Katie Griffiths

T00175748

Development Operations

Table of Contents

[Introduction 2](#_Toc513144367)

[Deploying Applications 3](#_Toc513144368)

[Building Applications 3](#_Toc513144369)

[Setting up Heroku 4](#_Toc513144370)

[Prepare the app 7](#_Toc513144371)

[Deploy the app 7](#_Toc513144372)

[View logs 9](#_Toc513144373)

[Define a Procfile 10](#_Toc513144374)

[Scale the app 10](#_Toc513144375)

[Declare app dependencies 11](#_Toc513144376)

[Run the app locally 13](#_Toc513144377)

[Push local changes 13](#_Toc513144378)

[Provision add-ons 15](#_Toc513144379)

[Start a console 16](#_Toc513144380)

[Define config vars 17](#_Toc513144381)

[Provision a database 18](#_Toc513144382)

[Continuous Integration Pipeline 21](#_Toc513144383)

[Pipeline 21](#_Toc513144384)

[Develop 22](#_Toc513144385)

[Sublime IDE 22](#_Toc513144386)

[Manage 23](#_Toc513144387)

[Git 23](#_Toc513144388)

[Database 23](#_Toc513144389)

[Alerts 24](#_Toc513144390)

[Build & Test 25](#_Toc513144391)

[BlazeMeter 25](#_Toc513144392)

[Nightwatch.js 27](#_Toc513144393)

[Host 31](#_Toc513144394)

[Heroku Deploy 31](#_Toc513144395)

[Summary and Conclusions 33](#_Toc513144396)

[References 34](#_Toc513144397)

GitHub Link: <https://github.com/katieg-itt/MongoDB-book_depo>

# Introduction

Heroku is an elastic, multi-language, multi-framework, platform as a service. It is part of the Salesforce Platform. Founded in 2007 enabling start-ups to become successful and a focus on high quality platform cloud service for developers, Heroku was acquired by Salesforce in 2010, supporting innovation in the enterprise and extending possibilities within the Salesforce platform. Heroku allows you to build your custom apps and deploy them using Git, you are able to create apps with a range of languages and databases and they all run in the Cloud, you can manage your choice of application framework – Rails, Nodejs, Spring, etc., it has elastic scalability so you can get all the resources you want, only when you want them. (Developers, 2013)

Heroku’s first class languages are Ruby, Node (JavaScript), Python and Java, it gives you developer driven deployment which means you can deploy using Git version control which is the most popular tool for developers using the command git push Heroku master. Heroku gives you instant and continuous deployment so you are able to deploy changes at the speed of business change. Repeatable and traceable deployment so you can easily manage versions across multiple environments (dev, test, staging, production), code commits drive application releases. Heroku gives you Dyno - an abstraction over infrastructure which is a simplifies definition of resources, runs one or more application processes and requires no patching or other maintenance a typical (virtual) server would need. Web and background processes can scale at different levels. You can instantly deploy your app over multiple Dynos and scale down during quiet periods to save costs. Customisable deployment though open source “buildpacks”, there is a 24x7 platform monitoring, automated patching and automated dyno management service. (Developers, 2013)

Heroku gives you fast Rollbacks allowing you to restore a previous version whilst you do root cause analysis and gives you a large collection of add-ons for you to build your application faster with on-demand services. (Developers, 2013)

For this development project I chose to follow Heroku’s own tutorial as a guide to building and deploying the development operations project. I felt this was the best tutorial to follow due to it being Heroku’s own Nodejs tutorial. Due to my project being quite different to the sample demonstrated by Heroku, I encountered a few difficulties which I was able to resolve by following additional tutorials I found online.

Deploying Applications

Git, a powerful, distributed version control system is the most common method used for deploying applications. In addition to the usual features that make it so great for developers, it also supports web hooks that can trigger a remote API when changes are commited to a branch. As an alternative, you can manually trigger a deploy from within Heroku (there are other ways to transport your source code to Heroku, including via an API). When you create an application on Heroku, it associates a new Git remote, typically named heroku, with the local Git repository for your application. (Center, 2018)

As a result, deploying code is just the familiar git push, but to the heroku remote instead:

$ git push heroku master

There are many other ways of deploying applications too. For example, you can enable GitHub integration so that each new pull request is associated with its own new application, which enables all sorts of continuous integration scenarios. Or you can use Dropbox Sync, which lets you deploy the contents of Dropbox folders to Heroku. Finally, you can also use the Heroku API to build and release apps. (Center, 2018)

Deployment then, is about moving your application from your local system to Heroku - and Heroku provides several ways in which apps can be deployed. (Center, 2018)

Building Applications

When the Heroku platform receives the application source, it initiates a build of the source application. The build mechanism is typically language specific, but follows the same pattern, typically retrieving the specified dependencies, and creating any necessary assets (whether as simple as processing style sheets or as complex as compiling code). (Center, 2018)

For example, when the build system receives a Rails application, it may fetch all the dependencies specified in the Gemfile, as well as generate files based on the asset pipeline. A Java application may fetch binary library dependencies using Maven, compile the source code together with those libraries, and produce a JAR file to execute. (Center, 2018)

In the case of my project, written using NodeJs, Heroku can follow the instructions provided by the package.json file that npm creates. This includes ensuring dependancies are installed and running any user defined npm actions.

