Praca z systemem kontroli wersji (VCS, ang. *Version Control System*): Github.

Nasze repozyturium: https://github.com/NidhoggProject/UEC2_Nidhogg

Czego będziemy potrzebować? GitBash: https://gitforwindows.org/

1. Jak sklonować repozytorium na swój lokalny komputer?

Odpalamy Git Bash. Następnie wchodzimy w interesujący nas katalog na komputerze (do którego sklonujemy nasze repozytorium) i używamy komendy: *git clone <adres repozytorium>*. Po chwili powinniśmy dostać informację, że nasze repozytorium zostało sklonowane do interesującej nas ścieżki.

```
Przemysław Kurzak@LAPTOP-269KMLDC MINGW64 /c/VivadoProjekt
$ git clone https://github.com/NidhoggProject/UEC2_Nidhogg
Cloning into 'UEC2_Nidhogg'...
remote: Enumerating objects: 23, done.
remote: Counting objects: 100% (23/23), done.
remote: Compressing objects: 100% (20/20), done.
remote: Total 23 (delta 4), reused 0 (delta 0), pack-reused 0
Unpacking objects: 100% (23/23), 522.49 KiB | 1.42 MiB/s, done.
```

Zdj.1: git clone <adres repozytorium>

2. Sprawdzenie statusu naszego repozytorium?

Przechodząc do folderu na komputerze, w którym znajduje się nasze sklonowane repozytorium wpisujemy komendę: *git status*. Widzimy też przy okazji na jakim jesteśmy aktualnie branch'u (tutaj głównym: *master*).

```
Przemysław Kurzak@LAPTOP-269KMLDC MINGW64 /c/VivadoProjekt/UEC2_Nidhogg (master)
$ git status
On branch master
Your branch is up to date with 'origin/master'.
```

Zdj.2: git status

3. Jak mogę dodać nowego brancha?

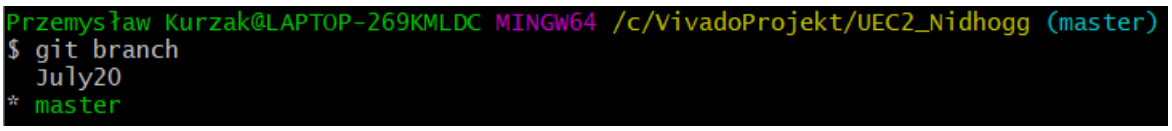
Jest na to kilka sposobów (np. https://stackabuse.com/git-create-a-new-branch/). Jednym z nich jest wpisanie komendy: *git branch <nazwa_brancha>*. Należy pamiętać, że ta komenda nie przenosi nas do nowo utworzonego brancha! Zostajemy w tym, w którym aktualnie byliśmy.

```
Przemysław Kurzak@LAPTOP-269KMLDC MINGW64 /c/VivadoProjekt/UEC2_Nidhogg (master)
$ git branch July20
```

Zdj.3: git branch <nazwa_brancha>

4. Jak sprawdzić w jakim aktualnie branchu się znajduję?

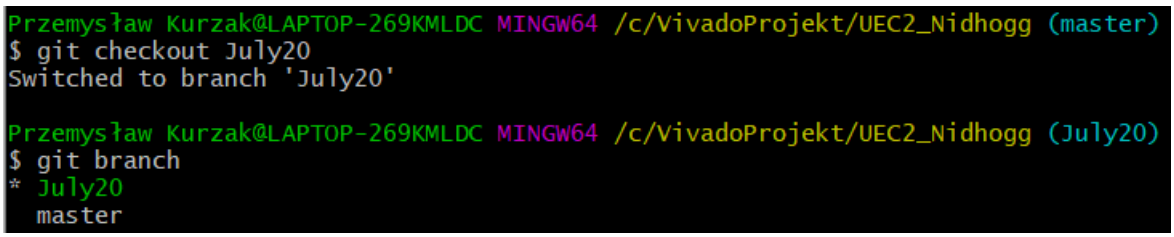
Wystarczy wpisać komendę: *git branch*, a dowiemy się w jakim jesteśmy aktualnie branchu.



