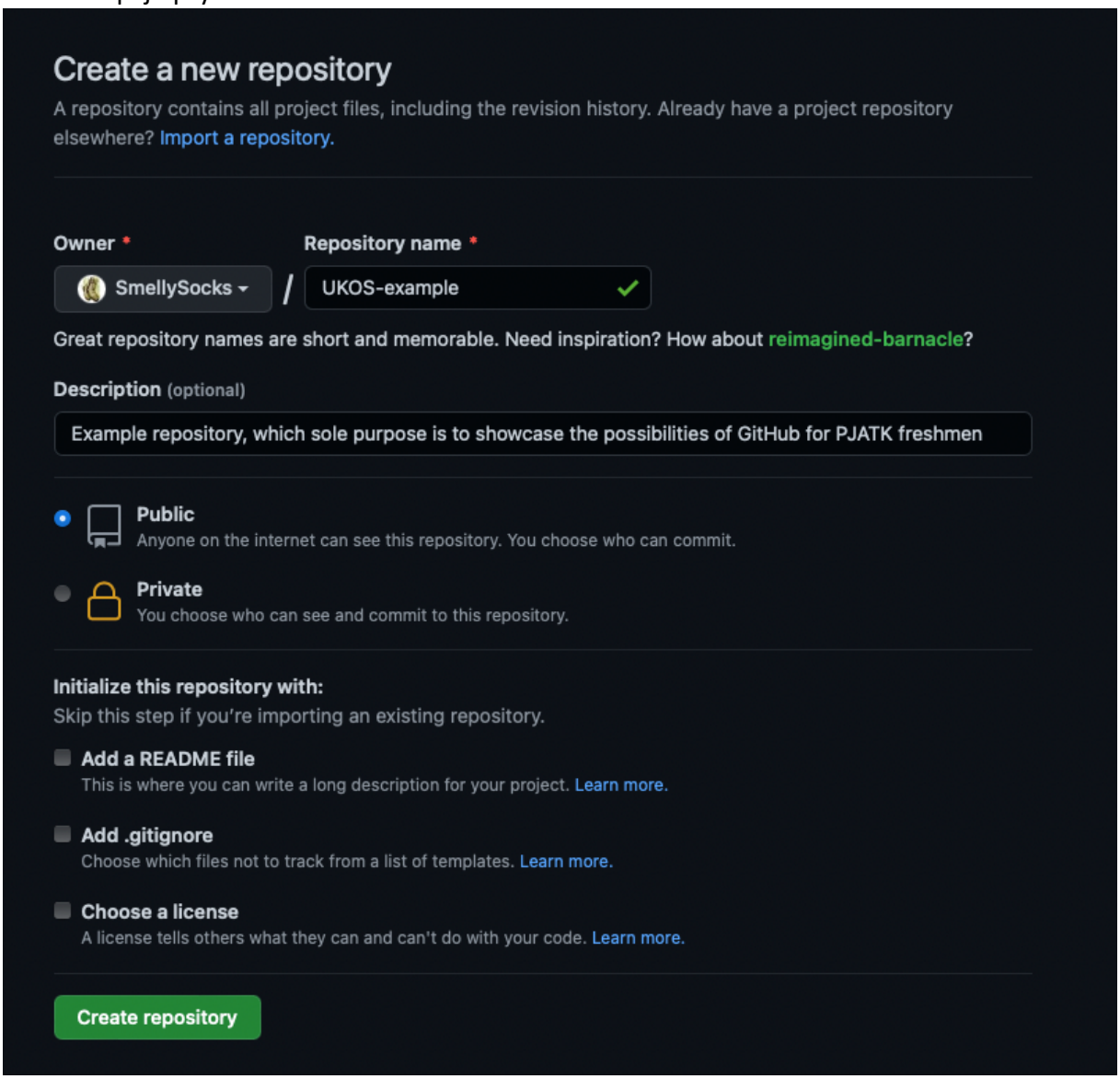
Git – praca z repozytorium zdalnym

Poprzednio zapoznaliśmy się z podstawami pracy z repozytorium git, ale jedynie w wersji lokalnej – dziś zajmiemy się pracą z repozytorium zdalnym. Repozytorium zdalne to w zasadzie kopia naszego repozytorium lokalnego, tylko utrzymywana w jakiejś zdalnej lokalizacji – może to być zarówno ogólnodostępny serwis jak GitHub, GitLab, Bitbucket, ale repozytorium zdalne możemy równie dobrze trzymać na Raspberry Pi schowanym w szafie.

Jak takie repozytorium stworzyć i skonfigurować?

Generalnie opcji jest kilka. Niektóre IDE po prostu mają możliwość wklejenia linku do repozytorium i automatycznie wszystko ustawiają. Do takich należy chociażby VS Code, lub wszystko, co stworzył zespół JetBrains (CLion, IntelliJ IDEA itd.). Ale jakoś ten link do repozytorium musimy zdobyć, nieprawdaż?

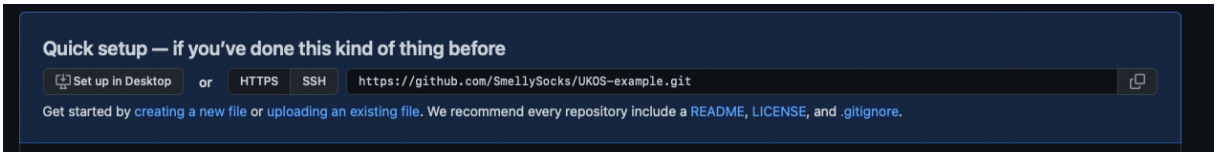
Aby stworzyć nowe, czyste (lub prawie czyste) repozytorium na GitHubie należy w sekcji *repositories* kliknąć *new*. Takie repozytorium należy oczywiście jakoś nazwać, dodać opis, ustawić opcje prywatności.



Poniżej znajdują się również tickboxy, które pozwalają na dodanie do repozytorium pustego pliku Readme.md (.md to format Markdown, prosty język znaczników) oraz .gitignore. Tego gitignore można oczywiście zainicjalizować pustego, ale można też wybrać sobie jeden z gotowych template'ów.

> Ważna informacja. Jeśli zaznaczymy którykolwiek z checkboxów, to zainicjalizowane repozytorium nie będzie puste, część rzeczy zostanie ustawiona domyślnie

W tym momencie możemy po prostu skopiować link do repozytorium



I wkleić go w odpowiednim miejscu w naszym IDE. Jeśli jednak chcemy dowiedzieć się jak to dokładnie działa, to musimy skorzystać z terminala. Załóżmy, że mamy już jakiś zalążek projektu i repozytorium lokalne.

```
git remote add [skrót] [url]
```

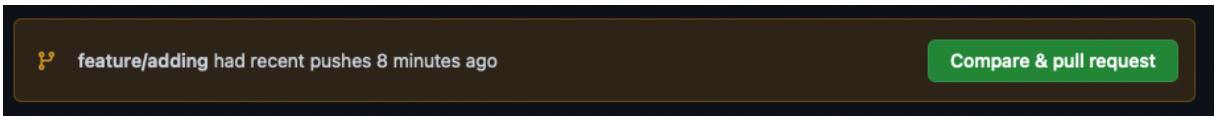
To polecenie doda nam alias do repozytorium zdalnego, np.

```
git remote add origin https://github.com/SmellySocks/UKOS-example.git
```

Natomiast poniższe polecenie wypchnie nasz lokalny branch main do zdalnego origin/main

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

I od teraz nasze życie się dopiero może skomplikować. Co prawda, w sytuacji w której pracujemy nad jakimś rozwiązaniem w pełni samodzielnie możemy po prostu wszystkie nasze lokalne zmiany wrzucać na maina i regularnie pushować, ale nie jest to najlepszą praktyką, lepiej skorzystać z mechanizmu branchy. Gdy pracujemy nad jakimś featurem i uznajemy, że jest on godzien włączenia go do gałęzi main, albo przynajmniej develop, udać musimy się na githuba do naszego zdalnego repozytorium, na którym już najduje się nasz branch. Nie wykonujemy merge'a lokalnie, zrobimy go poprzez githuba.

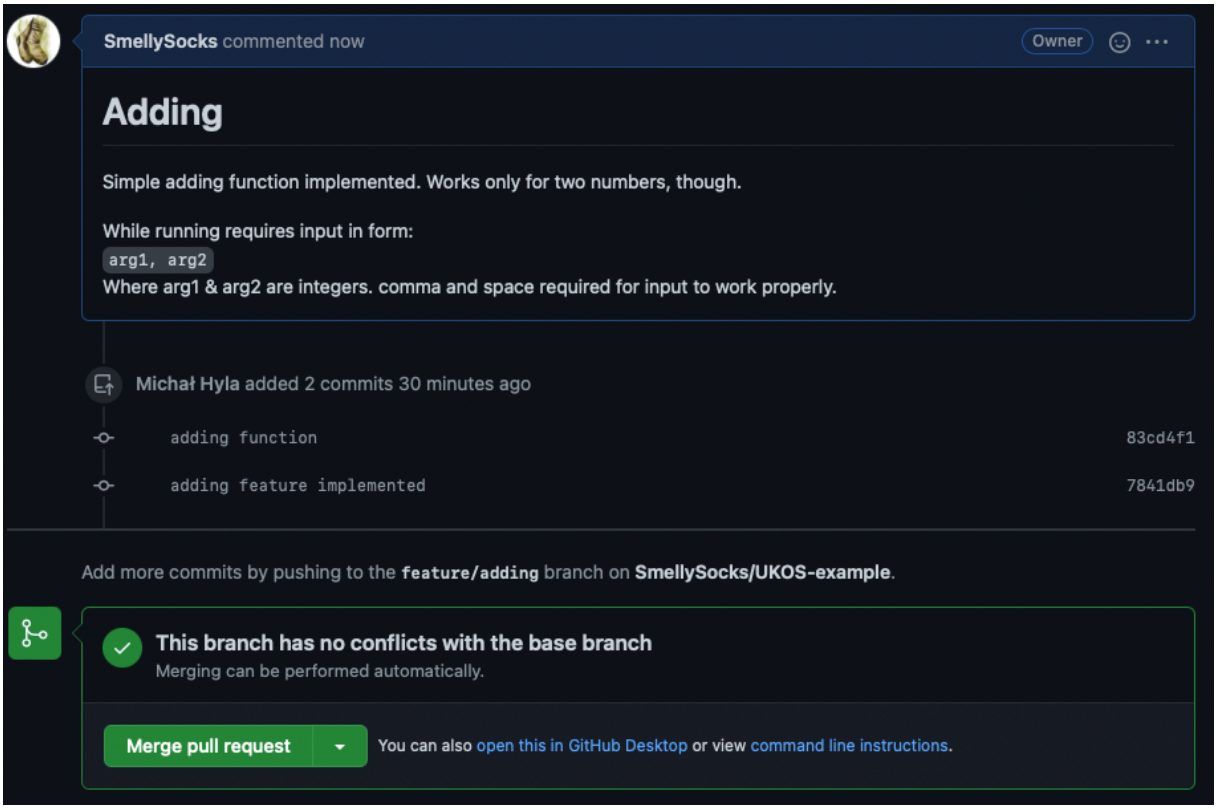


Na górze strony powinna znajdować się taka informacja, aby stworzyć *pull requesta* wystarczy kliknąć zielony guziczek – jeśli jej tam jednak nie ma, możemy go stworzyć wchodząc w zakładkę *Pull Requests* i klikając *new pull request*.

Czym jest ten Pull Request?

Generalnie jest to taka informacja puszczona do członków zespołu, mówiąca „Hej, skończyłem pracować nad taskiem, rzućcie okiem i dołączcie do głównego brancha jak wszystko będzie okej.

Na ekranie tworzenia pull requesta wybrać musimy branch do którego dołączane będą zmiany oraz branch z którego te zmiany będą pochodzić. Po kliknięciu Create Pull Request pojawi się okienko z miejscem, w którym powinniśmy pokrótce opisać wprowadzane zmiany. Po prawej stronie można natomiast przypisać zadanie do jakiegoś członka zespołu, tj. poprosić go o przeprowadzenie *code review*, wskazać, z którego projektu w ramach repozytorium pochodzi nasz pull request, opisać go odpowiednią etykietą itd.



Gdy pull request zostanie już zmerge'owany na repozytorium zdalnym oznaczać to będzie, że nasz branch develop jest już nieaktualny (tak naprawdę, kiedy pracujemy w zespole zdezaktualizować się on może w każdej chwili). Aby dociągnąć nowe zmiany, konieczne przed rozpoczęciem pracy nad nową funkcjonalnością, która wymagałaby stworzenia nowego brancha rozgałęziającego się właśnie od developa, musimy dociągnąć zmiany. W tym celu musimy na odpowiednim branchu wykonać polecenie

```
git pull origin develop
```

git pull ma niestety jeden problem – dociąga on zmiany dokładnie tak samo jak git fetch, ale wykonać może również merge commita. Generalnie, jeśli korzystamy z gita prawidłowo, to możemy coś takiego dopuścić. Całe drzewo zacznie się natomiast walić w sytuacji, w której na developie znajdzie się coś, czego nie powinno tam być.

Praktyk rozwiązujących ten problem jest sporo – niektórzy zalecają zawsze wykonanie pulla przed pushem. Zalecane też jest zamiast git pull wykonywanie

```
git pull –ff-only
```

To spowoduje wyrzucenie błędu, jeśli nie będziemy mogli wykonać pulla bez tworzenia nowego merge'a. Po więcej informacji zachęcam do zapoznania się z tym wpisem. To da nam okazję do zastanowienia się, co chcemy z tym fantem zrobić.