The source code for your application, together with the fetched dependencies and output of the build phase such as generated assets or compiled code, as well as the language and framework, are assembled into a slug. A slug is a bundle of your source, fetched dependencies, the language runtime, and compiled/generated output of the build system - ready for execution. (Center, 2018)

These slugs are a fundamental aspect of what happens during application execution - they contain your compiled, assembled application - ready to run - together with the instructions (the Procfile) of what you may want to execute. (Center, 2018)

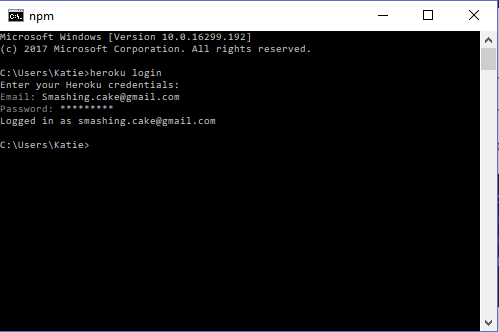
### Setting up Heroku

In this step you will install the Heroku Command Line Interface (CLI), formerly known as the Heroku Toolbelt. You will use the CLI to manage and scale your applications, to provision add-ons, to view the logs of your application as it runs on Heroku, as well as to help run your application locally. (Center, 2018)

Download the Heroku CLI



Once installed, you can use the heroku command from your command shell. On Windows, start Command Prompt (cmd.exe) or Powershell to access the comman shell. Login in using the email address and password that was used when creating your heroku account:



Authenticating is required to allow both the heroku and git commands to operate. Before you continue, check that you have the prerequisites installed properly. Type each command below and make sure it displays the version you have installed. (Your version may be different from the example.) If no version is returned, go back to the introduction of the tutorial. (Center, 2018)

All of the following local setup will be required to complete the “Declare app dependencies” and subsequent steps. (Center, 2018)

This tutorial will work for any version of Node greater than 4 or so - check that it’s there:

node -v

v5.9.1

npm is installed with Node, so check that it’s there. If you don’t have it, install a more recent version of Node:

npm -v

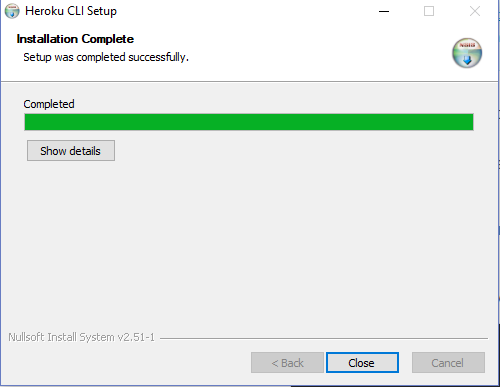
3.7.3

Now check that you have git installed. If not, install it and test again.

git --version

git version 2.2.1

Installation Complete



### Prepare the app

In this step, you will prepare a simple application that can be deployed. (Center, 2018)

To clone the sample application so that you have a local version of the code that you can then deploy to Heroku, execute the following commands in your local command shell or terminal:

$ git clone https://github.com/heroku/node-js-getting-started.git

$ cd node-js-getting-started

You now have a functioning git repository that contains a simple application as well as a package.json file, which is used by Node’s dependency manager. (Center, 2018)

### Deploy the app

In this step you will deploy the app to Heroku. (Center, 2018)

Create an app on Heroku, which prepares Heroku to receive your source code.

$ heroku create

Creating warm-waters-74555... done, stack is cedar-14

http://sharp-rain-871.herokuapp.com/ | https://git.heroku.com/warm-waters-74555.git

Git remote heroku added

When you create an app, a git remote (called heroku) is also created and associated with your local git repository. (Center, 2018)

Heroku generates a random name (in this case warm-waters-74555) for your app, or you can pass a parameter to specify your own app name. (Center, 2018)

Now deploy your code:

$ git push heroku master

Counting objects: 343, done.

Delta compression using up to 4 threads.

Compressing objects: 100% (224/224), done.

Writing objects: 100% (250/250), 238.01 KiB, done.

Total 250 (delta 63), reused 0 (delta 0)

remote: Compressing source files... done.

remote: Building source:

remote:

remote: -----> Node.js app detected

remote:

remote: -----> Creating runtime environment

remote:

remote: NPM\_CONFIG\_LOGLEVEL=error

remote: NPM\_CONFIG\_PRODUCTION=true

remote: NODE\_MODULES\_CACHE=true

remote:

remote: -----> Installing binaries

remote: engines.node (package.json): 5.9.1

remote: engines.npm (package.json): unspecified (use default)

remote:

remote: Downloading and installing node 5.9.1...

remote: Using default npm version: 2.7.4

....

remote: -----> Build succeeded!

remote: ├── ejs@2.4.1

remote: └── express@4.13.3

remote:

remote: -----> Discovering process types

remote: Procfile declares types -> web

remote:

remote: -----> Compressing... done, 9.4MB

remote: -----> Launching... done, v8

remote: http://sharp-rain-871.herokuapp.com deployed to Heroku

To https://git.heroku.com/nameless-savannah-4829.git

\* [new branch] master -> master

The application is now deployed. Ensure that at least one instance of the app is running:

$ heroku ps:scale web=1

Now visit the app at the URL generated by its app name. As a handy shortcut, you can open the website as follows:

$ heroku open

### Define a Procfile

Use a Procfile, a text file in the root directory of your application, to explicitly declare what command should be executed to start your app. (Center, 2018)

The Procfile in the example app you deployed looks like this:

web: node index.js

This declares a single process type, web, and the command needed to run it. The name web is important here. It declares that this process type will be attached to the HTTP routing stack of Heroku, and receive web traffic when deployed. (Center, 2018)

Procfiles can contain additional process types. For example, you might declare one for a background worker process that processes items off of a queue. (Center, 2018)

### Declare app dependencies

Heroku recognizes an app as Node.js by the existence of a package.json file in the root directory. For your own apps, you can create one by running npm init --yes. (Center, 2018)

The package.json file determines both the version of Node.js that will be used to run your application on Heroku, as well as the dependencies that should be installed with your application. When an app is deployed, Heroku reads this file and installs the appropriate node version together with the dependencies using the npm install command. (Center, 2018)

