Using Node on the Engineering Servers

Intro

Node.js is a light weight web server. Alone it does very little but write receive requests and write back some simple text responses. But it is very extensible and there are tons of modules for it that make it into a very capable and fully featured web server. This guide will walk you through setting it up on the OSU Engineering server and connecting it to an Engineering provided SQL database.

Overview

In general things will be very similar. There are a few important things to note in terms of Node.js on the Engineering server. The first is that your engineering account is not preconfigured with the required demo applications so you will need to get those yourself. The next is that *the only way you will be able to see your site is if you are on the OSU VPN or you are physically connected to the on campus internet*. If you are not connected to the VPN you will not be able to access your site. You can find information on connecting to the VPN here. The other piece is that your MySQL database that you will be using will need to be made for you. This is usually done as part of a class and your instructor will provide you with the appropriate addresses and credentials to access it.

Gitting Files

You should make a new directory in your OSU engineering space. Then from within that directory run the command `git clone https://github.com/wolfordj/CS290-Server-Side-Examples.git ./`. This will clone the class files from the class Git repository into that directory. In this case it clones CS290 class examples because that is generally the class that makes the most use of this tool.

Making Required Changes

First go to the `diagnostic` directory and run `npm install`, this will install the required node.js packages to get the example to work.

Next you will need to tweak some files to work with the engineering setup. Because these examples are general there are a few template values that need to be replaced.

In particular in the `diagnostic` directory there is a file called dbcon.js.template. Replace the dbcon.js file with this one and change the the values appropriately. Then you need to edit the file, it should look like the following code with the square brackets and their contents replaced with the appropriate content:

```
var mysql = require('mysql');
var pool = mysql.createPool({
  connectionLimit : 10,
  host            : 'classmysql.engr.oregonstate.edu',
  user            : '[cs290_yourengrusername]',
  password        : '[last-4-digits-of-your-osu-id-number]',
  database        : '[cs290_yourengrusername]'
});

module.exports.pool = pool;
```

These are the default settings for all students who were enrolled when the term started. The templates are designed to most closely match CS290. Again, these values should be provided by your instructor. An example might look like this:

```
var mysql = require('mysql');
var pool = mysql.createPool({
  connectionLimit : 10,
  host            : 'mysql.eecs.oregonstate.edu',
  user            : 'cs290_smithj',
  password        : '1234',
  database        : 'cs290_smithj'
});

module.exports.pool = pool;
```

With that file renamed with the proper info added we are almost ready to go.

Ports and Persistence

Because this is running on a shared machine everyone cannot use port 3000. If you look at the diagnostic.js file you will see that the port 3000 was replaced with `process.argv[2]`. This means it will take the argument immediately after the filename when you start Node.js and use that as the port. So running `node diagnostic.js 5345` will start node running on port 5345. Everyone will need to use a unique port otherwise you will get an error that the port is in use.

forever

Finally is the topic of persistence. If you want to use *forever* there will be a few changes. You need to install it on a per application basis. So to install it for the *diagnostic* folder, navigate to in that folder and run `npm install forever`. Note that this does *not* use the global flag.

Then you will need to run forever by accessing the forever binary directly. So from within the diagnostic directory you would run `forever` using the command `./node_modules/forever/bin/forever`. If you look at most `forever` it tells you to run `forever start diagnostic.js` because it is usually installed globally. However, here you will run `./node_modules/forever/bin/forever start diagnostic.js 5678`. So we specify the path to the forever binary and also add a port number as an argument. You could view the page being served by visiting http://access.engr.oregonstate.edu:5678 while you are VPNed into the OSU network.

The Many Flips

And as a closing note, if you log into access.engr.oregonstate.edu you will randomly be put on flip1, flip2 or flip3. You can see which flip you are using the command `hostname` then you can switch flips by using the command `ssh flipX` where X is 1, 2 or 3. You need to be sure to log into the same flip every time because the node instance will only be running on one of them. So if you ever run `forever list` and don't see your instance running it means you are probably on the wrong flip.

Activity

You should be able to run the diagnostic and access the page it servers while VPNed onto the server. It will display a message that MySQL is working.

Review

This should get you into a position where you have a web server running and it shows you are connected to a database. In addition you should be able to continue to access the site via a browser even if you end your SSH session provided you are on campus or logged into the VPN.