Ćwiczenie 6

System kontroli wersji GIT

6.1. Rozpoczęcie pracy

1. Uruchom konsolę i sprawdź poleceniami:
⇒`git config --global user.name`
⇒`git config --global user.email`
czy została wprowadzona globalna nazwa użytkownika i jego adres email. Jeżeli wyświetlany jest pusty tekst, to nazwy globalne można ustawić poleceniami:
⇒`git config --global user.name` *`student`*
⇒`git config --global user.email` *`d3stud@kia.prz.edu.pl`*
Wskazówka! Ustawienie danych użytkownika dla jednego repozytorium jest możliwe bez parametru `--global`, wówczas zmiany zapiszą się w konfiguracji bieżącego repozytorium.

2. Zidentyfikuj numer stanowiska na którym rozpoczynasz pracę. Jest on wyświetlany podczas logowania albo można go odczytać wydając polecenie konsoli:

⇒`echo $HOST`

Wówczas wyświetli się nazwa stanowiska, np.:

⇐`D3MAC-12`

wówczas **numerem stanowiska** jest **12**. Posłuży on do skonstruowania nazw repozytoriów wykorzystywanych podczas zajęć.

3. Ustal nazwy repozytoriów dla aktualnego stanowiska.
Dla studentów studiów **dziennych** repozytoria prywatne mają adresy:

`git@10.10.2.18:NDP_L`*`xx`*`_S`*`yy`*`.git`

gdzie *xx* to **numer grupy laboratoryjnej**, a *yy* to **numer stanowiska**.
Repozytoria wspólne dla grupy laboratoryjnej mają adresy:

`git@10.10.2.18:NDP_L`*`xx`*`.git`

gdzie *xx* to **numer grupy laboratoryjnej**.
Dla studentów studiów **niestacjonarnych** repozytoria prywatne mają adresy:

`git@10.10.2.18:NDP_Z`*`xx`*`_S`*`yy`*`.git`

gdzie *xx* to **numer grupy laboratoryjnej**, a *yy* to **numer stanowiska**.
Repozytoria wspólne dla całej grupy laboratoryjnej mają adresy:

`git@10.10.2.18:NDP_Z`*`xx`*`.git`

gdzie *xx* to **numer grupy laboratoryjnej**.

4. Zanotuj sobie w widocznym miejscu (np. kartka papieru, albo plik na komputerze) z ustalonymi nazwami repozytoriów. Dla przykładu dla studentów stacjonarnych grupy L02 i dla osoby pracującej na stanowisku 12 będą to:

prywatne (dalej *REPO_PRYWATNE*): `git@10.10.2.18:NDP_L02_S12.git`

wspólne (dalej *REPO_WSPÓLNE*): `git@10.10.2.18:NDP_L02.git`

Wskazówka do realizacji zajęć poza laboratorium

Osoby chcące zrealizować ćwiczenie bez dostępu do serwerów zlokalizowanych w sali laboratoryjnej (a nawet bez dostępu do internetu) mogą to zrobić tworząc samemu repozytorium w wybranym katalogu np. `~/cw6/repo` przez polecenia GIT-Bash:

```
mkdir -p ~/cw6/repo
cd ~/cw6/repo
git init --bare .
```

(kropka na końcu ostatniego polecenia). Wówczas *REPO_PRYWATNE* przyjmie postać:

```
file://$HOME/cw6/repo
```

a komunikacja będzie odbywała się bez wykorzystania sieci internet.

6.2. Inicjowanie repozytorium

5. Uruchomić konsolę i utworzyć katalogi robocze na prywatne repozytorium przykładowo `~/cw6/pryw`.

6. Przejść poleceniem `cd` do katalogu `pryw`.

7. Zainicjować repozytorium poleceniem:

⇒`git init .`

(kropka po `init` wskazuje katalog bieżący).

8. Skopiować do katalogu `pryw` początkowy plik raport.tex ze źródłem w LaTeXu pobrany ze strony z materiałami.

9. Poleceniem:

⇒`git add raport.tex`

dodać plik `raport.tex` do repozytorium.

10. Poleceniem

⇒`git commit`

zatwierdź aktualną wersję. Wówczas uruchomi się edytor VIM w którym trzeba wprowadzić opis zmian. Pusty plik (lub zakomentowany znakami `#`) powoduje przerwanie zatwierdzania. **Uwaga!** aby rozpocząć wpisywanie tekstu w VIM należy nacisnąć klawisz [i] (małe `i`). Wyjście z trybu edycji klawiszem [esc], a zapisanie i zamknięcie dokumentu przez [:wq].

