Nadayar Enegesi
Feb 26, 2017 · 6 min read · Listen

Heroku Fu: Multiple Servers on One Dyno

TL;DR Don't commit to an engineering decision before considering the problem, available tools, and tradeoffs. Or YOLO! Make the mistake, and learn something cool while at it!

Background

Every month, Andela accepts at least forty new developers into our developer Fellowship. The Fellowship kicks off with a series of immersive learning experiences focused on technical and professional skills designed and facilitated by our Learning team (which I'm part of).

As you can already figure, managing the flow of developers through our pipeline of programs all the way to when they start building products for our clients is a critical business function. That challenge proves to be more critical as we prepare to scale to other countries.

From the beginning, we had used spreadsheets to manage our pipeline. It was becoming tedious to manage all the data and transitions.

So a few months ago, my 10x team of software engineers turned Learning Facilitators took three days off to hack on a tool to upgrade our solution.

Man, I was stoked! Time to whip out my engineering chops for three days straight without any interruptions from other managerial or business operational responsibilities.

Mistakes Were Made

We decided to build a web app with a decoupled client and server architecture. Simple enough.

We were in explore mode, so we did not put much thought into what technologies we wanted to use to build the app. The front-end team agreed to use React.js because that's the new hotness in the javascript labyrinth of frameworks. Backend peeps decided to use Flask since they were all proficient in Python plus Flask is a lightweight framework. Nothing to fear here.

After some hours of spec'ing out APIs and wireframes, and initial implementations, it was time to set up our deployment to start seeing how the app was coming together. Makes sense.

One of the front-end team members suggested we deploy the client on DigitalOcean. My response was "yeah that makes sense but would take extra time to configure. Let's just use Heroku since it's very simple and this is a hackathon ¯_(ツ)_/¯". No big deal.

The Constraints

If I was going to deploy the client and server on Heroku, I had two options:

1. Deploy client and server as two separate Heroku applications
2. Deploy client and server on the same Heroku application but use two dynos

> *Challenge with Option #1: The code for client and server lived in the same repository.*

How's that a challenge? Let's go back to how Heroku does deployment out of the box.

Heroku has a buildpack module that sets up your dyno's environment by installing all the necessary dependencies before running your server. It runs a dependency file detection engine through your project's root directory to decide on what buildpack to use. E.g if it sees a requirements.txt file, it knows to use the Python buildpack, or if it sees a package.json file, it knows to use the Node.js buildpack.

Our project directory looked like this:

```
root
├── client
│   ├── src
│   └── package.json
├── server
│   ├── app.py
│   └── requirements.txt
└── README.md
```

No dependency file in root directory => no buildpack => no deployment. If we wanted to force a deployment anyways, we could just push the sub-directories using git's subtree module:

```
git subtree push --prefix pythonapp heroku master
```

YOLO!

As an avid practitioner of the YOLO way, I know that all things YOLO come with a caveat. So I thought, "well if we deployed the client and server as two separate applications, we'd have to manage two deployment pipelines, and this is just a hackathon. Ain't nobody got time fo dat". So I jumped to Option #2; deploy client and server on the same application and use two dynos.

> *Challenge with Option #2: Heroku's free plan only gives you 512 RAM with one web dyno and one worker dyno. You need another web dyno? Upgrade to a paid plan. Ain't nobody got money fo dat in a hackathon ¯_(ツ)_/¯*

I investigated deploying either the client or the server as a worker, but workers cannot be exposed on ports. You can't expose workers on ports using Heroku :(

Option #1 beckoned me with open arms. "Nope!"

Then I had one of those brain sparks that you should never listen to:

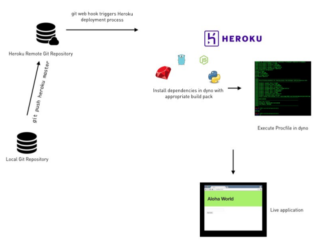
I DONT HAVE TO PAY IF I DEPLOY THE CLIENT AND SERVER ON THE SAME DYNO

So I made Option #3

> *Option #3: Figure out how to deploy two web servers on one Heroku Dyno.*

One of the first results after doing a Google search was this Quora post where the OP was trying to achieve what I wanted. He concluded that there was no solution, and ended up with my Option #1. This was my cue to give up. But no. What kind of Nad gives up just because there are no answers on Quora and StackOverflow? Not this limitless Nad!

I resumed my quest by thinking deeply about how Heroku triggers and manages deployments.


git web hook triggers Heroku deployment process
Heroku Remote Git Repository
git push heroku master
Local Git Repository
HEROKU
Install dependencies in dyno with appropriate build pack
Execute Procfile in dyno
Aloha World
Live application

Heroku deployment flow explained simply

It's important to know that most of the events in this process are automatic. The only step that requires a user's input is the Procfile step since Heroku executes whatever instruction a user enters in it. This was my in!

