

Deploy Python Apps

～Heroku を利用した Python アプリのデプロイ～

Yohei Munesada



about me

宗定 洋平
(むねさだ ようへい)

G's ACADEMY TOKYO メンター

ブログ => <https://www.yoheim.net>



course catalog

- ✓ Pythonとは
- ✓ Python基本編
- ✓ モジュールとパッケージ
- ✓ Webスクレイピング（基礎）
- ✓ Webスクレイピング（実践）
- ✓ Flaskを用いたWebアプリケーション（基礎）
- ✓ Flaskを用いたWebアプリケーション（実践） *Pre study*
- ✓ 手書き文字の判定アプリを作ろう（機械学習） *1st day*
- ✓ デプロイ
- ✓ 開発演習（任意提出） *2nd day*

a concept of the day

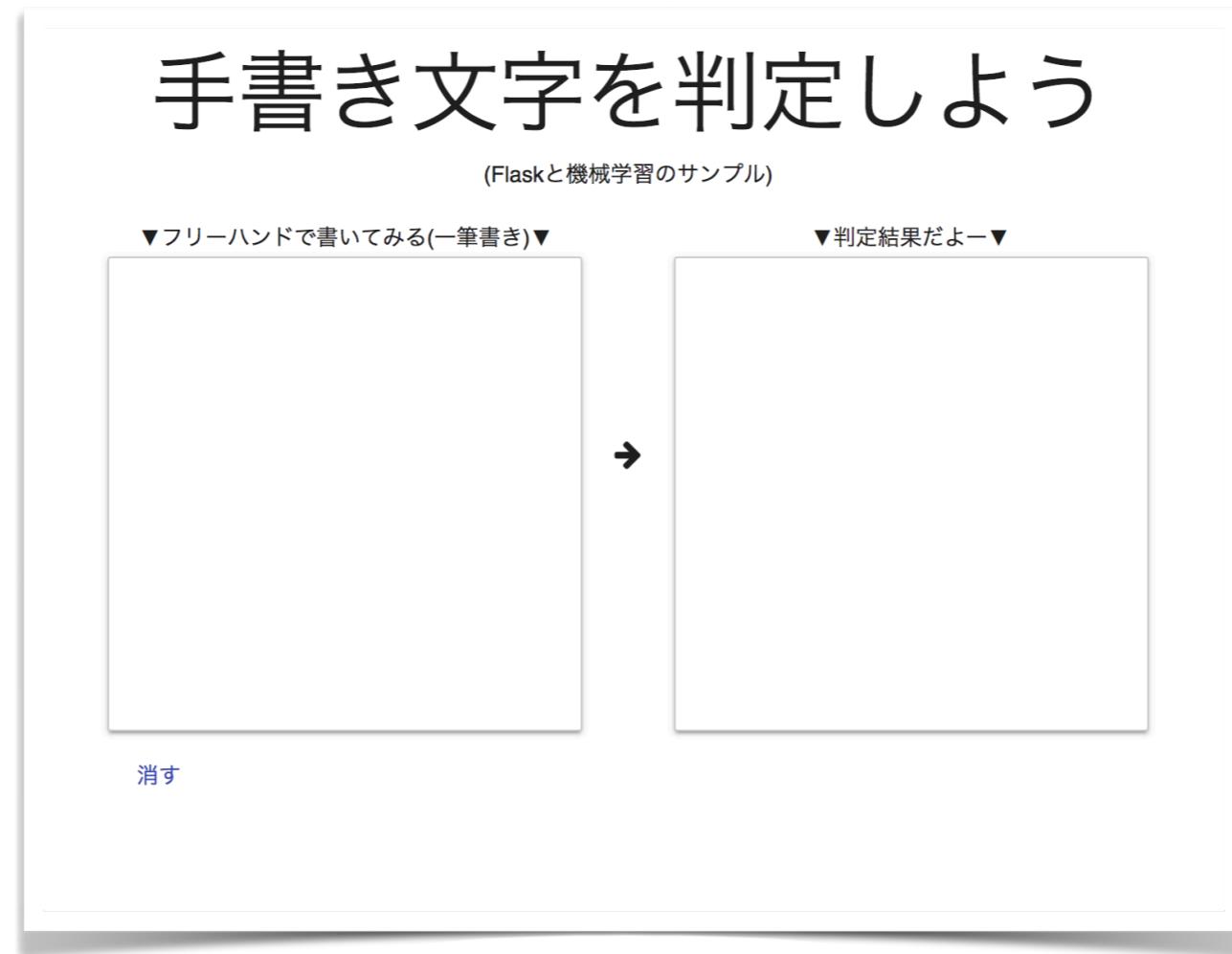
Heroku を用いてデプロイできるようになる

デプロイに使用する技術要素を理解する

開発演習にチャレンジ！



today's goal



手書き判定アプリを Heroku にデプロイしましょう



agenda

Heroku にデプロイしてみよう

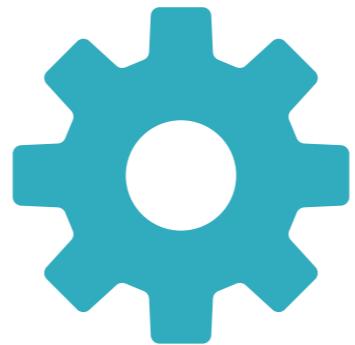
Heroku とは

デプロイに使用する技術要素

開発演習に取り組もう



Deploy to Heroku



anyway, Let's deploy your app.

まずは、Heroku にアプリをデプロイしてみましょう。

[公式] Heroku に Python アプリをデプロイする

<https://devcenter.heroku.com/ja/articles/getting-started-with-python?singlepage=true>

<諸注意>

- ✓ 2日目の演習素材を使います。
- ✓ 演習素材の取得時に「git clone」した場合は、現在の「.git」フォルダは削除しましょう。
 - ✓ git リポジトリのルートが、heroku デプロイの想定と異なるため。
- ✓ 「practice」フォルダにターミナルなどで移動した状態で作業しましょう。
- ✓ 事前に、ローカル環境で起動できる状態を確認しておきましょう。
 - ✓ 一連の「pip install」を行っておきましょう。



anyway, Let's deploy your app.

まずは、Heroku にアプリをデプロイしてみましょう。

<Steps 1/2>

1. heroku コマンドを使えるようにセットアップします。
2. 「[heroku/python-getting-started](#)」リポジトリを覗いてみよう。
