# Zobacz historię commitów
git log

# Krótsza wersja historii
git log --oneline

# Zobacz różnice w plikach (co się zmieniło)
git diff
```

Dodawanie Zmian

```
# Dodaj konkretny plik do stage'a
git add nazwa-pliku.txt

# Dodaj wszystkie zmienione pliki
git add .

# Dodaj wszystkie pliki z rozszerzeniem .js
git add *.js

# Dodaj wszystkie pliki z folderu
git add nazwa-folderu/
```

Zapisywanie Zmian (Commit)

```
# Wykonaj commit z wiadomością
git commit -m "Opis zmian w projekcie"

# Dodaj wszystko i od razu commituj (skrót)
git commit -am "Opis zmian"

# Commit z dłuższym opisem (otworzy edytor)
git commit
```

Dobre praktyki dla wiadomości commit:

- Używaj czasu teraźniejszego: “Dodaj funkcję” zamiast “Dodano funkcję”
- Bądź konkretny: “Napraw błąd w walidacji formularza” zamiast “Poprawki”

- Krótko (50-72 znaki)

Synchronizacja z GitHub

```
# Pobierz zmiany z GitHub (ale nie łącz)
git fetch

# Pobierz i połącz zmiany z GitHub
git pull

# Wyślij swoje zmiany na GitHub
git push

# Wyślij nową gałąź na GitHub
git push -u origin nazwa-galezi
```

Gałęzie (Branches)

```
# Zobacz wszystkie gałęzie
git branch

# Stwórz nową gałąź
git branch nazwa-galezi

# Przełącz się na inną gałąź
git checkout nazwa-galezi

# Stwórz i od razu przełącz się na nową gałąź (skrót)
git checkout -b nazwa-galezi

# Usuń gałąź lokalnie
git branch -d nazwa-galezi

# Połącz gałąź z aktualną
git merge nazwa-galezi
```

Cofanie Zmian

```
# Cofnij zmiany w pliku (przed dodaniem do stage)
git checkout -- nazwa-pliku.txt

# Usuń plik ze stage'a (zostaw zmiany w pliku)
git reset HEAD nazwa-pliku.txt

# Cofnij ostatni commit (zostaw zmiany w plikach)
git reset --soft HEAD~1

# Cofnij ostatni commit i usuń zmiany
git reset --hard HEAD~1
```

Usuwanie i Przenoszenie

```
# Usuń plik (z systemu i z Git)
git rm nazwa-pliku.txt

# Przenieś/zmień nazwę pliku
git mv stara-nazwa.txt nowa-nazwa.txt
```

Część 7: Typowy Workflow

Codzienna praca z projektem:

```
# 1. Zacznij od pobrania najnowszych zmian
git pull

# 2. Pracuj nad kodem, twórz/edytuj pliki
# (tu używasz swojego edytora/IDE)

# 3. Sprawdź, co się zmieniło
git status

# 4. Dodaj zmiany do stage'a
git add .

# 5. Wykonaj commit
git commit -m "Opisz swoje zmiany"

# 6. Wyślij na GitHub
git push
```

Praca z nową funkcjonalnością (feature branch):

```
# 1. Stwórz nową gałąź dla funkcjonalności
git checkout -b nowa-funkcja

# 2. Pracuj nad kodem
# ... edycja plików ...

# 3. Commituj zmiany
git add .
git commit -m "Dodaj nową funkcjonalność"

# 4. Wyślij gałąź na GitHub
git push -u origin nowa-funkcja

# 5. Wróć na główną gałąź
git checkout main

# 6. Połącz zmiany
git merge nowa-funkcja

# 7. Wyślij zaktualizowaną główną gałąź
git push
```

Część 8: Autentykacja GitHub (Ważne!)

Problem: GitHub nie akceptuje już haseł!

Od sierpnia 2021 GitHub wymaga **Personal Access Token** zamiast hasła przy operacjach przez HTTPS.

Rozwiązanie 1: Personal Access Token (PAT)

1. **Wygeneruj token na GitHub**:
    - Przejdź do: https://github.com/settings/tokens
    - Kliknij “Generate new token” → “Generate new token (classic)”
    - Nadaj nazwę tokenowi (np. “Mój laptop”)
    - Wybierz uprawnienia:
        - `repo` (pełny dostęp do repozytoriów)
        - `workflow` (jeśli używasz GitHub Actions)
    - Kliknij “Generate token”

- **WAŻNE**: Skopiuj token! **Nie będziesz go już więcej widział!**

2. **Użyj tokena zamiast hasła**:
   - Gdy Git poprosi o hasło, wklej token
3. **Zapisz token w Windows (aby nie wpisywać za każdym razem)** :

```
# Windows przechowa token w Credential Manager
git config --global credential.helper wincred
```

Rozwiązanie 2: GitHub CLI (gh) - Prostsze!

