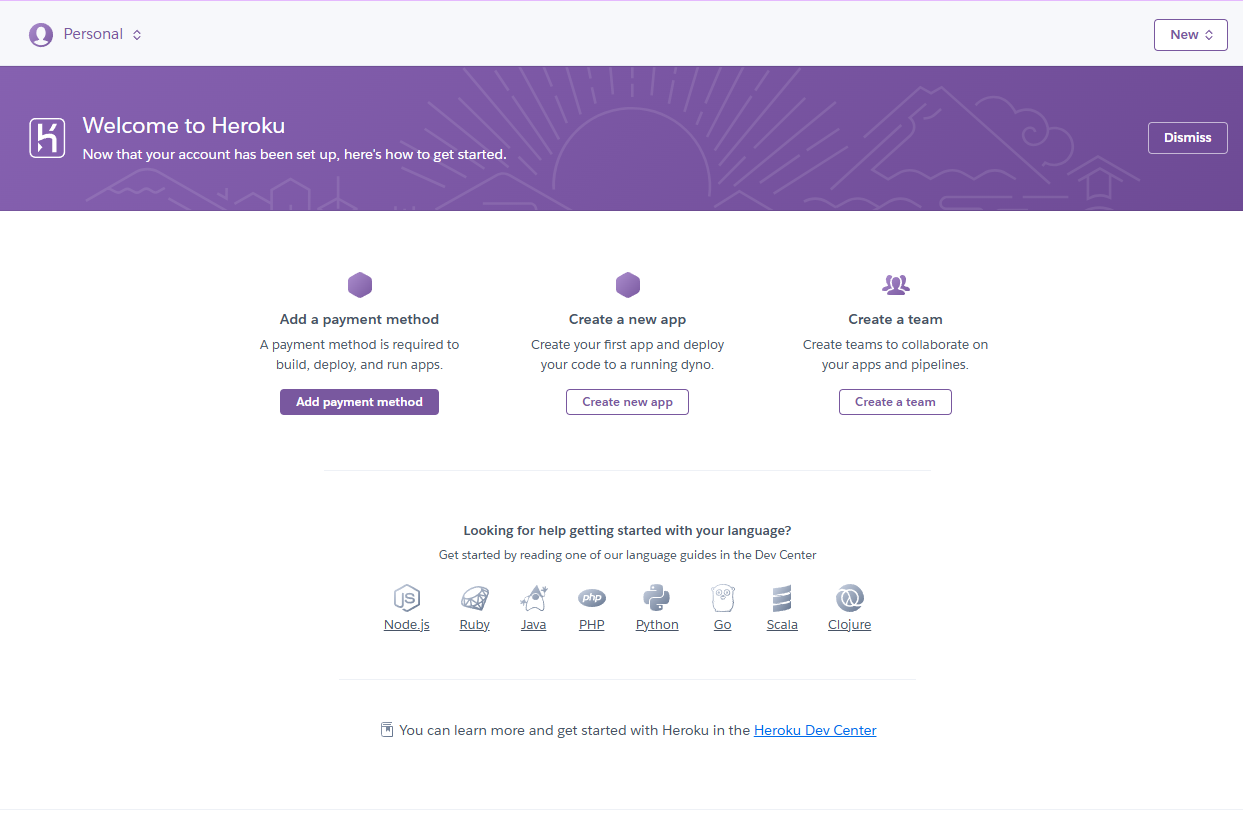
**Ustawianie aplikacji Flask na Heroku**

