INSTRUCTIONS – “Unleash the Power of Serverless Architecture”

1. Create AWS Free Tier Account
2. Login to AWS
3. Create ElasticSearch Service
   1. Name = beers-and-hobbies-search
   2. Set authentication to Open Public Access (no vpc)
4. Create DynamoDB table
   1. Table Name = beers-and-hobbies-people
      1. Primary Key = personID
      2. DynamoDB table Name and primary key must match ElasticSearch scripts
   2. Add data stream
      1. After table created > Click Manage Stream > New Image
5. Setup Roles in IAM
   1. Use Lambda service
   2. Add Create Role > AWSLambdaBasicExecutionRole
   3. Name role: geo-dev-role
   4. Add 3 Inline Policies to geo-dev-role
   5. Copy and paste each role. Replace the highlighted Resource ARN with the DynamoDB table ARN, DynamoDB Stream ARN and the ElasticSearch ARN.

|  |  |
| --- | --- |
|  | **Access DynamoDB Table**  { |
|  | "Version": "2012-10-17", |
|  | "Statement": [ |
|  | { |
|  | "Action": [ |
|  | "dynamodb:DescribeTable", |
|  | "dynamodb:Scan" |
|  | ], |
|  | "Effect": "Allow", |
|  | "Resource": [ |
|  | "**arn:aws:dynamodb:us-west-2:XXXXXXXXXXXX:table/elements**" |
|  | ] |
|  | } |
|  | ] |
|  | } |

**Access DynamoDB Stream**

|  |
| --- |
| { |
|  | "Version": "2012-10-17", |
|  | "Statement": [ |
|  | { |
|  | "Action": [ |
|  | "dynamodb:DescribeStream", |
|  | "dynamodb:GetRecords", |
|  | "dynamodb:GetShardIterator", |
|  | "dynamodb:ListStreams" |
|  | ], |
|  | "Effect": "Allow", |
|  | "Resource": [ |
|  | "arn:aws:dynamodb:us-east-1:XXXXXXXXXXXX:table/elements/stream/2017-09-14T14:16:12.788" |
|  | ] |
|  | } |
|  | ] |
|  | } |

**Access to ElasticSearch**

|  |
| --- |
| { |
|  | "Version": "2012-10-17", |
|  | "Statement": [ |
|  | { |
|  | "Action": [ |
|  | "es:ESHttpPost" |
|  | ], |
|  | "Effect": "Allow", |
|  | "Resource": "arn:aws:es:us-east-1:XXXXXXXXXXXXX:domain/elementsdomain/\*" |
|  | } |
|  | ] |
|  | } |

1. Create Lambda function from Scratch
   1. Name: index-beers-and-pastimes-job
   2. Runtime: Python 2.7
   3. Choose role just created: geo-dev-role
   4. Add DynamoDB trigger from table: beers-and-hobbies-people with Trim Horizon
   5. Create environment variable named SEARCH\_ENDPOINT using the elasticsearch endpoint URL for the value.
   6. Copy and paste geo-summit-lambda.py into Lambda function code.
   7. Save
2. Use Postman to create ElasticSearch mappings with the Geo\_Point indices
   1. Create URL and append URL end with repository name:
      1. **Format**: < elasticsearch endpoint url>/beers-and-hobbies-people
      2. **Example**:
   2. [https://search-geo-summit-backup-7azns2n3fe7kpksnfjr7go62ae.us-east-1.es.amazonaws.com/beers-and-hobbies-people](https://search-geo-summit-backup-7azns2n3fe7kpksnfjr7go62ae.us-east-1.es.amazonaws.com/geodev-summit-backup)
   3. Add Header: Key = Content-Type and Value = application/json
   4. Copy and paste below text in Body (raw) and hit send.

{

"mappings": {

"beers-and-hobbies-people\_type": {

"properties": {

"@timestamp": {

"type": "text"

},

"@SequenceNumber": {

"type": "text"

},

"personID": {

"type": "text"

},

"location": {

"type": "geo\_point"

},

"zipcode": {

"type": "text"

},

"city": {

"type": "text"

},

"name": {

"type": "text"

},

"gender": {

"type": "text"

},

"hobbies": {

"type": "text"

}

}

}

}

}

1. Now create value in DynamoDB

{

"personID": "100",

"location": "40.183450, -76.921845",

"zipcode": "17339",

"city": "Lewisberry",

"name": "Jordan Biskit",

"gender": "F",

"hobbies": "Tennis"

}

1. Verify value loaded into Elasticsearch
2. Now batch load using CLI
   1. Navigate to directory where file is located.

aws dynamodb batch-write-item --request-items <file://load-people.json>

1. Test the search service using Postman. For each call Add Header: Key = Content-Type and Value = application/json

**Get all values:** <Elastic Search URL Endpoint>/\_search?size=10&pretty=true&q=\*

**Get Person with Distance Query and Hobby:** <Elastic Search URL Endpoint>/\_search

* Paste below text in Postman Body (raw)

{

"query": {

"bool" : {

"must" : {

"match\_phrase" : {

"hobbies":"Yoga"

}

},

"filter" : {

"geo\_distance" : {

"distance" : "5mi",

"location" : {

"lat" : 40.261741,

"lon" : -76.880456

}

}

}

}

}

}

Bounding Box: <Elastic Search URL Endpoint>/\_search

{

"query": {

"bool" : {

"must" : {

"match\_phrase" : {

"hobbies":"Golf"

}

},

"filter" : {

"geo\_bounding\_box" : {

"location" : {

"top\_left" : "40.291740, -76.934653",

"bottom\_right" : "40.258449, -76.830500"

}

}

}

}

}

}