Deployment

Zajęcia Prymus 2020

Agnieszka Rudnicka, rudnicka@agh.edu.pl

Publikujemy naszą aplikacje

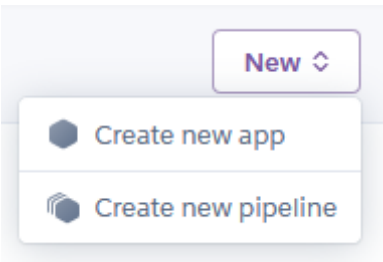
Aby nasze dzieło mogło zostać pokazane światu musimy umieścić je na jakimś serwerze. Oczywiście można by hostować stronę na własnym komputerze, ale w praktyce raczej nikt tego nie robi :) Jest wiele usług, które się tym specjalizują i to im warto powierzyć to zadanie.

Jeśli ktoś nie posiada konta na Heroku, to dobry moment, żeby takowe założyć: https://signup.heroku.com/

Można też zainstalować Heroku CLI: https://devcenter.heroku.com/articles/getting-started-with-python#set-up Przyda się między innymi do założenia użytkownika na zdalnym serwerze i logowania na niego, podglądania logów itp.

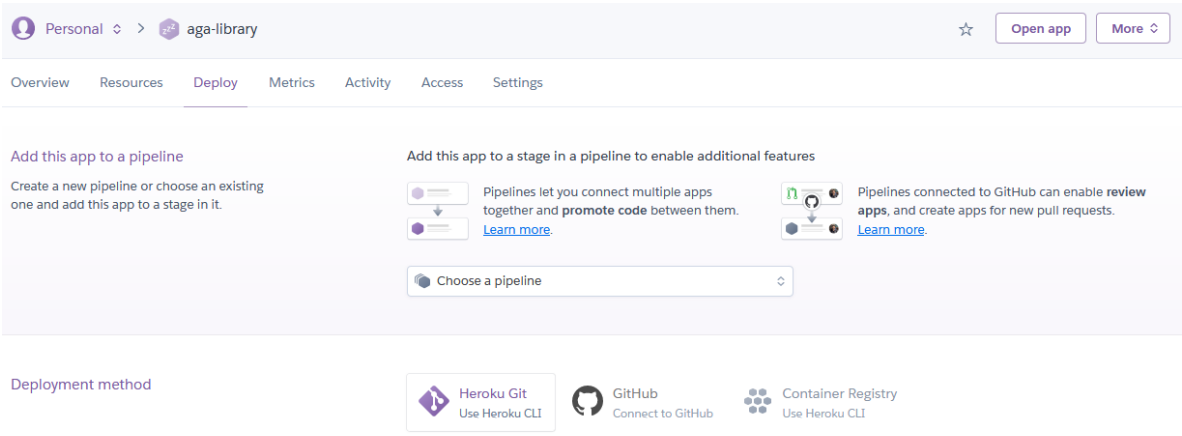
Tworzenie aplikacji na Heroku

Logujemy się do Heroku i dodajemy nową aplikację (https://dashboard.heroku.com/new-app).

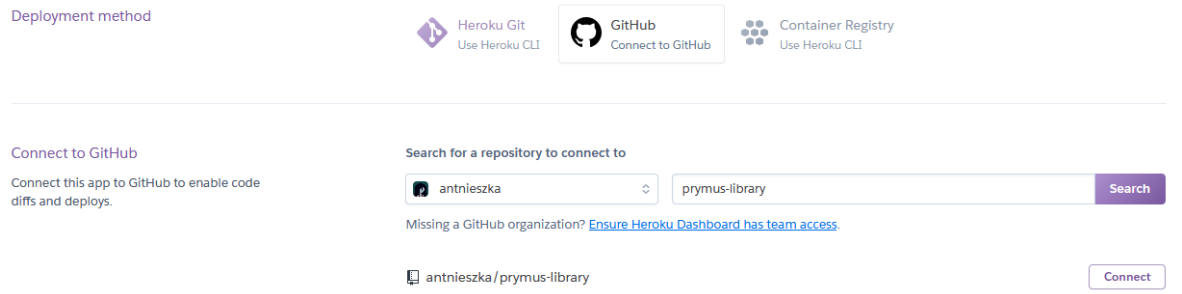


Następnie wpisujemy nazwę naszej aplikacji (domyślnie będzie to część adresu URL `nazwa-appki.heroku.com`). Jako region wybieramy oczywiście Europę.

następnie zostajemy przeniesieni do ekranu konfiguracji naszej nowo powstałej aplikacji:



Tutaj korzystając z integracji z GitHubem, wybieramy nasze repozytorium z kodem i klikamy `Connect`.



