**Installation Guide**

CIS 4911 - Senior Project (U01)

Event Driven Cloud Computing

Cory McAn

February 15, 2015

Jason Dettbarn

# Introduction

The purpose of this guide is to assist developers and users of Event Driven Cloud Computing backend with the installation of the project on their machines.

Instructions:

Step 1:

First we need to get the source files for the project into a directory where we can access them.

Open a shell program, navigate to the desired folder to store the project files, and execute the command to copy the project files into the folder:

Git clone <https://github.com/FIU-SCIS-Senior-Project-2015-Spring/Event-Driven-Cloud-Computing.git>

Step 2:

Now that we have the source files, we can view, edit, and upload them to Amazon Web Services where we will host our Lambda functions to be run on event triggers.

The accompanying User Manual for Event Driven Cloud Computing has the details related to getting AWS set up and using the Lambda functions in AWS. The actual ‘installation’ of the Lambda functions is similar for each piece of code in the Code/ folder. Please refer to the User Manual for this step.

Step 3:

Once you have set up Amazon Web Services, uploaded the Lambda functions and configured the functions to a bucket or buckets, you are ready to begin testing the functionalities.

The Lambda functions are written in Node.js, so it is necessary to find an IDE which can compile the code for IDE-side testing before doing a real run in AWS.

The IDE I used for this project is IntelliJ IDEA as it has strong debugging tools for node.js and plugins that integrate with AWS.

Navigate to this link and download the version that is relevant to your system: <https://www.jetbrains.com/idea/download/>

Step 4:

Next we need to install the Node.js interpreter, which you can get at the node website.

Navigate to: <https://nodejs.org/#download> and download the node framework onto your machine.

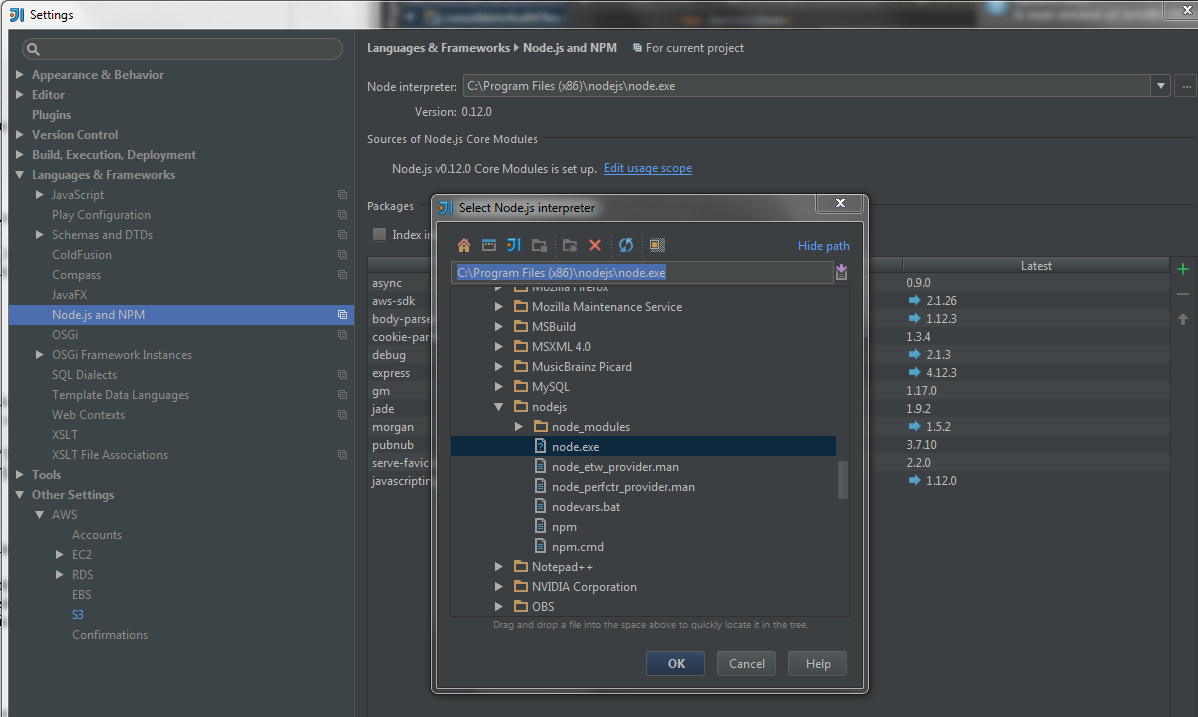
Step 5:

Now we will install the IDE and the Node Interpreter so that we can do any small tests using the IDE without needing to push new code to AWS.

With the IDE installed and open, File > Settings.

Under settings, click the arrow to the left of “Languages & Frameworks” > Node.js and NPM

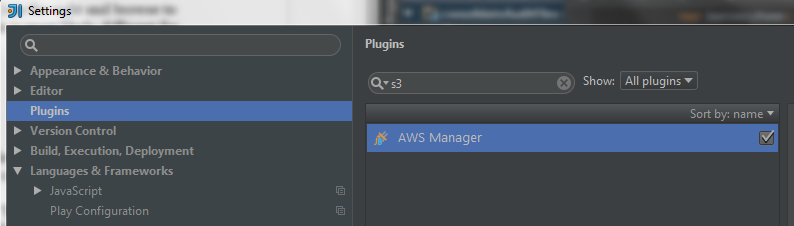
There is a box at the top for the Node Interpreter. Click the box (…) on the right and browse to node.exe (For Windows machines) and use it as the interpreter. (File is most likely different for other OS’s)



Step 6:

Now we can install the AWS S3 plugin for IntelliJ IDEA, which allows us to pass the AWS credentials through the plugin giving us a secure way of testing the code on the IDE-side.

In the same “Settings” window as in the previous step, navigate to “Plugins”. There is a search box at the top, simply search “s3” and the first result should be AWS Manager.

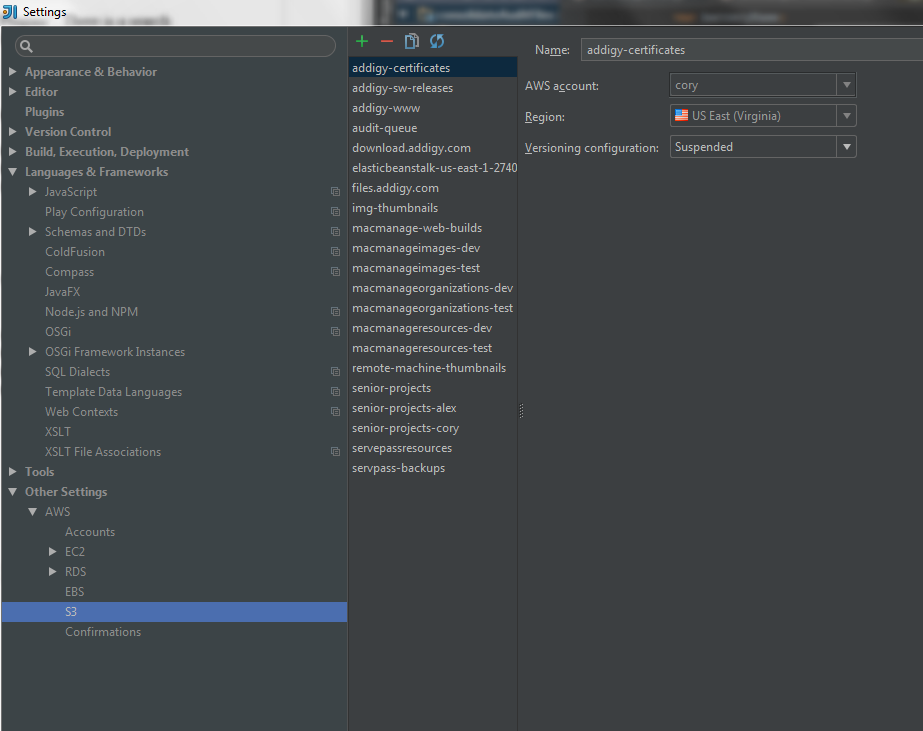


Step 7:

Click “Install JetBrains Plugin…”. Once installed, navigate to Settings > “Other Settings”.