### Provision add-ons

Add-ons are third-party cloud services that provide out-of-the-box additional services for your application, from persistence through logging to monitoring and more. (Center, 2018)

### Start a console

You can run a command, typically scripts and applications that are part of your app, in a one-off dyno using the heroku run command. It can also be used to launch a REPL process attached to your local terminal for experimenting in your app’s environment:

$ heroku run node

Running `node` attached to terminal... up, run.2132

Detected 512 MB available memory, 512 MB limit per process (WEB\_MEMORY)

Recommending WEB\_CONCURRENCY=1

>

If you receive an error, Error connecting to process, then you may need to configure your firewall. (Center, 2018)

When the console starts, it has nothing loaded other than the Node.js standard library. From here you can require some of your application files. (Center, 2018)

For example, you will be be able to run the following:

> var cool = require('cool-ascii-faces')

> cool()

( ⚆ \_ ⚆ )

To get a real feel for how dynos work, you can create another one-off dyno and run the bash command, which opens up a shell on that dyno. You can then execute commands there. Each dyno has its own ephemeral filespace, populated with your app and its dependencies - once the command completes (in this case, bash), the dyno is removed. (Center, 2018)

$ heroku run bash

Running `bash` attached to terminal... up, run.3052

~ $ ls

Procfile README.md composer.json composer.lock vendor views web

~ $ exit

exit

Don’t forget to type exit to exit the shell and terminate the dyno. (Center, 2018)

### Define config vars

Heroku lets you externalise configuration - storing data such as encryption keys or external resource addresses in config vars. (Center, 2018)

At runtime, config vars are exposed as environment variables to the application. For example, modify index.js so that it introduces a new route, /times, that repeats an action depending on the value of the TIMES environment variable:

app.get('/times', function(request, response) {

var result = ''

var times = process.env.TIMES || 5

for (i=0; i < times; i++)

result += i + ' ';

response.send(result);

});

heroku local will automatically set up the environment based on the contents of the .env file in your local directory. In the top-level directory of your project there is already a .env file that has the following contents:

TIMES=2

If you run the app with heroku local, you’ll see two numbers will be generated every time.

To set the config var on Heroku, execute the following:

$ heroku config:set TIMES=2

View the config vars that are set using heroku config:

$ heroku config

== sharp-rain-871 Config Vars

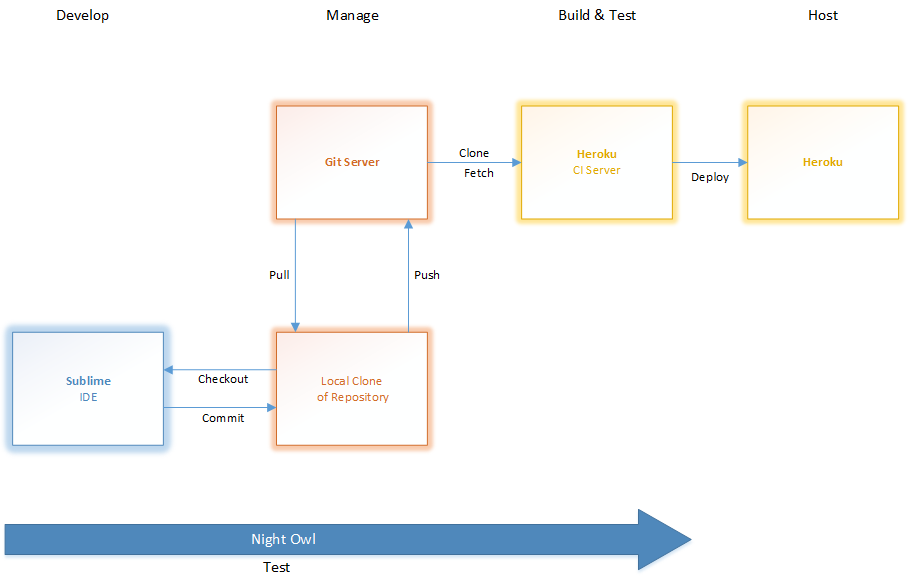
PAPERTRAIL\_API\_TOKEN: erdKhPeeeehIcdfY7ne

TIMES: 2

Deploy your changed application to Heroku and then visit it by running heroku open times. (Center, 2018)