6.3. Praca z repozytorium

11. Wywołać polecenie:

⇒`git status`

aby uzyskać informacje o aktualnej gałęzi repozytorium. Wyjście powinno być podobne do

poniższego:
⇐`On branch main`[1]
⇐`nothing to commit, working tree clean`
Oznacza to, że bieżąca gałąź to `main`.

12. W linii 14 dokumentu `raport.tex` zmień autorów na własne dane, a następnie w linii 24 wstaw nową sekcję zatytułowaną *Nowość* i dodaj dowolne zdanie, albo akapit.

13. Podobnie jak w zadaniu 9 poleceniem
⇒`git add raport.tex`
dodaj zmiany w pliku `raport.txt` jako przygotowane do zatwierdzenia.

14. Podobnie jak w zadaniu 10 poleceniem
⇒`git commit`
zatwierdź zmiany w repozytorium.

15. Poleceniem:
⇒`git log .`
wyświetl opis zmian wykonywanych w repozytorium. Wynik powinien być zbliżony do przedstawionego na rys. 6.1 (numery zatwierdzeń są generowane automatycznie i powinny być unikatowe na różnych stanowiskach).

```
jan@JanM pryw % git log .
commit 2a5592104e500ab3b680bf87cf6f4eeec2f87b16 (HEAD -> main)
Author: Jan <js@prz-rzeszow.pl>
Date:   Wed Dec 7 11:19:04 2022 +0100

    Drugie zmiany w pliku

commit 97402cc6e7abf622f5df6fd36d45fa1a25c975d2
Author: Jan <js@prz-rzeszow.pl>
Date:   Wed Dec 7 10:55:14 2022 +0100

    Zmiany początkowe
```

Rysunek 6.1. Stan repozytorium po dwóch zatwierdzeniach

16. Poleceniem
⇒`git tag MojaEtykieta`
nadaj etykietę tej wersji dokumentu. Wówczas wywołanie `git log .` wskaże etykietę (tag) na ostatnim zatwierdzeniu. Wówczas zostanie zdefiniowana „przyjazna nazwa" do tego zatwierdzenia, aby nie było konieczności pamiętania (zbyt długiej) sumy SHA1 do oznaczania zatwierdzeń.

17. Zmodyfikuj dokument `raport.tex`, i jednocześnie zatwierdź zmiany dokumentu w repozytorium poleceniem:
⇒`git commit -a`

6.4. Praca lokalna

18. Utwórz katalog `~/cw6/lok` i przejdź do niego poleceniem `cd`.

19. Wykonaj operację klonowania repozytorium lokalnego z katalogu `~/cw6/pryw` wydając polecenie:
⇒`git clone file://$HOME/cw6/pryw .`

[1] Dla poprzednich wersji GITa, bądź konfiguracji kompatybilnej wstecz gałąź może nazywać się również `master`.

Kropka na końcu polecenia wskazuje bieżący katalog do którego ma być sklonowane repozytorium, a protokół `file` wskazuje na repozytorium źródłowe zlokalizowane na bieżącym komputerze.

20. Utwórz katalog o nazwie `kod` i umieść w nim plik `start.c` zawierający program „Hello world" napisany w języku C.

21. Poleceniem
⇒`git add kod/start.c`
dodaj utworzony w poprzednim punkcie plik `start.c` do zmian w repozytorium, a następnie **zatwierdź zmiany**.

6.5. Praca zdalna

22. Poleceniem
⇒`git remote -v show`
wyświetl nazwy serwerów do synchronizacji kodu. Powinny pojawić się nazwy `origin` jak na rysunku 6.2.

```
jan@JanM lok % git remote -v show
origin  file:///Users/jan/cw6/pryw (fetch)
origin  file:///Users/jan/cw6/pryw (push)
```

Rysunek 6.2. Istniejące repozytoria zdalne

23. Dodaj zdalne repozytorium poleceniem:
⇒`git remote add serwer` *`REPO_PRYWATNE`*
gdzie *`REPO_PRYWATNE`* to nazwa prywatnego repozytorium ustalona w zadaniu 4.