Zdj.4: git branch

5. Jak przenieść się na inny branch?

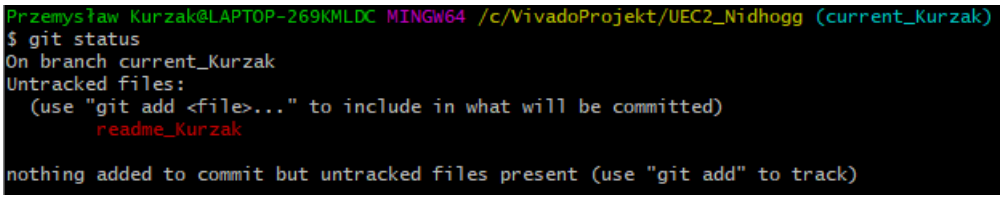
Należy wpisać: *git checkout <nazwa_brancha_na_ktory_chcemy_przejsc>*, po czym dostaniemy komunikat o przeniesieniu na podany branch. Aby sprawdzić, czy na pewno jesteśmy na innym branchu można skorzystać z *git branch*.



Zdj.5: git checkout <docelowa_nazwa_brancha>

6. Status Git'a?

Aby sprawdzić czy są zrobione jakieś zmiany, które należy zatwierdzić możemy skorzystać z polecenia: *git status*.



Zdj.6: git status

7. Jak dodać pliki na serwer?

Aby dodać pliki na serwer należy skorzystać kolejno z: *git add <file_name>*, *git add ** następnie *git commit -m „comment"* oraz wypchnąć te zmiany na serwer przez: *git push origin <branch_name>*. Trzymajmy się schematu, że gdy uzupełniamy komentarz to:

- w przypadku *dodania* pliku piszemy: *add <...>*,

- w przypadku *zmiany* pliku piszemy: *change <...>*,

- w przypadku *usunięcia* pliku piszemy: *delete <...>*.

```
Przemysław Kurzak@LAPTOP-269KMLDC MINGW64 /c/VivadoProjekt/UEC2_Nidhogg (July20)
$ git add ProjektGithub.pdf

Przemysław Kurzak@LAPTOP-269KMLDC MINGW64 /c/VivadoProjekt/UEC2_Nidhogg (July20)
$ git add *
```

*Zdj.7: git add, git add **

```
Przemysław Kurzak@LAPTOP-269KMLDC MINGW64 /c/VivadoProjekt/UEC2_Nidhogg (July20)
$ git commit -m "New test file"
[July20 f3d9581] New test file
 1 file changed, 0 insertions(+), 0 deletions(-)
 create mode 100644 ProjektGithub.pdf

Przemysław Kurzak@LAPTOP-269KMLDC MINGW64 /c/VivadoProjekt/UEC2_Nidhogg (July20)
$ git status
On branch July20
nothing to commit, working tree clean
```

Zdj.8: git commit -m „comment”

```
Przemysław Kurzak@LAPTOP-269KMLDC MINGW64 /c/VivadoProjekt/UEC2_Nidhogg (master)
$ git push origin master
Enumerating objects: 1, done.
Counting objects: 100% (1/1), done.
Writing objects: 100% (1/1), 226 bytes | 226.00 KiB/s, done.
Total 1 (delta 0), reused 0 (delta 0)
To https://github.com/NidhoggProject/UEC2_Nidhogg
   8475c26..b07e037  master -> master
```