# Continuous Integration Pipeline

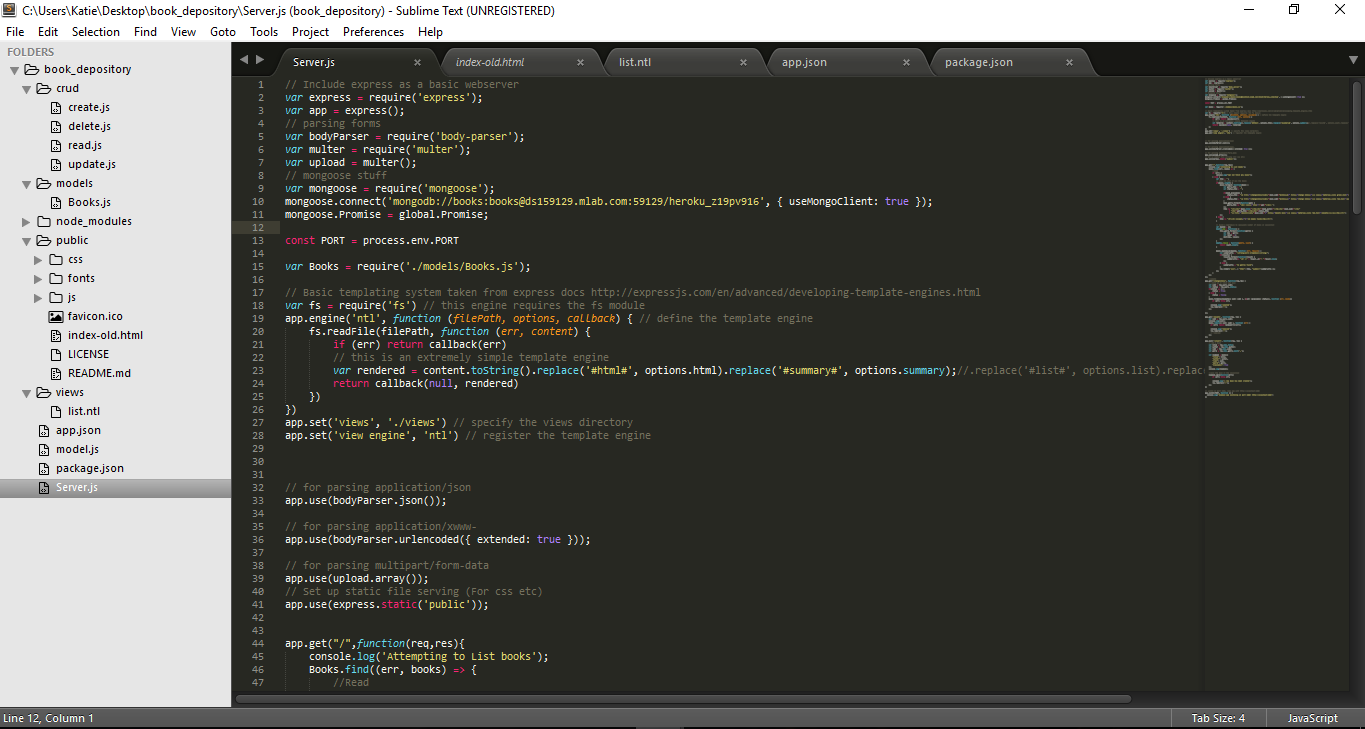
Pipeline



Develop

### Sublime IDE

I decided to create a web application in Node.js using Sublime Text. It is a proprietary cross-platform source code editor with a Python application programming interface. It natively supports many programming languages and mark-up languages, and functions can be added by users with plugins.



The application that I created is based on a book depository that enables the user to insert and delete books. I used Sublime to build the web application.



Manage

### Git

Git was used to produce a local clone of the repository. For this I entered the following commands into the command line tool:

$ git clone https://github.com/katieg-itt/MongoDB-book\_depo.git

$ cd MongoDB-book\_depo

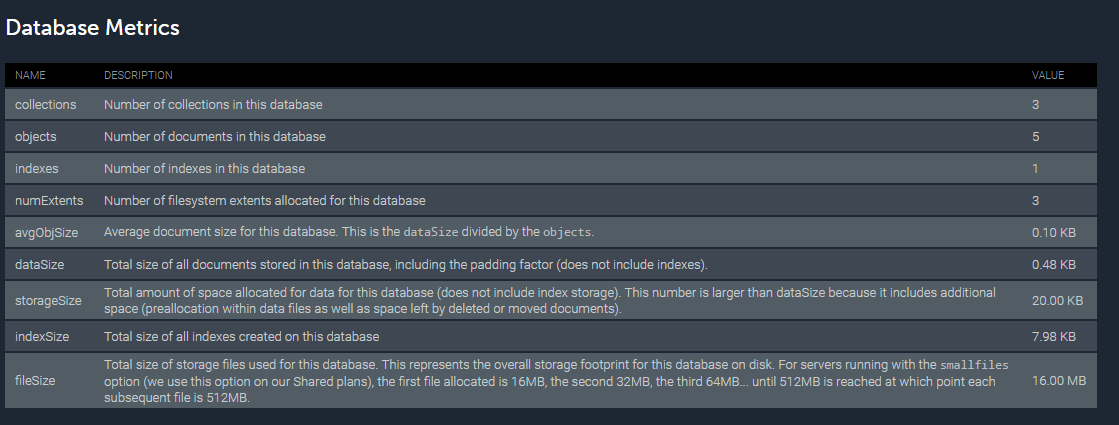
Once this was completed I had a local repository containing my application as well as a package.json file, this is used by Node’s dependency manager.

### Database

To run the app a database was configured through the heroku add-on to allow for the application to run. mLab is a fully managed cloud database service featuring automated provisioning and scaling of MongoDB databases, backup and recovery, 24/7 monitoring and alerting, web-based management tools, and expert support. mLab’s Database-as-a-Service platform powers hundreds of thousands of databases across AWS, Azure, and Google and allows developers to focus their attention on product development instead of operations. Through the mLab add-on, Heroku users can instantly have MongoDB databases running on Amazon EC2 and available for their Heroku applications. (mLab, 2018)

