

[medium.com](https://medium.com)

# Free Hosting a Python API - Trần Hoàng Long - Medium

Trần Hoàng Long

5–6 minutes



My choice for free python hosting 🏆

## Foreword


A few months ago, I encountered a problem while trying to deploy my backend API for a demo of my side project. **Which free service would host my python backend?**

The criterias/downsides that I had were:

- Must be fully free with no credit card requirement (student problem 🤔) → *So no AWS, Google or Azure or any other virtual machine services*
- I have a database so the server must be able to make DB connections instead of just communicating with frontend
- Of course speed is not my concern since we're going dirt-tier anyways

- Lastly, of course, the server must support python (duh!) (Django to be specific)

If you had some of the similar problems, this blog could be for you. I'll discuss some of my findings and workarounds below

 *It was suprisingly hard for me to find a completely free python hosting service. Took lots of trial/error. That's why I write this post for anyone that likes to save their time digging.*

## TOC

- [Foreword](#)
- [TOC](#)
- [Project Description](#)
  - [Database](#)
  - [Backend](#)
  - [Frontend](#)
- [Hosting Service](#)
- [Render Hosting](#)
  - [Build server](#)
  - [Environment Variables](#)
  - [Run server](#)
- [Demo](#)
- [Final words](#)

## Project Description

So here's a quick layout of my project at the time, called Geo - covid , which is basically a visualization website for the covid problem in USA, which consists of 3 parts

## Database

MongoDB is my goto database here due to the implementation simplicity and also the Mongo Atlas support for free DB hosting

*!! This (db connection from/to the backend) proven itself to be the trickiest part of the server hosting process*

## Backend

My API implemented with Python and Django REST API

*Why Django REST and not Fast API? Well cause I just wanna try Django out for once.*

## Frontend

Web visualization with the help of Tailwind CSS and D3.js (graph) and Leaflet (map)

We don't focus on the frontend here since free hosting for frontend JS is everywhere and it only make simple request back/forth to our API server.

## Hosting Service

After some trial and error, the two contenders that were the most promising were:

1. render.com
2. pythonanywhere.com

But in the end, pythonanywhere fell short of my need, despite being the more comprehensive and dev friendly service. All because of the previously mention DB connection problem 🙄

*You see on the **free tier**, pythonanywhere only allow external*

connection under the `https` protocol, so my Mongo Atlas connection is of `mongodb://` protocol which would simply not work ([Link](#))

Thankfully, render later prove to work properly, which is good enough 🙏

## Render Hosting

The hosting procedure is very detaily describe on Render's documentation ([Django Quickstart Doc](#)). So I'm just going to point out notable things from my project

The source code for my backend that's deployed could be found on my [production branch](#). And in that branch, I din't modify any code or add files to config deployment. All were done on Render itself (due to the simplicity of my project)

Important steps are:

1. Build: Setup python environment and (optional) build static files
2. Setup production environment variables
3. Run server

## Build server

### Python env

All we need here is a `requirement.txt` file that we'lll use to install packages when building our source code with `pip`

```
install -r requirement.txt
```

### Static files

Static files for Django server API GUI could be generated with `python manage.py collectstatic`

## Build

We put all the needed commands for server preping in the Build Command option for Render (found in Settings > Build & Deploy > Build Command). I set it as

```
pip install -r requirements.txt && python manage.py collectstatic
```

## Environment Variables

Found in Environment tab

First option to set is the PYTHON\_VERSION (I used 3.9.0)

For other variables related to my backend, I uploaded my .env file directly to Render instead.

This includes:

DEBUG=FALSE

DB\_URL="mongodb+srv://<username>:  
<password>@<cluster\_address>.mongodb.net  
/?retryWrites=true&w=majority"

DEPLOY\_ENV=production

Here's how the environment settings looks like

Render's Project Environment Settings

***[ ?** For anyone wondering how in the code I connected to Atlas,*

*here's that snippet*

```
from pymongo import MongoClient
client = MongoClient(
    host=DB_URL
)
```

## Run server

Settings > Build & Deploy > Run Command

```
gunicorn --bind :$PORT --workers 4 -k
uvicorn.workers.UvicornWorker geo_covid.asgi:application
```

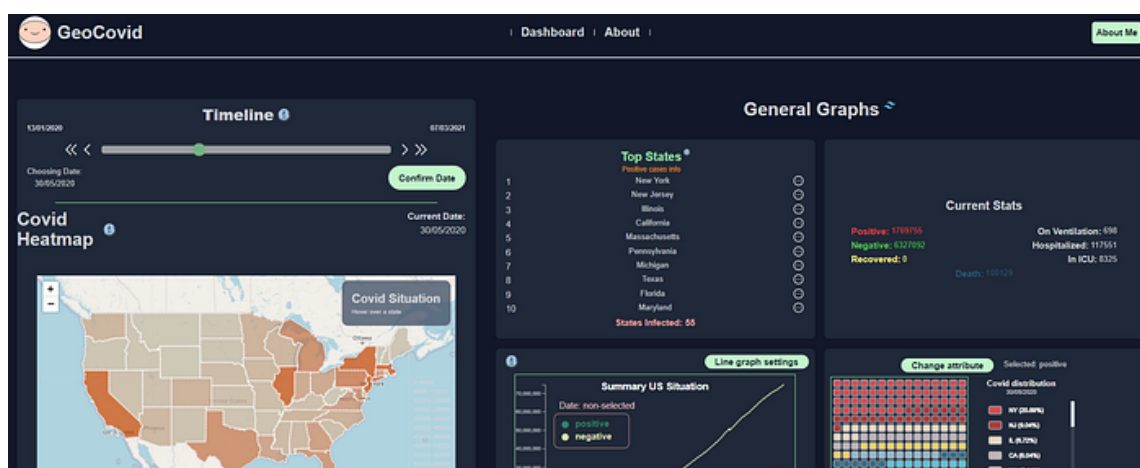
Key takeaway is that I ran it with gunicorn and some extra port, worker settings.

After all the settings, just run deploy with latest commit 🙏

## Demo

After getting the DB up and then the backend API online, I try making connections to it from my frontend and everything is nice/clean 🥳🥳🥳

The server might not be online anymore but as of now, my frontend is still online and you can visit it at <https://geo-covid-frontend.web.app/>





Demo

Backend source code can be found [here](#) and front-end [here](#)

## Final words

Those are my findings while trying to get my side project online. It's quite messy and erroneous overall but I'm quite happy that it's online.

I'll try to get a creditcard soon 💀

But please, do comment your thought on my topic and clap if it helped in any ways. Would love to hear from you ★

Thanks for reading and have a nice day 😎