Sprawozdanie
Narzędzia do automatyzacji budowy oprogramowania

Kowal Maciej
100400

January 2025

Spis treści

Spis rysunków

1 Repozytorium GitHub

Nazwa użytkownika: MaciejK11

Link do repozytorium github z pierwszych zajęć:
`https://github.com/MaciejK11/nauka_gita_mk_wsb.git`

Link do forka(żeby było wiadomo który jest mój):
`https://github.com/MaciejK11/2025_L_II_NWh_INFI_IAMm.git`

2 Komendy

Lista podstawowych komend wykorzystanych w ramach laboratorium

1. **git init** Tworzy nowe lokalne repozytorium Git w bieżącym katalogu, tworzy folder ukryty .git.
2. **git config** Konfiguracja ustawień globalnych gita, takich jak nazwa użytkownika czy email.
3. **git clone** Kopiuje istniejące repozytorium z serwera na PC. Przyjmuje m.in. adres https repozytorium.
4. **git add** Dodaje zmodyfikowane pliki do obszaru staging, faza przed commit
5. **git commit** Zapisuje zmiany z obszaru staging w historii repozytorium. Wymusza komentarz.
6. **git status** Wyświetla aktualny stan repozytorium, pokazując zmodyfikowane, dodane i nieśledzone pliki. Pomaga kontrolować, co zostanie zapisane w kolejnym commicie.
7. **git remote** Zarządza połączeniami z repozytoriami zdalnymi. Pozwala dodawać, usuwać i sprawdzać adresy repozytoriów takich jak origin.
8. **git push** Wysyła lokalne commity do repozytorium zdalnego. Umożliwia synchronizację zmian z serwerem, np. GitHubem. Wymusza zalogowanie do serwisu usługodawcy gita.
9. **git branch** Wyświetla listę gałęzi lub tworzy nową gałąź w repozytorium. Gałęzie pozwalają pracować nad różnymi wersjami projektu równolegle.
10. **git checkout** Przełącza między branchami.
11. **git merge** Łączy zmiany z jednej gałęzi do drugiej.

3 Konfiguracja lokalnego repozytorium

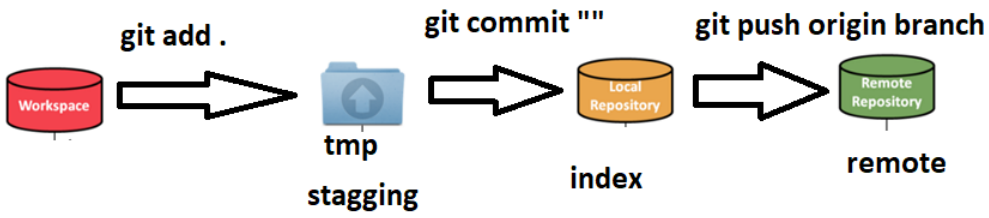
Konfigurujemy lokalne repozytorium git zgodnie z topologią.

```
vboxuser@ubuntu24:~/Desktop/MKowal/mkowal_fork/2025_L_II_NWh_INFI_IAMm$ git co
ig --list
user.email=wrx93174@student.wroclaw.merito.pl
user.name=Mkowal
core.editor=nano
core.repositoryformatversion=0
core.filemode=true
core.bare=false
core.logallrefupdates=true
remote.origin.url=https://github.com/MaciejK11/2025_L_II_NWh_INFI_IAMm.git
remote.origin.fetch=+refs/heads/*:refs/remotes/origin/*
branch.master.remote=origin
branch.master.merge=refs/heads/master
branch.master.vscode-merge-base=origin/master
```

Rysunek 1: Enter Caption

Jest to konfiguracja wykonana przy zadaniach z forkiem, origin to sklonowane repozytorium(git clone http). Na Windowsie miałem problem z ustawieniem notepad++ jako core.editor, git push zawieszał git bash, więc zmieniłem na notepad i działało, powyższy przykład config –list to linux bash, zatem edytor ustawiłem po prostu jako nano.