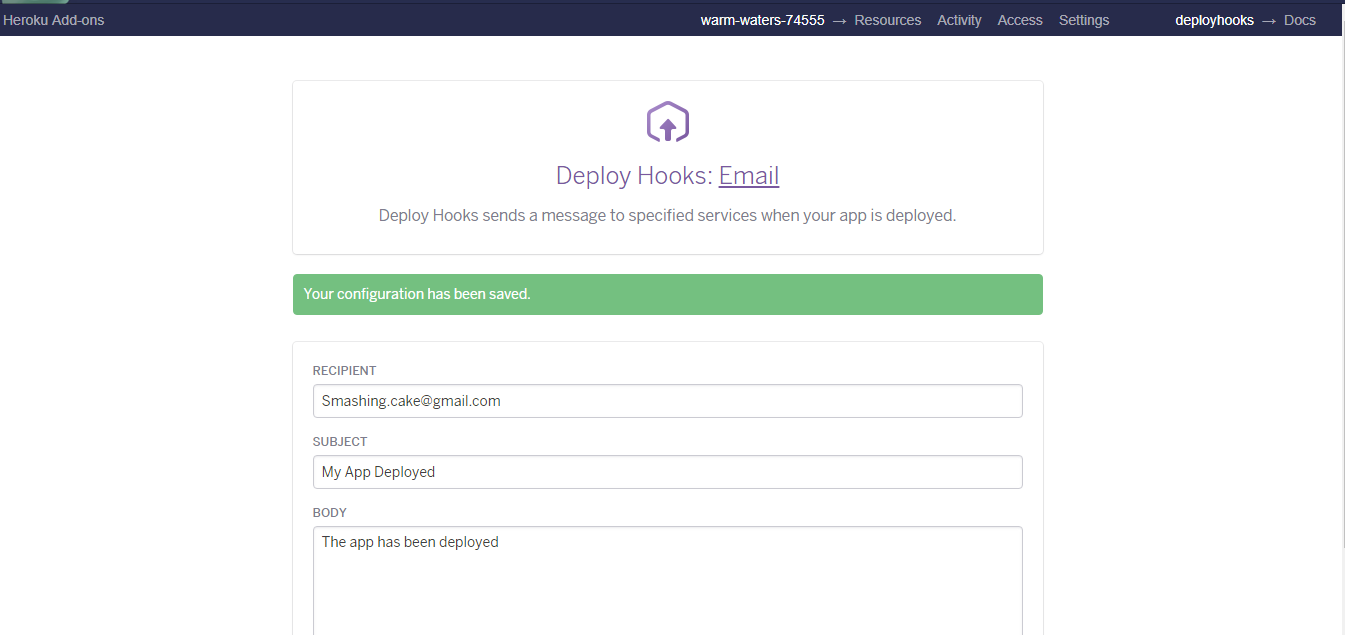
For mLab to be added to my heroku application I first had to add it though the heroku command using:

heroku addons:create mongolab



### Alerts

Once the application was running through heroku I set up an email alert system, to email my personal account each time the application was deployed. This was set up through a heroku add-on called Deploy Hooks this allows you to receive a notification whenever a new version of your app is pushed to heroku. It is a useful add-on to keep informed about deploys, it can also be used to integrate different systems together. Once the add-on is added to the heroku app, the dash board of Deploy Hooks can be configured to tailor the email that needs to be published. (Hooks, 2017)



Build & Test

### BlazeMeter

A Heroku add-on that provides an integrated self-service, on-demand, cloud-based, 100% JMeter compatible load and performance testing for your web and mobile apps. BlazeMeter’s self-service, web and mobile load testing platform (PaaS) provides developers an enterprise grade, out-of-the-box load testing solution that is 100% compatible with Apache JMeter™.

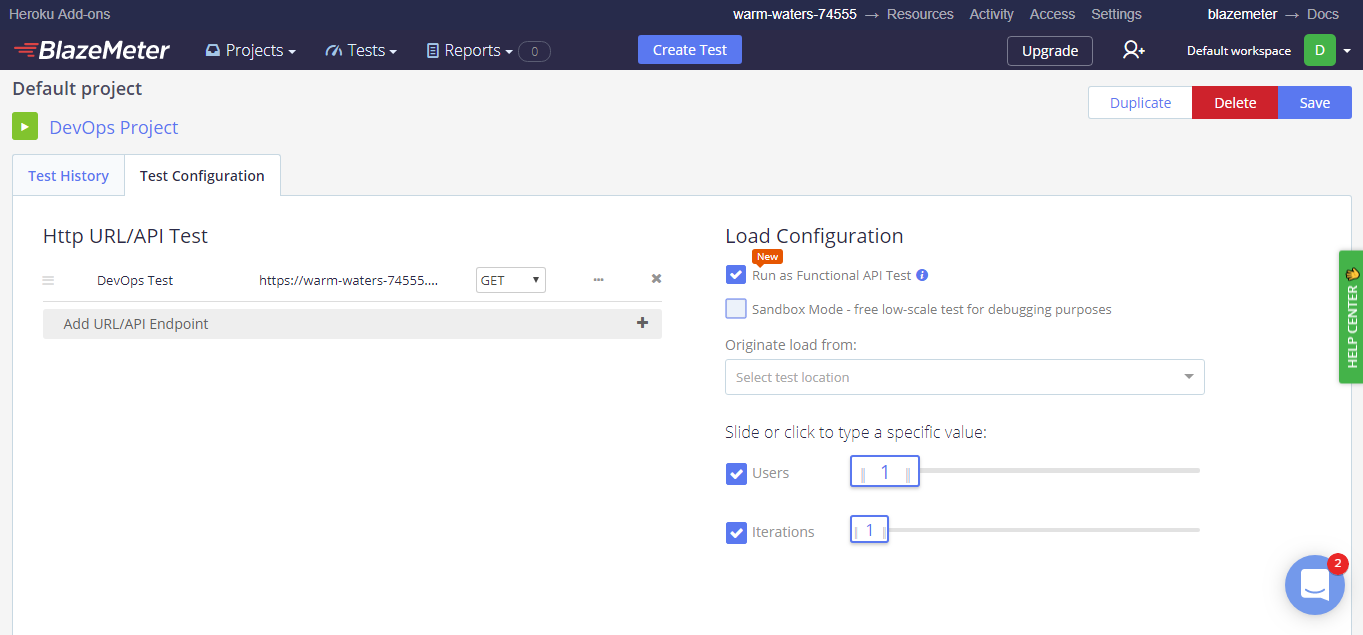
BlazeMeter provides developers simplified integration into their native development environment, realistic user simulation, advanced scripting capabilities, unlimited on-demand load testing capacity, comprehensive, interactive real-time reporting, sophisticated result analysis and geographically distributed testing across 8 different locations. Users can run multiple load tests in order to quickly locate and fix performance bottlenecks. BlazeMeter offers users a lifetime free tier of 10 hours and up to 50 concurrent users, per month. (See all BlazeMeter Monthly and Hourly (on-demand) pricing plans). (BlazeMeter, 2017)