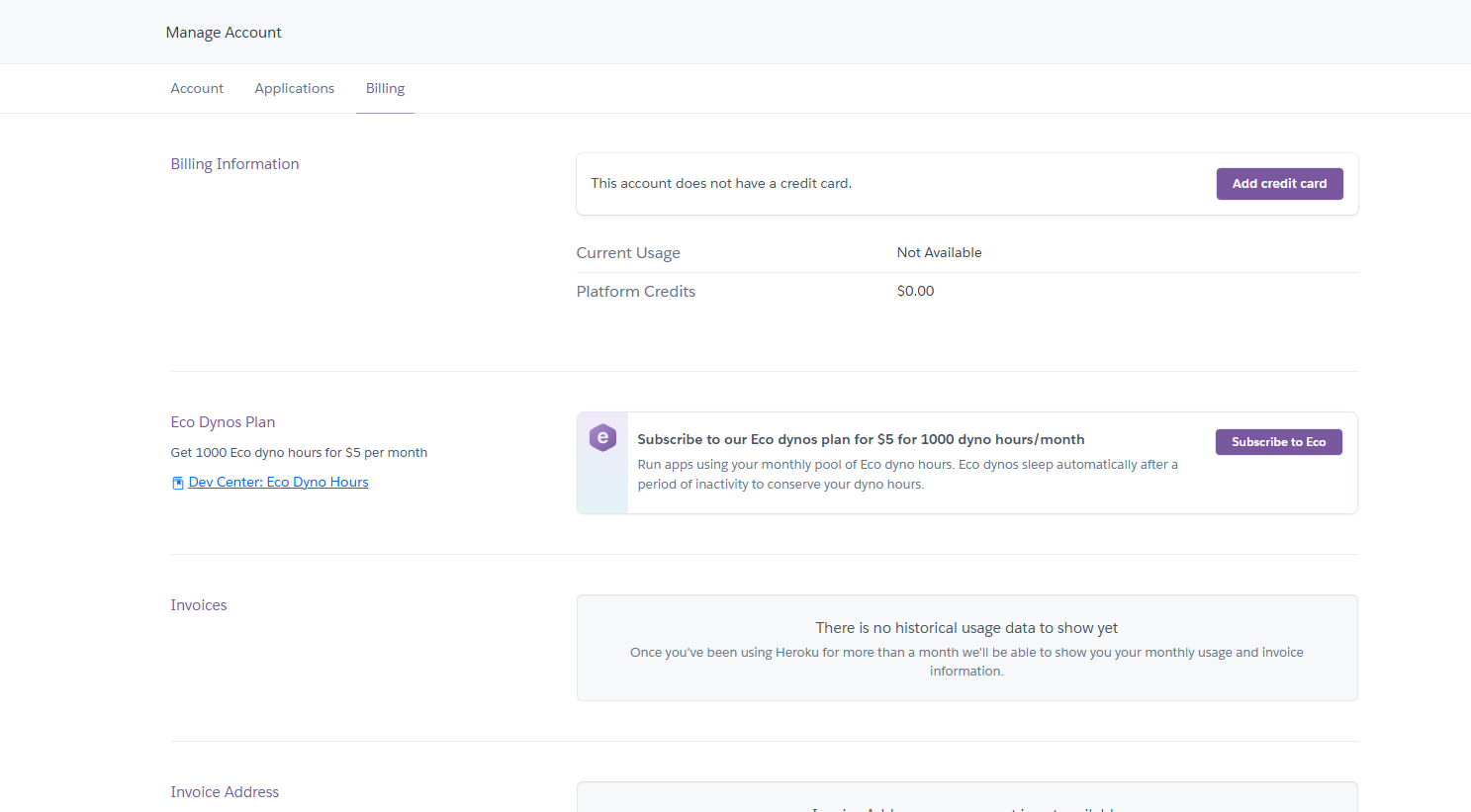
Krok 1: Tworzymy konto, jeżeli nie mamy i logujemy się do niego na stronie [Heroku](https://www.heroku.com/)

