

Deployment of Python Project

Deployment?

- Involves packaging up and putting it in a production environment
- Why is deployment necessary?
 - Application must live somewhere other than desktop or laptop.
 - A production environment is the canonical version of application

Deployment hosting options

- There are four options for deploying and hosting a web application:
 - "Bare metal" servers
 - Virtualized servers
 - Infrastructure-as-a-service
 - Platform-as-a-service

Deployment tools and resources

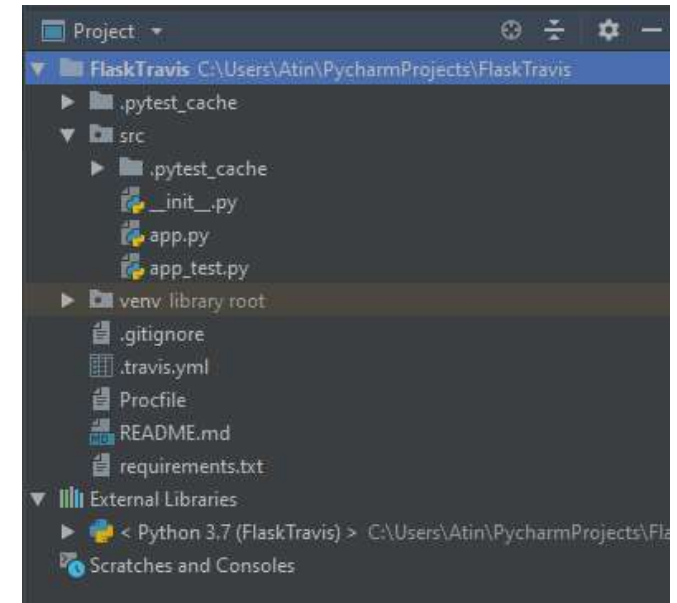
- GitHub
- Travis
- Heroku

Deployment learning checklist

- If tight on time look at the platform-as-a-service (PaaS) options.
 - You can deploy a low traffic project web app for free or low cost.
- No worry about setting up the operating system and web server.

Create The Python Application

- Develop a sample Flask Application
 - `import os`
 - `from flask import Flask`
 - `app = Flask(__name__)`
 - `@app.route('/')`
 - `def hello_world():`
 - `return 'Hello, World!'`
 - `if __name__ == '__main__':`
 - `# Bind to PORT if defined, otherwise default to 5000.`
 - `port = int(os.environ.get('PORT', 5000))`
 - `app.run(host='0.0.0.0', port=port)`
- Refer to the Pycharm project for complete details of the files



Create The Python Application

- `__init__.py`
 - Required for Python Project
- `app.py`
 - Main file containing the code
- `app_test.py`
 - Test code which would be executed everytime we push the code to Github.
 - Its used for automated testing which is required before making the build to deploy
- `.gitignore`
 - Got related file which specifies that which files from current directly should not be committed to Git

Create The Python Application

- .travis.yml
 - For travis CI
 - `dist: xenial`
 - `language: python`
 - `python:`
 - `- "3.7"`
 - `install:`
 - `- pip install -r requirements.txt`
 - `script: pytest`
- Procfile
 - For Heroku
 - `web: gunicorn src.app:app -b 0.0.0.0:$PORT`
- requirements.txt
 - List of Python libraries to be installed on the server
 - Flask
 - pytest
 - gunicorn

