Heroku

+ Node.JS

Samy Pessé

Published
with GitBook

Table of Contents

Heroku + Node.js

Learn how to build and deploy applications with Heroku and Node.js.

About Heroku

Heroku is a cloud platform as a service (PaaS) supporting several programming languages.

It's an easy and powerfull way to deploy applications on the cloud and scale it easily. Heroku supports many programming languages: Python, PHP, Java, Javascript (Node.js), Clojure and Scala.

In this book, we'll learn how to deploy Javascript/Node.JS application on Heroku.

About Node.JS

Node.js is a software platform for scalable server-side and networking applications. Node.js applications are written in JavaScript, and can be run within the Node.js runtime on Mac OS X, Windows and Linux with no changes.

In this book, we'll consider that you now the basics about We Applications and Javascript, and we'll be only focus on the heroku and deployment part.

Creating the Application

The first of this course is to create a Node.js application that can run on Heroku and locally (for testing).

Creation of a Git Repository

Our source code is going to be stored in a Git repository. Heroku uses Git for deployment, and it's also a better habit to use git.

To create a new empty repository, run on a terminal:

```
$ mkdir myapp
$ cd myapp
$ git init
```

We are now going to work on this `myapp` folder.

You can also host this repository on GitHub, this is optional but advised:

1. Create a repository on GitHub
2. On a terminal in the `myapp` folder run: `git remote set-url upstream <git url for the repository>`

The url for the repository on GitHub is in the format: `https://github.com/<username>/<repository>.git` .

Creation of a base for a Node.js application

Now that our git repository is ready, we can start working on writting our node.js application.

In a terminal in the `myapp` folder, run:

```
$ npm init
```

It will ask you for multiple questions and generate a `package.json` file. This file will contain the list of your dependencies (the librairies our program will depend on) and some others descriptives informations.

Commit our base

It's time to commit our first commit for this application:

```
$ git add package.json
$ git commit -m "Base package.json"
```

And push it to GitHub:

```
$ git push
```

Coding the application

Now that the git repository and the node module structure are setup, we are ready to start coding the application.

This application will be a simple hello world.

Instaling our dependencies

In node.js, the dependencies are listed in the **package.json** file. All dependencies from this file can be installed using the command `npm install .`.

An other dependency can be installed and added to the package.json using: `npm install name@version --save`.

Our application will depend on: `express@4.3.1`.

So install it using:

```
$ npm install express@4.3.1 --save
```

Hello World

Express is web application framework for node, it makes it really easy to write web application using node.

Write in a file named **main.js**:

```
var express = require("express");
var app = express();

app.get('/', function(req, res) {
  res.send('Hello World!');
});

var port = Number(process.env.PORT || 5000);
app.listen(port, function() {
  console.log("Listening on " + port);
});
```

This code will simply create an application using Express. Define an handling method for a get request on the root path. And start the web server on the port defined by the environment variable `PORT` or `5000`.

Run Script

Heroku convention needs a `Procfile` file that defines how to start the application. You can learn more about **Procfile** in the Heroku documentation.

Write in a file named **Procfile**:

```
web: node main.js
```

This file simply tell Heroku that for starting this applciation, it needs to start a web dyno by running the command `node main.js`.

Testing

Now that our application is coded, we need to run it locally for testing. To follow Heroku convention, we are not going to run `node main.js` directly but we are going to use **Foreman**.

Foreman is tool to run and manage procfile based applications.

Installation of Foreman

If you have...	Install with...
Ruby (MRI, JRuby, Windows)	$ gem install foreman
Mac OS X	Download and install: foreman.pkg

Running the application

Starting our applciation locally using foreman is really simple, run the command:

```
$ foreman start
```

You can now open a browser and take a look at **http://localhost:5000**.

Deployment on Heroku

We just test our application, and it's working fine. Now it's time to deploy it on Heroku to make it available for everybody.

Create an application on Heroku

If you haven't already, sign up on Heroku. It's free and easy!

Click on the button "Create a new app" and enter a name for your application. You can select the region that you want, it doesn't change anything for the deployment.