```
# 1. Zainstaluj GitHub CLI
# Pobierz z: https://cli.github.com/

# 2. Zaloguj się
gh auth login

# Wybierz:
# - GitHub.com
# - HTTPS
# - Yes (authenticate)
# - Login with a web browser
# - Postępuj zgodnie z instrukcjami w przeglądarce
```

Rozwiązanie 3: SSH (dla zaawansowanych)

```
# 1. Wygeneruj klucz SSH
ssh-keygen -t ed25519 -C "twoj-email@example.com"

# 2. Dodaj klucz do GitHub
# Skopiuj zawartość pliku: C:\Users\TwojaNazwa\.ssh\id_ed25519.pub
# Wklej na: https://github.com/settings/ssh/new

# 3. Używaj URL SSH zamiast HTTPS:
# git clone git@github.com:username/repo.git
```

Część 9: Przydatne Narzędzia

Git GUI (Graficzny interfejs)

Jeśli preferujesz interfejs graficzny zamiast wiersza poleceń:

1. **GitHub Desktop** (zalecane dla początkujących)
   - Pobierz: https://desktop.github.com/
   - Bardzo prosty w obsłudze
   - Integracja z GitHub
2. **GitKraken**
   - https://www.gitkraken.com/
   - Ładny interfejs
   - Wersja darmowa dla projektów publicznych
3. **SourceTree**
   - https://www.sourcetreeapp.com/
   - Darmowy
   - Od Atlassian

Edytory Kodu z Integracją Git

- **Visual Studio Code** (zalecany!)
  - https://code.visualstudio.com/
  - Wbudowana obsługa Git
  - Darmowy i potężny
- **IntelliJ IDEA / WebStorm / PyCharm**
  - Mają świetną integrację z Git

Część 10: .gitignore - Ignorowanie Plików

Czym jest .gitignore?

Plik .gitignore mówi Git, które pliki/foldery **nie** powinny być śledzone.

Kiedy używać?

Ignoruj:

- Pliki tymczasowe (.tmp, .log)
- Pliki konfiguracyjne z hasłami/kluczami
- Foldery z bibliotekami (node_modules/, venv/)
- Pliki IDE (.idea/, .vscode/)
- Pliki systemu operacyjnego (.DS_Store, Thumbs.db)

Jak utworzyć .gitignore?

1. **W katalogu głównym projektu utwórz plik .gitignore**:

   ```
   notepad .gitignore
   ```

2. **Przykładowa zawartość** (dla projektu webowego):

   ```
   # Foldery dependencies
   node_modules/
   bower_components/

   # Pliki środowiskowe
   .env
   .env.local

   # Logi
   *.log
   npm-debug.log*

   # Pliki systemu
   .DS_Store
   Thumbs.db

   # Foldery IDE
   .idea/
   .vscode/
   *.suo
   *.user

   # Pliki kompilacji
   dist/
   build/
   *.min.js
   ```

```
*.min.css
```

3. **Dodaj do repozytorium**:

```
git add .gitignore
git commit -m "Dodaj .gitignore"
git push
```

Gotowe szablony .gitignore

GitHub oferuje gotowe szablony dla różnych języków/frameworków:

- https://github.com/github/gitignore

Lub użyj generatora:

- https://www.toptal.com/developers/gitignore

Najczęstsze Problemy i Rozwiązania

Problem 1: “Permission denied (publickey)”

Rozwiązanie: Używaj HTTPS zamiast SSH, lub skonfiguruj klucz SSH (patrz Część 8)

Problem 2: “fatal: not a git repository”

Rozwiązanie: Jesteś w złym folderze lub nie zainicjowałeś repozytorium

```
git init
```

Problem 3: “Your branch is behind ‘origin/main’”

Rozwiązanie: Musisz pobrać najnowsze zmiany

```
git pull
```

Problem 4: Konflikty podczas merge

Rozwiązanie:

```
# 1. Git pokaże, które pliki mają konflikt
git status

# 2. Otwórz plik, znajdź sekcję:
# <<<<<<< HEAD
# twój kod
# =======
# kod z innej gałęzi
# >>>>>>> nazwa-galezi

# 3. Ręcznie zdecyduj, którą wersję zachować

# 4. Usuń znaczniki konfliktów (<<<<, ====, >>>>)

# 5. Dodaj rozwiązany plik
git add nazwa-pliku.txt

# 6. Zakończ merge
git commit -m "Rozwiąż konflikty"
```

Problem 5: “fatal: remote origin already exists”

Rozwiązanie: Usuń stare zdalne repozytorium i dodaj nowe