Dzięki tej operacji Heroku pozwoli nam na włączenie automatycznego wdrażania jak i manualnego. Musimy jedynie wybrać gałąź repozytorium i kliknąć przycisk.

Pierwszy deployment

Spróbujmy na dobry początek wykonać manualne wdrożenie klikając `Deploy Branch` w sekcji `Manual deploy`.

Jeśli konsolka wypisała poniższe, to jesteście o krok bliżej sukcesu.

```
-----> Python app detected
-----> Installing python-3.6.10
-----> Installing pip
-----> Installing SQLite3
-----> Installing requirements with pip

...
       django.core.exceptions.ImproperlyConfigured: You're using the staticfiles app
without having set the STATIC_ROOT setting to a filesystem path.
 !     Error while running '$ python manage.py collectstatic --noinput'.
       See traceback above for details.
```

```
       You may need to update application code to resolve this error.
       Or, you can disable collectstatic for this application:
          $ heroku config:set DISABLE_COLLECTSTATIC=1
       https://devcenter.heroku.com/articles/django-assets
 !     Push rejected, failed to compile Python app.
 !     Push failed
```

Środowisko jeżyka Python zostało automatycznie wykryte poprzez obecność pliku `requirements.txt` i nastąpiła automatyczna instalacja zależności, w tym frameworka Django, w której napisana jest nasza aplikacja.

Pliki statyczne

Jednak summa summarum wdrożenie zakończyło się błędem. Jeśli dobrze przyjrzymy się ostatnim linijkom błędu dostrzeżemy podpowiedź:

```
django.core.exceptions.ImproperlyConfigured: You're using the staticfiles app without
having set the STATIC_ROOT setting to a filesystem path.
```

Wygląda na to, że nie ustawiliśmy w `setting.py` czegoś co się nazywa `STATIC_ROOT`. Wróćmy więc do kodu naszego projektu i w pliku settings.py, na samym końcu dopiszmy brakującą ścieżkę do plików statycznych.

Skorzystamy też z biblioteki `whitenoise` do serwowania plików statycznych (https://devcenter.heroku.com/articles/django-assets#whitenoise). A w zasadzie z biblioteki `django_heroku`, która nam sama skonfiguruje `whitenoise` i inne rzeczy pod Heroku :)

https://devcenter.heroku.com/articles/django-app-configuration#settings-py-changes

Zainstalujemy więc ten dodatek:

```
pip install django-heroku
```

Do uruchomienia aplikacji na Heroku potrzebujemy też serwera Green Unicorn `gunicorn`.

Dodajmy do pliku `requirements.txt`:

```
asgiref==3.2.7
Django==3.0.5
pytz==2019.3
sqlparse==0.3.1

gunicorn  # NEW
django-heroku  # NEW
```

Na samym początku pliku `settings.py` dodajmy linijkę:

```
import django_heroku
```

A na samym końcu:

```
# Static files (CSS, JavaScript, Images)
# https://docs.djangoproject.com/en/3.0/howto/static-files/

STATIC_URL = '/static/'

# Activate Django-Heroku.
django_heroku.settings(locals())  # NEW
```

Reasumując, django-heroku:

1. Skonfiguruje odpowiednio ustawienia: `STATIC_ROOT` oraz `STATICFILES_STORAGE` używając biblioteki `whitenoise` aby można było wdrożyć aplikację na Heroku bez dodatkowego wysiłku.
2. Zmieni `ALLOWED_HOSTS` w `settings.py`, aby pozwalało na dostęp z naszej zdalnej domeny
3. Zainstaluje i obsłuży bazę danych PostgreSQL, która jest solidnym rozwiązaniem produkcyjnym
4. ... i kilka innych :)

Definiowanie aplikacji w Heroku

Po wpisaniu `django heroku` w Google szybko trafimy na stronę: https://devcenter.heroku.com/articles/django-app-configuration

W sekcji **The basics** dowiadujemy się, że każda aplikacja na Heroku wymaga pliku `Procfile`.