In the past, I had used Foreman to manage server processes. Foreman is a Ruby library that allows you to declare the processes that are necessary in order to run your app in a file called a `Procfile.` Looks familiar to my current problem!

All I needed to do was trick Heroku into thinking I was deploying a Ruby app, then tell it to run Foreman in the default Procfile, then Foreman will run my actual Procfile containing the startup instruction for the client and web servers. Win!

I quickly realized that on this dyno I would have Ruby, Python, and Node dependencies. Not good.

I prayed to the gods of open source via Google, and they showed me that Foreman had ports in different languages, including a Python port called Honcho.

With Honcho, I was set.

Search

Nadayar Enegesi
875 Followers
Alter Ego
Following

More from Medium

Michael Moreno
Styling React Components Using JSX

Elham
Demonstrating Recursion through call stack game!

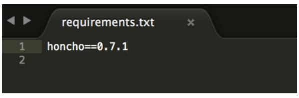
Dhruv Rawat
What is NPM ?

Jarry
How do you make JavaScript's array unique?There is 18 methods.

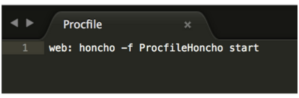
I modified the root directory to look like:

```
root
├── client
│   ├── src
│   └── package.json
├── server
│   ├── app.py
│   └── requirements.txt
├── requirements.txt
├── Procfile
├── ProcfileHoncho
└── README.md
```

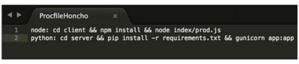
With the new directory structure and files, here's how Heroku behaves:



```
honcho==0.7.1
```



```
web: honcho -f ProcfileHoncho start
```



```
node: cd client && npm install && node index/prod.js
python: cd server && pip install -r requirements.txt && gunicorn app:app
```

Contents of new files in directory

1. Heroku sees requirements.txt and uses the Python buildpack to set up a Python environment.
2. Using the buildpack, Heroku installs the dependencies in the requirements => Honcho.
3. Heroku runs its Procfile which basically just tells Honcho to run using the ProcfileHoncho file
4. Honcho runs the node process to fire up the node client server, then runs the python process to fire up the python server!

I'm feeling pretty good at this point until I try deploying again only to realize that my node and python servers are inaccessible.

Redemption

After reading the logs line-by-line, I figured that the node and python servers were being exposed on random ports. To fix that and make the port assignments deterministic, I set the NODE_ENV environment variable on Heroku to `5000` (you can use whatever number you want). I set the python server to always be exposed on port `1337` by appending `-b 0.0.0.0:1337` to the end of the second line in `ProcfileHoncho.`

I try deploying again, and boom! I had the client and server running on one Heroku dyno! Profit!

I looked at the time; 6 am. I had spent way too much time bending the rules just to satisfy my curiosity.

Lesson learned: if it does not feel natural, forcing it would cause you pain. But, if you follow the white rabbit, you may discover something cool.

If you liked this, click the 💚 below so other people will see this here on Medium. Also, if you have any questions or observations, use the comment section to share your thoughts/questions.

Thanks to Mohini Ufeli

1,264 | 12 | ···

1,264 12

More from Nadayar Enegesi

Following

Alter Ego

Published in **Initiate** · Nov 30, 2016

Who Will Power Africa's Technology Revolution?

Africa is going to be the next economic powerhouse. Africa's momentum, combined with the current digital era, suggests that Africa could leapfrog all other continents by solving many of our problems with technology. Bu…

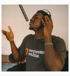
Africa 3 min read

Share your ideas with millions of readers. Write on Medium

May 27, 2016

Passionate Learning: From Zero To Mixtape in 7 Months

I don't really listen to rap music. However, there's something about freestyle rap and watching people create lyrics on the spot that fascinates…



Learning 5 min read

Nov 21, 2015

Bring The Love for Computer Science Back!

Just like everyone else reading this post, I take a lot of things for granted. So it is not a surprise that despite studying and practising Computer Science at the University of Waterloo for 5 years [which happens to be one of the best schools for that stuff], I have…

Education 2 min read

Sep 9, 2015

Competence breeds Company Success

If you are thinking of starting a company with a compelling mission, or if you're currently in one, you may be wondering about how you can attain success. There are many ways to go about that. However, there's one sure-fire way. A lot of companies with a noble mission do…

Culture 2 min read

Aug 31, 2015

The Tasty Pattern: Teaching People how to Code

So you're a programmer? Awesome! You have built complex applications from your own social networks to recommendation engines? Wonderful! You make countless contributions to open source? Fantastic! Now, you want to give back by teaching someone else how to be legendary like…

Developers 3 min read

Love podcasts or audiobooks? Learn on the go with our new app. Try Knowable