Zdj.9: git push origin <branch_name>

8. Jak usunąć niepotrzebne pliki z naszego brancha?

Należy użyć komendy: *git rm <file_name>*, następnie trzeba oczywiście jeszcze użyć *git add **, po czym można zrobić commit’a, którego opisujemy *delete: what_I_deleted* i oczywiście wypchnąć zmiany na serwer *git push origin <branch_name>*.

```
Przemysław Kurzak@LAPTOP-269KMLDC MINGW64 /c/VivadoProjekt/UEC2_Nidhogg (Kurzak_keyboard)
$ git rm ProjektGithub.pdf
rm 'ProjektGithub.pdf'
```

Zdj.10: git rm <file_name>

9. Jak usunąć brancha?

Należy pamiętać, że brancha możemy usunąć zarówno lokalnie jak również i z serwera. Żeby usunąć brancha z serwera posłuży nam do tego komenda: *git push origin –delete <branch_name>*, a do usunięcia lokalnie: *git branch -D <branch_name>*.

```
Przemysław Kurzak@LAPTOP-269KMLDC MINGW64 /c/VivadoProjekt/UEC2_Nidhogg (master)
$ git push origin --delete July20_new
To https://github.com/NidhoggProject/UEC2_Nidhogg
 - [deleted]         July20_new

Przemysław Kurzak@LAPTOP-269KMLDC MINGW64 /c/VivadoProjekt/UEC2_Nidhogg (master)
$ git branch -D July20_new
Deleted branch July20_new (was 50bb820).
```

Zdj.11: git push origin –delete <branch_name>, git branch -D <branch_name>

10. Nasz schemat działania z repozytorium?

Każdy powinien mieć swojego brancha: *current_Roza* i *current_Kurzak*. Od tych branchy robimy inne, które będą odpowiedzialne za poszczególne części projektu, np. Kurzak_keyboard -> wiadomo, że tutaj wrzucać będę pliki związane z obsługą klawiatury. Jeżeli program będzie działał, to dopiero wtedy zmiany można *mergować* do brancha *master*. Dlatego też główna hierarchia naszych branchy powinna wyglądać następująco:

- master,

- current_Roza (będąc na tym branchu tworzymy inny dla konkretnej funkcjonalności programu!),

- current_Kurzak (będąc na tym branchu tworzymy inny dla konkretnej funkcjonalności programu!).

Aby zobaczyć graficzną reprezentację naszego repozytorium można posłużyć się instrukcją (np. https://docs.github.com/en/github/visualizing-repository-data-with-graphs/viewing-a-repositorys-network). Będąc na naszym repozytorium przechodzimy do -> *Insights* -> *Network*.

Komendy Linuksa, które mogą się przydać do poruszania się po GitBash:

Inne komendy (np. https://www.astrouw.edu.pl/~jskowron/pracownia/komendy/)

- ls = wyświetla zawartość bieżącego katalogu
- mkdir <directory_name> = stworzenie folderu w danym katalogu o podanej nazwie
- cd = przenosi nas do katalogu domowego
- cd .. = przenosi nas do katalogu wyżej
- cd / = po slashu należy napisać ścieżkę, by przeniosło nas do interesującego nas katalogu
- rm -r <file_name> = usunięcie podanego pliku
- cp -R <source_folder> <destination_folder> = kopiuje zawartość folderu z jednej ścieżki do drugiej
- nano <file_name> ./ = stworzenie pliku tekstowego (i otwarcie w edytorze nano) w bieżącym katalogu

Inne materiały

Przewodnik Git'a:

https://rogerdudler.github.io/git-guide/index.pl.html

https://git-scm.com/book/pl/v2/Pierwsze-kroki-Wprowadzenie-do-kontroli-wersji

https://kot-zrodlowy.pl/programowanie/2017/10/11/korzystanie-z-git-i-githuba.html

Gałęzie (branche) Gita? Jak nazywamy branche?

https://git-scm.com/book/pl/v2/Ga%C5%82%C4%99zie-Gita-Podstawy-rozga%C5%82%C4%99ziania-i-scalania

https://stackoverflow.com/questions/273695/what-are-some-examples-of-commonly-used-practices-for-naming-git-branches

Github pages:

https://pages.github.com/