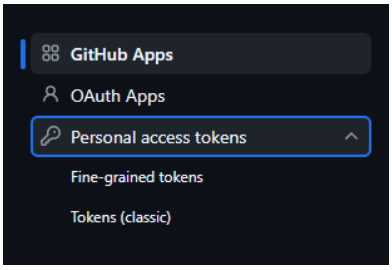
3.1 Schemat



Rysunek 2: Schemat pull request

Wszystkie zmiany jakie wykonamy w repozytorium zostają zapisane w przestrzeni roboczej(może być też tmp/temporary ale nazwałem tak stagging wiec niech tak zostanie), komenda git add example dodaje zmiany wykonane w expamle do folderu pośredniego nazwanego przeze mnie tmp (stagging), następnie indeksowanie polega na zapisaniu zmian z folderu pośredniego w repozytorium lokalnym, zmiany wykonane są na określonych gałęziach, które można tworzyć, mergować i zmieniać. Dopiero git push origin(http/ssh repozytorium na serwerze usługodawcy git) przenosi nas do strony logowania, a nasz commit do repozytorium zdalnego.

3.2 tokeny



Rysunek 3: Opcja utworzenia tokenu

W GitHubie można generować tokeny, które mogą być używane zamiast hasła, mają czas wygaśnięcia i są ogólnie bezpieczniejsze, ponieważ mogą zostać usunięte w dowolnym momencie, wyświetlają sie jednak tylko raz, dlatego są raczej jednorazowe.

3.3 Branch

Branch to wersja repozytorium lokalnego, którą można edytować niezależnie od innych gałęźi.

3.4 Konflikt

Konflikty podczas merge(scalania) mogą uniemożliwić wykonanie commita do momentu ich rozstrzygnięcia.

```
student-wroclaw@MX1-Docker-17 MINGW64 ~/Desktop/MaciejK11/se_nauka_gita (master)
$ git branch
  konflikt
* master

student-wroclaw@MX1-Docker-17 MINGW64 ~/Desktop/MaciejK11/se_nauka_gita (master)
$ git merge konflikt
Auto-merging avg.php
CONFLICT (content): Merge conflict in avg.php
Automatic merge failed; fix conflicts and then commit the result.
```

Rysunek 4: Konflikt gałęzi

Konflikt pomiędzy branchami powoduje modyfikację plików tkórych dotyczy i wygląda następująco:

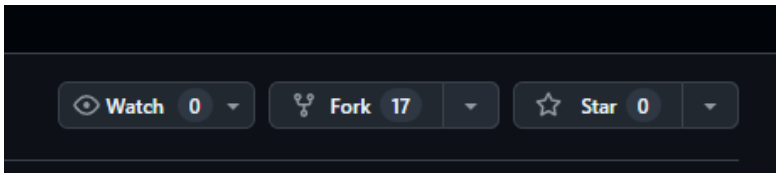
```
k? php

function srednia (a, b)
{
        return (a+b)/2

}

function wazona (a, b)
{
<<<<<<< HEAD
        return (5*a+9*b)/2
=======
        return (7*a+11*b)/2
>>>>>>> konflikt

}

?>
```

Rysunek 5: Komentarze po merge

W takim przypadku kod należy zmienić ręcznie.

4 Fork

Na laboratorium wykonany został fork(kopia) repozytorium podanego przez wykładowcę.



Rysunek 6: Przycisk Fork

Repozytoium zawiera program uruchamiający serwer w terminalu na danym porcie, do którego można się podłączyć z innego terminala(podając adres IP+port) i otrzymać tekst output zależny od wywołanej komendy.

4.1 Flask

Flask jest lekkim frameworkiem webowym w Pythonie(po prostu biblioteka). Służy do tworzenia aplikacji internetowych i API oraz pozwala definiować routing (@app.route), obsługiwać żądania HTTP i zwracać odpowiedzi (HTML, JSON). W naszym przypadku zwracany będzie .json.

4.2 Format .json

JSON to po prostu format zapisu danych. Plik z rozszerzeniem .json przechowuje dane w postaci struktury klucz:wartość.

4.3 Markdown

Format Markdown(.md) to format pliku do dokumentacji tekstowej, zawiera specjalne znaki określające pogrubienia oraz podobne elementy możliwe do reprezentacji graficznej.