24. Poleceniem:
⇒`git push --set-upstream serwer main`[2]
ustal domyślny cel wysyłania tej gałęzi do serwera. Wynik operacji powinien być podobny jak na rysunku 6.3

```
jan@JanM lok % git push --set-upstream serwer main
Enumerating objects: 13, done.
Counting objects: 100% (13/13), done.
Delta compression using up to 4 threads
Compressing objects: 100% (7/7), done.
Writing objects: 100% (13/13), 1.54 KiB | 1.54 MiB/s, done.
Total 13 (delta 2), reused 9 (delta 2), pack-reused 0
To msdn.prz-rzeszow.pl:REPO_PRYWATNE.git
 * [new branch]      main -> main
branch 'main' set up to track 'serwer/main'.
```

Rysunek 6.3. Przesłanie repozytorium do serwera zdalnego

25. Używając polecenia `cd` przejdź do katalogu `pryw` utworzonego w zadaniu 5.

26. Dodaj zdalne repozytorium podobnie jak w zadaniu 23 stosując polecenie:
⇒`git remote add origin` *`REPO_PRYWATNE`*
Zwróć uwagę na zmianę nazwy serwera z `serwer` na `origin` w stosunku do poprzedniej wersji polecenia.

[2]Zamiast `main` trzeba wpisać domyślną gałąź (wcześniej standardowo było to `master`)

27. Poleceniem:
⇒`git push --set-upstream origin main`
dodaj serwer zdalny do pracy z tym repozytorium, które zostanie odrzucone przez brak zsynchonizowanych zmian. Stąd wywołaj polecenie:
⇒`git fetch`
w celu pobrania zmian do lokalnej kopii, a następnie wykonaj polecenie:
⇒`git merge --ff-only origin/main`
aby uaktualnić (przewinąć) zmiany przesłane na serwer w tej kopii roboczej. Po tej operacji w katalogu `pryw` pojawi się katalog `kod` z plikiem `start.c` podobnie jak na rysunku 6.4.

```
[jan@JanM pryw % git fetch
 remote: Wymienianie obiektów: 5, gotowe.
 remote: Zliczanie obiektów: 100% (5/5), gotowe.
 remote: Kompresowanie obiektów: 100% (3/3), gotowe.
 remote: Razem 4 (delty 0), użyte ponownie 0 (delty 0), paczki użyte ponownie 0
 Unpacking objects: 100% (4/4), 374 bytes | 93.00 KiB/s, done.
 From msdn.prz-rzeszow.pl:REPO_PRYWATNE
  * [new branch]      main       -> origin/main
[jan@JanM pryw % git merge --ff-only origin/main
 Updating 7ad0776..fb9de3d
 Fast-forward
  kod/start.c | 8 ++++++++
  1 file changed, 8 insertions(+)
  create mode 100644 kod/start.c
```

Rysunek 6.4. Przewinięcie zmian w repozytorium lokalnym w celu synchronizacji

28. Zmodyfikuj plik `raport.tex` przez dopisanie przed linią `\end{document}` nowej sekcji i akapitu tekstu.

29. Dodaj zmiany do repozytorium
⇒`git commit -a` Aby przesłać je na serwer możemy wydać ponownie polecenie:
⇒`git push --set-upstream origin main`
parametr `--set-upstream` jest znowu potrzebny ponieważ polecenie w zadaniu 27 zakończyło się niepowodzeniem, więc konfiguracja została niezmieniona.

30. Przejdź do katalogu `lok`, zmodyfikuj plik `raport.tex` dodając kolejne linie tekstu przed `\end{document}`. Następnie zatwierdź zmiany i spróbuj wysłać je na serwer poleceniem:
⇒`git push`
Powinna zostać zwrócona odmowa podobna do tej z rysunku 6.5. Oznacza to, że najpierw trzeba

```
[jan@JanM pryw % git push
 To msdn.prz-rzeszow.pl:REPO_PRYWATNE.git
  ! [rejected]        main -> main (fetch first)
 error: failed to push some refs to 'msdn.prz-rzeszow.pl:REPO_PRYWATNE.git'
 hint: Updates were rejected because the remote contains work that you do
 hint: not have locally. This is usually caused by another repository pushing
 hint: to the same ref. You may want to first integrate the remote changes
 hint: (e.g., 'git pull ...') before pushing again.
 hint: See the 'Note about fast-forwards' in 'git push --help' for details.
```

Rysunek 6.5. Odmowa przesłania zmian na serwer

pobrać zmiany z serwera, scalić je a potem dopiero można przesłać zmodyfikowaną postać na serwer.