3. 「pip3 freeze > requirements.txt」を実行する。
4. Git リポジトリをローカルに作成する。
 1. git init
 2. git add .
 3. git commit -m “add files.”
5. 「heroku create」を実行する。
 1. リモートリポジトリの設定を確認（「git remote -v」）



anyway, Let's deploy your app.

まずは、Heroku にアプリをデプロイしてみましょう。

<Steps 2/2>

6. Procfile を作成します。

1. 「web: gunicorn app:app」を記載。

7. requirements.txt に 「gunicorn」を追加します。

1. pip3 install gunicorn

2. pip3 freeze > requirements.txt

8. Heroku へデプロイする。

1. git push heroku main

9. 動作確認をしましょう。

1. heroku open

2. heroku logs --tail



congrats !



how heroku works



agenda

- heroku とは
- get started
- heroku の仕組み
- heroku に必要なファイル群



what is heroku

Heroku の説明ページを読んでみましょう。

<https://jp.heroku.com/what>

The screenshot shows the top navigation bar of the Heroku website with links for '製品' (Products), 'マーケットプレイス' (Marketplace), '価格' (Pricing), 'ドキュメント' (Documentation), 'サポート' (Support), 'COVID-19', and 'その他' (Others). Below the navigation is a large purple banner with white text that reads: 'Heroku はアプリの構築、提供、監視、スケールに役立つクラウドプラットフォームで、アイデアを出してから運用を開始するまでのプロセスを迅速に進めることができます。また、インフラストラクチャの管理の問題からも解放されます。' (Heroku is a cloud platform for building, deploying, monitoring, and scaling applications. It allows you to move from idea to production quickly, and it also releases you from managing infrastructure issues.)

「そのためのアプリがあります」 – ある印象的なマーケティングキャンペーンの下で携帯電話の新しい使い方が提案されたのは、今からわずか数年前。現在では、アプリは多くの人々の生活の一部になっています。モバイルでも Web でも、私たちは日々アプリやその基盤となる API を使用し、雑務をこなしたり、何かを購入したり、他人と交流したり、情報を収集したり、顧客とやり取りを交わしたりしているのです。

開発者の方は [チュートリアル](#) に進むか、 [Heroku のしくみ](#)をご覧ください。開発者以外の方はこのまま読み進めてください。

アプリの重要性

顧客がアプリを使い始めると、そのアプリによって世界が変わり始めます。アプリをすばやくインターネットに配信し、更新していくことは、企業を生み出すことに

get started

さまざまな言語で Get started が用意されています。

<https://devcenter.heroku.com/start>



Getting Started on Heroku

Step-by-step guides for deploying your first app and mastering the basics of Heroku



Node.js



Ruby



Java



PHP



Python



Go



Scala



Clojure

how heroku works

heroku 稼働の仕組みを理解しましょう。

<https://devcenter.heroku.com/ja/articles/how-heroku-works>

Heroku Dev Center ⚡ Get Started ▾ 📄 ドキュメント ⏲ Changelog More ▾

Dev Center を検索

CATEGORIES

- Heroku のアーキテクチャ
 - Dyno (アプリコンテナ)
 - スタック (オペレーティングシステムイメージ)
 - ネットワーキングと DNS
 - プラットフォームポリシー
 - プラットフォームの原則
- コマンドライン
- デプロイ
- 継続的デリバリー
- 言語サポート
- データベースとデータ管理
- モニタリングとメトリクス
- アプリのパフォーマンス
- アドオン
- 共同作業
- セキュリティ
- Heroku Enterprise

Heroku のアーキテクチャ / Heroku の仕組み

Heroku の仕組み

最終更新日 2020年06月09日(火)

日本語 – [Switch to English](#)

Table of Contents

- アプリケーションを定義する
- 実行内容を明確にする
- アプリケーションをデプロイする
- アプリケーションをビルドする
- dyno でアプリケーションを実行する
- 環境設定
- リリース
- Dyno Manager
- アドオン
- ログと監視
- HTTP ルーティング
- まとめ

heroku デプロイに必要な、いくつかのファイルを説明します。

- Procfile
- runtime.txt
- requirements.txt



procfile

Procfile を用いてアプリを実行するコマンドを指定します。

<https://devcenter.heroku.com/ja/articles/procfile>

Heroku Dev Center ⚡ Get Started ▾ 📄 ドキュメント ⏪ Changelog More ▾ Dev Center を検索 🔎 ⌂ 🏠

CATEGORIES

- Heroku のアーキテクチャ
 - Dyno (アプリコンテナ)
 - スタック (オペレーティングシステムイメージ)
 - ネットワーキングと DNS
 - プラットフォームポリシー
 - プラットフォームの原則
- コマンドライン
- デプロイ
- 継続的デリバリー
- 言語サポート
- データベースとデータ管理
- モニタリングとメトリクス
- アプリのパフォーマンス
- アドオン
- 共同作業
- セキュリティ
- Heroku Enterprise

Heroku のアーキテクチャ / Procfile

Procfile

⌚ 最終更新日 2020年03月09日(月) ☰ 日本語 – [Switch to English](#)

Table of Contents

- Procfile の名前と場所
- Procfile の形式
- ローカルでデプロイする
- Heroku にデプロイする
- プロセスタイプをスケールする
- その他のプロセスタイプの例
- Procfile と heroku.yml
- 参考情報

Heroku アプリには、起動時にアプリで実行するコマンドを指定する **Procfile** があります。 Procfile を使用して、次のようなさまざまなプロセスタイプを定義できます。

runtime.txt

実際の開発では必ず、Python バージョンを指定しましょう。

<https://devcenter.heroku.com/ja/articles/python-runtimes>

The screenshot shows the Heroku Dev Center interface. The top navigation bar includes links for 'Get Started', 'Documentation', 'Changelog', and 'More'. A search bar and a user profile icon are also present. The left sidebar has a 'CATEGORIES' section with links for Heroku architecture, command-line tools, deployment, continuous delivery, language support (Node.js, Ruby, Python, Django, Python background jobs, Java, PHP, Go, Scala, Clojure), databases, and monitoring. The 'Python' link under 'Language Support' is currently selected. The main content area displays the article 'Python ランタイムの指定' (Specifying a Python Runtime). It includes a breadcrumb trail ('言語サポート / Python / Python ランタイムの指定'), a 'Last updated' date (August 30, 2021), a 'Table of Contents' (Selecting a runtime and Supported runtime versions), and two paragraphs of text about default runtimes and specifying runtimes for specific applications. At the bottom, there's a section titled 'ランタイムを選択する' (Selecting a runtime) with a note about declaring the correct version in a runtime.txt file.

Heroku Dev Center Get Started ▾ ドキュメント Changelog More ▾ Dev Center を検索 日本語 – [Switch to English](#)

CATEGORIES

- Heroku のアーキテクチャ
- コマンドライン
- デプロイ
- 継続的デリバリー
- 言語サポート
 - Node.js
 - Ruby
 - Python
 - Django の使用
 - Python でのバックグランドジョブ
 - Java
 - PHP
 - Go
 - Scala
 - Clojure
- データベースとデータ管理
- モニタリングとメトリクス

言語サポート / Python / Python ランタイムの指定

Python ランタイムの指定

最終更新日 2021年08月30日(月)

Table of Contents

- ランタイムを選択する
- サポートされているランタイムバージョン

デフォルトでは、Heroku 上の新しい Python アプリケーションは「[Python バージョンの指定](#)」に示されている Python ランタイムを使用します。

別のサポートされるランタイムが必要な Python アプリケーションを実行している場合や、単純にアップグレードの準備ができるまでパッチの更新に対してプロジェクトをロックしたい場合は、アプリにどのランタイムを使用するかを指定できます。

ランタイムを選択する

Python ランタイムを指定するには、使用する正確なバージョン番号を宣言する `runtime.txt` ファイルをコードのルートディレクトリに追加します。

requirements.txt

該当ファイルを用いて、アプリの依存ライブラリを定義できます。

<https://note.nkmk.me/en/python-pip-install-requirements/>

How to Use

1. requirements.txt の作成

1. pip freeze > requirements.txt

2. requirements.txt に定義された依存関係をインストール

1. pip install -r requirements.txt



re:agenda

- heroku とは
- get started
- heroku の仕組み
- heroku に必要なファイル群



wsgi and gunicorn



agenda

- wsgi とは
- gunicorn とは



what is wsgi

WSGI は、Python を Web サーバーとして動かす仕様です。

<https://wsgi.readthedocs.io/en/latest/>

Description

1. WSGI は Web Server Gateway Interface の略です。
2. ぶっちゃけ、内容が分からなくてもOKです（笑）。
3. メジャーなライブラリ（Flask, Django, etc）がサポートしています。
 1. <https://wsgi.readthedocs.io/en/latest/frameworks.html>
4. WSGI 仕様をサポートするサーバーは複数あります。
 1. gunicorn, uwsgi, など
 2. <https://wsgi.readthedocs.io/en/latest/servers.html>



try to write a wsgi app

WSGI仕様に則った、アプリを書いてみましょう。

```
# ここではファイル名を「my_wsgi.py」とします。
```

```
def app(environ, start_response):
    print('environ:', environ)
    start_response('200 OK', [('Content-Type', 'text/plain')])
    body = 'hello from wsgi app.'
    return [body.encode('utf-8')]
```

実装したら、以下コマンドで実行してみましょう。

```
gunicorn my_wsgi:app
```



flask and wsgi

Flask の場合、「app」が wsgi の仕様を満たしています。

```
from flask import Flask  
→ app = Flask("app")  
  
@app.route("/")  
def index():  
    return "Hello from flask"  
  
if __name__ == '__main__':  
    app.run()
```

以下のように実行できます（ファイル名が「my_app.pyの場合）

```
gunicorn my_app:app
```



what is gunicorn

gunicorn は、WSGI を動作させるためのサーバーです。

<https://gunicorn.org/>

Latest version: [20.1.0](#)



gunicorn

Gunicorn 'Green Unicorn' is a Python WSGI HTTP Server for UNIX. It's a pre-fork worker model. The Gunicorn server is broadly compatible with various web frameworks, simply implemented, light on server resources, and fairly speedy.

