Assignment: Serverless AWS Webserver - PacMan

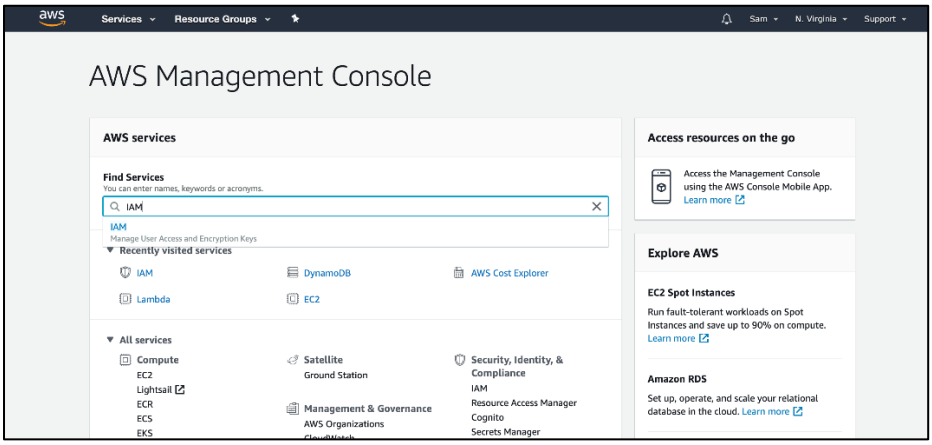
Overall Steps:

1. *Set Up the AWS CLI/CDK authentication*

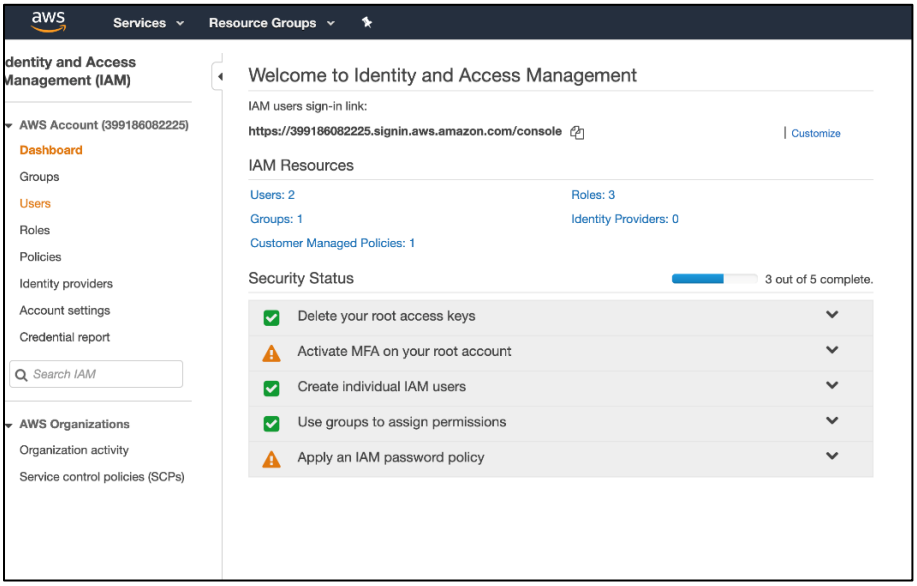
Before we can use the command line interface to launch serverless architecture, we need to link it to our AWS account that we have already set up. To do this follow the following steps on the computer that you have admin access to (such that you can install programs on your computer).

Navigate to aws.amazon.com (these instructions come from here)

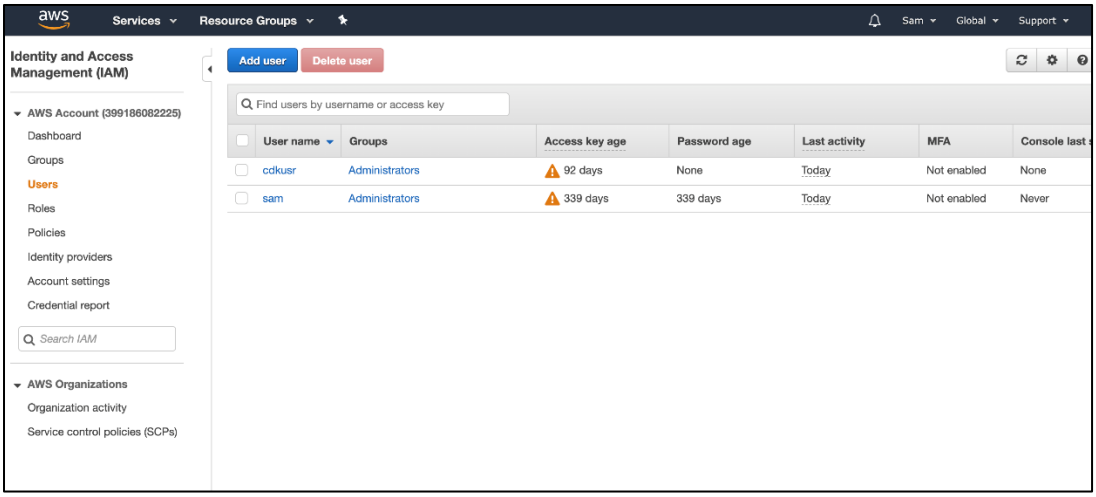
- Sign into the Console
- Search for IAM