4.4 Błędy

lista znalezionych błędów

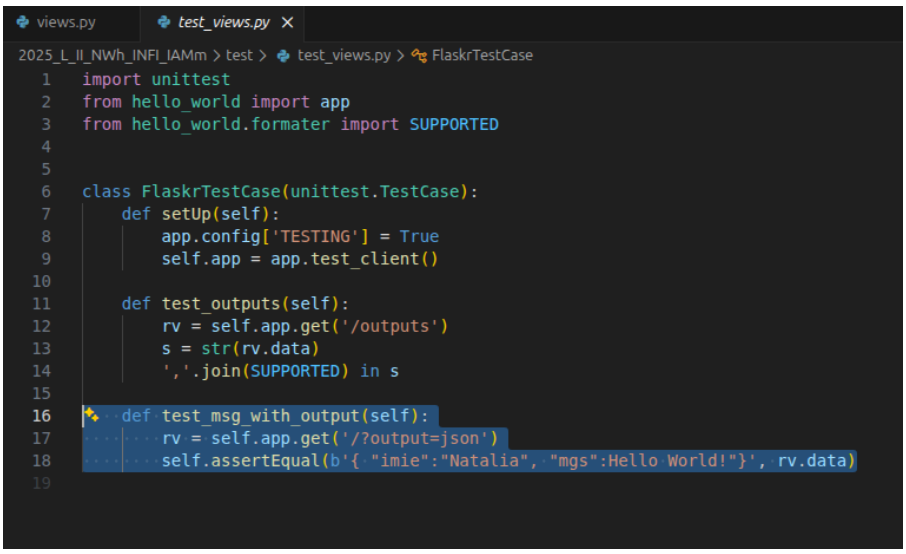
1. format_to_json - zwraca 'mgs' co jest niezgodne z formatem .json zatem zostaje zmienione na 'msg'.
2. stała moje_imie - nie jest błędem ale jej wartość zostaje zmieniona.
3. test_msg_with_output - w wersji programu z 'mgs' .assertEqual działało poprawnie, jednak w nowej wersji zostaje zmienione zgodnie z format_to_json
4. '**README.md** - błędna ścieżka aktywacji .venv, w Source nie ma funkcji activate, znajduje się ona w bin dla linuxa oraz Scripts dla Win10/11.

views.py

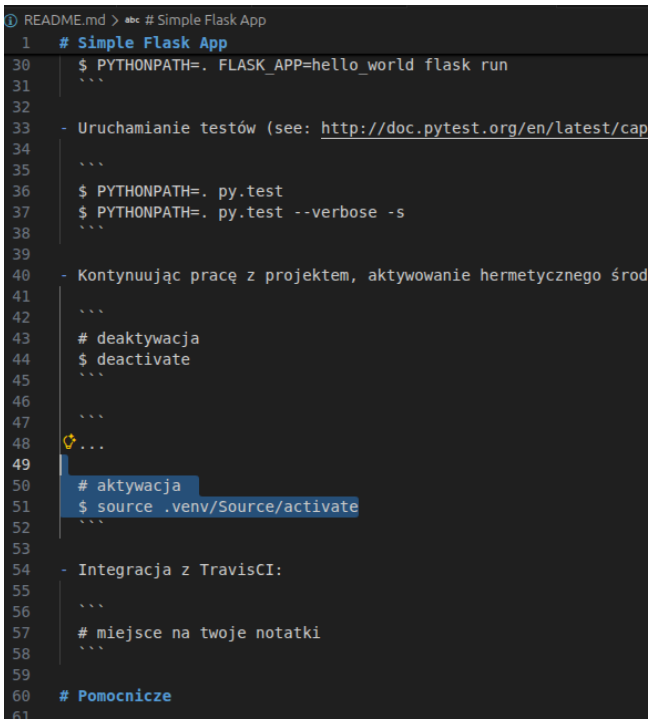
2025_L_II_NWh_INFI_IAMm > hello_world > views.py > moje_imie

```python
from hello_world import app
from hello_world.formater import get_formatted
from hello_world.formater import SUPPORTED, PLAIN
from flask import request

moje_imie = "Natalia"
msg = "Hello World!"

@app.route('/')
def index():
    output = request.args.get('output')
    if not output:
        output = PLAIN
    return get_formatted(msg, moje_imie,
                         output.lower())

@app.route('/outputs')
def supported_output():
    return ", ".join(SUPPORTED)
```