31. Wykonać pobranie zmian z serwera poleceniem
⇒`git pull`
W przypadku, gdy GIT nie został domyślnie skonfigurowany w jaki sposób radzić sobie ze zmianami wyświetla podobny komunikat jak na rysunku 6.6. Wówczas można jednorazowo

```
[jan@JanM pryw % git pull                                                    ]
remote: Wymienianie obiektów: 5, gotowe.
remote: Zliczanie obiektów: 100% (5/5), gotowe.
remote: Kompresowanie obiektów: 100% (3/3), gotowe.
remote: Razem 4 (delty 0), użyte ponownie 0 (delty 0), paczki użyte ponownie 0
Unpacking objects: 100% (4/4), 383 bytes | 191.00 KiB/s, done.
From msdn.prz-rzeszow.pl:REPO_PRYWATNE
   2a55921..62dec3e  main       -> origin/main
hint: You have divergent branches and need to specify how to reconcile them.
hint: You can do so by running one of the following commands sometime before
hint: your next pull:
hint:
hint:   git config pull.rebase false  # merge
hint:   git config pull.rebase true   # rebase
hint:   git config pull.ff only       # fast-forward only
hint:
hint: You can replace "git config" with "git config --global" to set a default
hint: preference for all repositories. You can also pass --rebase, --no-rebase,
hint: or --ff-only on the command line to override the configured default per
hint: invocation.
fatal: Need to specify how to reconcile divergent branches.
```

Rysunek 6.6. Pobranie zmian z serwera

skonfigurować to repozytorium, aby domyślnie scalało zmiany przez wywołanie polecenia:
⇒`git config pull.rebase false`
i ponowienie pobrania zmian poleceniem:
⇒`git pull`
wówczas pojawi się edytor pozwalający na nadanie opisu do zatwierdzenia scalania zmian (edytor VI jest wypełniony standardowym komunikatem o scalaniu) **albo** komunikat o konflikach scalania jak na rysunku 6.7. **W przypadku konfliktu scalania** należy go rozwiązać edytując

```
[jan@JanM lok % git config pull.rebase false
[jan@JanM lok % git pull
Auto-merging raport.tex
CONFLICT (content): Merge conflict in raport.tex
Automatic merge failed; fix conflicts and then commit the result.
```

Rysunek 6.7. Informacja o konflikcie scalania

problematyczny dokument i usuwając fragmenty składające się ze znaczników `<<<<<<<`, `=======` i `>>>>>>>` oraz pozostawiając właściwy tekst jaki ma znaleźć się w docelowym dokumencie. Zakończenie rozwiązywania problemów scalania w pliku oznacza się przez wywołanie polecenia:
⇒`git add` *`ścieżka_skonfliktowanego_pliku`*
Sprawdzenie, które pliki są jeszcze w stanie konfliktu odbywa się poleceniem:
⇒`git status`
Przykładowe statusy plików w repozytorium są widoczne na rysunku 6.8. Rozwiązanie wszystkich konfliktów powoduje, że można zatwierdzić scalenie poleceniem:
⇒`git commit`
wówczas uruchomi się edytor komunikatów podobnie jak w przypadku braku konfliktów.

W przypadku braku konfliktów należy uzupełnić opis scalenia (jak przy zwykłym zatwierdzeniu) i repozytorium zostaje scalone.

32. Scalone repozytorium należy przesłać na serwer poleceniem:
⇒`git push`

6.6. Praca zespołowa (zadania dla ambitnych)

33. W pracy zespołowej zazwyczaj rozpoczyna się od sklonowania repozytorium wspólnego dla całego zespołu. Dlatego na początku należy utworzyć katalog w którym praca zespołowa będzie

```
[jan@JanM lok % vi raport.tex
[jan@JanM lok % git status
On branch main
Your branch and 'serwer/main' have diverged,
and have 1 and 1 different commits each, respectively.
  (use "git pull" to merge the remote branch into yours)

You have unmerged paths.
  (fix conflicts and run "git commit")
  (use "git merge --abort" to abort the merge)

Unmerged paths:
  (use "git add <file>..." to mark resolution)
        both modified:   raport.tex

no changes added to commit (use "git add" and/or "git commit -a")
[jan@JanM lok % git add raport.tex
[jan@JanM lok % git status
On branch main
Your branch and 'serwer/main' have diverged,
and have 1 and 1 different commits each, respectively.
  (use "git pull" to merge the remote branch into yours)

All conflicts fixed but you are still merging.
  (use "git commit" to conclude merge)

Changes to be committed:
        modified:   raport.tex
```

Rysunek 6.8. Informacje o statusie plików

realizowana, np. `~/cw6/wsp`. Następnie należy przejść do niego poleceniem `cd`.

