Leonardy Kristianto Follow
PM @ Taralite; Crowd research with @StanfordHCI
Nov 19, 2015 · 4 min read

How to Run a Meteor.js Application on Heroku in 10 Steps

This is an idiot proof method of deploying a Meteor.js application to Heroku. I had trouble setting it up at first so I think everybody deserves to know the correct way of setting it up.

1. Install Meteor.js and Heroku toolbelt

```
For Mac: curl https://install.meteor.com/ | sh
For Windows: Meteor installer
Heroku: Heroku Toolbelt
```

2. Create your Meteor application

```
meteor create foobar
cd foobar
```

3. Initialize the directory as a git managed repository

```
git init
git add .
git commit -m "My first commit!"
```

If you're wondering about "*what is this thing called git?*", give this thread a read. It'll help a lot.

4. Create your Heroku instance

```
heroku login
heroku apps:create foobar
```

5. Set a Meteor buildpack for your Heroku instance

```
heroku buildpacks:set https://github.com/AdmitHub/meteor-
buildpack-horse.git
```

What is a **buildpack**? It's a collection of scripts that prepares your code for execution by the Heroku dyno manager. Heroku's *cedar* stack has no default language/framework support, so we use a *buildpack* to determine/specify what kind of framework we wanted to build on.

What is **cedar** then? It's a polyglot environment, which means it has native support for many popular languages and frameworks (e.g. Rails, Node.js, Java, PHP). It also serves as Heroku's default runtime stack (cedar-14).

Edit: Replaced **jordansissel's buildpack** with **AdmitHub's** as the former was no longer maintained (thanks Swyx and Ajit Goel)

6. Create a new mLab* instance

```
heroku addons:create mongolab:sandbox
```

Why **mLab**? It's free and it comes with ~500MB storage (the *sandbox* parameter above signifies the free tier). Compose is available too but you need to fork out $18 for it. You can also use an existing external MongoDB database if you wish to. Look for the tips below.

*Edit: As of February 29, 2016, MongoLab has changed their name to mLab

7. Get your MongoLab URI

```
heroku config | grep MONGODB_URI
```

```
// Alternatively, run "heroku config" to display all your
configuration variables, but truthfully we only need the
MONGODB_URI
// Be careful of running "heroku config" and leaving your
console in the open since it displays all your important
environment variables like Stripe API keys
```

Edit: Replaced MONGOLAB_URI to MONGODB_URI (Credits to Tyler Brown Cifu Shuster and Adam Hadlock)

8. Set the configurations of your Meteor app running on Heroku

```
heroku config:add MONGO_URL=<MONGODB_URI value>
heroku config:add ROOT_URL=https://foobar.herokuapp.com
```

You can also add them through your Heroku dashboard at **https://dashboard.heroku.com/apps/your_app_name/settings** and select “Reveal Config Vars”.

9. Check your remotes to ensure heroku is there

```
git remote -v
```

```
// The output should look like this
heroku https://git.heroku.com/foobar.git (fetch)
heroku https://git.heroku.com/foobar.git (push)
```

10. Deploy the app

```
git push heroku master
```

Really useful tips:

In Heroku, install an add-on called Papertrail to access your application log in real time. It helps in debugging whatever errors that come your way. Or you can track these logs through your console with commands below. The nice thing about Papertrail is that it allows you to filter out the type of logs.

```
heroku logs --tail
```

If you have multiple Heroku apps in one folder (e.g. when you have production & staging in 1 folder):

```
// Add a suffix to Heroku commands
--app yourAppName
```

```
e.g.
heroku buildpacks:set ___ --app foobar-staging
heroku config:add ___ --app foobar-production
```

If you're pushing from a branch:

```
git push heroku branchName:master
```

```
// If you are working in a branch called "load-test"
git push heroku load-test:master
```

Important: If you're working with more than one web dynos, you might want to add session affinity to your app to ensure that Meteor Sessions can still work correctly.

> *In a system without session affinity, if the programmer developed a way to store user data in memory (and nowhere else), this would work fine on one node. However, as soon as the system moved to two nodes or more,*

> *different servers may handle different requests, meaning that different requests will have different views of server-local data. For example, if someone stores profile photos on disk on one server and the next request hits a different server, the photo won't be there. (Heroku)*

Using external MongoDB service

If you want to use an existing external database (say, you're already using a service beside mLab), you only need to obtain the standard MongoDB URI. It looks like this:

```
mongodb://<dbUser>:<dbPassword>@<service>.com:
<port>/<databaseName>
```

For development, you only need to append it before running Meteor.

```
MONGO_URL=mongodb://<dbUser>:<dbPassword>@<service>.com:
<port>/<databaseName> meteor
```

Or you can also add them inside your application codes by directly adding it to the environment variables.

```
// In server
console.log(process.env);
process.env.MONGO_URL=mongodb://<dbUser>:
<dbPassword>@<service>.com:<port>/<databaseName>
```

Settings.json

If you want to use settings.json (to store API tokens for many services, like Amazon S3):

```
// For development
meteor --settings settings.json
```

```
// For Heroku
heroku config:add METEOR_SETTINGS="$(cat settings.json)"
```

```
// Please remember to .gitignore it if you're using a public
repository
```

Also, consider joining the Meteor Chef Slack group or codebuddies Meteor.js channel.