Materiały do nauki Git i GitHub (Notebook)

Wprowadzenie do Git i GitHub

Git to rozproszony system kontroli wersji, który pozwala na śledzenie zmian w kodzie, współpracę z innymi deweloperami oraz przechowywanie historii projektu. GitHub to platforma oparta na Git, która umożliwia hostowanie repozytoriów w chmurze oraz współpracę nad projektami.

Notebook ten pomoże Ci zrozumieć podstawowe koncepcje i nauczy Cię praktycznego korzystania z Git i GitHub. Każda sekcja zawiera przykłady i zadania do wykonania.

---

1. Instalacja Git

1.1 Instalacja Git

1. **Windows**: Pobierz instalator z git-scm.com i zainstaluj Git. Podczas instalacji możesz pozostawić domyślne ustawienia.
2. **macOS**: Użyj Homebrew:

   ```
   brew install git
   ```

3. **Linux**: Użyj menedżera pakietów, np. dla Ubuntu:

   ```
   sudo apt-get install git
   ```

1.2 Konfiguracja Git

Po zainstalowaniu Gita skonfiguruj swoje dane użytkownika, aby żadania commitów miały odpowiedni podpis:

```
git config --global user.name "Twoje Imię i Nazwisko"
git config --global user.email "twojemail@example.com"
```

Sprawdź konfigurację poleceniem:

```
git config --list
```

---

2. Podstawowe Pojęcia w Git

- **Repozytorium (Repository)**: Miejsce przechowywania kodu źródłowego oraz historii zmian.
- **Commit**: Zapis zmian w repozytorium.
- **Gałąź (Branch)**: Oddzielna linia rozwoju kodu. Domyślna gałąź to **main** lub **master**.
- **Merge**: Scalanie zmian z jednej gałęzi do drugiej.
- **Fork**: Rozwidlenie repozytorium - kopia na Twoim koncie GitHub.
- **Clone**: Kopia repozytorium na lokalnym komputerze.
- **Remote**: Zdalne repozytorium (np. na GitHub).

---

3. Podstawowe Operacje w Git

3.1 Tworzenie Repozytorium

- **Utworzenie nowego repozytorium w lokalnym folderze:**

  ```
  git init
  ```

- **Klonowanie istniejącego repozytorium:**

  ```
  git clone https://github.com/uzytkownik/nazwa-repozytoriu
  m.git
  ```

3.2 Dodawanie Plików i Tworzenie Commitów

- **Dodanie plików do staging area:**

```
git add nazwa_pliku
```

- **Commitowanie zmian:**

```
git commit -m "Opis zmian"
```

3.3 Sprawdzanie Statusu Repozytorium

- **Sprawdzenie stanu repozytorium (plików zmodyfikowanych, dodanych, etc.):**

```
git status
```

3.4 Historia Commitów

- **Wyświetlenie historii commitów:**

```
git log
```

---

4. Gałęzie (Branches)

4.1 Tworzenie i Przełączanie Gałęzi

- **Tworzenie nowej gałęzi:**

```
git branch nazwa_galezi
```

- **Przełączanie się na inną gałęź:**

```
git checkout nazwa_galezi
```

- **Tworzenie i przełączanie w jednym kroku:**

```
git checkout -b nazwa_galezi
```

4.2 Scalanie Gałęzi

- **Scalanie gałęzi do `main`:**

```
git checkout main
git merge nazwa_galezi
```

---

5. Praca z GitHub

5.1 Tworzenie Repozytorium na GitHub

1. Zaloguj się na GitHub.
2. Kliknij przycisk **New** lub **Create a new repository**.
3. Wprowadź nazwę repozytorium i opcjonalny opis.
4. Wybierz, czy repozytorium ma być publiczne czy prywatne.
5. Kliknij **Create repository**.

5.2 Wysyłanie Lokalnego Repozytorium na GitHub

- **Dodanie zdalnego repozytorium:**

```
git remote add origin https://github.com/uzytkownik/nazwa-repozytorium.git
```

- **Wysyłanie zmian do zdalnego repozytorium:**

```
git push -u origin main
```

5.3 Klonowanie Repozytorium

- **Klonowanie istniejecego repozytorium z GitHub:**

```
git clone https://github.com/uzytkownik/nazwa-repozytorium.git
```

---

6. Współpraca z Inniami

6.1 Forkowanie i Pull Requests

- **Forkowanie repozytorium:**
  - Przejdź do repozytorium na GitHub i kliknij **Fork**.
- **Tworzenie Pull Request:**
  - Po wprowadzeniu zmian w swoim forku kliknij **New pull request** w oryginalnym repozytorium, aby zaproponować zmiany.