Po zalogowaniu zobaczymy ten ekran

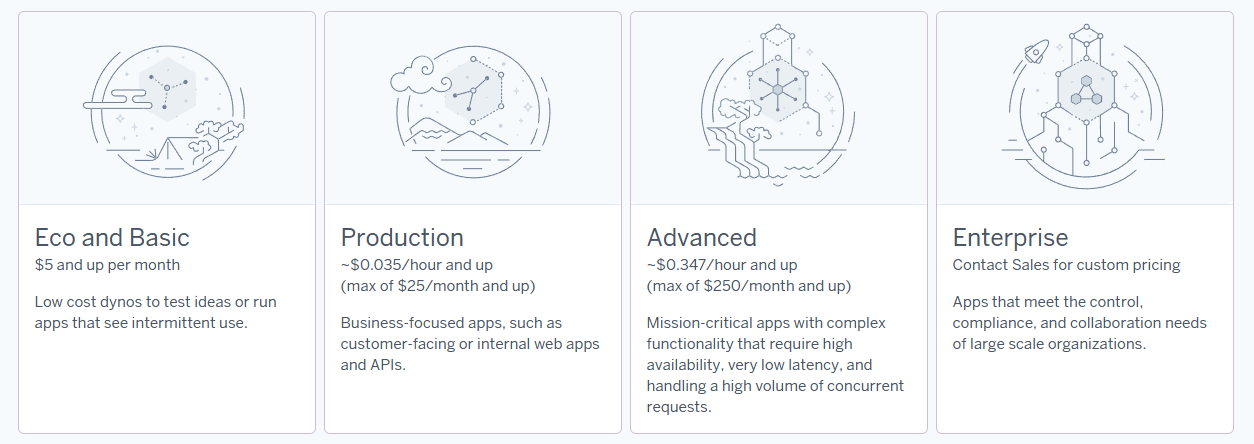


Klikamy w przycisk „Add payment method”

Krok 2: Musimy ustawić plan rozliczeń i dodać dane karty bankowej. Heroku niestety jest płatną usługą więc musimy to zrobić.



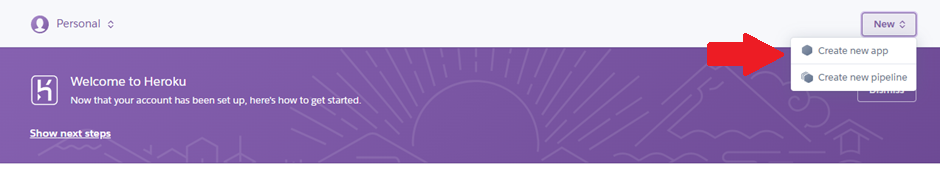
Heroku oferuje cztery różne plany



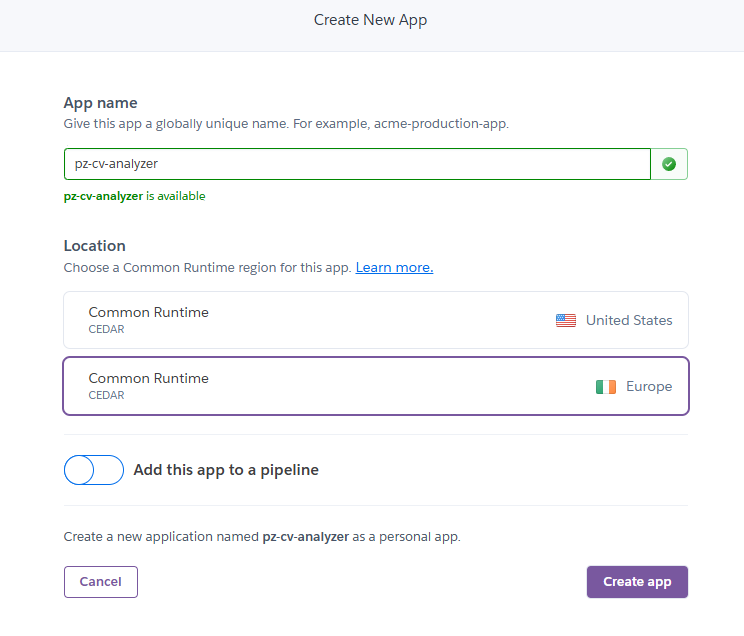
W naszym przypadku wybieramy plan Eco and Basic, jako że nie spodziewamy się dużego ruchu i nie potrzebujemy większego pakietu.

