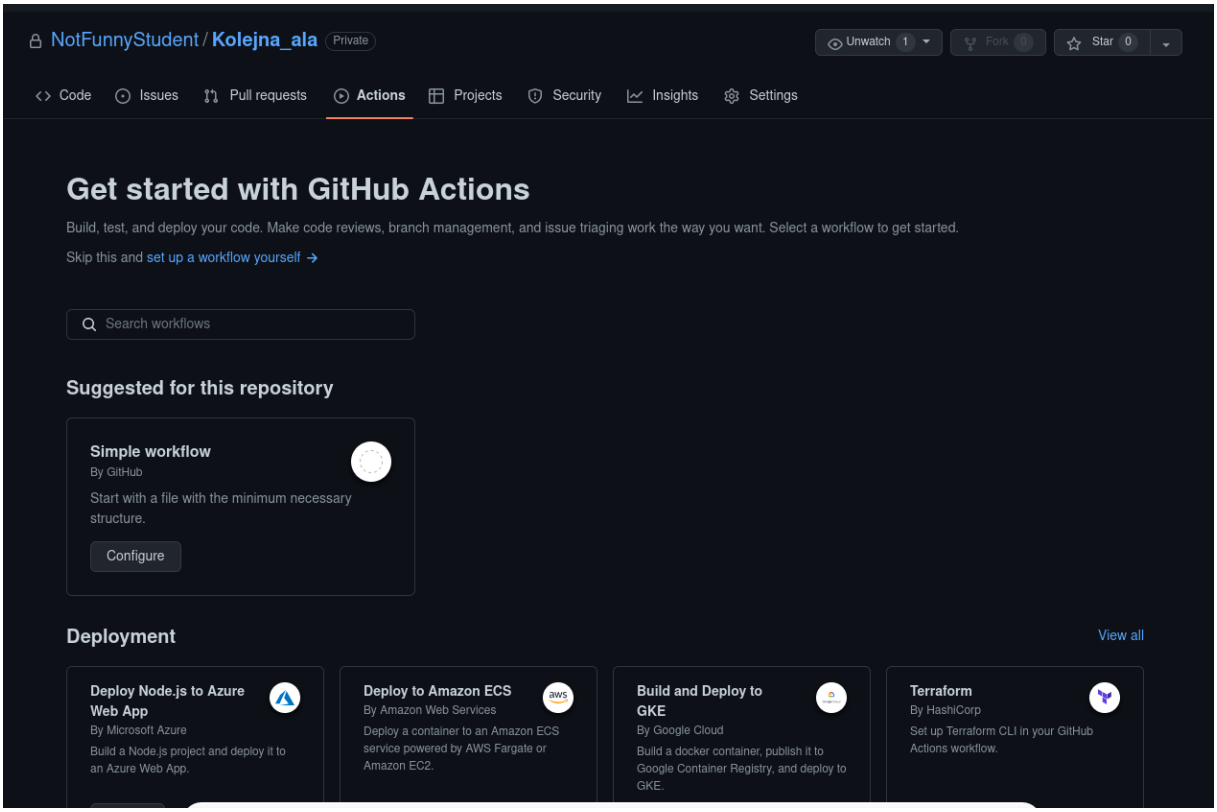
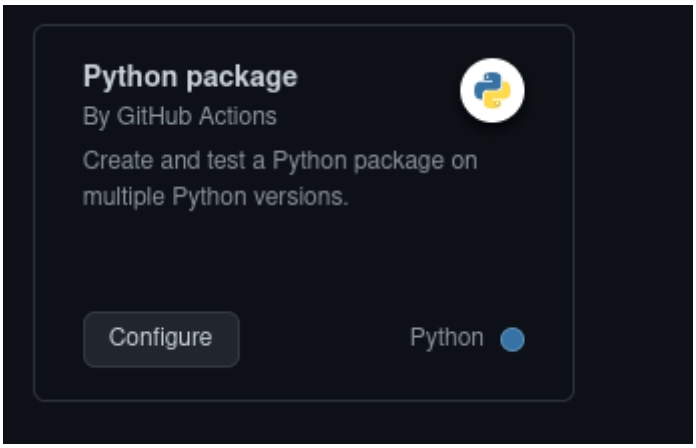
