Rok akademicki 2024/2025
- Informatyka w Biznesie –

Technologie Webowe w Biznesie
Laboratorium 23/24
Git – system kontroli wersji

Proszę zapoznać się z prezentacją i następnie wykonać następujące zadania:
Przed wykonywaniem zadania, proszę się upewnić, czy mają Państwo zainstalowanego gita, aby to sprawdzić w wierszu poleceń (terminal) należy wpisać polecenie: **git**.

Proszę przygotować sprawozdanie z wykonanych zadań, dołączając zdjęcia dokumentujące realizację poszczególnych etapów i zamieścić je na platformie Teams.

Zadanie 1 (2 pkt)
Podstawowe operacje

1. **Utwórz nowe repozytorium:**
   - Otwórz terminal.
   - Przejdź do folderu, w którym chcesz utworzyć repozytorium.
   - Wykonaj poniższe polecenie: **git init**

2. **Dodaj plik do repozytorium:**
   - Utwórz nowy plik (np. **index.html**) w folderze repozytorium. (touch index.html)
   - Dodaj plik do śledzenia przez Git: **git add nazwa_pliku**
   - Uruchom polecenie **git status** i zobacz, czy Twój plik jest już śledzony

3. **Zrób kilka commitów, opisując zmiany:**
   - Zatwierdź dodany plik jako pierwszy commit: **git commit -m "Dodano nowy plik index.html"**
   - Wprowadź kolejne zmiany i zatwierdzaj je jako kolejne commity.

4. **Stwórz nowego brancha i wprowadź zmiany tylko na tym branchu:**
   - Stwórz nowego brancha (np. **nowy-branch**): **git branch nowy-branch**
   - Przełącz się na nowo utworzony branch: **git checkout nowy-branch**
   - Wprowadź zmiany w pliku (np. dodaj nowe linie kodu).
   - Dodaj i zatwierdź zmiany na nowym branchu: **git add nazwa_pliku** oraz **git commit -m "Dodano nowe funkcje na nowym branchu"**

   Teraz macie Państwo repozytorium z historią commitów, nowym plikiem i dodatkowym branchem z wprowadzonymi zmianami.

   Proszę sprawdzić historię commitów (**git log**). Następnie jako potwierdzenie zrealizowania zadania, należy zrobić screenshota (zrzut ekranu) historii commitów.

Zadanie 2 (3 pkt)
Rozwiązywanie konfliktów

1. **Zmodyfikuj ten sam plik na dwóch różnych branchach:**
   - Stwórz nowego brancha (np. **branch-A**): **git branch branch-A git checkout branch-A**
   - Wprowadź zmiany w pliku (np. dodaj nową linijkę tekstu).
   - Dodaj i zatwierdź zmiany: **git add nazwa_pliku git commit -m "Zmiany na branchu-A"**
   - Przełącz się na główny branch (np. **main**): **git checkout main**
   - Stwórz drugiego brancha (np. **branch-B**): **git branch branch-B git checkout branch-B**
   - Wprowadź inne zmiany w tym samym pliku co na **branch-A**.
   - Dodaj i zatwierdź zmiany: **git add nazwa_pliku git commit -m "Zmiany na branchu-B"**

2. **Spróbuj zmergować te branchy, aby spowodować konflikt:**
   - Przełącz się z powrotem na **main**: **git checkout main**
   - Zmerguj **branch-A** do **main**: **git merge branch-A**
   - Następnie, zmerguj **branch-B** do **main**: **git merge branch-B**

   Teraz powinno pojawić się ostrzeżenie o konflikcie.

3. **Rozwiąż konflikt i dokończ proces mergowania:**
   - Otwórz konfliktowy plik w edytorze tekstowym.
   - Znajdź oznaczenia konfliktu (<<<<<<, ====, >>>>>>).
   - Zdecyduj, które zmiany zostaną zachowane.
   - Usuń oznaczenia konfliktu i zapisz plik.
   - Dodaj zmodyfikowany plik: **git add nazwa_pliku**
   - Zakończ proces mergowania: **git merge --continue**

   Teraz konflikt powinien być rozwiązany, a zmiany z obu branchy będą zmergowane do **main**.