Rysunek 7: Zmiana imienia



Rysunek 8: Zmiana Ensure

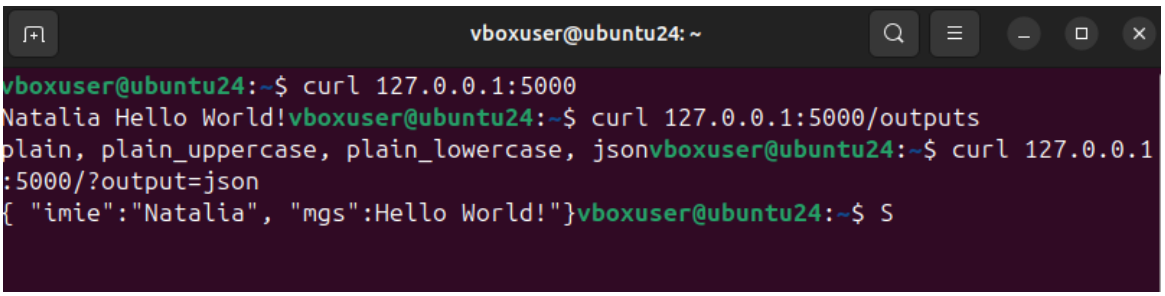


Rysunek 9: Markdown bug

Powyższy przykład to nie jest do końca błąd w kodzie zatem go nie zmieniłem, jest to błąd dokumentacji.

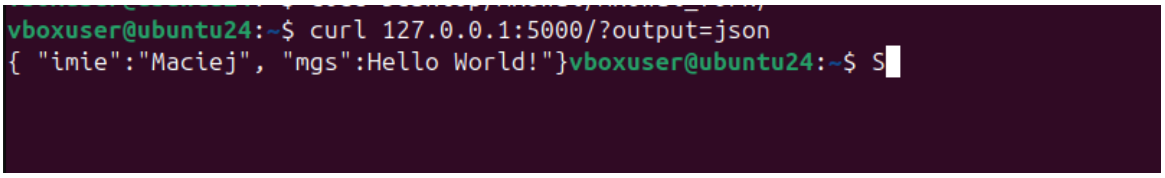
4.5 Działanie

Do uruchomienia programu wykorzystujemy albo przeglądarkę podając IP oraz nr portu, lub narzędzie curl spełniające tę samą funkcję (zwraca dane tekstowe, w układzie JSON).



Rysunek 10: Aplikacja przed zmianami stałych

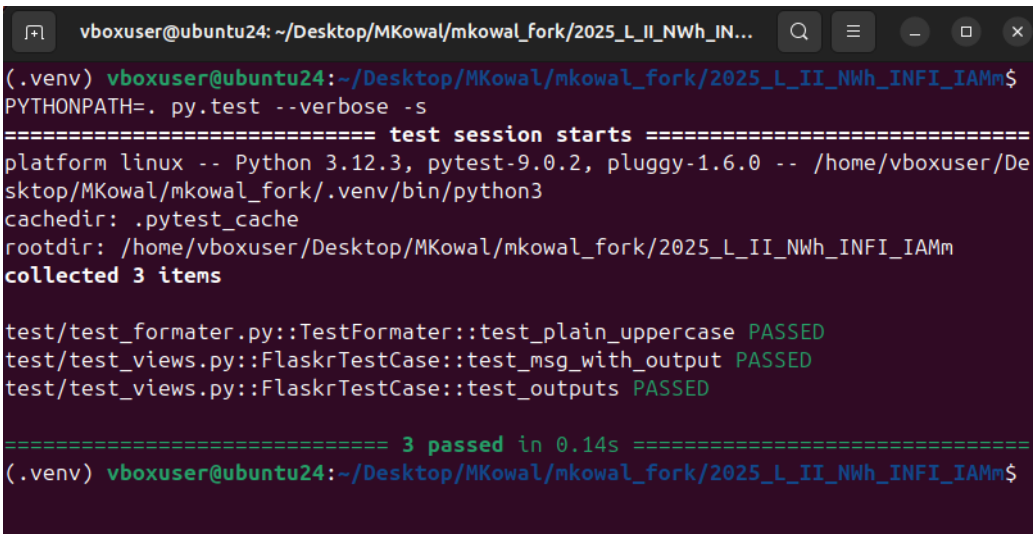
Rysunek 11: Aplikacja po zmianach stałych



Rysunek 12: Zwrócony format .json(mgs jeszcze nieporawione)

4.6 Testowanie

Rysunek 13: Test nr 1



Rysunek 14: Test nr2

5 Makefile

Makefile jest typem pliku umożliwiającym uruchamianie określonych komend, służących do kompilacji oraz testowania programów. Wymaga instalacji programu make, skrpyt przechowywany jest w Makefile. **Elementy naszego Makefile**