BlazeMeter also has an integration with New Relic which provides application level monitoring while you run a performance test. (BlazeMeter, 2017)

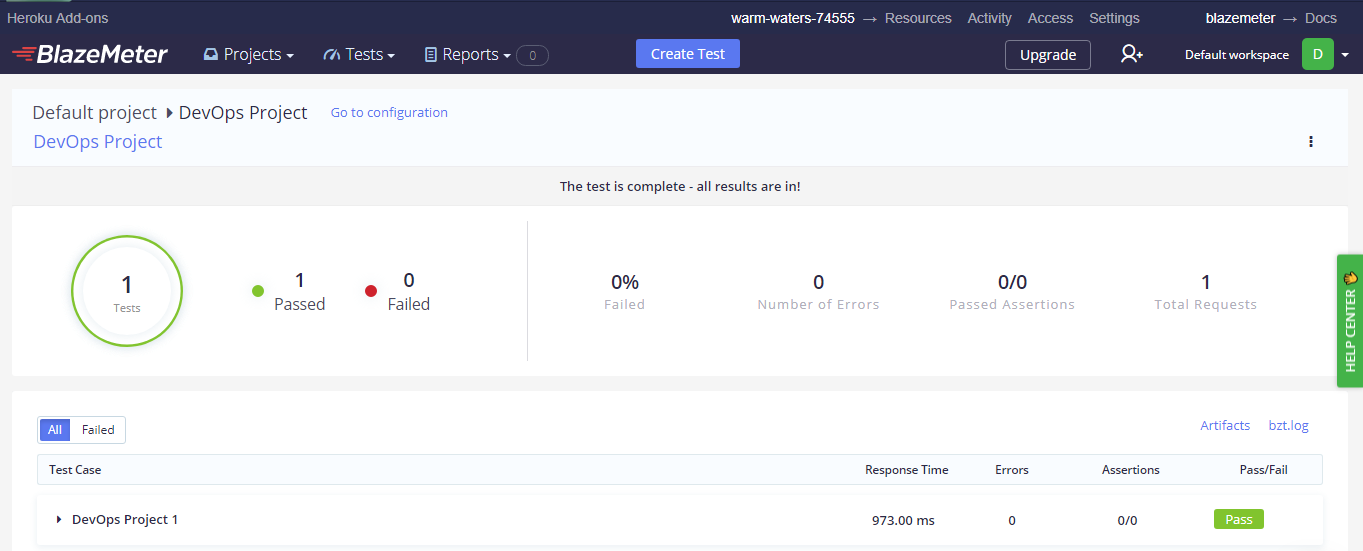
Installing BlazeMeter to heroku app:

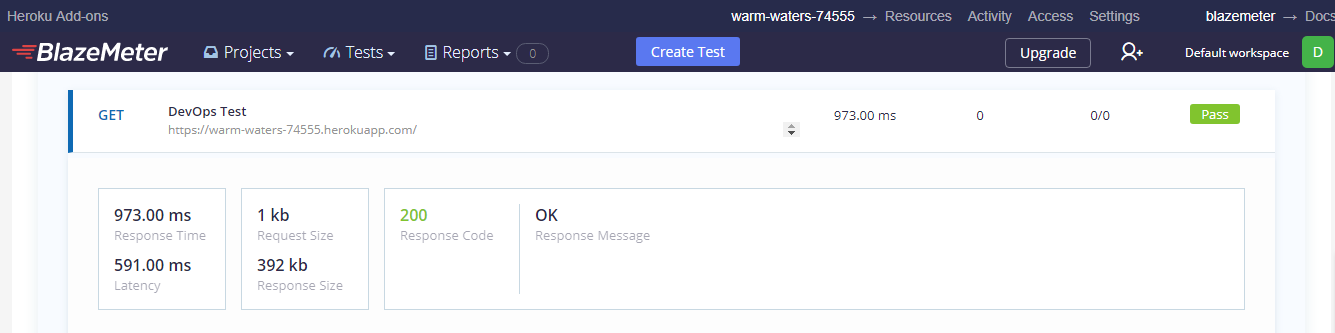
heroku addons:create blazemeter

Once BlazeMeter is installed I nagivated to the BlazeMeter dashboard. Once there I clicked on “Add Test”. This then allows me to create a new test and to complete the test configuration. After the test has been created. I created a single GET test that tests in retrieving the application from the specified URL given. (BlazeMeter, 2017)



Results of the test:







### Nightwatch.js

This is an automated testing framework for web applications and websites. It is written in Node.js and uses the W3C WebDriver API (formally Selenium WebDriver). (Nightwatch.org, 2015)

It is a complete end to end testing solution which aims to simplify the process of setting up Continuous Integration and the writing of automated tests. The Web Driver is a general purpose library for automating web browsers, Nightwatch uses the web driver api to perform the browser automation related tasks e.g. opening windows or clicking on links. (Nightwatch.org, 2015)

I installed nightwatch on my system by using the following command:

$ npm install nightwatch --save

Nightwatch primerally operates using Selenium, but it also requires individual drivers to be installed for each browser, as the tool runs tests through an automated browser session. For this project I opted to just use the Chrome driver as I was not too concerned about the site not working on all browsers. In production of a real site though, this could be very important as sometimes there are differences between how different browsers treat javascript or css. To install Selenium and the chrome web driver (to allow testing of the site through chrome) I entered the following command:

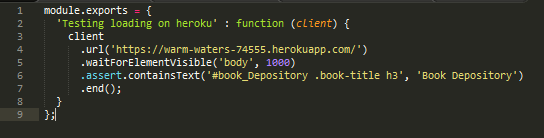
$ npm install selenium chromedriver --save

The save option in the commands make sure that the package is added to the package.json file, which makes it easy to set up the project as typing npm install first checks this file and then installs all the dependencies, which is vital to ensure that the project works as expected when setting it up on a new system (for example, on a Heroku instance).