Run Python Tests

```
C:\Windows\System32\cmd.exe
(env) E:\gitWorkspace\FlaskHelloWorld>pytest
===== test session starts =====
platform win32 -- Python 3.7.4, pytest-5.0.1, py-1.8.0, pluggy-0.12.0
rootdir: E:\gitWorkspace\FlaskHelloWorld
collected 2 items

src\app_test.py .. [100%]

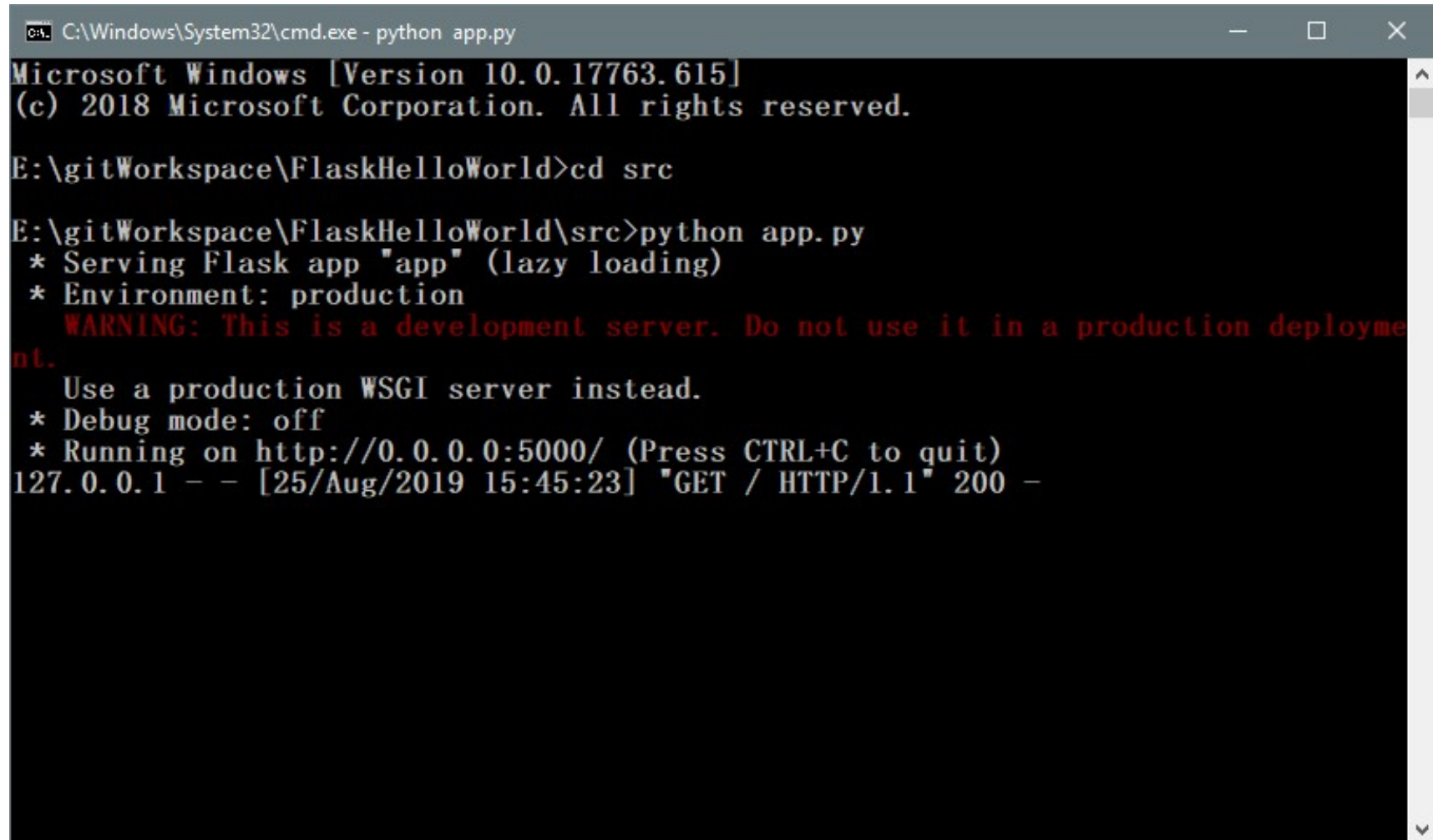
===== warnings summary =====
env\lib\site-packages\jinja2\utils.py:485
  e:\gitworkspace\flaskhelloworld\env\lib\site-packages\jinja2\utils.py:485: DeprecationWarning: Using or importing the ABCs from 'collections' instead of from 'collections.abc' is deprecated, and in 3.8 it will stop working
    from collections import MutableMapping

env\lib\site-packages\jinja2\runtime.py:318
  e:\gitworkspace\flaskhelloworld\env\lib\site-packages\jinja2\runtime.py:318: DeprecationWarning: Using or importing the ABCs from 'collections' instead of from 'collections.abc' is deprecated, and in 3.8 it will stop working
    from collections import Mapping

-- Docs: https://docs.pytest.org/en/latest/warnings.html
===== 2 passed, 2 warnings in 0.12 seconds =====

(env) E:\gitWorkspace\FlaskHelloWorld>
```

Run the application

A screenshot of a Windows command prompt window. The title bar shows the path 'C:\Windows\System32\cmd.exe - python app.py'. The window contains the following text:

```
Microsoft Windows [Version 10.0.17763.615]
(c) 2018 Microsoft Corporation. All rights reserved.

E:\gitWorkspace\FlaskHelloWorld>cd src

E:\gitWorkspace\FlaskHelloWorld\src>python app.py
* Serving Flask app "app" (lazy loading)
* Environment: production
  WARNING: This is a development server. Do not use it in a production deployment.
  Use a production WSGI server instead.
* Debug mode: off
* Running on http://0.0.0.0:5000/ (Press CTRL+C to quit)
127.0.0.1 - - [25/Aug/2019 15:45:23] "GET / HTTP/1.1" 200 -
```

Push Code to Repository

- Use Github to push project.

Create Pipelines Application

- Create Travis CI for continuous integration
- Prior to getting to the hosting however, set up Travis to test application.
- Goto <https://travis-ci.org>
- Set up an account and “enable” the functionality for the repository we want to have tested with travis.
- Next, we’ll need a .travis.yml file with the following content:
 - dist: xenial
 - language: python
 - python:
 - - "3.7"
 - install:
 - - pip install -r requirements.txt
 - script: pytest

Deploy Steps

- Deploy the application to Heroku
- Project is now ready, set up Heroku to serve our website
- The set up of a heroku account is pretty straight forward
 - Just need to point heroku to github repo
 - Then tell it to auto deploy pushes to master
- Procfile
 - `web: gunicorn src.app:app -b 0.0.0.0:$PORT`

Getting Started on Heroku with Python

- The Heroku CLI requires Git, the popular version control system.
- Ubuntu:
 - `sudo snap install heroku --classic`
- Windows:
 - <https://cli-assets.heroku.com/heroku-x64.exe>
- Once installed, you can use the heroku command from your command shell.
 - `heroku login`
- `git clone https://github.com/atingupta2005/FlaskTravis`
- `Cd FlaskTravis`

Deploy the app

- `heroku create`
- `git push heroku master`
- `heroku ps:scale web=1`
- `heroku open`
- `heroku logs -tail`

Thanks