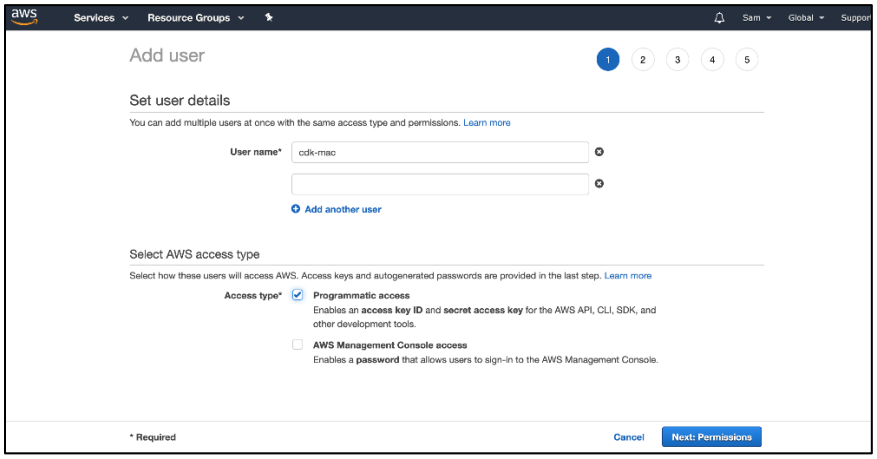
- Click on Users



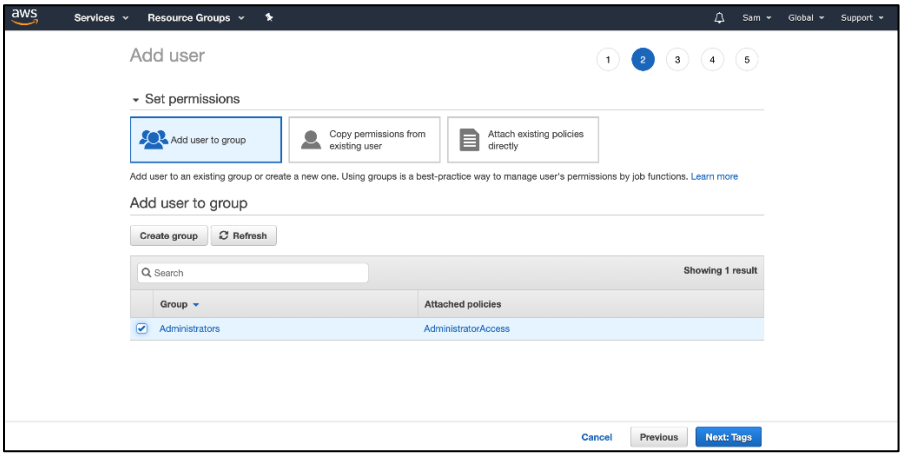
- Click on Add User



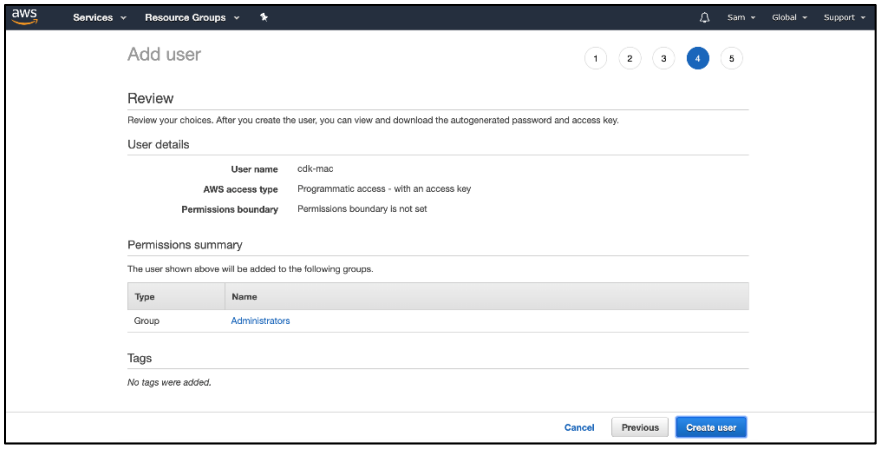
- Give the user a name
- Select programmatic access