34. Sklonować repozytorium wspólne stosując polecenie:
⇒`git clone` ***`REPO_WSPÓLNE`*** `.`
pamiętając o kropce wskazującej bieżący katalog w którym zostanie zamieszczona kopia repozytorium oraz o zamianie ***`REPO_WSPÓLNE`*** na nazwę ustaloną w zadaniu 4. Pierwsza osoba, która to zrobi otrzyma komunikat:
⇐`Cloning into '.'...`
⇐`warning: You appear to have cloned an empty repository.`
Pozostałe osoby otrzymają listę zmian już przesłanych na serwer. Jeżeli pojawi się komunikat:
⇐`warning: remote HEAD refers to nonexistent ref, unable to checkout.`
To należy ręcznie ustawić gałąź stosując polecenie:
⇒`git checkout main`[3]
Celem repozytorium jest wspólne przygotowanie książki, gdzie autorami rozdziałów będą osoby biorące udział w zajęciach.

35. Jeżeli w katalogu `wsp` nie ma pliku `wspolna.tex`, to skopiuj go ze strony z materiałami do zajęć. A następnie zastąp parametr ***`indeks`*** w poleceniu `\include{`***`indeks`***`/tresc.tex}` swoim numerem indeksu[4]. Jeżeli plik `wspolna.tex` został pobrany z repozytorium, to dodaj kolejny wpis `\include{`***`indeks`***`/tresc.tex}` zamieniając ***`indeks`*** swoim numerem indeksu (albo nickiem).

36. Utwórz katalog z numerem indeksu[4] i utwórz w nim plik `tresc.txt`, który zostanie wypełniony zawartością:

[3]Zamiast `main` należy wpisać domyślną gałąź (wcześniej standardowo było to `master`)
[4]Gdy numer indeksu nie jest znany można użyć unikatowy nick

```
1 \chapter{Mój tytuł rozdziału}
2 \section{Nazwa punktu}
3
4 A tu można wstawić dowolny tekst czy rysunek włącznie z lorem ipsum.
```

37. Zatwierdź zmiany poleceniami:
⇒`git add wspolna.tex indeks/tresc.tex`
⇒`git commit -m "Opis zmian"`
i prześlij na serwer:
⇒`git push`

38. Utwórz nową gałąź o nazwie `vindeks`[4] poleceniem:
⇒`git checkout -b vindeks`
Zmodyfikuj plik `indeks/tresc.tex`, zatwierdź zmiany:
⇒`git commit -a` i wyślij na serwer poleceniem:
⇒`git push --set-upstream origin vindeks`
w celu dodania nowej gałęzi na serwerze.

39. Poleceniem:
⇒`git checkout main`[5]
przełącz się do gałęzi `main`[5]. Następnie pobierz zmiany z serwera, które mogły zostać w międzyczasie wprowadzone poleceniem:
⇒`git pull`
Scal zmiany z gałęzi `vindeks`[4] poleceniem:
⇒`git merge vindeks`[4]

40. Rozwiąż możliwe konflikty i wyślij zmiany na serwer poleceniem
⇒`git push`

41. Użyj polecenia
⇒`git blame wspolna.tex`
do uzyskania informacji, która linia pliku została zatwierdzona przez którego użytkownika.

42. Podobnie jak w zadaniu 41 sprawdź autora zmian linii dla pliku `vindeks`[4]`/tresc.tex`.

43. Przełącz się do gałęzi `vindeks`[4] jak w zadaniu 39 i scal ją z gałęzią `main`[5], a następnie wyślij na serwer.

44. Sprawdź dostępne gałęzie na serwerze poleceniem:
⇒`git ls-remote`

45. Przełącz się do gałęzi `main`[5] i usuń lokalnie gałąź `vindeks`[4]:
⇒`git checkout main`
⇒`git branch -d vindeks`[4]

6.7. Zakończenie ćwiczenia

46. Zarchiwizować we własnym zakresie otrzymane pliki, a następnie usunąć z dysku katalog `cw6` na komputerze będącym stanowiskiem laboratoryjnym w celu dostarczenia czystego środowiska pracy dla kolejnej grupy.

[5]Zamiast `main` trzeba będzie użyć `master`, gdy domyślna gałąź na serwerze to `master`.