Install the Heroku Command Line Tool

First, install the Heroku Toolbelt on your local workstation. You can find it at toolbelt.heroku.com.

Commit your changes

The first is to commit your changes (main.js and Procfile):

List all the changes that needs using:

```
$ git status
```

You can see that the folder `node_modules` is contained in the list. This folder contains all the dependencies, we installed using **NPM**. We need to add this folder to a `.gitignore` file:

Copy and paste the content of GitHub Node .gitignore to a file named `.gitignore`.

If you run `git status` again, you can see that `node_modules/` is no longer present in the list.

You can now commit the other changes using:

```
$ git add .
$ git commit -m "Base code"
```

Pushing to Heroku

It is now time to push to Heroku. In the configuration or homepage of your Heroku application, you can see a GIT url with the following format:

```
git@heroku.com:{{ application name }}.git
```

To deploy a new release of your application, simply run:

```
$ git push heroku master
```

Heroku will log the installation of the node dependencies and the launch of your application.

Once it's done you can open your application using:

```
$ heroku open
```

Configuration

In this chapter, we'll learn how to manage different configurations for our applications (locally and in production).

Environment variables

The best way to configure an application on Heroku is to use **environment variables**. It's a key value storage managed by the system that can affect the way running processes will behave on a computer.

Exemple of an env variable:

```
MESSAGE=Hello World
```

You can list all current environment variables using the command `env`.

Heroku defines by default 2 environement variables:

- `PORT` which equals the port our application should be running on.
- `DYNO` which gives you a id/name for the current process dyno.

With Node.js

In Node.js, it's really easy to read environment variable, the varibale `process.env` is an object containing all current env variables.

We already used it to start our application on the right port: `var port = Number(process.env.PORT || 5000);`.

Notice that environment variables are always **string**.

Modifying our application

We are going to change our application to show a message instead of "Hello World" that will be stored in a en environment variables.

Edit the **main.js** file to change the `app.get` to:

```
app.get('/', function(req, res) {
  res.send(process.env.MESSAGE || 'Default message!');
});
```

If you run the application using `foreman start` and access `http://localhost:5000`, you'll see : `Default message!`.

But you can test changing the value of **MESSAGE** in your terminal and running the application with:

```
$ export MESSAGE=Hello
$ foreman start
```

Storing a fixed configuration for foreman

You don't want to define using `export` our all configuration each time you want to start working on your application!

So we need to store our configuration in a file. By default foreman use a file named `.env` but we are going to use this file for your production configuration.

So we'll store our configuration in a file named `.env.local`:

```
MESSAGE=Hello from the local version
```

And we need to update foreman configuration by writting the file `.foreman`:

```
port: 5000
env: .env.local
```

You can then test using `foreman start` and see teh output: `Hello from the local version`.

Deployment of a production configuration

We are going to store our production configuration in a file named `.env`:

```
MESSAGE=Hello from the production version
```

Then we need to commit all these changes and deploy the last update of our code to Heroku:

```
# Commit changes
$ git add .
$ git commit -m "Use environment variables as configuration"

# Deploy to heroku
$ git push heroku master
```

But if you take a look at your application (using `heroku open`), you can see that the message is still "Default message!". It's because we didn't pushed your configuration to heroku yet.

For this we are going to use the plugin heroku-config, install it using:

```
$ heroku plugins:install git://github.com/ddollar/heroku-config.git
```

And then we can push our all configuration using:

```
$ heroku config:push
```

Now take a look at your application and you'll see "Hello from the production version".

Managing Heroku configuration by hand

I want to...	Command
List all my configuraton	`heroku config`
Get a variable value	`heroku config:get MESSAGE`

Set a variable value	`heroku config:set MESSAGE=Test`
Delete a variable	`heroku config:unset MESSAGE`

And with the plugin **heroku-config**:

I want to...	Command
Push my .env to heroku	`heroku config:push`
Update my .env with my heroku config	`heroku config:pull`
Rewrite my .env with my heroku config	`heroku config:pull --overwrite`