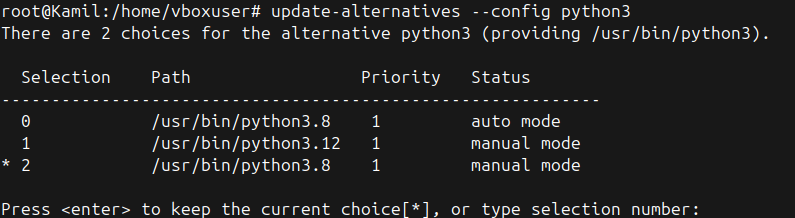
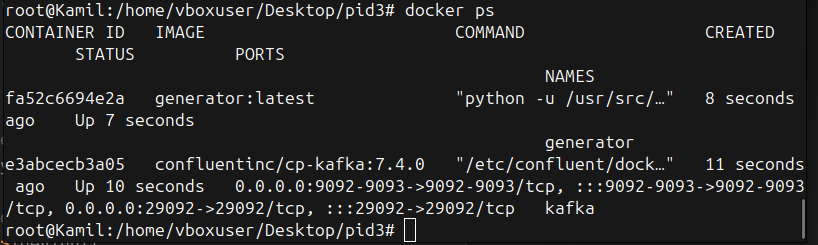
**Uruchomienie skryptu**

Uruchomienie skryptu skutkowało utworzeniem wyłącznie jednego kontenera (drugi miał status Exited). Uruchomienie skryptu generatora skutkowało błędem z jednym z modułów Kafki. Aby go naprawić zainstalowałem starszą wersję Pythona (3.8), kilkukrotnie przeinstalowałem Kafkę, zmieniłem w skrypcie folder plików, zmieniłem początkowy obraz dockerowy i edytowałem konfigurację Pythona i pakietów z nim związanych. Skrypt zaczął działać po tym jak wymusiłem instalację Kafki globalnie (łamiąc przy tym wytyczne systemu) używając wersji Pythona 3.12 (3.8 na to nie pozwalała), a potem zmianę domyślnej wersji Pythona na 3.8:



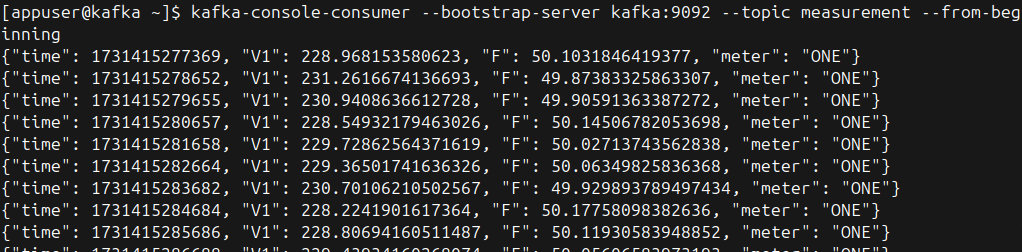


Dopiero po tym wszystkim uruchomienie generatora nie skutkowało błędem (program oczekiwał w nieskończoność), a wywołanie skryptu uruchomiło oba kontenery:



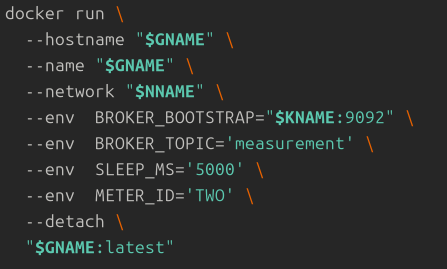
**Sprawdzenie przepływu danych**

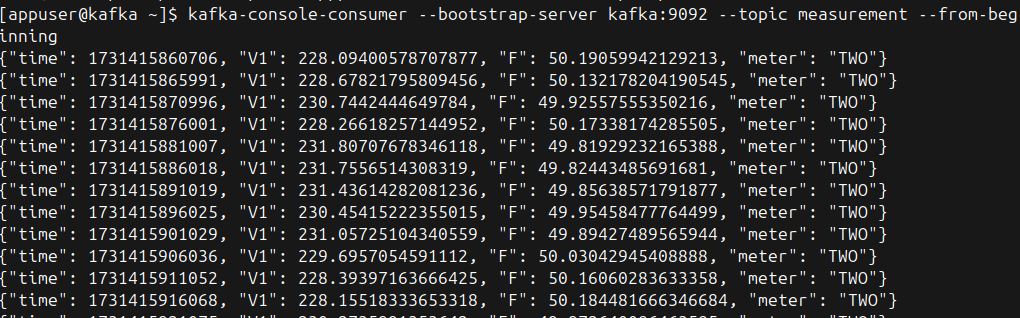
Utworzyłem konsumenta, który zgodnie z oczekiwaniami zaczął odbierać dane:



**Modyfikacja generatora**

Zmieniłem identyfikator na „TWO” oraz częstotliwość na 5 sekund. Nowe pomiary są generowane w odpowiednich interwałach:



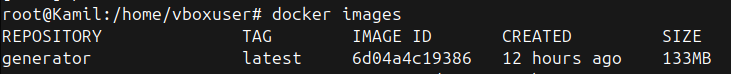


**Wytworzenie obrazu generatora**

Tworzenie obrazu generatora jest już zawarte w skrypcie:

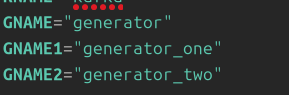


Nie było więc wymagane edytowanie kodu. Wywołanie komendy docker images:

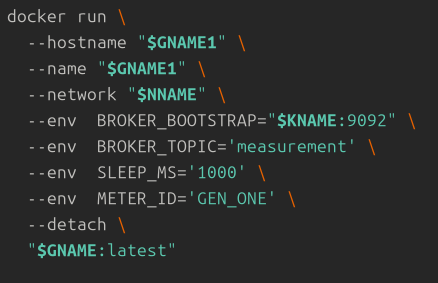
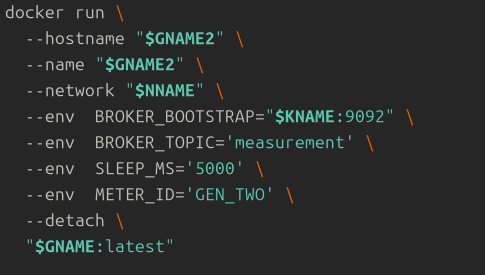


**Uruchomienie dwóch generatorów**

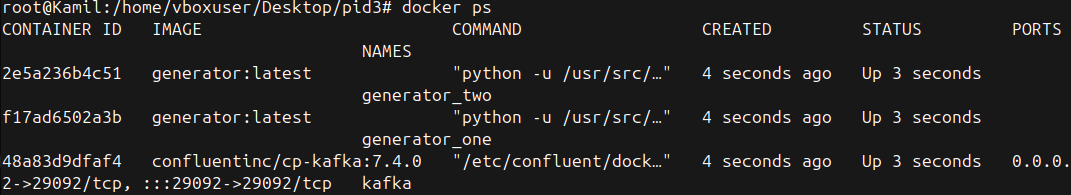
Aby uruchomić dwa generatory jednocześnie zmodyfikowałem niektóre fragmenty kodu:



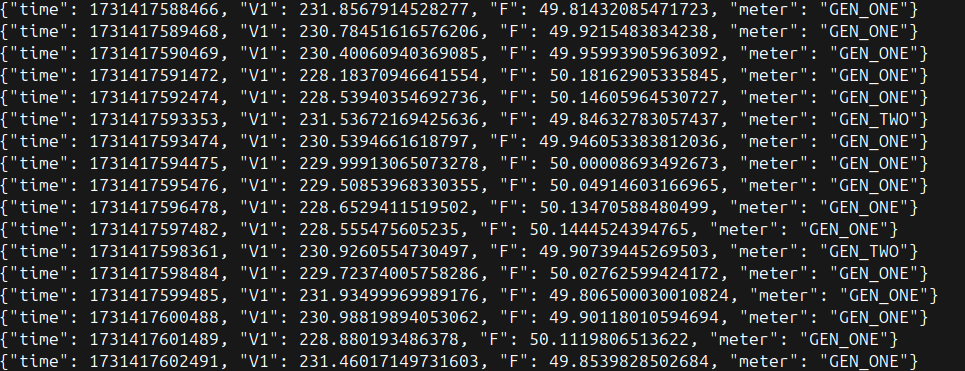
GNAME pozostało, ponieważ wciąż jest używane do tworzenia obrazu generatora.

Wywołanie skryptu poprawnie utworzyło dwa generatory z tego samego obrazu:



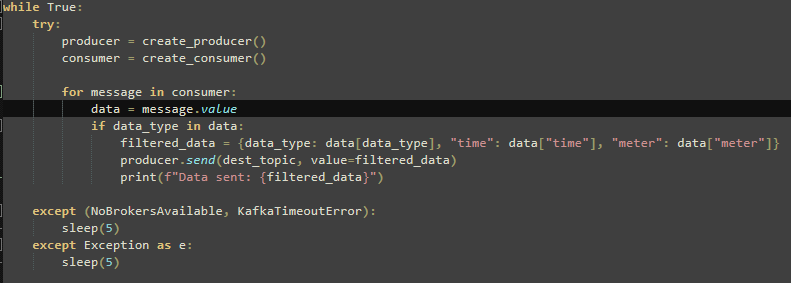
Uruchomiłem też konsumenta, by potwierdzić, że otrzymywane są komunikaty z obu generatorów:



**Zbudowanie filtru**

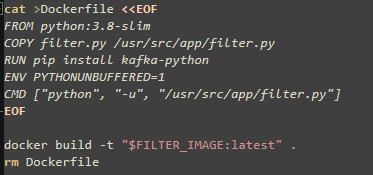
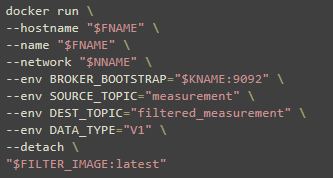
Przy tworzeniu skryptu mocno sugerowałem się generatorem. Stworzyłem konsumenta pobierającego dane z generatora i producenta, który po przefiltrowaniu tych danych wysyłał je na inny temat:



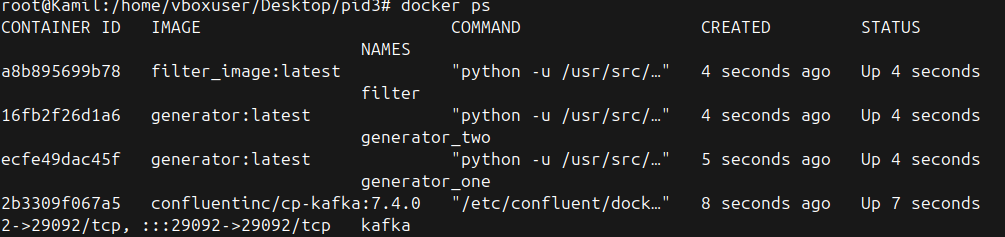


Napisałem również nowy skrypt, który tworzy również kontener wywołujący ten skrypt:



W zipie przesyłam zarówno skrypt bez filtru (po zmianach), jak i ten z filtrem (pod nazwą tiny-generator\_with\_filter.sh). Po wywołaniu poprawnie tworzone są 4 kontenery.



Utworzyłem konsumenta nasłuchującego na temacie filtered\_measurement. Filtr zadziałał poprawnie, JSON zawiera tylko jedną metrykę:

