Node Deployment Workshop

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

February 28, 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 and those provided in the deployment section of this handout, 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, Node.JS and Socket.IO. The following section on deployment will show you how to load the application you have created (or the one from our repository) onto a cloud service or on your own Raspberry Pi.

4 Deployment

Koding.com is great, but (on the free tier) your server will power-down after 60 minutes of inactivity. In order to keep your application running, the server also needs to stay on - this is where cloud environments are often used. Here we will be discussing using DigitalOcean droplets, but any cloud service (Heroku, Amazon Web Services, Google Compute Engine, etc) will work just as well.

Keeping a small web server running is also a perfect application for a Raspberry Pi! However, running a web server from your own home to be accessible on the wider Internet (on a Raspberry Pi or desktop PC) has other complications as detailed here: `http://bit.ly/1AhPEK8`

The Koding system uses an operating system called Ubuntu which is similar to that of the Raspberry Pi. When we talk about cloud service providers we will assume that you have chosen to use Ubuntu as the operating system for your server. All of the commands supplied below are written for use on Ubuntu but are compatible with the Raspbian distribution commonly used on the Raspberry Pi.

4.1 Accessing the remote terminal via SSH

Koding.com provided us a 'terminal' tab in order to execute commands on our web server. When using traditional cloud services or running on a Raspberry Pi, this convenience is not available and so you will need to SSH into the server. The recommend programs to use for each operating system are given below:

- 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.

Note that it is also possible to connect to a Koding VM via SSH however the process requires the use of SSH keys. This is explained in detail here: `http://learn.koding.com/guides/ssh-into-your-vm`

4.1.1 Making the connection

You'll need the following information to SSH into your server:

- The server hostname or IP address e.g. "102.234.215.48", "ec2-102-234-215-48.eu-west-1.compute.amazonaws.com"
- The username to login as e.g. "root", "ubuntu", "glaming"

- The password either emailed to you or previously input by yourself at some point

Type into the terminal window:

```
ssh <username>@<hostname or ip address>
```

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

```
Last login: Thu Jun 5 12:22:54 from 54.236.18.124
ubuntu@ip-10-147-243-78:~$
```

When you're finished, to exit, type:

```
exit
```

If you're unfamiliar with using the terminal, a good place to go for help for basic Linux/Unix help is here: http://cssbristol.co.uk/tutorials/linux-unix/

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). It is available here: 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 authbind
sudo ln -s /usr/bin/nodejs /usr/bin/node
```

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
```

However, 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. If you used your own zip replace "node-deployment-workshop" with the name of your zip.

```
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

Once you've generated your creation you'll want to host it online. There are many options and we could write about this for days. Instead we will give you an overview of the options available to you.

- Node — A lovely little express server
- Apache — http://bit.ly/1ACEogv
- Python — http://bit.ly/1JWWFeB

We have included a simple express server serving on port 3001 in the project directory you can run it with:

```
node express.js
```

To run on the default port 80 you have to configure authbind:

```
sudo touch /etc/authbind/byport/80
sudo chown user /etc/authbind/byport/80
sudo chmod 755 /etc/authbind/byport/80
authbind node express.js
```