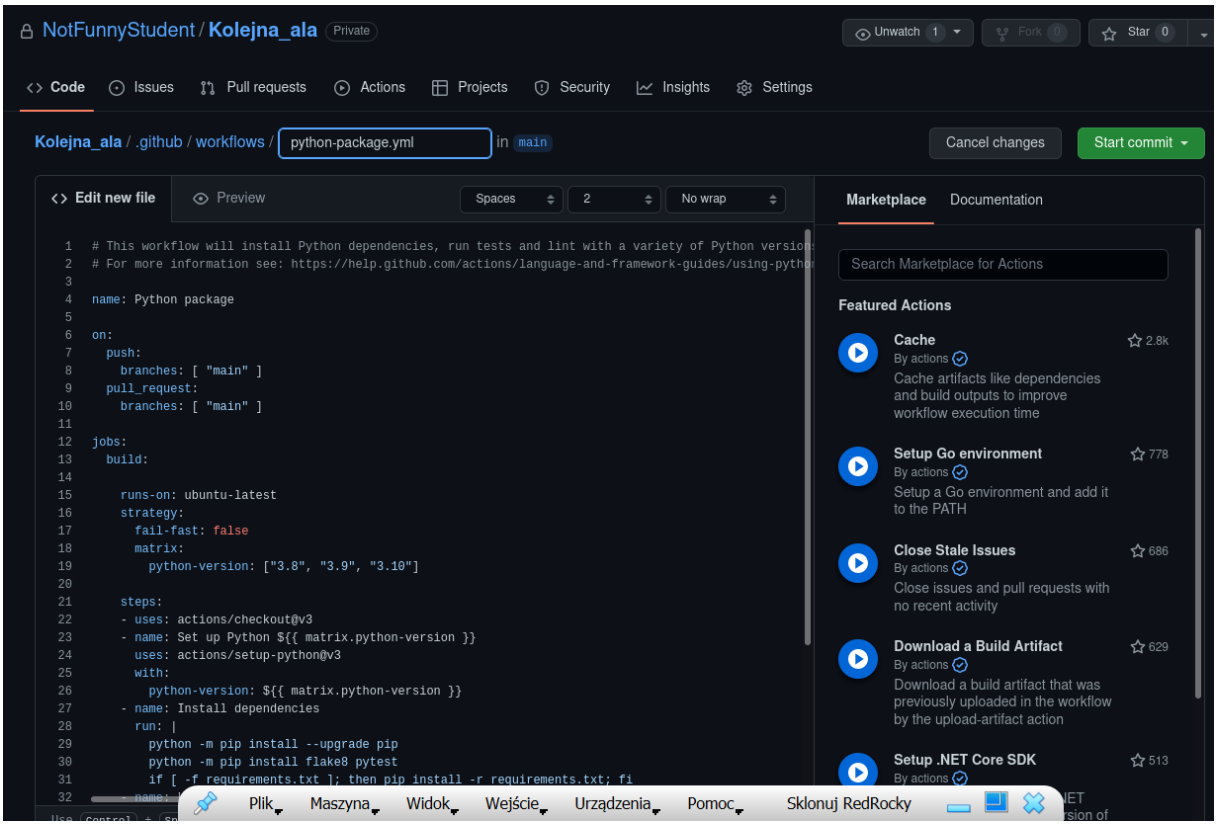
Ustawianie automatycznego testowania kodu z wykorzystaniem GitHub Actions