```
git remote remove origin
git remote add origin https://github.com/username/repo.git
```

Problem 6: Przypadkowo scommitowałem wrażliwe dane (hasła, klucze API)

UWAGA: Usunięcie pliku w nowym commicie **NIE usuwa go z historii**!

Rozwiązanie:

1. **Natychmiast zmień skompromitowane hasło/klucz!**
2. Użyj `git filter-branch` lub `BFG Repo-Cleaner` do wyczyszczenia historii
3. Na przyszłość: używaj `.gitignore` i `.env` plików

Przydatne Komendy - Ściąga

```
# === PODSTAWY ===
git init                    # Inicjuj repozytorium
git clone <url>             # Sklonuj repozytorium
git status                  # Sprawdź status
git add .                   # Dodaj wszystkie zmiany
git commit -m "wiadomość"   # Commituj zmiany
git push                    # Wyślij na serwer
git pull                    # Pobierz ze serwera

# === GAŁĘZIE ===
git branch                  # Lista gałęzi
git branch <nazwa>          # Utwórz gałąź
git checkout <nazwa>        # Przełącz gałąź
git checkout -b <nazwa>     # Utwórz i przełącz
git merge <nazwa>           # Połącz gałęzie
git branch -d <nazwa>       # Usuń gałąź

# === HISTORIA ===
git log                     # Historia commitów
git log --oneline           # Skrócona historia
git log --graph --all       # Graficzna historia
git diff                    # Pokaż różnice

# === COFANIE ===
git checkout -- <plik>      # Cofnij zmiany w pliku
git reset HEAD <plik>       # Usuń ze stage
git reset --soft HEAD~1     # Cofnij commit (zostaw zmiany)
git reset --hard HEAD~1     # Cofnij commit (usuń zmiany)

# === ZDALNE REPOZYTORIA ===
git remote -v               # Lista zdalnych repozytoriów
git remote add origin <url> # Dodaj zdalne repo
git push -u origin main     # Wyślij i ustaw upstream
git fetch                   # Pobierz zmiany
git pull origin main        # Pobierz i połącz

# === INNE ===
git stash                   # Schowaj zmiany tymczasowo
git stash pop               # Przywróć schowane zmiany
git rm <plik>               # Usuń plik
git mv <stary> <nowy>       # Zmień nazwę
```

Dodatkowe Zasoby do Nauki

Dokumentacja

- Oficjalna dokumentacja Git: https://git-scm.com/doc
- GitHub Docs: https://docs.github.com/
- GitHub Skills (interaktywne kursy): https://skills.github.com/

Interaktywne Tutoriale

- Learn Git Branching: https://learngitbranching.js.org/?locale=pl
- Git-it (desktop app): https://github.com/jlord/git-it-electron

Kursy Video (YouTube)

- Traversy Media - Git & GitHub Crash Course
- freeCodeCamp - Git and GitHub for Beginners

Cheat Sheets

- GitHub Git Cheat Sheet: https://education.github.com/git-cheat-sheet-education.pdf
- Atlassian Git Cheat Sheet: https://www.atlassian.com/git/tutorials/atlassian-git-cheatsheet

---

Checklist dla Studentów

Po przejściu przez ten przewodnik, powinieneś umieć:

- ☐ Utworzyć konto na GitHub
- ☐ Zainstalować Git na Windows
- ☐ Skonfigurować Git (user.name, user.email)
- ☐ Utworzyć repozytorium na GitHub
- ☐ Sklonować repozytorium na swój komputer
- ☐ Wykonać podstawowe operacje: add, commit, push, pull
- ☐ Pracować z gałęziami (branch, checkout, merge)
- ☐ Rozwiązywać konflikty
- ☐ Używać .gitignore
- ☐ Autentykować się na GitHub (PAT lub GitHub CLI)

Wskazówki Na Zakończenie

1. **Commituj często** - małe, logiczne zmiany są lepsze niż jeden wielki commit
2. **Pisz dobre wiadomości commitów** - przyszły Ty będzie wdzięczny
3. **Zawsze `git pull` przed `git push`** - unikniesz konfliktów
4. **Twórz gałęzie dla nowych funkcji** - nie pracuj bezpośrednio na `main`
5. **Używaj `.gitignore`** - nie commituj śmieci
6. **Backupuj ważne rzeczy** - Git nie jest backupem (ale GitHub może być)
7. **Nie panikuj** - prawie wszystko da się cofnąć w Git
8. **Praktykuj** - Git robi się intuicyjny z czasem