Now AWS is available, and we can configure our login to AWS in the IDE.

Click Accounts and type in your credentials. Once you have the credentials click Test Connection. If your credentials are correct you should be able to click “S3” and see all the buckets available to your AWS account.



Step 8:

Now that we’ve fully set up our IDE with access to AWS, any of the code in the Code folder can be tested without interacting directly with the S3 web portal.

IMPORTANT: The code naming convention used for this project is simple: Either the code is the IDE version or the Lambda version for uploading to S3 and use in buckets. If the code has the suffix “Lambda” then it is configured for being used in a bucket, otherwise it is for testing in the IDE. Statements such as “context.” must be used in the bucket context, but are not recognized by the node framework in the IDE.

Step 9:

The user manual also has details for getting connected to PubNub for the purpose to testing the “multipleRunTester” which uploads many files to the bucket and gets alerts in PubNub console.

Here I will describe a method for accomplishing the multipleUpload to S3.

For this project I used CyberDuck, which has support for AWS and S3.

Navigate to <https://cyberduck.io/> and download CyberDuck for your OS.

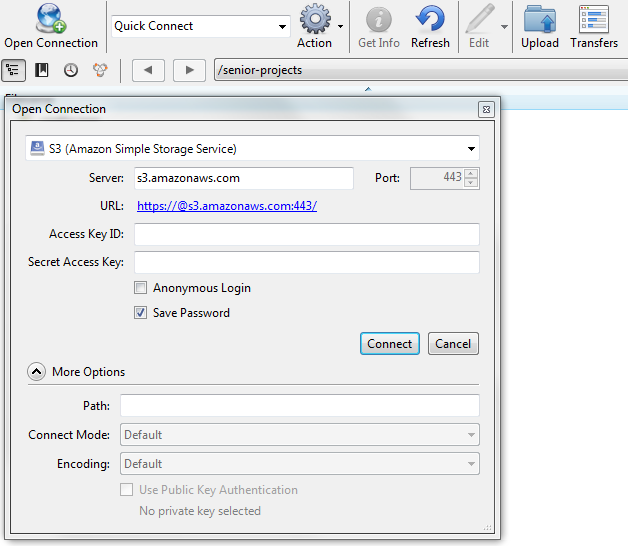
Once installed, all we need to do is configure our AWS login to be able to use CyberDuck for multipart uploads.

Click File > Open Connection…

In the drop down menu there are many different upload options. We want S3 (Amazon Simple Storage Service).

The server will be automatically changed for you. Now just put in your AWS credentials and click “Connect”.

Once you are connected, you will have a view with all the buckets linked to your AWS account.

Navigate to a bucket and now we can attempt a multi part upload!

Step 10:

If you’re at this step you should already have checked the User Manual for the details about logging into PubNub and using the Dedicated Console for testing the multiRunTester.

Have the PubNub Console open so you can see the alerts, and let’s try uploading sample files to the bucket! (Test files are located in Code/Test)

Click “Upload” near the top of CyberDuck, and navigate to the Test files. Choose one of the test folders and “Select All (Ctrl + A)” the files in the folder to upload all of them simultaneously.

CyberDuck will ask if you want to overwrite the files in the bucket, click OK.

Now look at the PubNub Console and you should see the alerts related to the files we have uploaded.

**Conclusion**

This concludes the Installation Manual for Event Driven Cloud Computing! Thank you for using this software and I hope this manual was helpful!