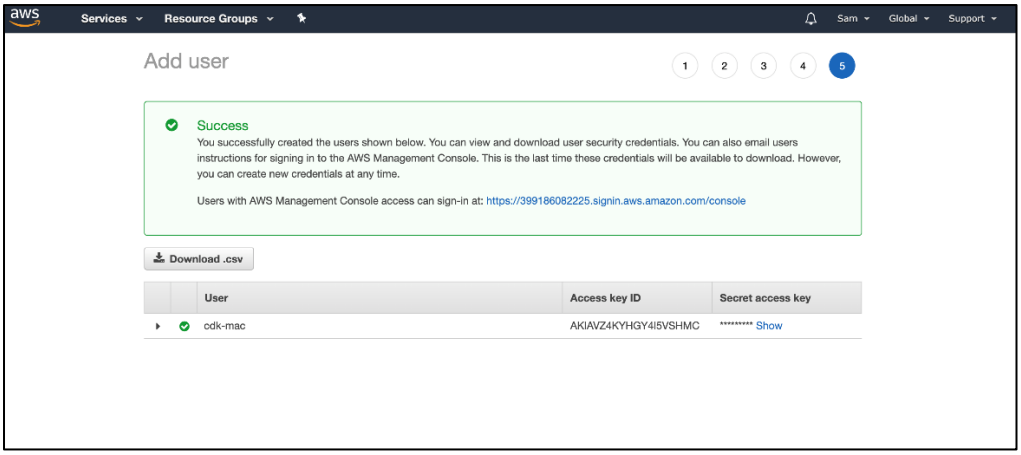
- Click on Next: Permissions
- Select the Administrators Group



- Click on Next: Tags
- Don't create any new Tags
- Click on Next: Review



- Click on Create user



- Click on Download.csv to a place that is easy to access
- Open the file that was downloaded
- Create a new folder in your **home directory** and name it ".aws" (without the quotes but with the period in front)
    - To do this on windows follow the steps here:
      https://autechtips.com/create-folder-name-starting-with-a-dot-in-windows/1996/
(if this already exists, delete the files in there as we need to make new ones)

Home directory (Windows: C:/Users/*yourusername*/ | macOS: /Users/*yourusername*/)

- Create a file with the following lines (only type what is after the colon for each line):
    - 1: [default]
    - 2: region = us-east-1
    - 3: output = json
    - 4: aws_access_key_id=YOUR ACCESS KEY ID FROM THE DOWNLOADED FILE
    - 5: aws_secret_access_key=YOUR SECRET ACCESS KEY FROM THE DOWNLOADED FILE
- If your region by default is different to us-east-1, then change it so this matches what you have in the AWS console.
- Save this file with the name config (with no extension) into the. aws folder you just created
- In addition, you will need to create an empty file in the same folder and name it "credentials" with no extension

2. *Download and install the AWS Command Line Interface*

Now that we have set up our account to link with our computer, we can install the CLI package to talk directly to the AWS console.

Windows:

https://docs.aws.amazon.com/cli/latest/userguide/install-windows.html
- Click on the install AWS MSI Installer
- Download the 64bit Installer
- Accept the defaults and click through to install (you will need to be able to install programs on your machine (ie. admin access))
- Once the installation finishes, open a command prompt and navigate to the .aws folder
- type "***aws --version***" and ensure that the system responds with a version (eg. "aws-cli/1.16.221 Python/3.6.0 Windows/10 botocore/1.12.211")

MacOS:

https://docs.aws.amazon.com/cli/latest/userguide/install-macos.html

curl "https://s3.amazonaws.com/aws-cli/awscli-bundle.zip" -o "awscli-bundle.zip"
unzip awscli-bundle.zip
sudo ./awscli-bundle/install -i /usr/local/aws -b /usr/local/bin/aws

To test that this command executed correctly, type "***aws --version***" and ensure that the system responds with a version (eg. "aws-cli/1.16.221 Python/3.6.0 Windows/10 botocore/1.12.211")

3. *Download the code for the AWS templates and the PacMan Code*

Download and unzip the code for this assignment. You will find two folders: aws_serverless and pacman_serverless. The first folder will let us build the serverless architecture using the cli we just installed. The second folder will show a demonstration of how you can use this architecture to interact with a website that we have used previously in this course.

