Roger Stringer
I'm Roger Stringer, a DevOps engineer, developer, author, foodie, speaker, dad. Founder of Flybase.

Make Your Own Heroku with Dokku and DigitalOcean

May 13, 2015 **Code** , **Devops**

If you've ever used Heroku or Github Pages, you know how easy and convenient it is to be able to deploy a quick static site by pushing to a remote. Being able to type `git push origin gh-pages` and have your site immediately live on a publicly accessible domain is hugely satisfying and you end up experimenting way more, pushing up micro-sites and project pages because it's easy.

GitHub pages does have a limitation, though: there is no concept of a server. While you might not notice that on smaller projects or experiments, it is inevitable that you will run into it eventually. Sometimes you just want a server.

Heroku does enables you to have the concept of a server, with a similar work-flow: make something, then push to a remote with git and it's on the internet. But Heroku costs money. It's free at first, (as long as you don't mind your site being down for up to 6 hours a day) but as you scale, it begins to get more expensive fairly quickly. Add a couple more dynos and you're already up to $70.

Enter Dokku, an open source mini-Heroku.

With Dokku, you can run your own (albeit slightly less feature-rich) Heroku. With a $5 / month Digital Ocean droplet, this DIY solution is pretty inexpensive and easy to set up. Dokku is a Docker powered mini-Heroku.

I'm actually using the Dokku-alt fork, as it includes plugins already to go, so setup is even quicker.

1. Get a Domain Name

I recommend Hover only because I have experience with them and they're not Godaddy...

2. Create a Digital Ocean Account

Create an account with Digital Ocean: digitalocean.com.

3. Create a Droplet

Make a new droplet in DigitalOcean. Be sure to name the droplet exactly what your domain is. Select the smallest size, and whatever region is closest to you.

In the `Select Image` section, click the `Distributions` tab and choose the `Ubuntu 14.04` option.

Click `Create Droplet` .

4. Configure the DNS

To get your domain to point to your new droplet, you need to change the DNS records. Use the following host name and subdomain settings in your name registrar:

HOST NAME	IP ADDRESS/URL	RECORD TYPE	MX PREF	TTL
@	your.droplet.ip.address	A (Address)	n/a	60
www	your.domain	CNAME	n/a	60
*	your.droplet.ip.address	A (Address)	n/a	60

The * entry for subdomains allows you to make any number of apps on different subdomains.

5. Generate an SSH Key Pair

I'll describe what these are in more detail below in the `Set up SSH Keys` section. Don't worry too much about it right now, just generate a pair like this:

1. Go into your .ssh directory: `cd ~/.ssh`

2. Type: `ssh-keygen`
3. When prompted, enter a name for the pair (I used rslab)
4. If you'd like to password protect it enter a password (I left this blank)
5. After it's done, you should have two files: `{name}` and `{name}.pub`

6. Install Dokku-alt

Login to your Droplet and install dokku-alt using the handy one line script:

```
sudo bash -c "$(curl -fsSL bit.ly/dokku-alt)"
```

Check `virtual host naming` if you'd like your Dokku apps to be formatted like app-name.your.domain.

Follow the instructions on screen and you'll be set up.

After a minute or two, it will prompt you to open your browser to configure via a web page, when you do, you should see a screen that has several fields. Add a public ssh key from the previous step (that's the one that ends in .pub).

Hostname should be your domain name exactly.

Check `virtual host naming` if you'd like your Dokku apps to be formatted like `app-name.your.domain` .

Click `Finish Setup`

7. Set up SSH Keys

To make it so you don't have to type your password every time you connect to your droplet or push to Dokku, you can set up ssh keys. Basically, there are two files, one that is on your machine, and one that is on the Digital Ocean droplet. The droplet checks to make sure you have the key that matches it's key, and automatically connects you without a password. We created the keys a few steps back, now let's use them to connect to our droplet.

Make Sure you Have the Keys

1. Go into your .ssh directory: `cd ~/.ssh`
2. Ensure your keys you generated previously are still there. You should see them listed if you type `ls -l` .
3. If they aren't there, generate a new pair (see above).
4. Create a Config File

Create a file named config inside your `.ssh/` directory so your computer knows to use the new ssh keys for your domain. My config looks like this:

```
Host rslab rslab.me
Hostname rslab.me
IdentityFile ~/.ssh/rslab
```

Basically, this is how ssh knows to use that particular key for that host. Obviously change `rslab.me` to your domain, and `.ssh/rslab` to whatever you named your ssh keys.

Put the Public Key on your Droplet

To upload the public key to your droplet, just type:

```
cat ~/.ssh/{name}.pub | ssh root@your.domain "cat >>
~/.ssh/authorized_keys"
```

Replacing the {name} with your key name and your.domain with your domain name.