1. deps: pobiera pakiety pythona określone w plikach
2. lint: linter, nie testuje oprogramowania, ale znajduje potencjalne błędy(m. in. uznaje import views za niepotrzebny import)
3. test: wywołuje testy programu
4. run: uruchamia program
5. docker_build: tworzy obraz dockera o nazwie hello-world-printer w tym repozytorium na podstawie Dockerfile(omówienie w dalszej czesci)

```
1   .PHONY: test
2
3   deps:
4       pip install -r requirements.txt; \
5       pip install -r test_requirements.txt
6   lint:
7       flake8 hello_world/ test/
8   test:
9       $ PYTHONPATH=. py.test
10      $ PYTHONPATH=. py.test --verbose -s
11  run:
12      PYTHONPATH=. FLASK_APP=hello_world flask run
13
14  docker_build:
15      docker build -t hello-world-printer .
```

Rysunek 15: Makefile

5.1 Dlczego ważne jest single point of entry?

Ogólnie sprowadza się to do tego, że cała komunikacja w pogramie przechodzi przez jedno miejsce, dzięki czemu możliwa jest walidacja argumentów oraz weryfikacja uprawnień czy testowanie(te same testy są wywoływane dla różnych instancji, dodatkowa walidacja).

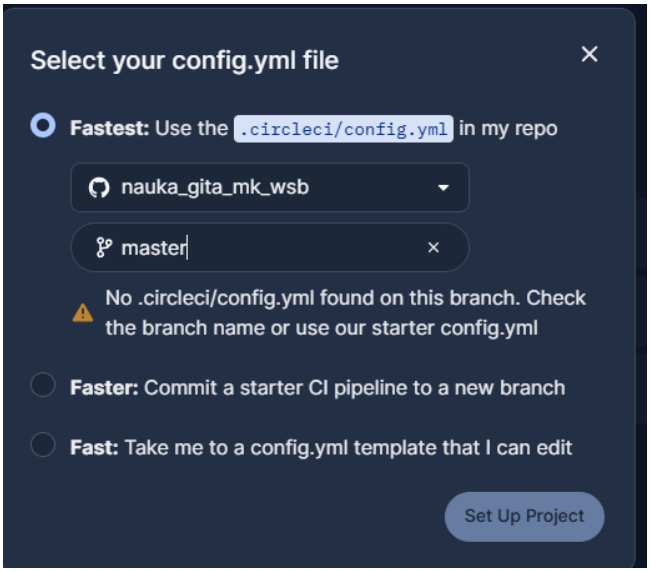
6 CirlceCI

CircleCi jest serwisem umożliwiającym automatyczne testowanie oraz buildowanie(budowanie?) aplikacji na podstawie podanego repozytorium. Jest zintegorwany m. in. z makiem(Makefile) oraz Dockerem.

6.1 YAML

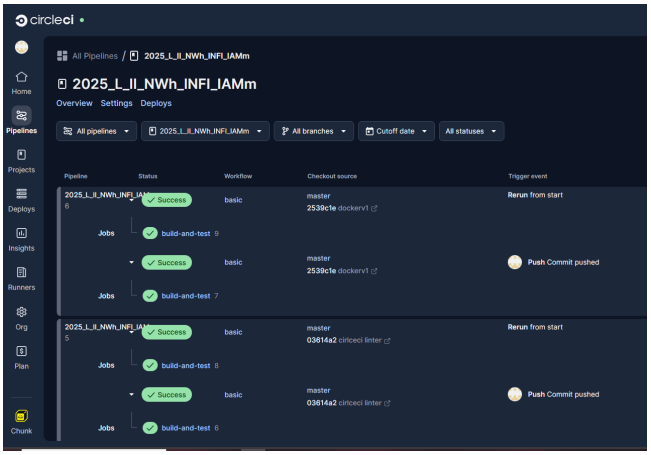
Format .yml to czytelny dla człowieka format do zapisu danych w postaci tekstowej. Używa się go do konfiguracji aplikacji