Krok 3: Tworzymy nową aplikację na Heroku:

Mamy do dyspozycji dwa sposoby na stworzenie aplikacji: pierwszy korzystając tylko z ich strony internetowej, drugi używając [Heroku CLI](https://devcenter.heroku.com/articles/heroku-cli) czyli narzędzia do wpisywania komend w terminal za pomocą którego możemy tworzyć i zarządzać naszymi aplikacjami Heroku.

Pomimo tego że będę pokazywał głównie jak postawić aplikację na Heroku poprzez ich stronę internetową, to jednak nie wszystko możemy zrobić tylko przez stronę, dlatego warto pobrać aplikację Heroku CLI.

Na stronie głównej Heroku klikamy w przycisk „New” -> „Create new app”



Otworzy nam się okno w którym możemy nazwać swój projekt i wybrać lokalizację serwera na którym będzie postawiona aplikacja. Wpisujemy nazwę aplikacji, ustawiamy serwer na Europę i naciskamy przycisk „Create app”.

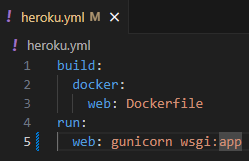
Krok 4: Przed publikacją programu na serwer musimy zrobić kilka zmian w naszym projekcie:

1. Zmienić budowę folderu z projektem, tak żeby Heroku wszystko mógł sobie wszystko znaleźć

|  |  |
| --- | --- |
| Główny folder z projektem | |
| Przed | Po |
|  |  |

Poprzez usunięcie folderu analyzer\_cv i rozłożenie wszystkich elementów do pierwszego folderu projektu Heroku poprawnie teraz będzie nam wczytywać wszystkie pliki i zależności między nimi.

1. Dodać plik heroku.yml do struktury naszego programu



Tutaj przykładowa struktura heroku.yml

Jest to plik konfiguracji dla aplikacji, używa się go do definiowania procesu budowy i uruchamiania aplikacji z wykorzystaniem Dockerfile.

Struktura struktury:

1. **build:**  
   Tutaj definiujemy, jak budować obrazy Dockera:

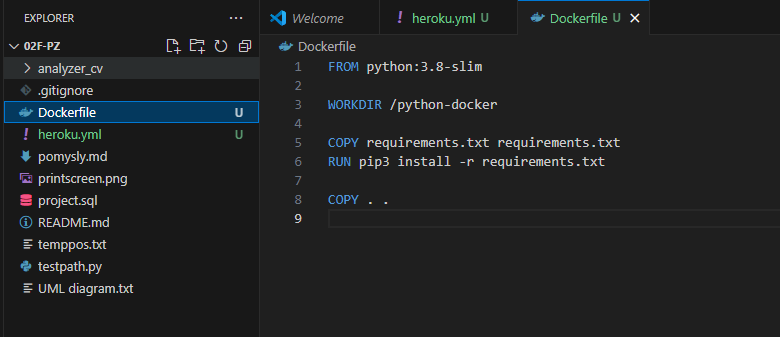
**docker:** wskazuje, że proces budowy opiera się na pliku Dockerfile.

**web:** oznacza usługę webową, która będzie budowana za pomocą tego Dockerfile.

1. **run:**  
   Sekcja określa, jak uruchomić proces po zbudowaniu obrazu:

**web:** definiuje komendę startową dla usługi webowej.  
W tym przypadku używany jest serwer aplikacji **Gunicorn**, który uruchamia aplikację Python zdefiniowaną w pliku wsgi.py

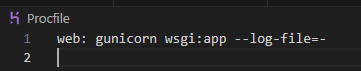
1. Dodać plik Dockerfile (bez żadnego rozszerzenia) do struktury naszego programu



**Dockerfile** – widoczny plik bez rozszerzenia, zawierający instrukcje do budowy obrazu Dockera. Kod w nim zawarty:

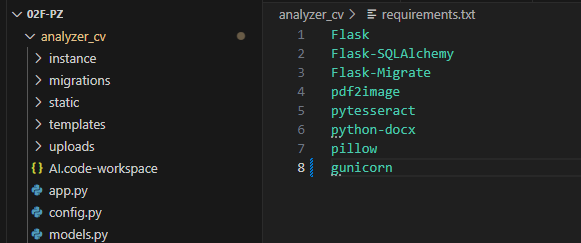
* Ustawia katalog roboczy na /python-docker.
* Kopiuje plik requirements.txt do obrazu.
* Instaluje zależności z requirements.txt za pomocą pip3.
* Kopiuje resztę zawartości projektu do obrazu Dockera.