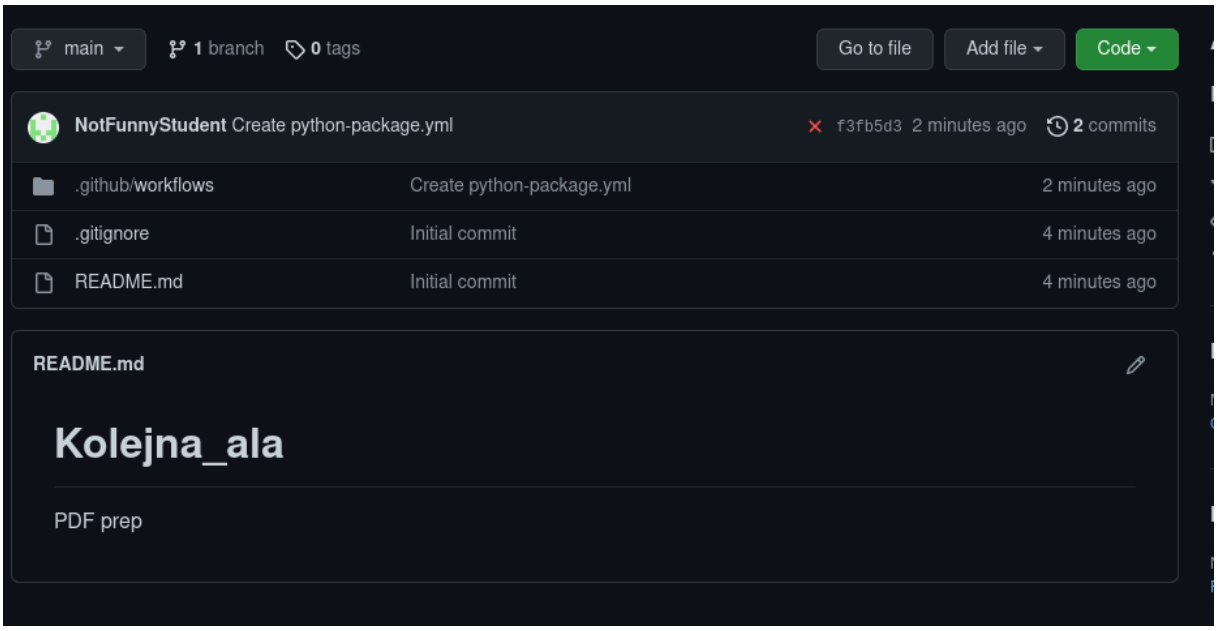


Szukamy takiego cuda



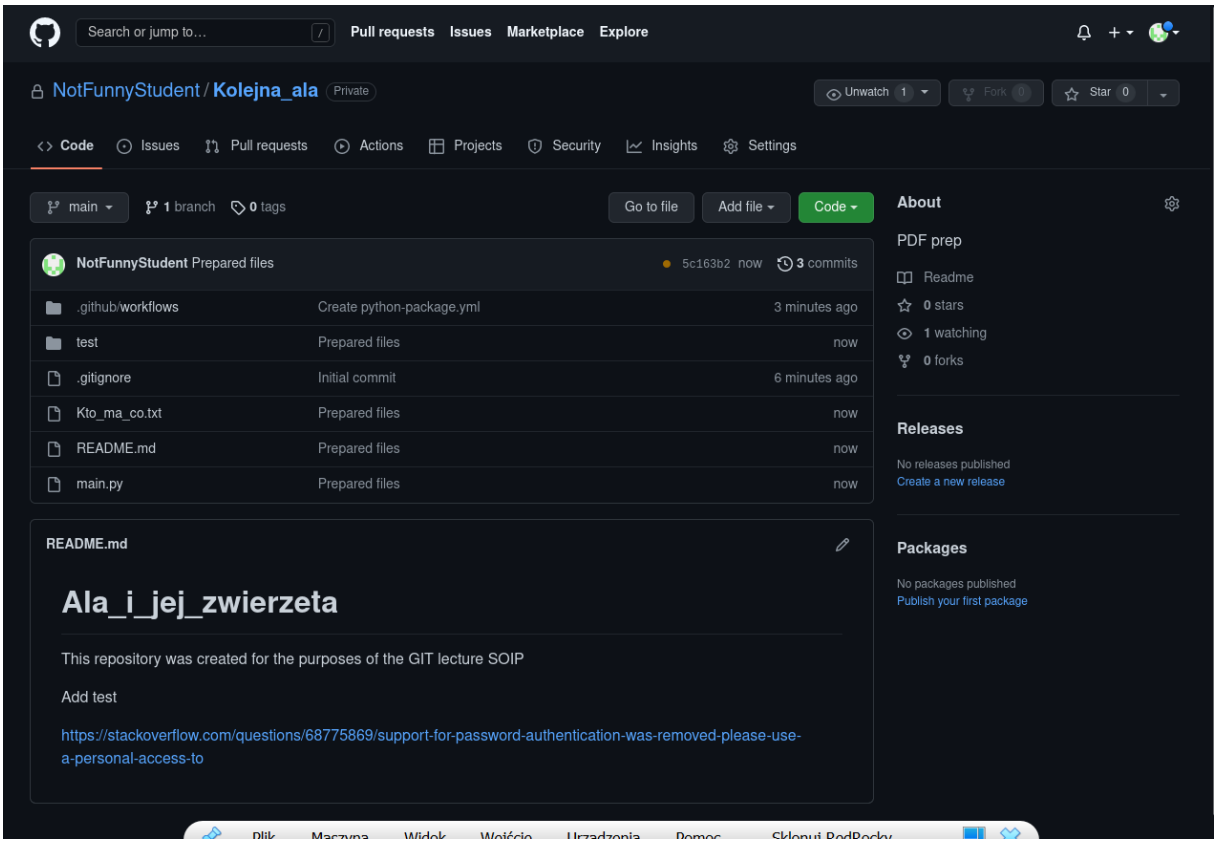
Pytest’a można też zainstalować z łapki, tak samo jak i stworzyć całkowicie