Serverless Framework Walk-Thru Demo

Deploy a REST API using serverless

1. Install serverless
    - a. npm i serverless -g
    - b. Have AWS credentials for your environment configured AWS credentials.
    - c. Serverless needs your AWS credentials
        - i. Easiest way to set a IAM user profile in your AWS account with appropriate access.
        - ii. To config serverless for default access:
            1. $ serverless config credentials --provider aws --key XXXXXX --secret XXXXXX
2. Create a new dir with package.json file in it
    - a. $ mkdir my-express-application && cd my-express-application
    - b. $ npm init -f
3. Install express and serverless-http
    - a. $ npm install --save express serverless-http
    - b. serverless-http package is a handy piece of middleware that handles the interface between your Node.js application and the specifics of API Gateway.
4. Create an index.js file with the follow code:

```
// index.js
const serverless = require('serverless-http');
const express = require('express')
const app = express()
app.get('/', function (req, res) {
  res.send('Hello World!')
})
module.exports.handler = serverless(app);
```

    - a. This is a very simple application that returns "Hello World!" when a request comes in on the root path /.
    - b. It's straight out of the Express documentation with two small additions. First, we imported the serverless-http package at the top. Second, we exported a handler function which is our application wrapped in the serverless package.
5. Create a serverless.yml file and put the following code in it

```
# serverless.yml

service: my-express-application

provider:
  name: aws
  runtime: nodejs6.10
  stage: dev
  region: us-east-1

functions:
  app:
    handler: index.handler
    events:
      - http: ANY /
      - http: 'ANY {proxy+}'
```

   a. This is a pretty basic configuration. We've created one function, app, which uses the exported handler from our index.js file. Finally, it's configured with some HTTP triggers.
   b. We've used a very broad path matching so that all requests on this domain are routed to this function. All of the HTTP routing logic will be done inside the Express application.

6. Deploy the funcion
   a.
   ```
   $ sls deploy
   ```
7. After a minute, the console will show your endpoints in the Service Information section.
8.