6.2 Setup

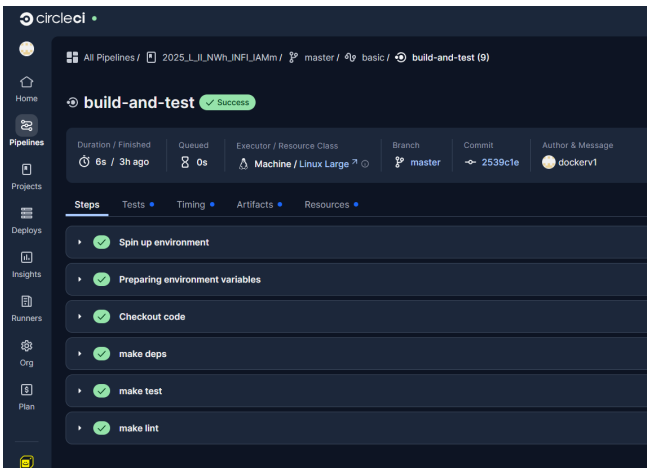


Rysunek 16: Przykładowy setup na CircleCi

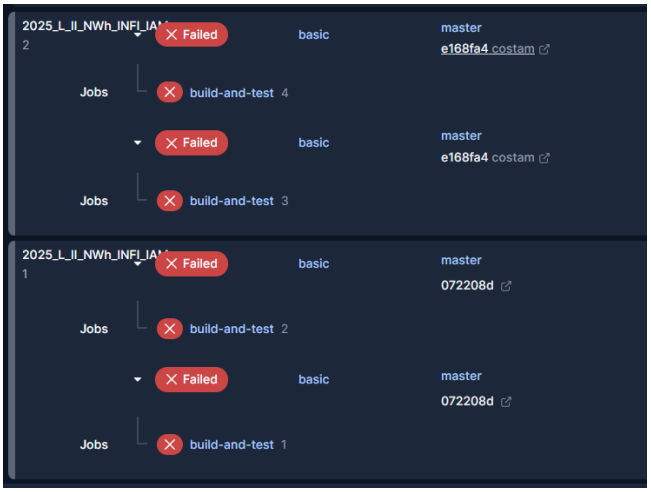
Powyżej przykładowy setup projektu na CircleCi, akurat w tym projekcie nie mam config.yml.



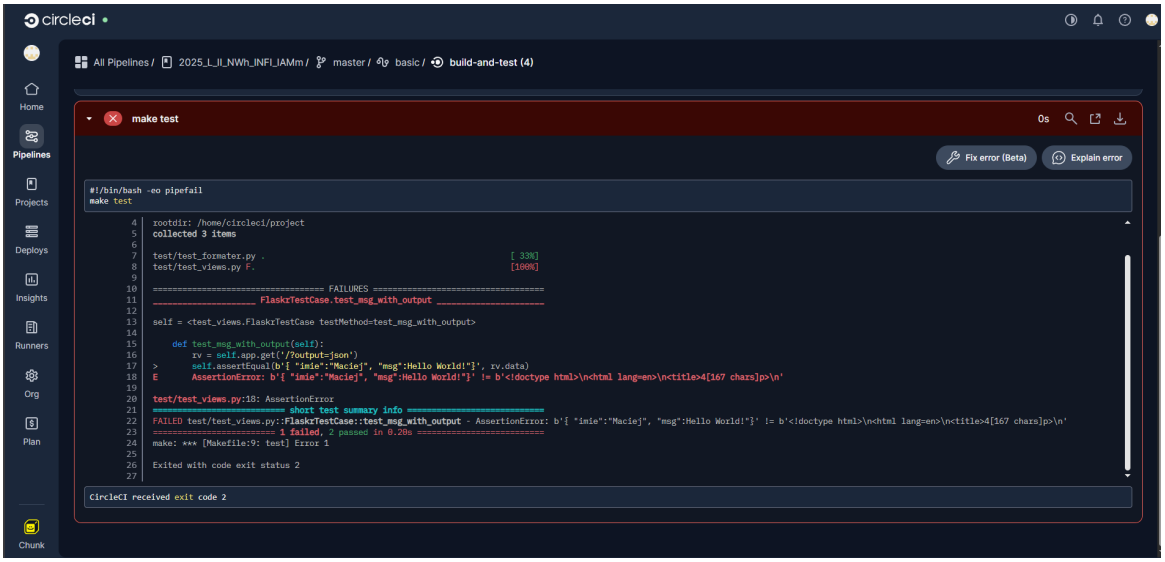
Rysunek 17: Udane testy projektu



Rysunek 18: Udane testy projektu



Rysunek 19: Nieudane testy



Rysunek 20: Logi nieudanego testu

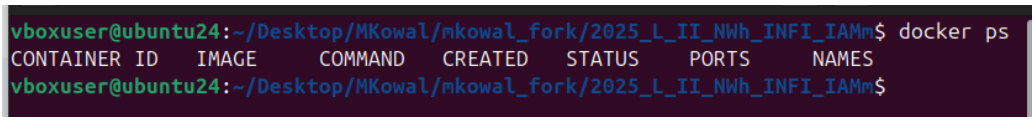
Wiele buildów nie udało się na początku z powodu złej kombinacji znaków białych w plikach Makefile oraz config.yml(znak biały po \ dodatkowa spacja przed nazwą argumentu).