6.2 Rozwiązywanie Konfliktów

- **Konflikty mogą wystąpić podczas scalania zmian.**
- Aby je rozwiązać:
  1. Git wskaże pliki z konfliktami.
  2. Otwórz plik i zdecyduj, które zmiany zachować.
  3. Po rozwiązaniu konfliktów dodaj pliki do staging area:

     ```
     git add nazwa_pliku
     ```

  4. Kontynuuj scalanie:

     ```
     git commit
     ```

6.3 Informacje o Pull Request (PR) i Code Review

- **Pull Request (PR)**: Pull Request to propozycja wprowadzenia zmian z jednej gałęzi do innej (np. z Twojego forka do głównego repozytorium instruktora). PR jest kluczowym elementem współpracy w zespołach programistycznych, ponieważ umożliwia innym programistom przeglądanie i komentowanie Twoich zmian przed ich scaleniem do głównej gałęzi.
- **Code Review**: To proces przeglądania kodu przez innych członków zespołu, aby upewnić się, że wprowadzone zmiany są zgodne z najlepszymi praktykami, mają odpowiednią jakość i nie wprowadzają błędów. Code Review

to ważna część pracy zespołowej, która pomaga utrzymać wysoką jakość kodu.

---

7. Synchronizacja z Repozytorium Instruktora (Upstream)

W trakcie kursu będziemy korzystać z repozytorium instruktora, które będzie zawierać wszystkie materiały, ćwiczenia oraz aktualizacje. Każdy uczestnik będzie pracował na swoim forku, a poniżej opisany jest cały proces, jak będziemy działać.

7.1 Dodanie Repozytorium Instruktora jako `upstream`

Po utworzeniu forka repozytorium instruktora, należy dodać repozytorium instruktora jako `upstream`, aby móc pobierać najnowsze zmiany:

```
git remote add upstream https://github.com/instruktor/nazwa-repozytorium.git
```

7.2 Pobieranie Aktualizacji z `upstream`

Instruktor będzie regularnie aktualizował repozytorium, dodając nowe materiały i ćwiczenia. Aby mieć pewność, że pracujesz na najnowszej wersji, musisz regularnie pobierać zmiany z repozytorium instruktora.

- **Pobierz zmiany z repozytorium instruktora:**

  ```
  git fetch upstream
  ```

- **Scal zmiany z gałęzi `main` instruktora z Twoją lokalną gałęzią `main`:**

  ```
  git checkout main
  git merge upstream/main
  ```

7.3 Rozwiązywanie Konfliktów Podczas Scalania

Podczas scalania mogą wystąpić konflikty, zwłaszcza jeśli wprowadzałeś zmiany w tych samych plikach, które zostały zaktualizowane przez instruktora. W takiej sytuacji Git wskaże pliki z konfliktami, a Ty będziesz musiał je ręcznie rozwiązać.

- **Kroki rozwiązywania konfliktów:**
  1. Otwórz pliki z konfliktami w edytorze tekstu lub IDE. Konflikty będą oznaczone znacznikami `<<<<<<<`, `=======` i `>>>>>>>`.
  2. Przeanalizuj zmiany i zdecyduj, które z nich zachować (możesz połączyć obie wersje, jeśli to konieczne).
  3. Usuń znaczniki konfliktu i zapisz plik.
  4. Dodaj plik do staging area:

     ```
     git add nazwa_pliku
     ```

  5. Kontynuuj scalenie, wykonując commit:

     ```
     git commit
     ```

7.4 Wysyłanie Zmian na Swoje Repozytorium (Fork)

Po scaleniu zmian z `upstream` i rozwiązaniu ewentualnych konfliktów, możesz wysłać zaktualizowaną wersję na swoje zdalne repozytorium na GitHubie:

```
git push origin main
```

7.5 Praca nad Ćwiczeniami na Oddzielnych Gałęziach

Aby uniknąć konfliktów i ułatwić sobie pracę, zaleca się tworzenie oddzielnych gałęzi do pracy nad ćwiczeniami lub nowymi funkcjonalnościami.

