# Abstract

The purpose of this document is to demonstrate how developers can create a meaningful cloud native application using an agile and developer friendly framework. I’m calling this development stack “DyNode”. It is composed of: Angular, Node.js, ExpressJS, and DynamoDB.

To demonstrate this framework, I have created an application called AWSMovies. This application allows a user to search for movie data stored in an Amazon Dynamo database and display the results in both tabular and raw JSON format using JavaScript and Angular. The middle tier is implemented using Node.js and ExpressJS.

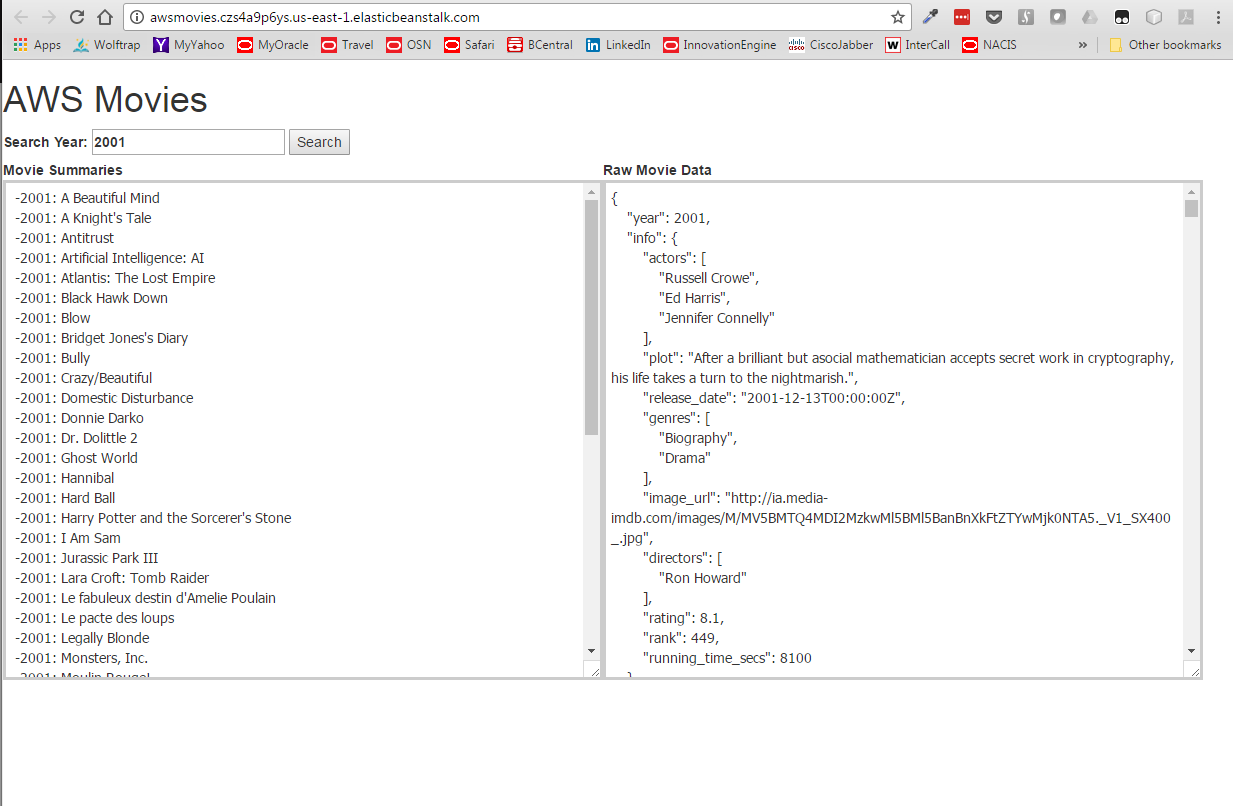


Figure 1 AWSMovies - A Sample DyNode Application

The rest of this document describes a step-by-step guide on how to recreate this sample application using the DyNode stack.

I developed this application as a way to learn the AWS cloud services. I hope that others new to AWS will find this project useful. This project can be used as a stepping-stone for learning additional AWS services such as RDS, Polly, and others. For further information, assistance, or comment, please contact me on my [GitHub page](https://github.com/sfurlong).