[View source](#)

[Download](#)



QUICKSTART



DEPLOYMENT



COMMUNITY



DOCUMENTATION

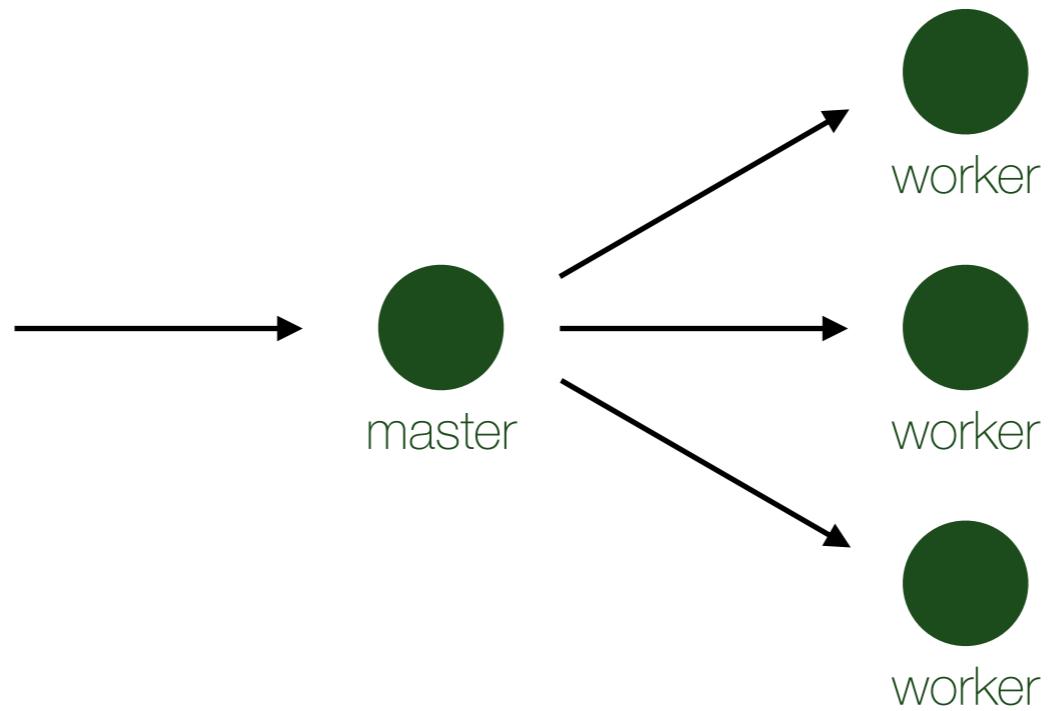
why gunicorn

gunicorn を使う理由は、本番環境などで動作を高速に、安定させるためです。

Reason (1 / 2)

1. gunicorn は Master / Worker モデルで稼働します。

- ✓ 「gunicorn —workers=3 app:app」
- ✓ <https://docs.gunicorn.org/en/stable/run.html>



why gunicorn

gunicorn を使う理由は、本番環境などで動作を高速に、安定させるためです。

Reason (2 / 2)

2. nginx などの Web サーバーとの連携もできます。

✓ <https://docs.gunicorn.org/en/stable/deploy.html#nginx-configuration>

3. サーバー (EC2など) 起動時に、自動起動もできます (systemdの利用) 。

✓ <https://docs.gunicorn.org/en/stable/deploy.html#systemd>



re:agenda

- wsgi とは
- gunicorn とは



practice



開発演習について

講義の集大成として、Python アプリ制作に取り組みましょう。

What to do

1. 作りたい Python アプリを考え、Slackに記載してください。
2. 時間の許す限り、制作しましょう。
✓全ては作り切れないと思いますが、この講義内で1歩目を踏み出してほしいと考えています。
3. (若干名) 制作内容の発表をお願いします。



Course Summary



course catalog

- ✓ Pythonとは
- ✓ Python基本編
- ✓ モジュールとパッケージ
- ✓ Webスクレイピング（基礎）
- ✓ Webスクレイピング（実践）
- ✓ Flaskを用いたWebアプリケーション（基礎）
- ✓ Flaskを用いたWebアプリケーション（実践） *Pre study*
- ✓ 手書き文字の判定アプリを作ろう（機械学習） *1st day*
- ✓ デプロイ
- ✓ 開発演習（任意提出） *2nd day*

Thank you



<http://www.yoheim.net>



@yoheiMune

<https://flic.kr/p/mzmQK2>