Musimy w nim umieścić następujące:

```
release: python manage.py migrate
web: gunicorn libraryproj.wsgi
```

Pamiętajmy dodać `gunicorn` do pliku `requirements.txt` jeśli go jeszcze nie mamy. Uwaga, mój katalog z plikiem `wsgi.py` nazywa się `libraryproj`, trzeba tutaj wpisać nazwę swojego!

Zróbmy to więc, a następnie ponownie git add/commit/push i deploy na Heroku :)

Appendix

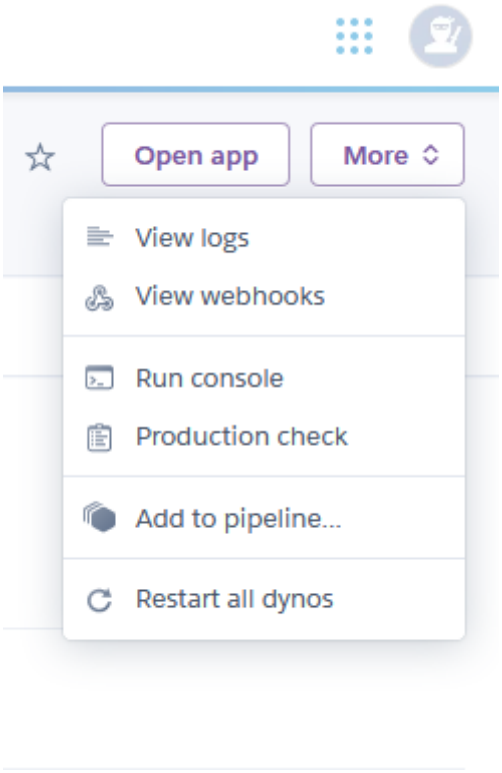
Logowanie do Heroku CLI

Poniższe polecenie otworzy w przeglądarce okno logowania, a następnie uwierzytelni terminal do wykonywania operacji na serwerze :)

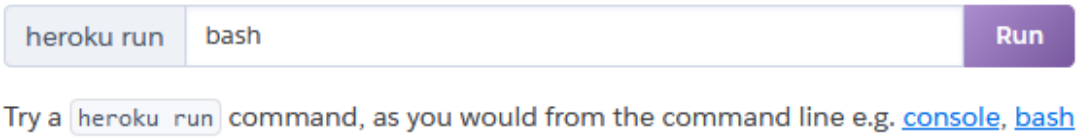
```
heroku login
```

Konsola przez przeglądarkę

Jeśli nie chcemy instalować dodatkowego oprogramowania możemy przetestować konsolę dostępną przez przeglądarkę. W widoku naszej aplikacji, w prawym górnym roku powinna być dostępna opcja `More`.