4. *Run the pacman example with your name as the entry*

The pacman_serverless code has a pre-existing serverless architecture linked to it. So for this first phase, we will edit the code so we send our name to this server. To do this:
Open server.js in the pacman_serverless and edit the code at line 27 so that it changes from just sending “x” to sending a name (for example the name sam is shown here).

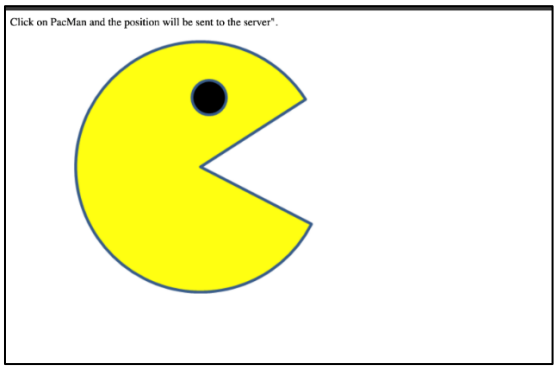
```
24  app.get('/set', function (req, res) {
25      console.log(JSON.stringify(req.query));
26      var record = {
27          x: req.query.x
28      };
```

```
24  app.get('/set', function (req, res) {
25      console.log(JSON.stringify(req.query));
26      var record = {
27          x: req.query.x,
28          name: "sam"
29      };
```

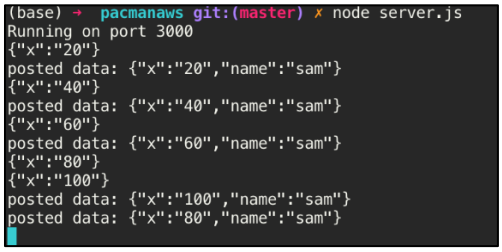
Now you just need to access this folder in your terminal, run the commands:
npm install

node server.js

This will create a webserver running on port 3000. Head to your browser and enter the url: localhost:3000



After clicking on the image and seeing it move, you will notice that the terminal output will look something like the following:



We will be able to see your accessing of this site to in our AWS database. To empower you to do this, the next section will let you build your own.

5. *Create your own Cloud serverless infrastructure using the CDK*

In the downloaded aws_serverless folder, run the following commands in the command prompt:

npm install -g aws-cdk

npm install

npm run build

These commands will create the necessary files needed to deploy the cloud serverless services. Now you will be ready to launch the lambda functions and the database. Type the following into the command line:

cdk bootstrap

cdk deploy

This will create the cloud templates and ask you to confirm if this is what you want, select “y” to begin building them.

```
(base) ➜  pacman_aws_setup cdk deploy
This deployment will make potentially sensitive changes according to your current security approval level (--require-approval broadening).
Please confirm you intend to make the following modifications:

IAM Statement Changes
```

	Resource	Effect	Action	Principal	Condition
+	${createItemFunction.Arn}	Allow	lambda:InvokeFunction	Service:apigateway.amazonaws.com	"ArnLike": { "AWS:SourceArn": "arn:${AWS::Partition}:execute-api:${AWS::Region}:${AWS::AccountId}:${itemsApi28111E1C}/test-invoke-stage/POST/items" }
+	${createItemFunction.Arn}	Allow	lambda:InvokeFunction	Service:apigateway.amazonaws.com	"ArnLike": { "AWS:SourceArn": "arn:${AWS::Partition}:execute-api:${AWS::Region}:${AWS::AccountId}:${itemsApi28111E1C}/${itemsApi/DeploymentStage.prod}/POST/items" }
+	${createItemFunction/ServiceRole.Arn}	Allow	sts:AssumeRole	Service:lambda.amazonaws.com	
+	${deleteItemFunction.Arn}	Allow	lambda:InvokeFunction	Service:apigateway.amazonaws.com	"ArnLike": { "AWS:SourceArn": "arn:${AWS::Partition}:execute-api:${AWS::Region}:${AWS::AccountId}:${itemsApi28111E1C}/test-invoke-stage/DELETE/items/{id}" }
+	${deleteItemFunction.Arn}	Allow	lambda:InvokeFunction	Service:apigateway.amazonaws.com	"ArnLike": { "AWS:SourceArn": "arn:${AWS::Partition}:execute-api:${AWS::Region}:${AWS::AccountId}:${itemsApi28111E1C}/$

```
IAM Policy Changes
```