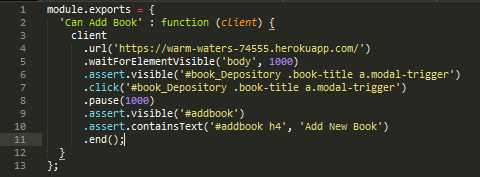


The nightwatch.conf.js file is used for the configuration set up, containing the src folder where the tests are located as well as the information for the selenium and location of the drivers.

The first test that was written was to check that the web application would load, the test contains the URL for the site then waiting a second then it makes sure that when the page loads that the page contains a title called Book Depository. Nightwatch tests were quite intuitive to write, as their asserts can use css selectors directly.



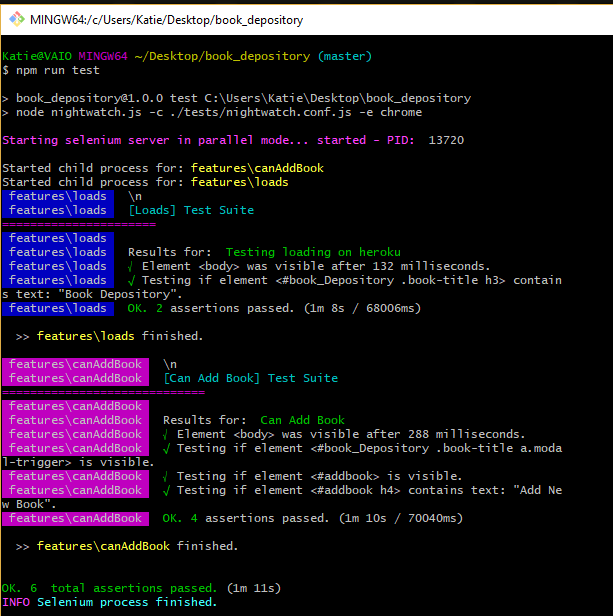
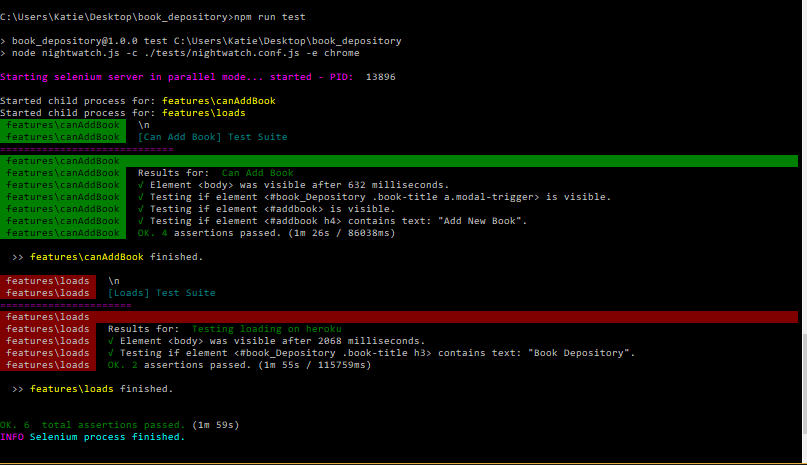
The second test contains a function that checks that the site loads from the URL location, then waits for a second to make sure the title is visible before clicking on the button to add a book, then checks that panel has the title Add New book.



For the tests to be instantiated they must be called in the package.json file under scripts, it also contains the script to start the server.



