Node Deployment Workshop

Thomas Mortensson and Graham Laming
Computer Science Department,
University of Bristol,
tm0797@bristol.ac.uk, gl1646@bristol.ac.uk

February 27, 2015

1 Introduction to Node - What is it?

Node.js is a platform built on Chrome's JavaScript runtime for easily building fast, scalable network applications. Node.js uses an event-driven, non-blocking I/O model that makes it lightweight and efficient, perfect for data-intensive real-time applications that run across distributed devices. One such application could be a socket based reactive chat application. In this deployment session we will aim to build and deploy a Node.js based web application using Socket.IO on top of the brilliant Koding.com hosting environment. With the skills gained from this workshop you should be equipped to go out to build and deploy your own reactive web applications on a hosted environment such as Koding or in Cloud environments such as DigitalOcean or locally on your own Raspberry Pi!

2 Getting the source

All content explained in this workshop is contained in an easily deployable repository online. This can be accessed at:

`https://github.com/thomasmortensson/node-deployment-workshop`

You can browse through this repository at your own leisure if you wish to go back over things we have covered in the workshop or just wish to have a simple base application to start with. Using the instructions in the README.md file you should be able to easily deploy this application on Koding.com or in a DigitalOcean droplet.

3 What's next?

In this workshop we have learnt some of the basics of web development. We have touched upon HTML, Javascript and CSS as well as some back-end technologies such as Apache serving, Node.JS and Socket.IO. The following deployment sections will show you how to load the application you have created (or the one from our repository) onto a cloud service or your own Raspberry Pi server.

4 Deployment

The Koding system uses an operating system called Ubuntu which is similar to that of the Raspberry Pi. All of the commands supplied below are compatible with the Raspbian distribution bundled on the Raspberry Pi.

4.1 Accessing the SSH remote terminal

- Windows — Download and install Git. `http://git-scm.com/download/win`. Open Programs → Git Bash
- Mac — Navigate to /Applications/Utilities/Terminal.
- Raspberry Pi — Navigate to Accessories → LxTerminal.

4.1.1 Making the connection

- Your hostname —"XXX.koding.com"
- Username — ubuntu
- Password — emailed to you

Type into the terminal window:

```
ssh ubuntu@10.147.243.78
```

Hit enter. You will now be prompted to type a password. Type this in (You will not see the characters on screen) and hit enter. You should be seeing something similar to this:

```
Last login: Thu Jun  5 12:22:54 2014 from
    195.246.108.112
ubuntu@ip-10-147-243-78:~$
```

To exit type:

```
exit
```

A good place to go for help for basic Linux/Unix help is here:

`http://cssbristol.co.uk/tutorials/linux-unix/`

For an in depth overview of how to use these types of tools look at: `http://datatoolkit.org/documentation.pdf`

4.2 Uploading and Downloading files

To upload and download files from the machine we will use SFTP. This is a Secure version of the old FTP you may be used to. We will be using Filezilla although any SFTP client will work.

Using SFTP means you don't have to bother with command lines, so it's much easier for beginners. We recommend

downloading a program called Filezilla, It's available for free on Mac, Linux and Windows. (I would like to point out that any SFTP client will work just as well).

`https://filezilla-project.org/download.php?type=client`
This program should be pretty self explanatory, you will need the same information to connect as with SSH.

4.3 Ubuntu and apt-get

To install packages within Ubuntu or Debian we use apt-get from the command line. To install the packages we will need for the project open a terminal and type:

```
sudo apt-get update
sudo apt-get install npm git unzip
```

4.4 Getting our code on the server

If you just want a working copy of our code you can issue the following command to get a clean copy of the deployment:

```
git clone https://github.com/thomasmortensson/node-
    deployment-workshop.git
```

To upload your own code to the server, first create a zip folder with your work. Once you have this use SFTP to upload the file to the home directory. Once uploaded, login to the server with SSH and type:

```
unzip nameOfZip.zip
```

We can now change directories so we can interact with our code.

```
cd node-deployment-workshop
```

4.5 Node and npm

As we have generated a package.json file you can install all Node dependencies for the project with npm by typing:

```
npm install
```

You can add dependencies as you go by typing:

```
npm install --save <package>
```

4.6 Bower

To install Bower type:

```
sudo npm install -g bower
```

As we have generated the bower.json and .bowerrc files you can install all javascript dependencies for the project with Bower by typing:

```
bower install
```

These will be installed in the www/components directory.

You can add dependencies as you go by typing:

```
bower install --save <package>
```

4.7 Running Node server

We can run our Node.JS server by issuing the command:

```
npm start
```

The start script is defined in the package.json file as: "node server.js"

4.8 Web Hosting