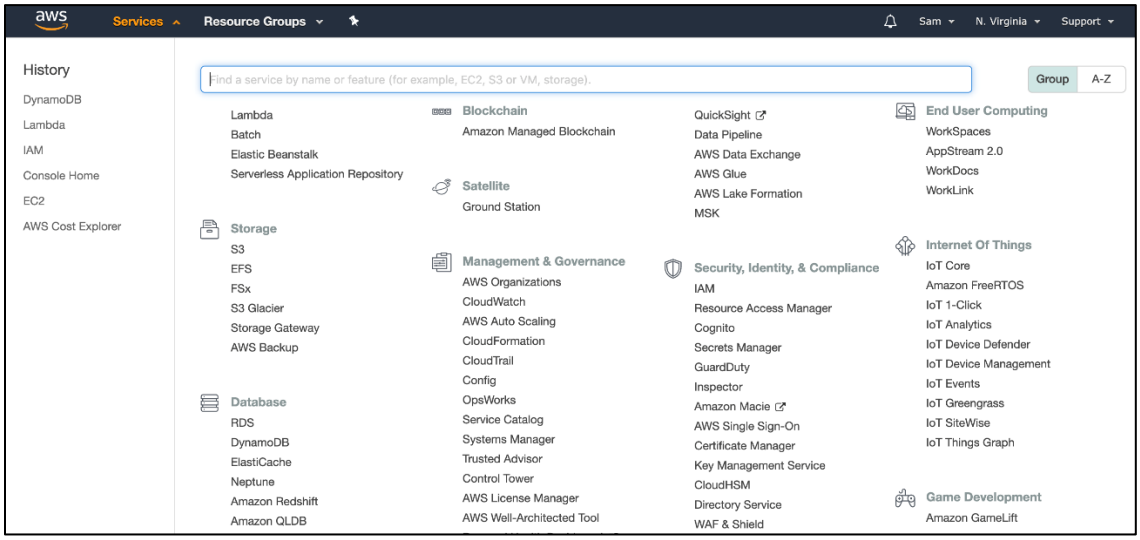
	Resource	Managed Policy ARN
+	${createItemFunction/ServiceRole}	arn:${AWS::Partition}:iam::aws:policy/service-role/AWSLambdaBasicExecutionRole
+	${deleteItemFunction/ServiceRole}	arn:${AWS::Partition}:iam::aws:policy/service-role/AWSLambdaBasicExecutionRole
+	${getAllItemsFunction/ServiceRole}	arn:${AWS::Partition}:iam::aws:policy/service-role/AWSLambdaBasicExecutionRole
+	${getOneItemFunction/ServiceRole}	arn:${AWS::Partition}:iam::aws:policy/service-role/AWSLambdaBasicExecutionRole
+	${itemsApi/CloudWatchRole}	arn:${AWS::Partition}:iam::aws:policy/service-role/AmazonAPIGatewayPushToCloudWatchLogs
+	${updateItemFunction/ServiceRole}	arn:${AWS::Partition}:iam::aws:policy/service-role/AWSLambdaBasicExecutionRole

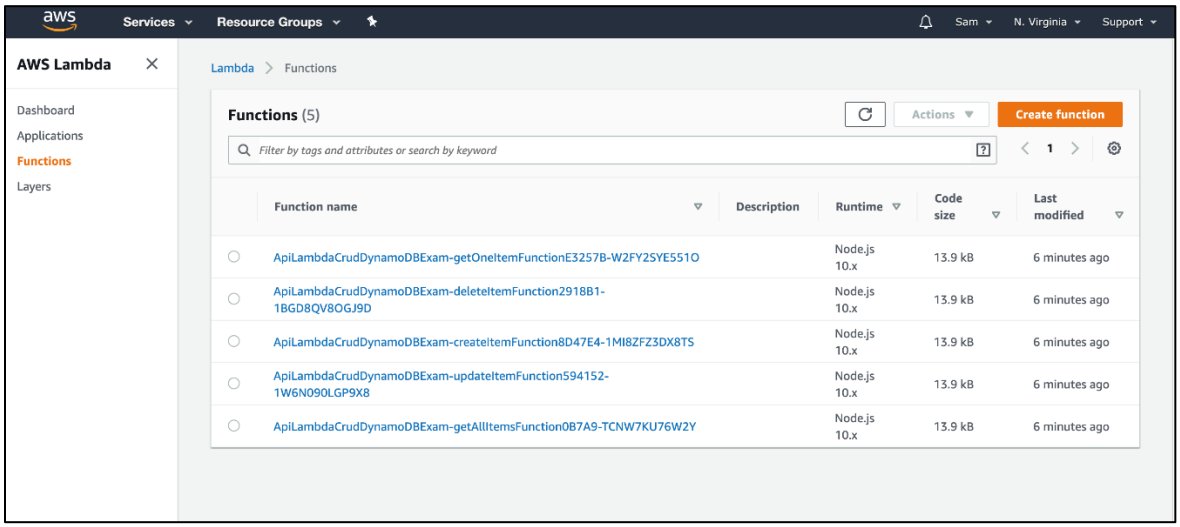
```
(NOTE: There may be security-related changes not in this list. See https://github.com/aws/aws-cdk/issues/1299)

Do you wish to deploy these changes (y/n)? y
```