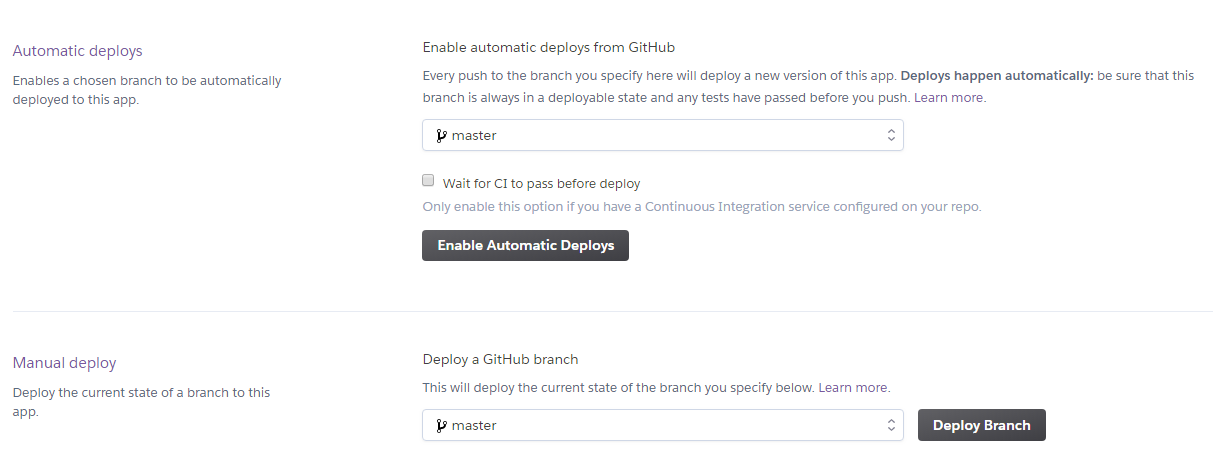
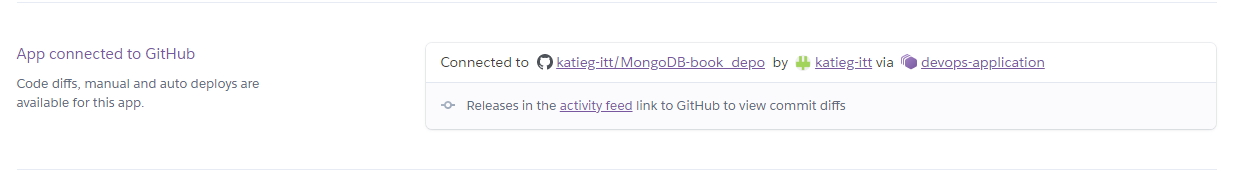
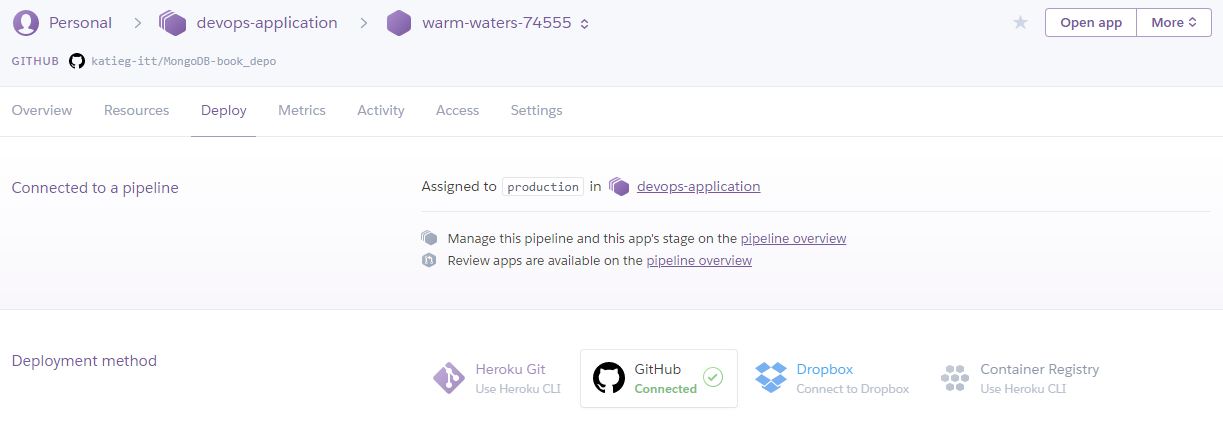
The results of the tests being run locally are as follows. Both tests are loaded through the command npm run test, this then loads both tests and runs them.



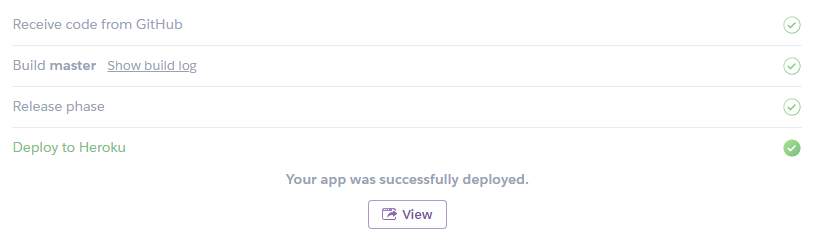
Host

### Heroku Deploy

For an app to deploy on Heroku, it first must be on the production pipeline, this can be done through the Heroku dashboard. Once the pipeline has been altered, the application named warm-waters-74555 (which is given random) can then be deployed through the deploy section in the dashboard. This is available once you set up an account and connect through GitHub. The deploy retrieves the source code, builds and if a successful build it will then deploy automatically. When tests are present, the application can be set to deploy and test automatically.



Once the deploy button has been selected, Heroku retrieves the code from the GitHub repository it then builds on the master and deploys to Heroku. Image shows the deploy as being successful.



# Summary and Conclusions

The first issue is you can't just pick a port and listen on it like you do normally.

Heroku dynamically assigns a port, which can change between individual runs. This required learning about environmental variables, and using them with express.

Heroku relies on start-up scripts which you don't need locally when you are typing node server.js, so these had to be stored, along with the node and nmp versions in my package.json file. Finally using a mongo database required attaching an add-on to my account and I needed to change the connection info within my app to reflect this.

As when I originally created my app i already had info in my mongo database I had to modify the code slightly to resolve a number of crashes that were caused by empty or non-existent collections.

Overall, I found quite a lot of conflicting information out there, much of which did not work, this was likely not helped by my lack of Node experience. While I was able to run nightwatch locally, I had difficulty including it as part of my CI flow as Selenium is written in Java, which a node instance does not appear to support. I did find some guides relating to this, and Heroku states that Selenium works there, but I was not able to get a solution working.

I can see fantastic benefits to using a system like Heroku, once any initial setup issues are solved, deployments are as simple as pushing to git, which is something that’s part of your workflow anyway. By ensuring that your project has good test coverage, you can avoid accidentally deploying buggy code.

# References

BlazeMeter, 2017. *BlazeMeter.* [Online]   
Available at: https://a.blazemeter.com/app/#/accounts/220925/workspaces/214858/dashboard  
[Accessed 2018].

Center, H. D., 2018. *Getting Started on Heroku with Node.js.* [Online]   
Available at: https://devcenter.heroku.com/articles/getting-started-with-nodejs#set-up  
[Accessed 2018].

Developers, S., 2013. *Introduction to Heroku.* [Online]   
Available at: https://www.slideshare.net/developerforce/df13-introduction-to-heroku  
[Accessed 20 April 2018].

Hooks, D., 2017. *Deploy Hooks.* [Online]   
Available at: https://deploy-hooks.herokuapp.com/  
[Accessed 2018].

mLab, 2018. *mLab.* [Online]   
Available at: https://www.mlab.com/databases/heroku\_z19pv916#stats  
[Accessed 2018].

Nightwatch.org, 2015. *Nightwatch.* [Online]   
Available at: http://nightwatchjs.org/gettingstarted/  
[Accessed 1 May 2018].