własną konfigurację, ale powiedzmy, że na spotkaniu zajmiemy się tylko podstawami.





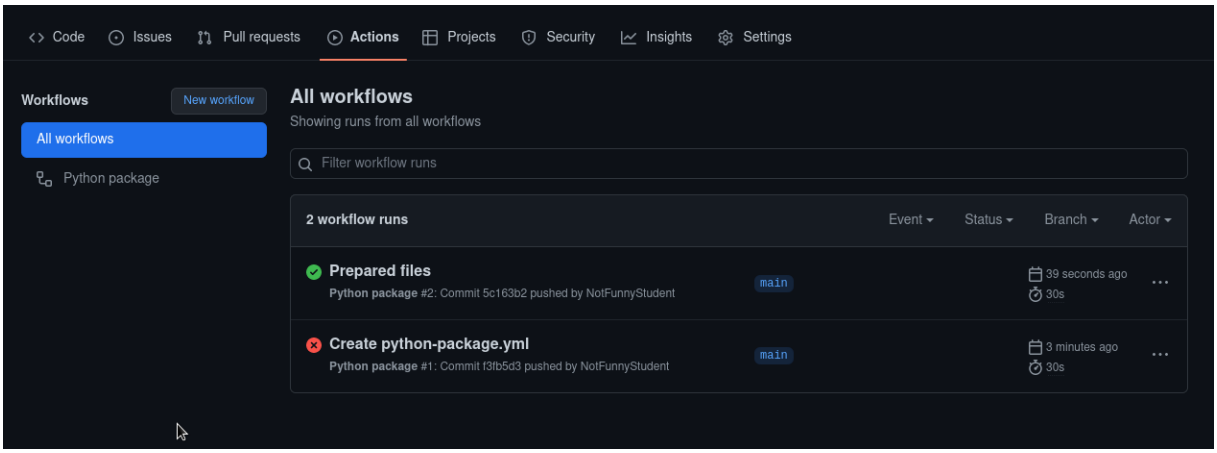
Zostaliśmy z takim repozytorium i co dalej?