- **Tworzenie nowej gałęzi do pracy nad ćwiczeniem:**

  ```
  git checkout -b cwiczenie-1
  ```

- Po zakończeniu pracy nad ćwiczeniem możesz scalić zmiany z gałęzią `main`:

```
git checkout main
git merge cwiczenie-1
```

- Następnie wypchnij zmiany na GitHub:

```
git push origin main
```

7.6 Podsumowanie Procesu

1. **Forkowanie repozytorium instruktora** na swoje konto GitHub.
2. **Klonowanie forka** na lokalny komputer.
3. **Dodanie repozytorium instruktora jako** ********** `upstream`, aby móc pobierać zmiany.
4. *Regularne pobieranie zmian z ** `upstream` i scalanie ich z lokalną gałęzią `main`.
5. **Rozwiązywanie ewentualnych konfliktów** podczas scalania.
6. **Praca nad ćwiczeniami na oddzielnych gałęziach** i scalanie ich z `main` po zakończeniu.
7. **Wysyłanie zaktualizowanej wersji** na swoje zdalne repozytorium na GitHub.

Dzięki temu procesowi będziesz na bieżąco z materiałami kursu, jednocześnie zachowując swoje zmiany i pracując w sposób uporządkowany.

---

8. Rozwiązywanie Konfliktów w Praktyce

Przykład Konfliktu

Załóżmy, że plik `przyklad.txt` został zmodyfikowany zarówno przez Ciebie, jak i przez instruktora.

Konflikt w Pliku `przyklad.txt`:

```
To jest wspólny tekst.
```

```
<<<<<<< HEAD
Twoja wersja zdania.
=======
Wersja instruktora.
>>>>>>> upstream/main
```

Rozwiązanie Konfliktu:

1. Wybierz, którą wersję zachować lub połącz obie.
2. Usuń znaczniki (`<<<<<<<`, `=======`, `>>>>>>>`).
3. Zapisz plik i dodaj go do staging area:

   ```
   git add przyklad.txt
   ```

4. Kontynuuj scalenie:

   ```
   git commit
   ```

---

9. Integracja Git/GitHub z VCS (Visual Studio Code)

9.1 Instalacja Visual Studio Code

Visual Studio Code (VS Code) to popularne, darmowe IDE, które wspiera integrację z Git i GitHub. Możesz pobrać VS Code ze strony https://code.visualstudio.com/.

9.2 Konfiguracja Git w VS Code

Po zainstalowaniu VS Code, możesz korzystać z wbudowanej integracji z Git:

1. **Otwórz folder projektu** w VS Code.
2. **Panel Źródła (Source Control)**: Kliknij ikonę "Source Control" po lewej stronie, aby zobaczyć status repozytorium.
3. **Commity i Gałęzie**: Możesz dodawać zmiany do staging area, tworzyć commity, przełączać się między gałęziami, wszystko za pomocą interfejsu VS

Code.

4. **Rozwiązywanie konfliktów**: VS Code wizualnie pomaga rozwiązywać konflikty, pokazując różnice między wersjami plików.

9.3 Wtyczki do VS Code

- **GitLens**: To popularna wtyczka do VS Code, która wzbogaca doświadczenie pracy z Git. Pozwala na łatwe śledzenie zmian w kodzie, podgląd historii commitów i współpracę z zespołem.

---

10. Dodatkowe Zasoby

- **Dokumentacja Git**: https://git-scm.com/doc
- **Samouczek GitHub**: https://guides.github.com/activities/hello-world/
- **Interaktywny Kurs Git**: https://learngitbranching.js.org/?locale=pl_PL

Podsumowanie

Git i GitHub są kluczowymi narzędziami do współpracy nad projektami programistycznymi. W tym notebooku nauczyłeś się podstawowych operacji w Git, jak korzystać z repozytoriów zdalnych, jak rozwiązywać konflikty oraz jak pracować nad projektami zespołowymi za pomocą GitHub. Praktyka jest kluczem do opanowania tych narzędzi, więc zachęcam do regularnego korzystania z Git w swoich projektach.