7 Docker

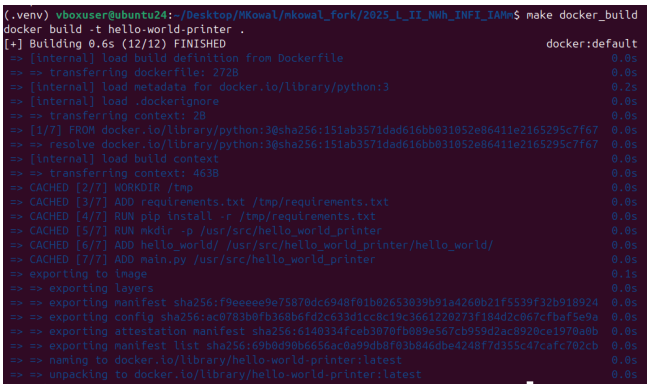
Docker to platforma do tworzenia, uruchamiania i zarządzania kontenerami. Kontener to lekka, przenośna jednostka oprogramowania, która zawiera aplikację wraz ze wszystkimi zależnościami, takimi jak biblioteki i konfiguracje, dzięki czemu działa identycznie na każdym systemie.

7.1 Dockerfile

W naszym projekcie tworzymy Dockerfile, czyli instrukcję jak zbudować obraz dockera(obraz to kopia której nie mozna zmieniać).

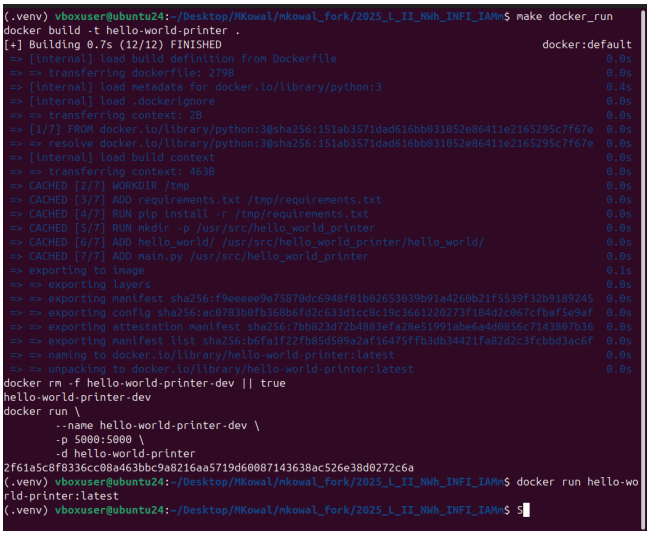


Rysunek 21: Lista kontenerów dockera



Rysunek 22: Docker build

Make docker_build poprawnie tworzy obraz dockera.