# Architecture Overview – Dev Stack

The DyNode stack is comprised of Angular JS, Node.js, ExpressJS, and the Amazon Dynamo database platform. This stack represents a modern development stack where JavaScript is used on the client-tier, middle-tier, and database tier (using JSON document objects). This allows the full stack developer to have language and toolset fidelity through all tiers of an application development cycle.



The best source for more information on each of these components can be found here:

* Angular JS – <https://angularjs.org>
* Node JS - <https://nodejs.org>
* Express JS – <http://expressjs.com>
* Amazon DynamoDB - [DynamoDB](https://aws.amazon.com/dynamodb/?sc_channel=PS&sc_campaign=acquisition_US&sc_publisher=google&sc_medium=dynamodb_hv_b&sc_content=dynamodb_e&sc_detail=dynamodb&sc_category=dynamodb&sc_segment=101621314602_control_q3_2016&sc_matchtype=e&sc_country=US&s_kwcid=AL!4422!3!101621314602!e!!g!!dynamodb&ef_id=VpPqsQAAAbRP9KpG:20170605214119:s)

# Architecture Overview – Tools & Infrastructure

# https://image.slidesharecdn.com/app202-141117151635-conversion-gate02/95/app202-deploy-manage-scale-apps-w-aws-opsworks-aws-elastic-beanstalk-aws-reinvent-2014-6-638.jpg?cb=1416237488

I use MS Visual Studio Code as an IDE.

# Installing the DyNode Stack

This section provides a step-by-step guide for installing the minimal components to develop using the DyNode stack. Only the strict minimum components are used to provide the developer with a solid understanding of what layers of the stack contribute to specific architectural capabilities.

## Install Node.js

* Download node.js install binaries from <https://nodejs.org> and install like any other application for your platform (i.e. exe, pkg, rpm, etc)
* Once you've installed node, you will have an “npm” command (node package manager) available to install the remaining components of the stack. Test the installation:
  + $ node --version
* Create a directory for the AWSMovies application
  + $ mkdir awsmovies
  + $ cd awsmovies
* Create a package.json file in the new directory. This file holds various metadata relevant to the project and gives information to npm that allows it to identify the project as well as handle the project's dependencies. OK to just take the default prompts for this command
  + $ npm init

## Install Angular JS

* Install Angular using the Node Package Manager in the directory “./awsmovies”. This will create an entry in the “node\_modules” subdirectory.
  + $ npm install angular –save

## Install AWS SDK Node.js Client

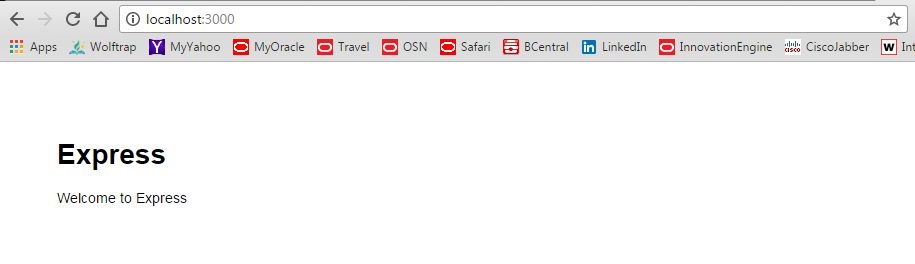
* Install the AWS-SDK Node.js Client Module in the awsmovies directory. This module is used by NodeJS to communicate with the AWS Services
  + $ npm install aws-sdk --save

## Install Express

* Install Express using the Node Package Manager in the directory “./awsmovies”. This will create an entry in the “node\_modules” subdirectory.
  + $ npm install express –save
  + $ npm install express-generator –save
  + Create the Express/Node application directory structure:
    - $ express myapp
    - $ npm install #installs all dependencies

## Test the Node.js/Express Environment

* From the awsmovies directory..
  + $ npm start
* From browser you should see:



## Install AWS “EB CLI”, Elastic Beanstalk CLI toolkit

<http://docs.aws.amazon.com/elasticbeanstalk/latest/dg/eb-cli3-install-windows.html>

**To configure static files and add a new page to your Express application**

1. Add a second configuration file with the following content:

**node-express/.ebextensions/staticfiles.config**

option\_settings:

aws:elasticbeanstalk:container:nodejs:staticfiles:

/public: /public

1. Comment out the static mapping in node-express/app.js. This step is not required, but it is a good test to confirm that static mappings are configured correctly.