Connect Without a Password

You should now be able to now connect to your droplet without a password like this: `ssh root@your.domain`

If that worked, you can now (optionally) disable password login to your droplet:

Connect to your droplet: `ssh root@your.domain` Open your config: `sudo nano /etc/ssh/sshd_config` Find the PermitRootLogin line and edit it so it reads: `PermitRootLogin without-password` Reload your ssh: `reload ssh`

8. Deploy your first App

Now you can deploy apps on your Dokku. Let's deploy the Heroku Node.js sample app. All you have to do is add a remote to name the app. It's created on-the-fly.

```
$ git clone https://github.com/heroku/node-js-sample
$ cd node-js-sample
$ git remote add deploy dokku@my.dokku-alt.com:node-js-app
$ git push deploy master
Counting objects: 296, done.
Delta compression using up to 4 threads.
Compressing objects: 100% (254/254), done.
Writing objects: 100% (296/296), 193.59 KiB, done.
Total 296 (delta 25), reused 276 (delta 13)
-----> Building node-js-app ...
       Node.js app detected
-----> Resolving engine versions
... blah blah blah ...
-----> Application deployed:
       http://node-js-app.YOURDOMAIN.com
```

You're done!

Right now Buildstep supports buildpacks for Node.js, Ruby, Python, PHP, and more.

It's not hard to extend the list with your own buildpack. Please check the documentation for your particular build pack as you may need to include configuration files (such as a Dockerfile or Procfile) in your project root.

9. Enable manager

Dokku-alt includes a handy manager that you can use to manage your apps from a web interface, to set this up use:

It's a web interface to easily manage your dokku-alt instance. To install manager, simply run:

```
dokku manager:install
```

You can enable or disable it anytime:

```
dokku manager:enable
dokku manager:disable
```

Or even uninstall if you prefer command line access (it will also wipe used database):

```
dokku manager:uninstall
```

Once set up, you can access the manager from
`http://dam.YOURDOMAIN.com`

10. Set up SWAP

This bit of preemptive work helps out a lot if you use a box with 512MB of memory, to prevent these messages:

```
runtime: panic before malloc heap initialized
fatal error: runtime: cannot allocate heap metadata
```

Run the following (it will create 512MB swap file, you can adjust it for your needs):

```
dd if=/dev/zero of=/extraswap bs=1M count=512
mkswap /extraswap
```

Add it to /etc/fstab:

```
/extraswap         none            swap    sw              0       0
```

Turn it on:

```
swapon -a
```

11. Configuring apps to run on another domain

It's also pretty easy to use another domain and point it to Dokku.

Let's say you are building a little site for a domain you own, we'll call it `my-special-domain.com`, but you set up Dokku on another domain like `dokku-domain.com`. When you decide to deploy your site, all you need to do is change the DNS records for `my-special-domain`. In your DNS registrator, change the `@` and `CNAME` records like this:

HOST NAME	IP ADDRESS/URL	RECORD TYPE	MX PREF	TTL
@	your.droplet.ip.address	A (Address)	n/a	60
www	your.domain	CNAME	n/a	60

Then when you add your remote, you would add it like this:

```
git remote add dokku dokku@dokku-domain.com:my-special-domain.com
```

Once the domain is pointing to your droplet's IP and you've pushed to your remote, you should see your app on `my-special-domain.com`.

12. Hosting a static website

If you want to deploy a static website, there are two methods:

Deploying a Static App in a Subfolder

Deploying a static app actually took a bit of hunting around to find, but once I figured it out, it's dead simple. Essentially you just include an empty `.nginx` file in the root level of your project, and put all your static content in a `www` directory and it will be served on an nginx server automatically. For example, if I had a simple `index.html` file and a `style.css` file I wanted to serve statically, my project would look something like this:

```
.nginx
www
  - index.html
  - style.css
```

That's it! You can have as many folders and files inside the `www` directory as you need and it will serve it up statically for you. This is really useful if you use a static site generator to produce a folder of html as you can just set it to build to `/www` and your site will automatically work.

Deploying a Static App From the Root Folder

If you have no need for a subfolder, you can deploy a static app from the root of your project by adding an empty `.htaccess` file. The folder structure would look something like this:

```
.htaccess
index.html
style.css
```

Dokku should automagically understand that you're writing an app that uses Apache and give you a simple static site after you push.

13. Deleting an App

If you want to remove an app, first, connect with ssh:

```
ssh root@your.domain
```

If you set up ssh keys, this should work without a password. Then simply run:

```
dokku delete app_name
```

Where `app_name` is the name of the app you'd like to delete.

If you set up the manager then you can also delete an app from the manager itself.

Finishing up

That's it, you've now built your own mini Heroku using Dokku.

I've got my main Dokku server running PHP, Node.js, Static and a Jekyll site and they all run nicely together.

Amazon EC2 works well with this as well, actually any server running Ubuntu will work well, but I specifically used Digital Ocean in this post due to being a Digital Ocean user.

Previously: Marco Arment on Redesigning Overcast's Apple Watch app

Next: B. B. King, defining bluesman for generations, dies at 89

Roger Stringer spends most of his time solving problems for people, and otherwise occupying himself with being a dad, cooking, speaking, learning, writing, reading, and the overall pursuit of life. He lives in Penticton, BC.

CONNECT: TWITTER | GOOGLE+

Follow @freekrai on Twitter if you'd like.

TOPICS OF INTEREST

LINKS