Once you select “y” it will begin creating the lambda functions and database tables in your aws account:

```
ApiLambdaCrudDynamoDBExample: deploying...
Updated: asset.042525c37b32d38ed1d843b55c7104416913be76b53f50c876e3428091ae9f01 (zip)
ApiLambdaCrudDynamoDBExample: creating CloudFormation changeset...
  0/42 | 1:45:13 PM | CREATE_IN_PROGRESS   | AWS::DynamoDB::Table          | items (items07D08F4B)
  0/42 | 1:45:13 PM | CREATE_IN_PROGRESS   | AWS::CDK::Metadata            | CDKMetadata
  0/42 | 1:45:13 PM | CREATE_IN_PROGRESS   | AWS::IAM::Role                | updateItemFunction/ServiceRole (updateItemFunctionServiceRole40035396)
  0/42 | 1:45:13 PM | CREATE_IN_PROGRESS   | AWS::IAM::Role                | deleteItemFunction/ServiceRole (deleteItemFunctionServiceRole5C201FCC)
  0/42 | 1:45:13 PM | CREATE_IN_PROGRESS   | AWS::ApiGateway::RestApi      | itemsApi (itemsApi28111E1C)
  0/42 | 1:45:13 PM | CREATE_IN_PROGRESS   | AWS::IAM::Role                | getAllItemsFunction/ServiceRole (getAllItemsFunctionServiceRoleCC084440)
  0/42 | 1:45:13 PM | CREATE_IN_PROGRESS   | AWS::IAM::Role                | getOneItemFunction/ServiceRole (getOneItemFunctionServiceRoleCFD54796)
  0/42 | 1:45:13 PM | CREATE_IN_PROGRESS   | AWS::IAM::Role                | itemsApi/CloudWatchRole (itemsApiCloudWatchRoleB5C7B431)
  0/42 | 1:45:13 PM | CREATE_IN_PROGRESS   | AWS::IAM::Role                | updateItemFunction/ServiceRole (updateItemFunctionServiceRole40035396) Resource creation Initiated
  0/42 | 1:45:13 PM | CREATE_IN_PROGRESS   | AWS::IAM::Role                | createItemFunction/ServiceRole (createItemFunctionServiceRole1BBF2178)
  0/42 | 1:45:13 PM | CREATE_IN_PROGRESS   | AWS::IAM::Role                | deleteItemFunction/ServiceRole (deleteItemFunctionServiceRole5C201FCC) Resource creation Initiated
  0/42 | 1:45:13 PM | CREATE_IN_PROGRESS   | AWS::DynamoDB::Table          | items (items07D08F4B) Resource creation Initiated
  0/42 | 1:45:13 PM | CREATE_IN_PROGRESS   | AWS::IAM::Role                | getAllItemsFunction/ServiceRole (getAllItemsFunctionServiceRoleCC084440) Resource creation Initiated
  0/42 | 1:45:13 PM | CREATE_IN_PROGRESS   | AWS::ApiGateway::RestApi      | itemsApi (itemsApi28111E1C) Resource creation Initiated
  0/42 | 1:45:14 PM | CREATE_IN_PROGRESS   | AWS::IAM::Role                | getOneItemFunction/ServiceRole (getOneItemFunctionServiceRoleCFD54796) Resource creation Initiated
  0/42 | 1:45:14 PM | CREATE_IN_PROGRESS   | AWS::IAM::Role                | itemsApi/CloudWatchRole (itemsApiCloudWatchRoleB5C7B431) Resource creation Initiated
  0/42 | 1:45:14 PM | CREATE_IN_PROGRESS   | AWS::IAM::Role                | createItemFunction/ServiceRole (createItemFunctionServiceRole1BBF2178) Resource creation Initiated
  1/42 | 1:45:14 PM | CREATE_COMPLETE      | AWS::ApiGateway::RestApi      | itemsApi (itemsApi28111E1C)
  1/42 | 1:45:15 PM | CREATE_IN_PROGRESS   | AWS::CDK::Metadata            | CDKMetadata Resource creation Initiated