1. Dodać plik Procfile



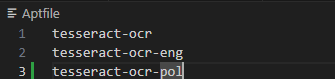
Jest to plik tekstowy używany na platformie Heroku do określenia komend, które mają być wykonywane przy starcie aplikacji, pokazuję np. jaką komendą uruchomić aplikację.

1. Dodać moduł gunicorn do requirements.txt



Teraz moduł gunicorn który służy do odpalenia serwera zostanie poprawnie zainstalowany na dockerze.

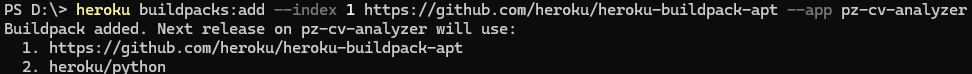
1. Ponieważ korzystamy z modułu pytesseract musimy go odpowiednio najpierw [skonfigurować](https://stackoverflow.com/questions/53363547/how-to-deploy-pytesseract-to-heroku)
   1. Dodajemy plik Aptfile do folderu głównego



Plik ten służy do określenia pakietów, które mają zostać zainstalowane do użycia przez Tesseract OCR.

* 1. W naszym wybranym terminalu, z zainstalowanym Heroku CLI, wpisujemy poniższą komendę

heroku buildpacks:add --index 1 <https://github.com/heroku/heroku-buildpack-apt> --app [nazwa aplikacji]

  
Służy to do dodawania nowego buildpacka do aplikacji na platformie Heroku. Buildpacki pozwalają określić dodatkowe zależności lub konfiguracje, np. te które mamy w pliku Aptfile.

* 1. Dodać odpowiednią ścieżkę dla zmiennej TESSDATA\_PREFIX



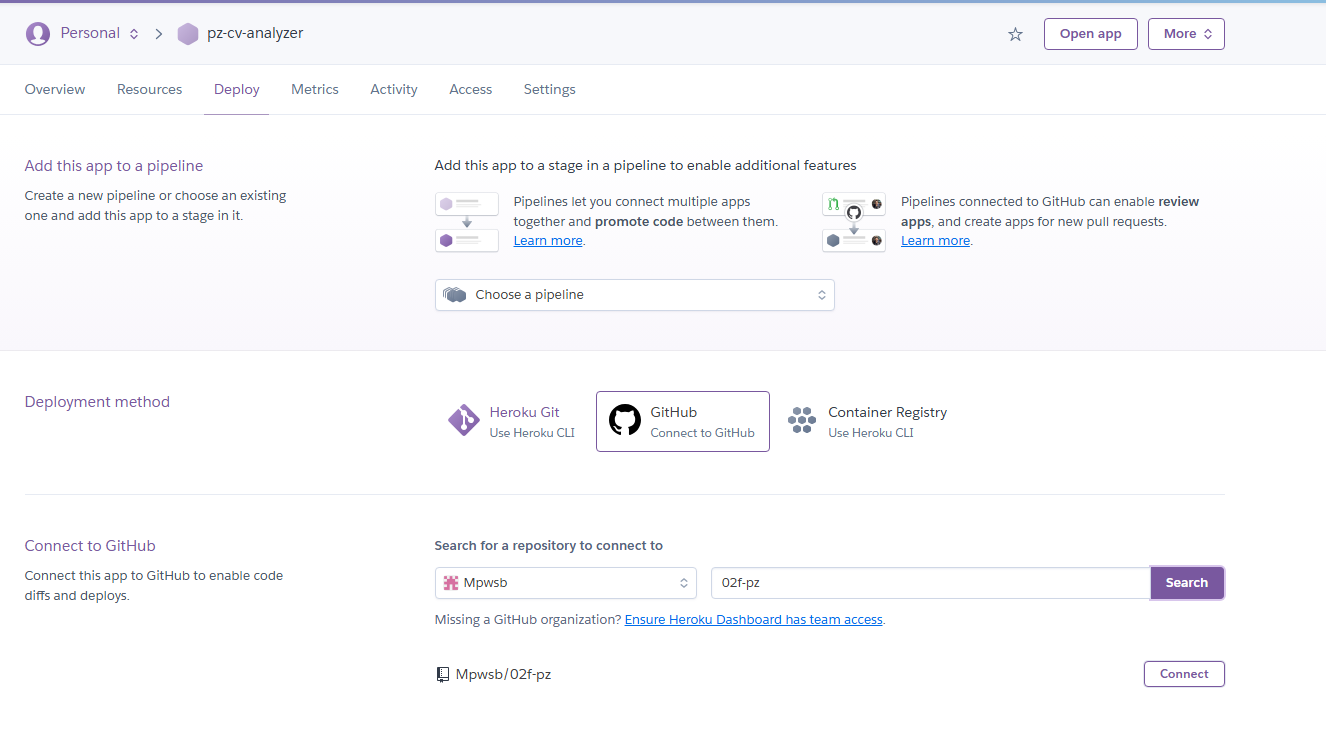
Po wdrożeniu tych zmian i dodaniu nowych plików możemy zaktualizować nasze repozytorium