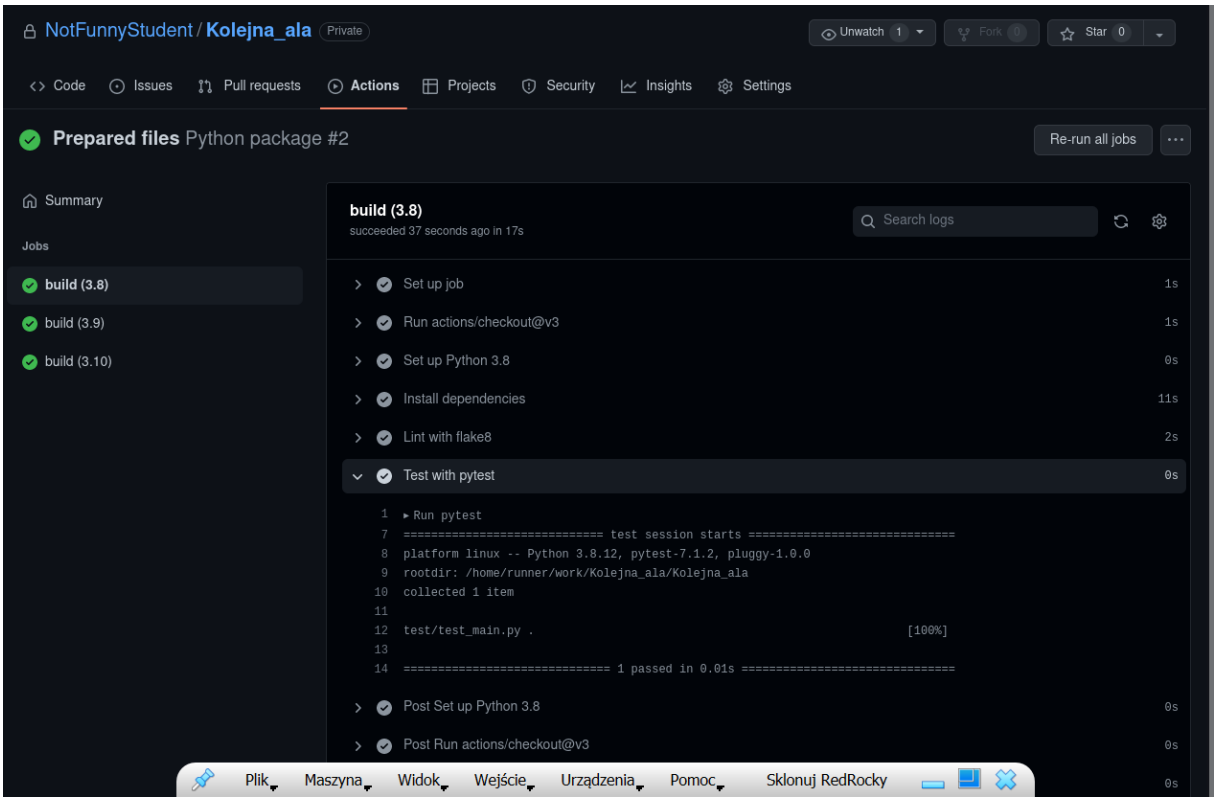
Robimy commita naszymi pliczkami



I jak widzicie czerwony x nam gdzieś zaginął, jakaś żółta kropka się pojawiła obok hasha

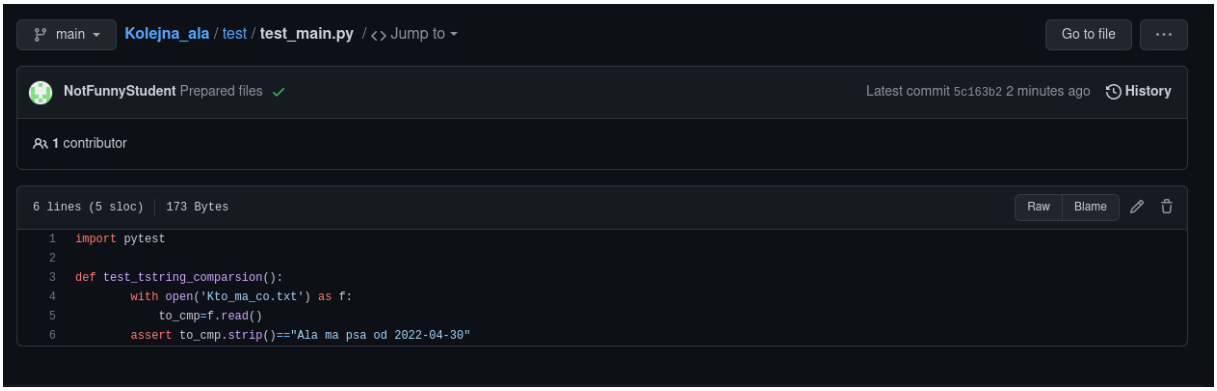
Skoczmy sobie do zakładki Actions



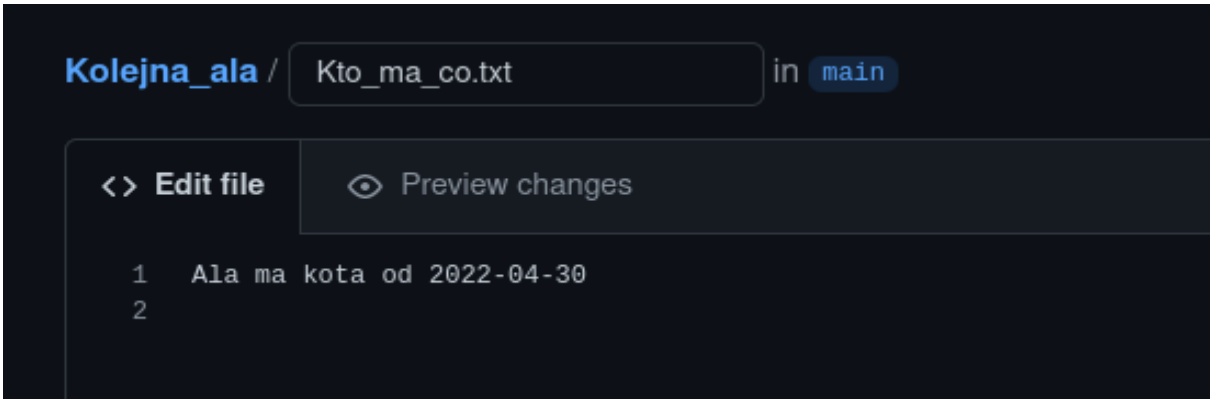


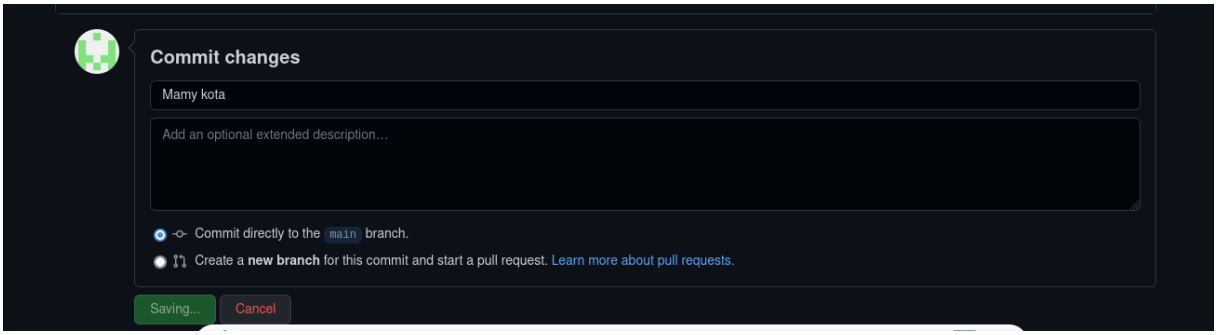
I co się okazuje, że coś nam się potestowało, a przynajmniej nie rzuciło błędów.