**//** app.use(express.static(path.join(\_\_dirname, 'public')));

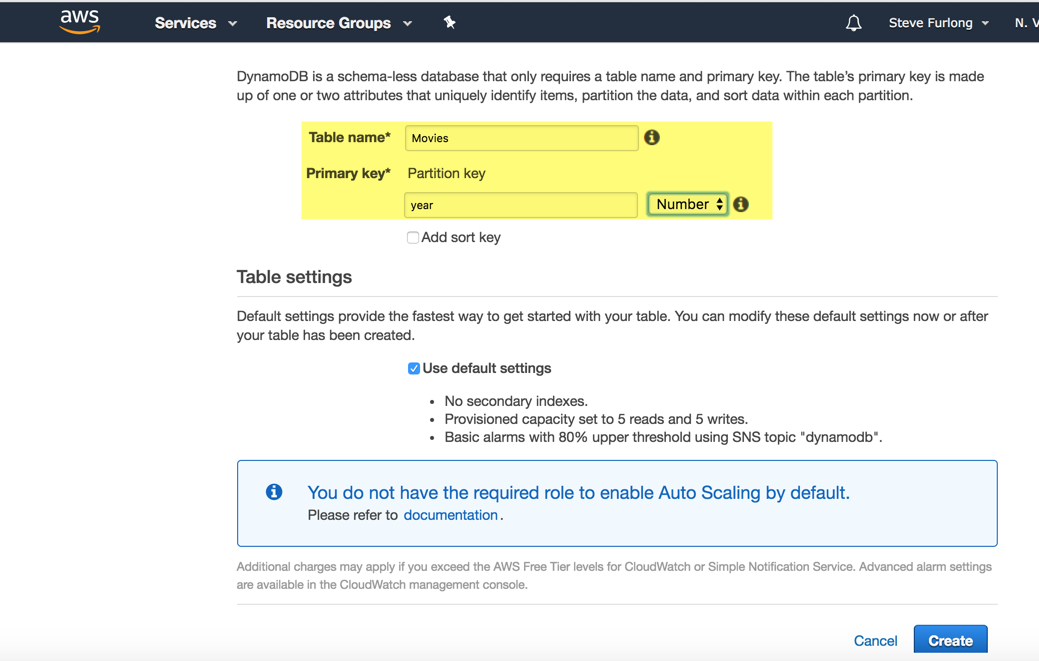
1. Add your updated files to your local repository and commit your changes.
2. ~/node-express$ **git add .ebextensions/ app.js**

~/node-express$ **git commit -m "Serve stylesheets statically with nginx."**

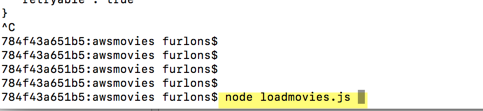
## Install Bootstrap CSS Style Sheets (OPTIONAL)

* Bootstrap is a very common client side UI style sheet framework developed by Twitter and now in the public domain. This will help beautify out browser based UI page.
  + $ npm install bootstrap

## Configure the DyanamoDB Service

* Create a DynamoDB table with the attributes as shown below
* 

Load Data into the Table:



## Other Tools for Developing ANEML Applications

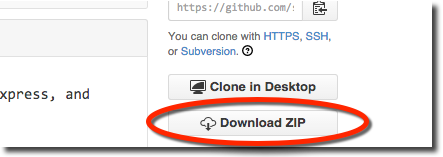
Additional application development tools that are helpful for developing your ANEML applications:

* A JavaScript aware editor such as Sublime or Atom
* JavaScript browser code debugging with the Chrome Developer Tools and JavaScript Console
* Git and GitHub for source control