   Proszę sprawdzić historię commitów. Następnie jako potwierdzenie zrealizowania zadania, należy zrobić screenshota (zrzut ekranu) historii commitów.

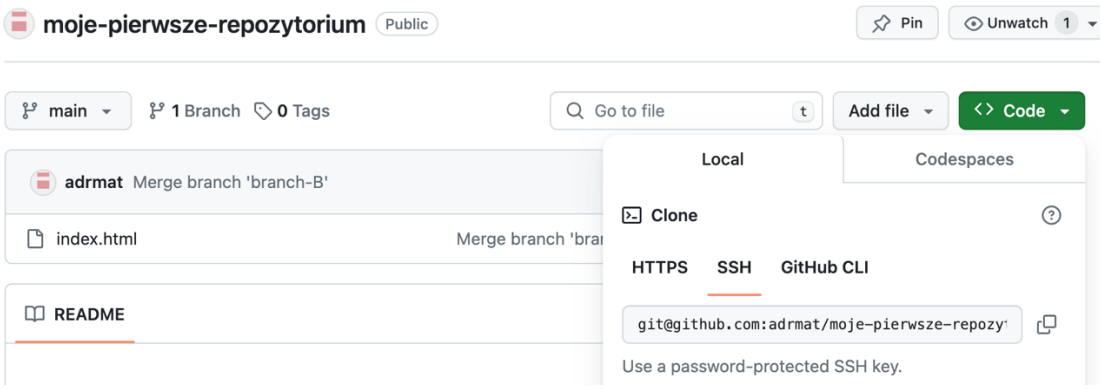
Zadanie 3 (3 pkt)
Zdalne repozytorium

1. **Stwórz konto na GitHub:**
   - Jeśli nie masz jeszcze konta, załóż je na stronie GitHub.

2. **Utwórz nowe zdalne repozytorium:**
   - Zaloguj się na swoje konto GitHub.
   - Na stronie głównej kliknij przycisk "New" w górnym prawym rogu, aby utworzyć nowe repozytorium.
   - Nadaj mu nazwę (np. **moje-pierwsze-repozytorium**), dodaj krótki opis i wybierz opcję "Initialize this repository with a README".

- Kliknij przycisk "Create repository".
- Repozytorium powinno być **publiczne**

3. **Dodaj zdalne repozytorium do swojego lokalnego projektu:**
   - Przejdź do folderu swojego istniejącego lokalnego repozytorium.
   - Dodaj zdalne repozytorium (zastępując **<link_do_twojego_repozytorium>** odpowiednim adresem): **git remote add origin <link_do_twojego_repozytorium>**
     - Code – SSH
     - Należy wygenerować klucz SSH: https://docs.github.com/en/authentication/connecting-to-github-with-ssh/generating-a-new-ssh-key-and-adding-it-to-the-ssh-agent



   - Sprawdź, czy zdalne repozytorium zostało dodane poprawnie: **git remote -v**

4. **Wypchnij swoje zmiany na zdalne repozytorium:**
   - Wprowadź jakieś zmiany w swoim lokalnym repozytorium (np. dodaj nowy plik).
   - Dodaj i zatwierdź zmiany: **git add . git commit -m "Dodano nowy plik"**
   - Wypchnij zmiany na zdalne repozytorium: **git push -u origin main**

5. **Sprawdź zdalne repozytorium na GitHub:**
   - Odwiedź swoje zdalne repozytorium na GitHub i upewnij się, że twoje lokalne zmiany zostały wgrane.

Następnie jako potwierdzenie zrealizowania zadania, należy do Teamsa wstawić link do Państwa repozytorium.

Zadanie 4 (2 pkt)
Historia i logi

1. **Zrób kilka commitów:**
   - Wprowadź kilka zmian w plikach w swoim repozytorium.
   - Dodaj i zatwierdź każdą zmianę jako nowy commit.

