

Ask Django

Heroku 배포

알립니다.

- 본 자료는 2017년 11월 15일 버전의 Heroku 공식문서를 참고했습니다.
- 후에 본 자료를 참고하실 경우, Heroku 최신변경내역을 반영하지 못할 수도 있습니다.

Heroku #Pricing

AWS 플랫폼에서 서비스되고 있는 PaaS 플랫폼

- Free
 - 최대 5개 앱
 - 550시간/월 제공 + (신용카드 정보 등록시) 450시간 추가 지원
 - US/EU 지역에 대해서만 인스턴스 생성 지원
 - 30분동안 수신 트래픽이 발생하지 않으면 Sleep ^{#ref}
- 이 외 : Hobby (\$7/월), Standard/Performance (\$25~\$500/월 플랜)
- VCS통해 관리되지 않는 파일은 재배포시에 삭제됩니다.

배포할 내용

- 장고는 Heroku 상에서 gunicorn을 통한 실행
- STATIC 파일은 장고에서 서빙 (비추)
 - AWS S3/CDN을 활용하는 것이 좋습니다.
- 데이터베이스는 Heroku 기본 지원 PostgreSQL 데이터베이스

장고 프로젝트 배포 기본 설정 #Heroku 공식문서

runtime.txt 파일 생성 #지원하는파이썬버전

- 지원하는 파이썬 버전
 - python-2.7.14
 - **python-3.6.2** : 파이썬3를 쓰겠습니다.

```
셸> cat runtime.txt  
python-3.6.2
```

Procfile 파일 생성

gunicorn을 통한 기본옵션 실행

```
web: gunicorn askdjango.wsgi
```

프로젝트/wsgi.py 를 gunicorn을 통해 실행하려 합니다.

Heroku 필수 패키지

- `dj-database-url`
 - Heroku에서는 `DATABASE_URL` 환경변수를 통해 `DATABASE` 접속정보를 제공합니다.
 - 이를 파싱하여 `dict`으로 변환해줍니다.
- `gunicorn` 혹은 `uwsgi`
 - Python WSGI HTTP Server
- `psycopg2`
 - Heroku에서 PostgreSQL 지원

다음과 같이 **requirements.txt** 구성해봅시다.

Heroku에서는 requirements.txt 을 찾아서 파이썬 패키지를 설치합니다. 전 reqs/prod_heroku.txt 경로에 다음과 같이 정의하고, requirements.txt가 이 파일을 바라보게 만들 것입니다. 필요한 패키지가 있다면 추가해주시면 됩니다.

```
dj-database-url  
django<2  
gunicorn  
psycpg2  
pillow
```

`dj-database-url` 활용 예시

```
>>> import dj_database_url

>>> dj_database_url.parse('postgres://u-----:w-----@ec2-111-222-333-444.'
                        'compute-1.amazonaws.com:5431/d8r82722r2kuvn')

{'CONN_MAX_AGE': 0,
 'ENGINE': 'django.db.backends.postgresql_psycopg2',
 'HOST': 'ec2-111-222-333-444.compute-1.amazonaws.com',
 'NAME': 'd8r82722r2kuvn',
 'PASSWORD': 'w-----',
 'PORT': 5431,
 'USER': 'u-----'}
```

dj-database-url 활용토록 settings 편집

askdjango/settings/prod_heroku.py 에 추가

장고 애플리케이션에는 hobby-dev 데이터베이스가 자동제공됩니다¹.
데이터베이스 접근정보는 DATABASE_URL 환경변수를 통해 제공됩니다.

```
import dj_database_url
db_from_env = dj_database_url.config(env='DATABASE_URL', conn_max_age=500)

# 기존 DATABASES
DATABASES['default'].update(db_from_env)
```

¹ [Heroku Postgres Price Tier](#) : 가격정보

외부 웹서버 정적인 파일 서빙을 쓰지 않는다면 간단하게 **static/media** 파일을 장고에서 직접 서빙토록 설정하기

```
# askdjango/urls.py 끝에 추가
from django.conf import settings
from django.views import static

static_list = [
    (settings.STATIC_URL, settings.STATIC_ROOT),
    (settings.MEDIA_URL, settings.MEDIA_ROOT),
]

for (prefix_url, root) in static_list:
    if '://' not in prefix_url: # 외부 서버에서 서빙하는 것이 아니라면
        prefix_url = prefix_url.lstrip('/')
        url_pattern = r'^' + prefix_url + r'(?P<path>.+)'
        pattern = url(url_pattern, static.serve, kwargs={'document_root': root})
        urlpatterns.append(pattern)
```

배포를 수행해봅시다.

다양한 배포 방법 #공식문서

- **Git을 통한 직접 배포** #공식문서
- Github를 통한 배포 #공식문서
- Dropbox Sync #공식문서

Git을 통한 직접 배포

로컬에 **git** 저장소 생성

장고 프로젝트 최상위 경로에서 아래 명령을 통해 **git** 저장소 생성 및 커밋

CLI 명령어로는 다음과 같이 하실 수 있습니다.

윈도우에서는 *Heroku CLI*를 설치하면, *git CLI* 프로그램이 포함되어 있습니다.

```
git init
git add .
git commit -m "initial commit"
```

GUI 프로그램으로는 Github Desktop이나 Source tree를 추천합니다.

Tip: 프로젝트에 루트에 .gitignore 파일 생성

최소한 다음 3가지를 포함시켜주시면, 불필요한 파일을 관리 대상에서 제외하실 수 있습니다.

