# Deploy FastAPI on Heroku

## 1. Everything you need to know about Heroku

### 1.1 About Heroku

Heroku is a container-based cloud Platform as a Service (PaaS). Developers use Heroku to deploy, manage, and scale modern apps.Heroku is fully managed, giving developers the freedom to focus on their core product without the distraction of maintaining servers, hardware, or infrastructure.

### 1.2 Why use Heroku when AWS is present?

Actually Heroku runs on Amazon Web Services (AWS), so why don’t we deploy to AWS and bypass Heroku entirely? First of all, Heroku and AWS are not the same things.

AWS is an Infrastructure as a Service(IaaS) provider, meaning they are responsible for managing large, shared data centers. These data centers are what we call “the cloud”. Companies like AWS, Azure, and Google have all created IaaS so that developers can pay to host their applications in these data centers instead of building servers themselves.

Heroku, on the other hand, is a Platform as a Service that sits on top of AWS to provide an experience that is specifically designed to make developers' lives easier. For example, in order to keep an application running at scale on Heroku, it only takes knowledge of a few commands on the Heroku CLI and Dashboard. These commands can easily be found in Heroku’s documentation.

## 2. Deploy FastAPI on Heroku

### Prerequisite:

* Go to [Heroku](https://signup.heroku.com/login) website and create a new account.
* Download and install Heroku CLI
* Open Terminal and repeat the following process

### Method1: Deploy FastAPI on Heroku without Docker Container

1. Check Heroku CLI correctly installed

$ heroku --version

2. Login and create an app through command line

$ heroku create

Notes: you can specify the app name following heroku create, if creating an app through Heroku website, need to run $ heroku login later in the command line

3. Push the code from the local repository's main branch to your heroku remote.

$ git push heroku main

Notes: If the code is located in a sub file folder, then you use the command:

$ git subtree push —prefix FOLDER-NAME heroku main

If the code is not on the main branch, then you need to run

git push heroku BRANCH-NAME:main

# replace FOLDER-NAME with your main folder name; replace BRANCH-NAME with your branch name

Prerequisite: you need to create a file with the name *Procfile* without any extensions. Thus, Heroku knows where to get started. A sample content of Procfile:

web: uvicorn app.main:app --host=0.0.0.0 --port=${PORT}

### Common issues:

#### 1. No such app when you git push as follows:

git push using: heroku main

remote: ! No such app as xxxxxxx

Solution: relinking the app

# Check for the current url

git remote -v

# remove remote url

git remote rm heroku

# re-add the remote url

git remote add heroku [git@heroku.com](mailto:git@heroku.com):boiling-inlet-6957.git

# heroku.com:boiling-inlet-6957.git should be replaced by the name you have for the app.

##### 2. Compiled slug size is too big (over 500 mb)

Optimize your requirements.txt to shrink the size of the packages installed or use docker container to deploy on Heroku to bypass that size limit

### Method2: Deploy FastAPI on Heroku using Docker Container

1. First login to the heroku CLI

$ heroku login

2. Create an app through the command line

$ heroku create APPNAME

# create an app with a specified name

3. Then login to heroku container registry

$ heroku container:login

4. Push the container to heroku

$ heroku container:push web --app YOURAPPNAME

# replace YOURAPPNAME with your app name

5. In the end release the container

$ heroku container:release web --app YOURAPPNAME

Notes:

View logs: $ heroku logs --tail

## 3. Pros & Cons of Heroku

Advantages:

* Easy setup - as a PaaS you don't need to know how to install and configure Apache, nginx, unicorn, passenger, MySQL, Postgres etc
* Easier to scale initially - spin up more dynos, size up DBs etc
* Great plugin support for third party apps

Disadvantages:

* Price - Heroku Dynos are not cheap; for almost the same price as a Heroku “2x” Dyno with 1GB RAM, you can have a 3.large EC2 instance with 3.75GB RAM on AWS , monitoring addon doubles the price.
* Performance - Not surprisingly , multiple Heroku Dyno's are running on the same EC2 instances under separate namespaces (containers); on large scale infrastructures ,with serious traffic this can cause a serious performance problem, which eventually can impact your business.
* Dyno is a "Black Box" ; you cannot SSH to your Dyno and debug a memory/cpu/disk io/network bottleneck.
* Lack of GPU support