2. **Sprawdź historię commitów:**
- Użyj polecenia **git log**, aby wyświetlić historię commitów.
- Zobacz informacje takie jak SHA commita, autor, data i opis zmian.

3. **Filtruj historię commitów:**
   - Użyj różnych opcji polecenia **git log**, aby przefiltrować historię commitów.
     - Na przykład, użyj **git log --author=<autor>** aby zobaczyć commit'y tylko tego autora.
     - Użyj **git log --grep=<słowo_kluczowe>** aby znaleźć commit'y zawierające dane słowo kluczowe.

4. **Przywróć poprzedni stan:**
   - Użyj **git checkout** lub **git revert**, aby przywrócić jeden z poprzednich commitów.
   - Spróbuj przywrócić poprzedni stan pliku.

5. **Utwórz tag dla konkretnego commita:**
   - Stwórz tag dla jednego z commitów w historii.
   - Użyj polecenia **git tag**.

6. **Przeglądaj historię w interaktywny sposób:**
   - Użyj polecenia **git log --oneline --graph --all --decorate**, aby uzyskać bardziej czytelną i interaktywną historię commitów.

Proszę sprawdzić historię commitów (**git log**). Następnie jako potwierdzenie zrealizowania zadania, należy zrobić screenshota (zrzut ekranu) historii commitów.

Zadanie 5 (2 pkt)
Tagi i wersjonowanie

1. **Utwórz repozytorium:**
   - Utwórz nowe repozytorium na swoim koncie GitHub lub lokalnie na komputerze.

2. **Dodaj pliki do repozytorium:**
   - Dodaj kilka plików do repozytorium (np. pliki kodu źródłowego, dokumentację).

3. **Utwórz tag dla pierwszej wersji:**
   - Utwórz tag oznaczający pierwszą wersję twojego projektu. **git tag -a v1.0 -m "Pierwsza wersja"**

4. **Zaktualizuj kod i utwórz nową wersję:**
   - Wprowadź kilka zmian w kodzie lub dodaj nowe funkcje.
   - Zaktualizuj wersję w plikach projektu.
   - Utwórz nowy tag dla tej wersji: **git tag -a v1.1 -m "Druga wersja z nowymi funkcjami".**

5. **Przeglądaj dostępne tagi:**
   - Użyj polecenia **git tag**, aby zobaczyć dostępne tagi.
   - Możesz również użyć opcji **-l** do filtrowania tagów.

6. **Przełącz się na konkretną wersję za pomocą taga:**
   - Użyj polecenia **git checkout** z nazwą taga, aby przełączyć się na konkretną wersję: **git checkout v1.0**

7. **Zaktualizuj tag (opcjonalnie):**
   - Jeśli masz potrzebę zaktualizować tag (np. poprawić opis), możesz to zrobić za pomocą poniższego polecenia: **git tag -a -f v1.0 -m "Poprawiona pierwsza wersja"**

Proszę sprawdzić historię commitów (**git log**). Następnie jako potwierdzenie zrealizowania zadania, należy zrobić screenshota (zrzut ekranu) historii commitów.

Zadanie 6 (5 pkt)
Na podstawie prezentacji i wykonanych zadań, proszę odpowiedzieć na następujące pytania:

1. Jakie są główne zalety korzystania z systemu kontroli wersji Git w porównaniu do innych systemów?
2. W jaki sposób tworzy się nowe repozytorium w Git?
3. Co to jest **commit** w kontekście systemu Git?
4. Jakie są główne różnice między **branch** a **fork** w Git?
5. W jaki sposób można zintegrować istniejące repozytorium Git z serwisem zdalnym, takim jak GitHub?
6. Co to jest konflikt scalania (**merge conflict**) w Git i jak go rozwiązać?
7. Jakie są główne różnice między poleceniami "**git pull**" a "**git fetch**"?
8. Jak używać gałęzi (**branch**) w Git do równoczesnej pracy nad różnymi funkcjonalnościami projektu?
9. Jak przywrócić poprzednią wersję pliku (**commit**) w Git, jeśli coś poszło nie tak?