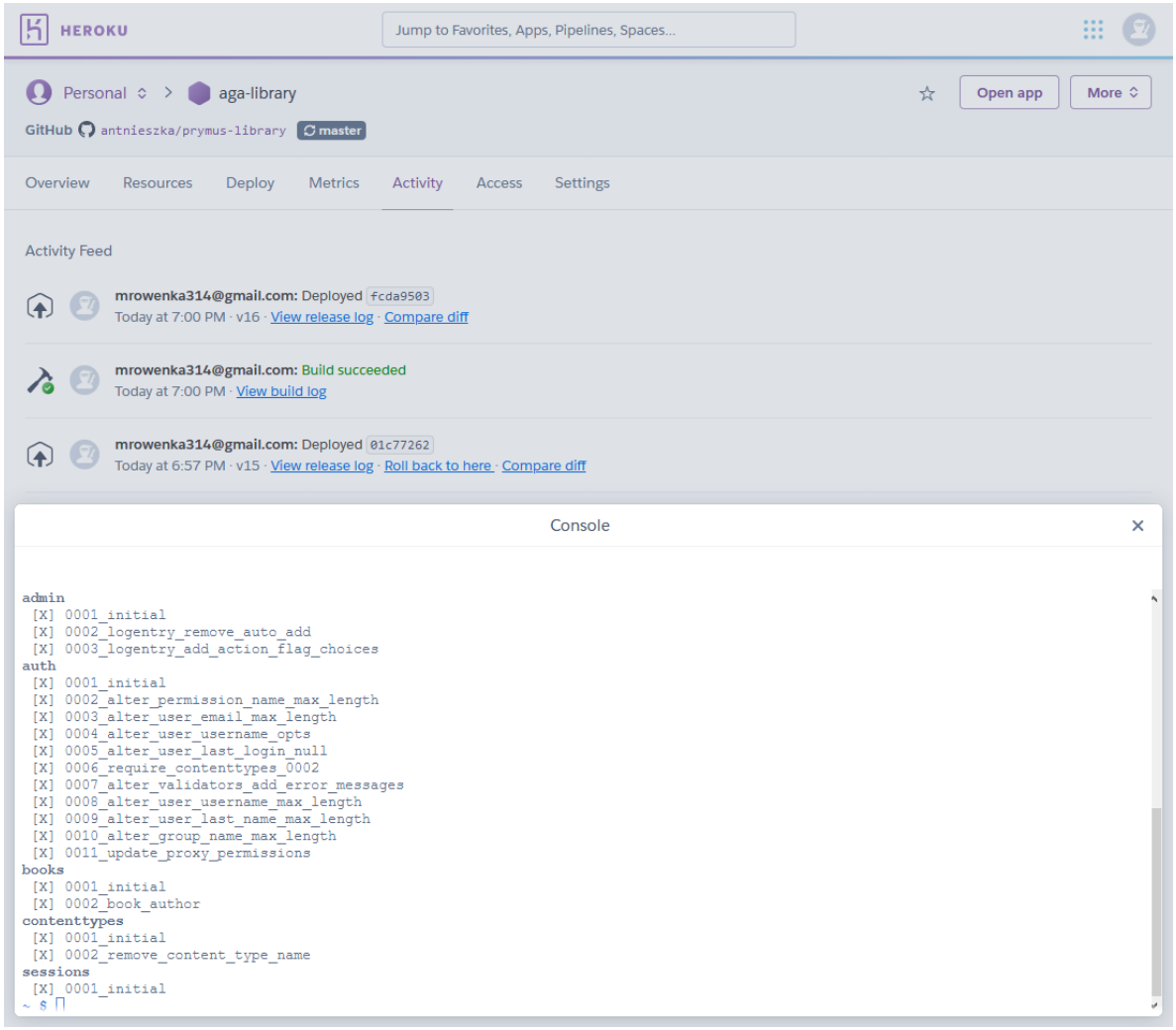


Można tutaj wybrać `Run console` a następnie wpisać `bash`.



Try a `heroku run` command, as you would from the command line e.g. console, bash

Dzięki temu możemy wykonywać polecenia takie jak (`python manage.py showmigrations`) pokazane poniżej.



Tworzenie super-użytkownika

Podobnie jak w przypadku lokalnej instalacji potrzebny nam będzie użytkownik administracyjny. Tutaj sprawa się komplikuje, bo musimy albo podłączyć się jakoś do bazy danych, albo wykonać polecenie na serwerze. Łatwiej jest zrobić to drugie, z wykorzystaniem Heroku CLI to jedno polecenie:

```
heroku run python manage.py createsuperuser
```

Teraz powinniśmy mieć możliwość zalogować się do panelu administracyjnego na naszej wdrożonej stronie!

PS: czasem może być wymagane podanie nazwy swojej aplikacji:

```
heroku run -a prymus-django python manage.py createsuperuser
```

Dodatkowe źródła

- https://devcenter.heroku.com/articles/django-app-configuration
- https://devcenter.heroku.com/articles/release-phase#specifying-release-phase-tasks
- https://tutorial-extensions.djangogirls.org/en/heroku/
- https://devcenter.heroku.com/articles/django-app-configuration#settings-py-changes

Last resort

Gdyby deployment dalej nie działał można porównać zmiany w kodzie dostępne pod tym linkiem na GitHubie: https://github.com/antnieszka/prymus-library/compare/django-lab-2...django-lab-3