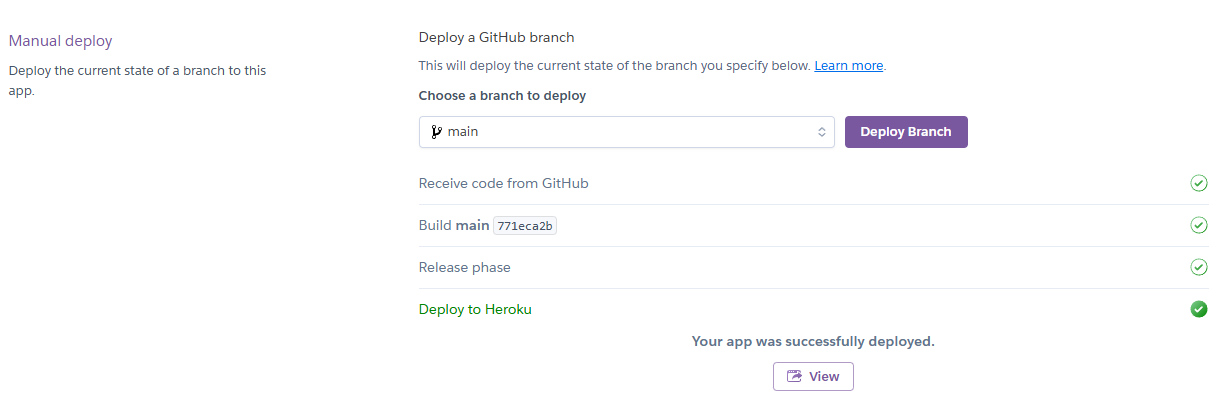
Git commit -m “heroku added”

Git push

Krok 5: W końcu możemy przejść do wdrożenia naszej aplikacji na platformie Heroku

Przechodzimy do strony z naszym projektem i w zakładce “Deploy” klikamy przycisk “Connect to GitHub”



Będziemy poproszeni o zalogowanie się do GitHuba i wybranie repozytorium które chcemy wysłać na Heroku. Następnie możemy wybrać opcję “Manual Deploy” i “Deploy Branch” żeby platforma wdrożyła project. 

Po deploymentcie powinien się wyświetlić komunikat „Your app was successfully deployed.”, po klinięciu w przycisk „View” odpali się nasza aplikacja.