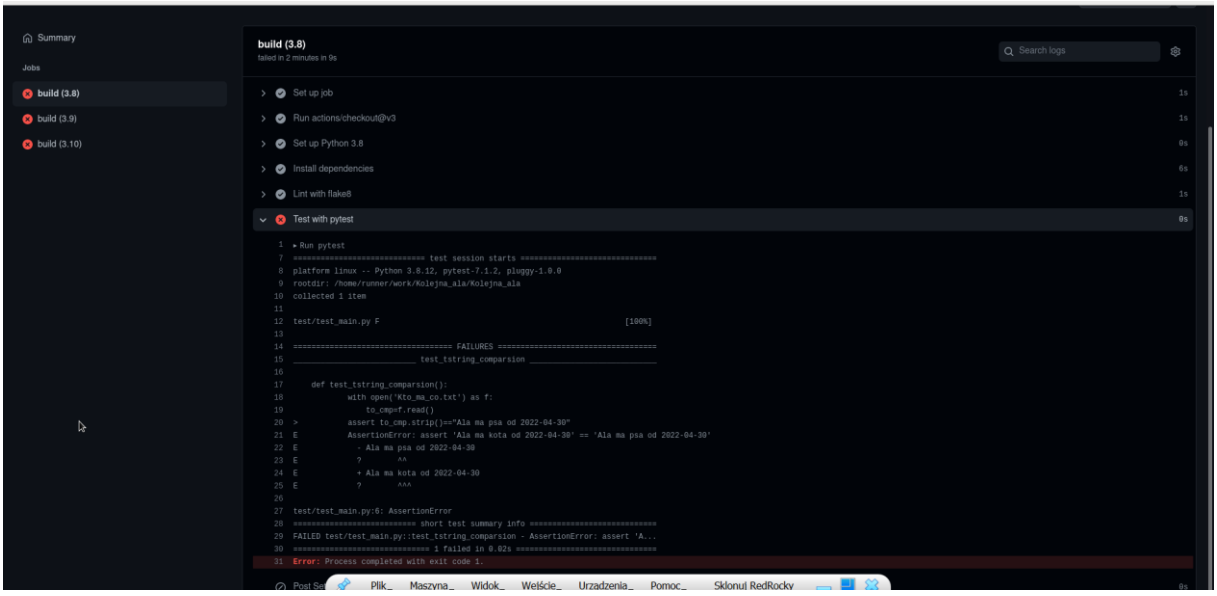
No, ale teraz tak warto się zastanowić, czym jest ten nasz test



I co się stanie jeśli spróbujemy wpisać do pliku inny string i zasymulujemy zdarzenie on_push?