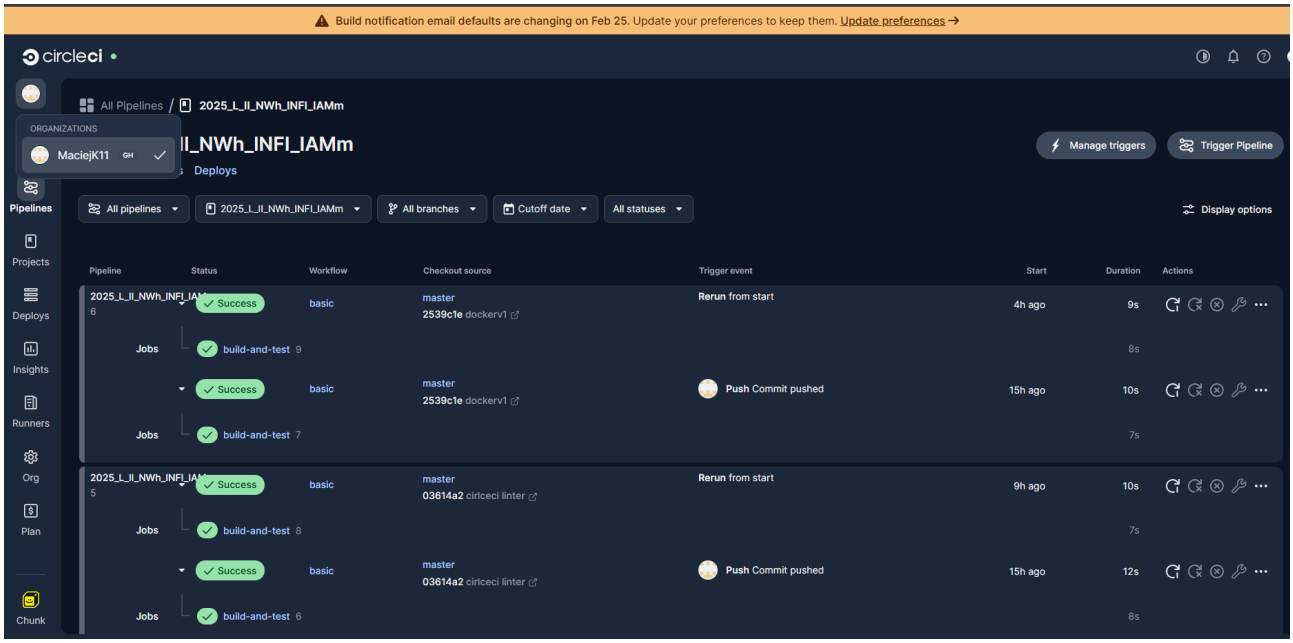


Rysunek 23: Docker nie uruchamia aplikacji

Jendak docker nie uruchamia zbudowanej aplikacji, przez co dlasza część zadania jest nie możliwa do wykonania. Próbowałem zmian znaków białych, które mogły być przyczyną błędów na początku, jednak ostatecznie wykonuje obraz tworzony jest bez błędów, co nie zmienia faktu, że cały build jest bez sensu, ponieważ nie udało mi się utworzyc działającej aplikacji.

8 Screenshoty

8.1 CircleCI

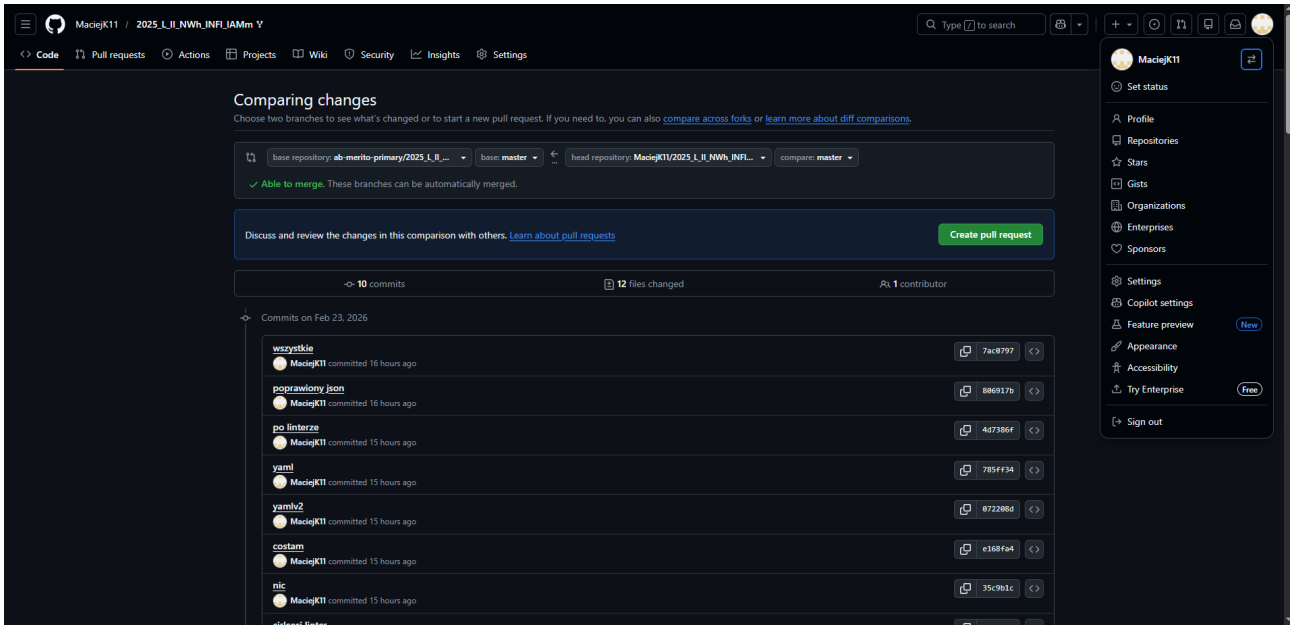


Rysunek 24: CircleCI screenshot

8.2 Dockerhub

Nie wykonano przez problemy.

8.3 Commity Github



Rysunek 25: Github screenshot