  2/42 | 1:45:15 PM | CREATE_COMPLETE      | AWS::CDK::Metadata            | CDKMetadata
  2/42 | 1:45:17 PM | CREATE_IN_PROGRESS   | AWS::ApiGateway::Resource     | itemsApi/Default/items (itemsApiitems9015DBED)
  2/42 | 1:45:17 PM | CREATE_IN_PROGRESS   | AWS::ApiGateway::Resource     | itemsApi/Default/items (itemsApiitems9015DBED) Resource creation Initiated
  3/42 | 1:45:17 PM | CREATE_COMPLETE      | AWS::ApiGateway::Resource     | itemsApi/Default/items (itemsApiitems9015DBED)
  3/42 | 1:45:20 PM | CREATE_IN_PROGRESS   | AWS::ApiGateway::Resource     | itemsApi/Default/items/{id} (itemsApiitemsidA29927C2)
  3/42 | 1:45:20 PM | CREATE_IN_PROGRESS   | AWS::ApiGateway::Resource     | itemsApi/Default/items/{id} (itemsApiitemsidA29927C2) Resource creation Initiated
  3/42 | 1:45:21 PM | CREATE_IN_PROGRESS   | AWS::ApiGateway::Method       | itemsApi/Default/items/OPTIONS (itemsApiitemsOPTIONSB46B4D53)
  4/42 | 1:45:21 PM | CREATE_COMPLETE      | AWS::ApiGateway::Resource     | itemsApi/Default/items/{id} (itemsApiitemsidA29927C2)
  4/42 | 1:45:21 PM | CREATE_IN_PROGRESS   | AWS::ApiGateway::Method       | itemsApi/Default/items/OPTIONS (itemsApiitemsOPTIONSB46B4D53) Resource creation Initiated
  5/42 | 1:45:22 PM | CREATE_COMPLETE      | AWS::ApiGateway::Method       | itemsApi/Default/items/OPTIONS (itemsApiitemsOPTIONSB46B4D53)
  5/42 | 1:45:23 PM | CREATE_IN_PROGRESS   | AWS::ApiGateway::Method       | itemsApi/Default/items/{id}/OPTIONS (itemsApiitemsidOPTIONS62BD91D0)
  5/42 | 1:45:24 PM | CREATE_IN_PROGRESS   | AWS::ApiGateway::Method       | itemsApi/Default/items/{id}/OPTIONS (itemsApiitemsidOPTIONS62BD91D0) Resource creation Initiated
  6/42 | 1:45:25 PM | CREATE_COMPLETE      | AWS::ApiGateway::Method       | itemsApi/Default/items/{id}/OPTIONS (itemsApiitemsidOPTIONS62BD91D0)
  7/42 | 1:45:26 PM | CREATE_COMPLETE      | AWS::IAM::Role                | deleteItemFunction/ServiceRole (deleteItemFunctionServiceRole5C201FCC)
  8/42 | 1:45:26 PM | CREATE_COMPLETE      | AWS::IAM::Role                | getAllItemsFunction/ServiceRole (getAllItemsFunctionServiceRoleCC084440)
  9/42 | 1:45:26 PM | CREATE_COMPLETE      | AWS::IAM::Role                | updateItemFunction/ServiceRole (updateItemFunctionServiceRole40035396)
 10/42 | 1:45:26 PM | CREATE_COMPLETE      | AWS::IAM::Role                | getOneItemFunction/ServiceRole (getOneItemFunctionServiceRoleCFD54796)
 11/42 | 1:45:26 PM | CREATE_COMPLETE      | AWS::IAM::Role                | createItemFunction/ServiceRole (createItemFunctionServiceRole1BBF2178)
 12/42 | 1:45:26 PM | CREATE_COMPLETE      | AWS::IAM::Role                | itemsApi/CloudWatchRole (itemsApiCloudWatchRoleB5C7B431)
 12/42 | 1:45:29 PM | CREATE_IN_PROGRESS   | AWS::ApiGateway::Account      | itemsApi/Account (itemsApiAccount591BF3E1)
 12/42 | 1:45:30 PM | CREATE_IN_PROGRESS   | AWS::ApiGateway::Account      | itemsApi/Account (itemsApiAccount591BF3E1) Resource creation Initiated
 13/42 | 1:45:30 PM | CREATE_COMPLETE      | AWS::ApiGateway::Account      | itemsApi/Account (itemsApiAccount591BF3E1)
```

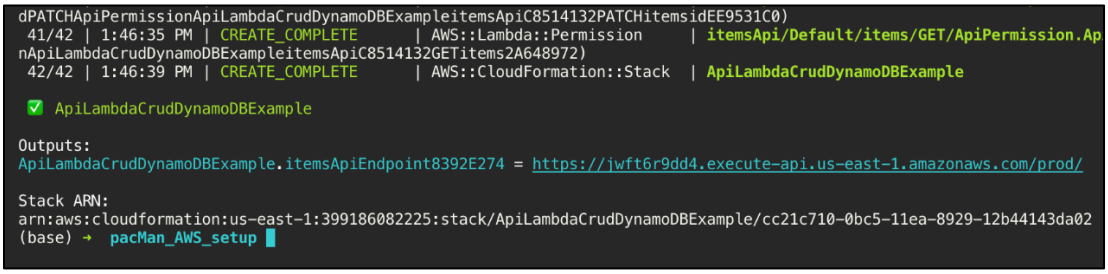
Once this finishes you can go to your AWS console and look for the lambda functions to confirm they were built.





6. *Rerun the pacman example with your own API*

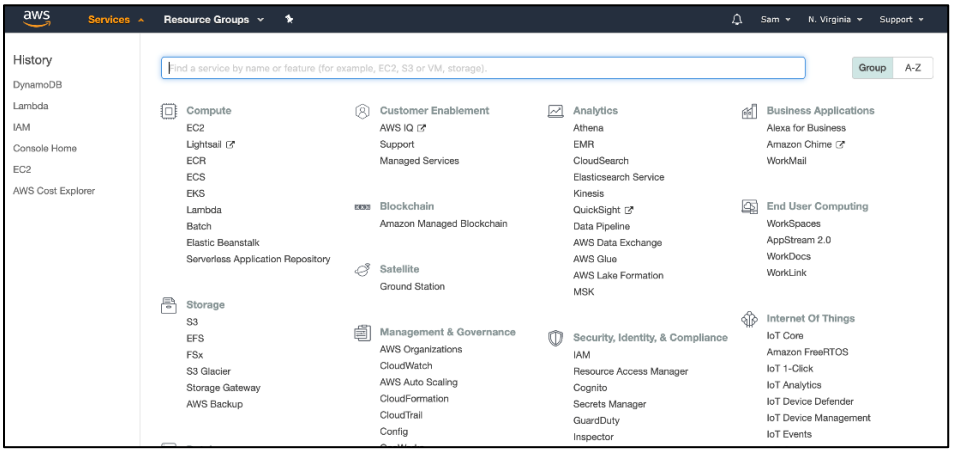
To start using these lambda functions and to create entries into your own database, you will need to take the API endpoint from the command line output (starting with “https” and ending in “prod/”)



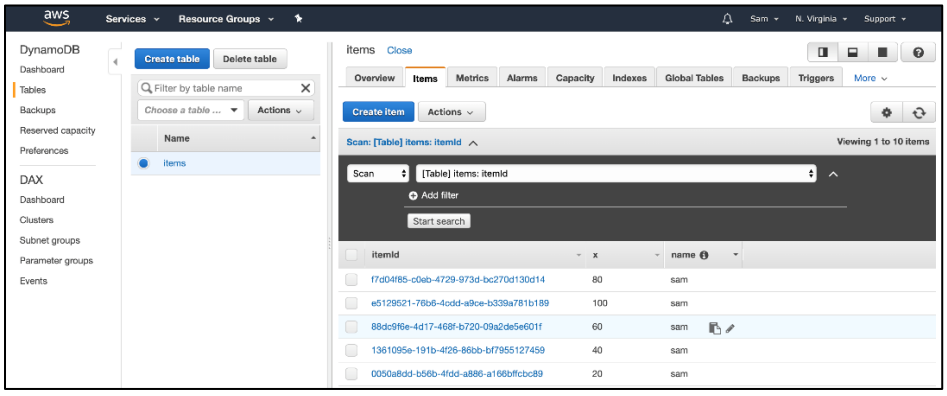
Then paste this in line 8 of the server.js file from the pacman_serverless folder:

```
var express = require('express');
var bodyParser = require('body-parser');
var path = require('path');
var app = express();
const fetch = require('node-fetch');

// const url = 'https://11jskt4u1d.execute-api.us-east-1.amazonaws.com/prod/items/';
const url ='https://jwft6r9dd4.execute-api.us-east-1.amazonaws.com/prod/items'
// used to serve static files from public directory
app.use(express.static('public'));
accounts = [];
app.use(bodyParser.json()); // support json encoded bodies
app.use(bodyParser.urlencoded({
    extended: true
}));

```

Make sure to keep the "/items" at the end of the url so that it knows to access the right table. Rerun the node server.js command inside the pacman_serverless folder with the updated url and click on the pacman icon in the browser (you may need to refresh the page if you rerun the node command). Now head to the aws console and look for the DynamoDB link.



Click on the Tables and the items tab on the items table and you will see the entries populated from your clicking of the pacman image.



Congratulations! You just created a fully serverless backend managed on AWS with a website that can be hosted anywhere, connected to this backend.