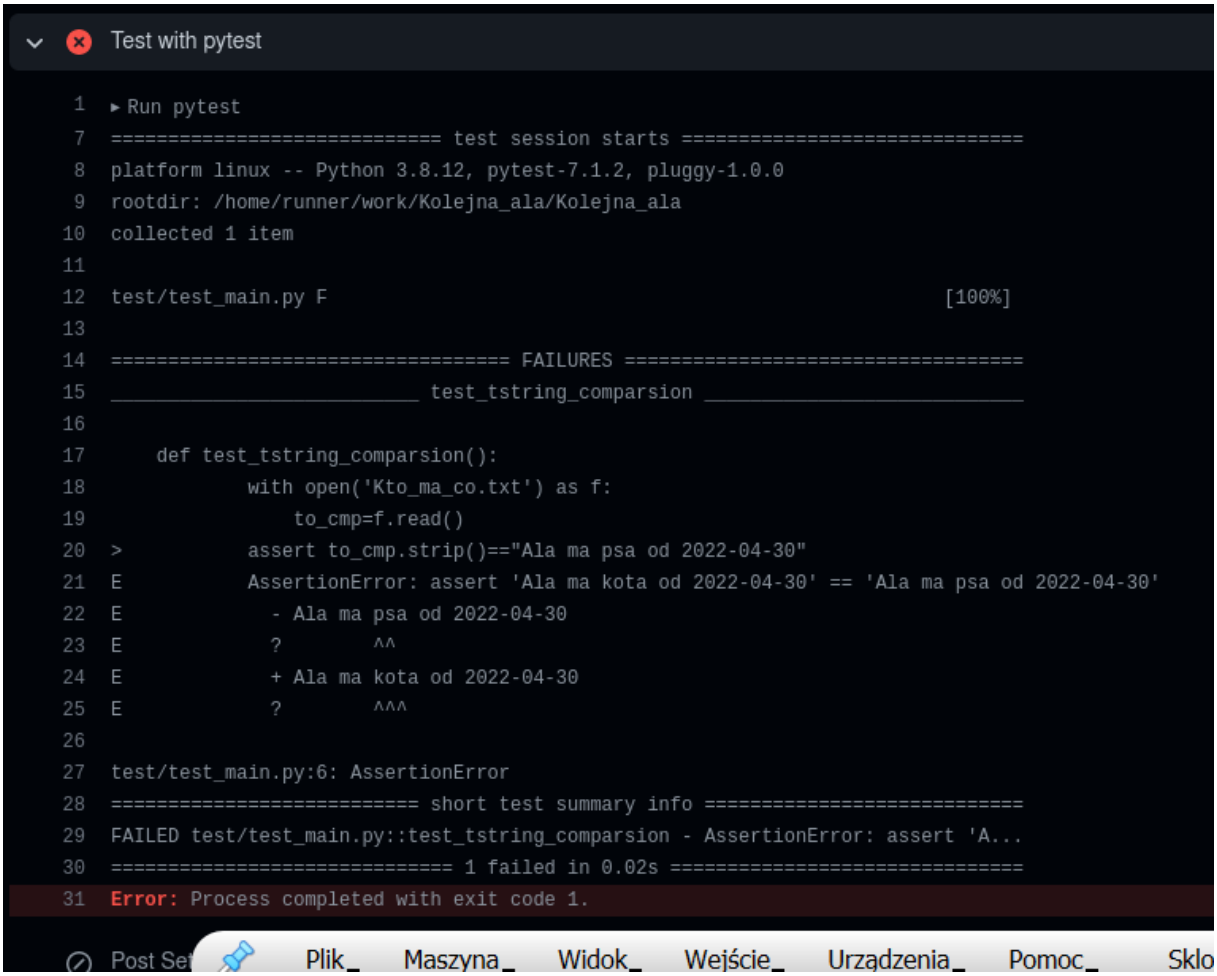






Jak widzimy w raporcie program nam się wysypał. Także otwieramy szampana, nasz kod ma pokrycie równe 100% i jesteśmy bezpieczni.

Zwróćcie uwagę na to, że raprot mówi nam co dokładnie poszło nie tak



W funkcji takiej to a takiej, to i to mi się nie zgadza. Co prawda sami również możemy trochę modyfikować jakie dane zwraca nam raport, ale przyznacie, że jest to już potężne ułatwienie.

*Test failed w domyślnym ustawieniu śle nam na maila powiadomienie, że oj coś poszło nie tak. Z jednej strony fajnie, z drugiej szybko można sobie zaśmiecić pocztę jeśli tego nie zauważymy.

Co powinniśmy zapamiętać:

- Te nazwy jak i struktura to nie jest całkowity przypadek, aby to działało poprawnie jednak musimy współpracować z wytycznymi dokumentacji
- Testy jednostkowe powinny testować mały fragment - przeważnie jedną funkcję.
- Testy w naszej przestrzeni zwracają tylko dwa stany True/False
- Testy powinny być szybkie i niezależne