__pycache__

*.pyc

db.sqlite3

Heroku CLI 설치 #공식문서

- 윈도우
 - [32비트 Installer](#)
 - [64비트 Installer](#)
- 맥
 - [Installer](#)를 통해 설치
 - Homebrew를 통한 설치 : 셸> `brew install heroku/brew/heroku`
- Ubuntu
 - 셸> `wget -qO- https://cli-assets.heroku.com/install-ubuntu.sh | sh`
 - wget 프로그램이 설치되어있지 않다면, `sudo apt-get install wget` 명령으로 설치해주세요.

이 외 운영체제는 위 공식문서를 참고하세요.

Heroku CLI 설치/버전 확인

셸> `heroku --version`

`heroku-cli/6.14.39-adddc925 (darwin-x64) node-v9.2.0`

출력되는 내용은 운영체제/버전에 따라 다를 수 있습니다.

이제, **Heroku** 계정이 필요합니다.

Free 계정에 가입해주세요.

Heroku CLI 툴을 통한 인증 수행

```
셸> heroku login
```

```
Enter your Heroku credentials.
```

```
Email: 헤로쿠_계정_이메일주소_입력
```

```
Password: 암호_입력
```


```
Logged in as me@nomade.kr
```

```
. . .
```

heroku app 생성

장고 프로젝트 최상위 경로에서 다음 명령을 통해 Heroku app 생성

```
셸> heroku apps:create 프로젝트명
```

```
Creating  프로젝트명... done
```

```
https://프로젝트명.herokuapp.com/ | https://git.heroku.com/프로젝트명.git
```

웹브라우저로 생성된 <https://프로젝트명.herokuapp.com/> 주소로 접속하실 수 있습니다.
아직 장고 코드는 배포하지 않았습니다.

필요한 환경변수 설정하기

```
셸> heroku config:set DJANGO_SETTINGS_MODULE=askdjango.settings.prod_heroku
```

사용하실 환경변수를 설정해주세요.

해당 Heroku app 배포 수행

```
셸> git push heroku master

Counting objects: 25, done.
Delta compression using up to 4 threads.
Compressing objects: 100% (21/21), done.
Writing objects: 100% (25/25), 3.86 KiB | 1.93 MiB/s, done.
Total 25 (delta 8), reused 0 (delta 0)
remote: Compressing source files... done.
remote: Building source:
remote:
remote: -----> Python app detected
remote: -----> Installing python-3.6.3
remote: -----> Installing pip
remote: -----> Installing requiremets with pip
remote:      Collecting gunicorn (from -r /tmp/build_ef0d1e6373956db17c1932280d9602f6/requirements.txt (line 1))
remote:      Downloading gunicorn-19.7.1-py2.py3-none-any.whl (111kB)
remote:      Collecting django (from -r /tmp/build_ef0d1e6373956db17c1932280d9602f6/requirements.txt (line 2))
remote:      Downloading Django-2.0-py3-none-any.whl (7.1MB)
remote:      Collecting pytz (from django->-r /tmp/build_ef0d1e6373956db17c1932280d9602f6/requirements.txt (line 2))
remote:      Downloading pytz-2017.3-py2.py3-none-any.whl (511kB)
remote:      Installing collected packages: gunicorn, pytz, django
remote:      Successfully installed django-2.0 gunicorn-19.7.1 pytz-2017.3
remote:
remote: -----> $ python manage.py collectstatic --noinput
remote:      118 static files copied to '/tmp/build_ef0d1e6373956db17c1932280d9602f6/static'.
remote:
remote: -----> Discovering process types
remote:      Procfile declares types -> web
remote:
remote: -----> Compressing...
remote:      Done: 48.1M
remote: -----> Launching...
remote:      Released v4
remote:      https://askdjango-heroku-demo.herokuapp.com/ deployed to Heroku
remote:
remote: Verifying deploy... done.
To https://git.heroku.com/askdjango-heroku-demo.git
 * [new branch]      master -> master
```

주의 : 배포 중에 collectstatic 명령을 수행하므로, settings.STATIC_ROOT설정이나 다른 static 관련 설정이 꼭 필요합니다.

페이지에 접속해보세요. :D

- 페이지 접속이 잘 되시나요?
- 데이터베이스 migrate가 필요하진 않으신가요? :D

필요한 장고 **manage** 명령을 **Heroku** 인스턴스 상에서 수행할 수 있습니다.

다음 명령으로 Heroku shell을 띄웁니다.

```
로컬 셸> heroku run bash
```

이제 입력하는 명령은 Heroku 인스턴스에서 실행됩니다. 필요한 명령을 입력해주세요.

```
Heroku 셸> python --version
Heroku 셸> python manage.py --version
Heroku 셸> python manage.py showmigrations
Heroku 셸> python manage.py migrate
Heroku 셸> python manage.py createsuperuser
```

혹은 다음과 같이, 로컬 셸에서 직접적으로 명령을 실행할 수 있습니다.

```
로컬 셸> heroku run -- python manage.py showmigrations
```

그 외 명령

- Heroku App 삭제: 쉘> `heroku apps:delete`
- 로그 확인: 쉘> `heroku logs --tail`
- 도메인 설정: 쉘> `heroku domains`

이 외에도 수많은 명령이 있습니다.

A person wearing a white tank top and a straw hat stands with their back to the camera, arms raised in a 'V' shape. They are overlooking a dense cityscape with many buildings and a body of water in the distance under a clear blue sky.

*Life is short,
use Python3/Django.*