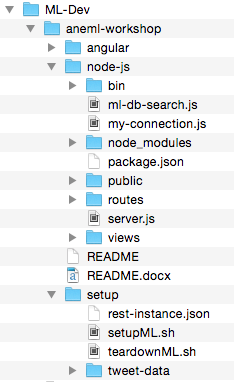
# AWSMovies Application Code Deployment

## Fetch Code from GitHub

* In your favorite browser, navigate to: https://github.com/sfurlong/awsmovies
* Download the zipfile containing the code



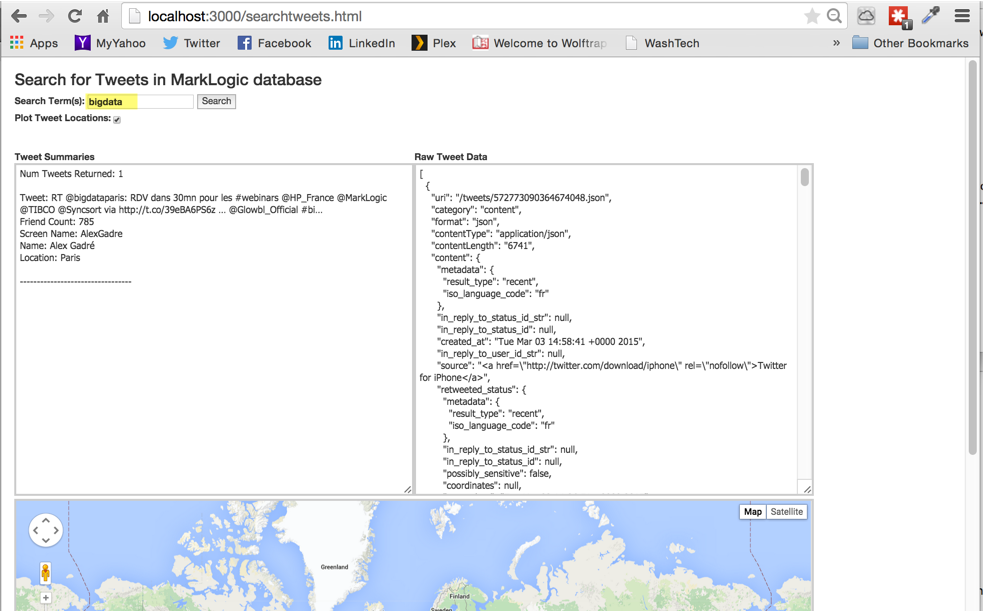
* Unzip the file into the your /awsmovies directory. Note: This should overlay your existing /awsmovies, while keeping its existing contents
* Your directory structure should now look like this:



## 

# To Run The Demo

## Start the Node.js Server

* Start the Node JS server using the following command. This will start the serve up the web pages.
  + $ npm start
* Open a browser and navigate to <http://localhost:3000/>. 
* Generate a new Google Maps API Key by following the instructions here: <https://developers.google.com/maps/documentation/business/mobile/ios/auth>
* Update the Google Maps Key in the file /aneml-workshop/node-js/public/search-tweets.html

# Appendix

## Working with Git Command Line

Summary of useful Git command line

* Initialize a git repository
  + $ git init
* Add files to ignore from git management
  + $ vi .gitignore
  + Add any files to be ignored
* Recursively add all files to your local git repository
  + $ git add -A
* Commit files to the local repository
  + $ git commit –m “<<comment>>”
* Add a remote origin to your local git repository
  + $ git remote add origin <https://sfurlong@github.com/sfurlong/aneml.git>
* Push all local changes to the remote repository
  + $ git push -u origin master
* Status commands
  + $ git status
  + $ git log
  + $ git diff head
  + $ git push

## Future Enhancement Opportunities

1. Add more AWS Services!!
2. Add pagination to the UI when > 100 tweets are returned
3. AuthN via LDAP
4. AuthZ based on user Role

# Change History

|  |  |  |
| --- | --- | --- |
| Date | Who | Changes |
| 6/1/17 | Stephen Furlong | Initial Revision |
|  |  |  |