Lambda functions / Serverless architecture

Serverless Architecture?

*“ [...] applications that **significantly or fully depend on 3rd party applications / services** (‘in the cloud’) to manage server-side logic and state. [...] These types of services have been previously described as **‘Backend as a Service’**, and I’ll be using **‘BaaS’** [...] “*

http://martinfowler.com/articles/serverless.html

Serverless Architecture?

“ [...] applications where some amount of server-side logic is still written by the application developer but unlike traditional architectures is run in ***stateless*** *compute* ***containers*** *that are* ***event-triggered****, ephemeral (may* ***only last for one invocation****), and fully* ***managed by a 3rd party****. [...] One way to think of this is ‘****Functions as a service / FaaS****’ [...] “*

http://martinfowler.com/articles/serverless.html

Ventajas

- Escalamiento prácticamente ilimitado y automático
- Bajos costos
- Facilita servicios geolocalizados
- Alta disponibilidad
- Cero administración
- Event Driven
- Aislamiento y seguridad

Desventajas

- Dependencia a una plataforma
- Complejidad de la solución
- Multitenancy
- Startup Latency
- Execution duration

No es para todos...

Ejemplos

- File Processing
- Data & Analytics
- Websites
- Mobile Applications

Cloud Functions
Google Compute Engine

Lambda Functions
AWS

Azure Functions
Azure

Lambda Functions

AWS

Precios

P = # Veces * F(Memoria, Tiempo) + Otros

Ej:

- 128 MB de memoria
- 30 millones de veces
- 200 ms cada vez

= 5,83 USD

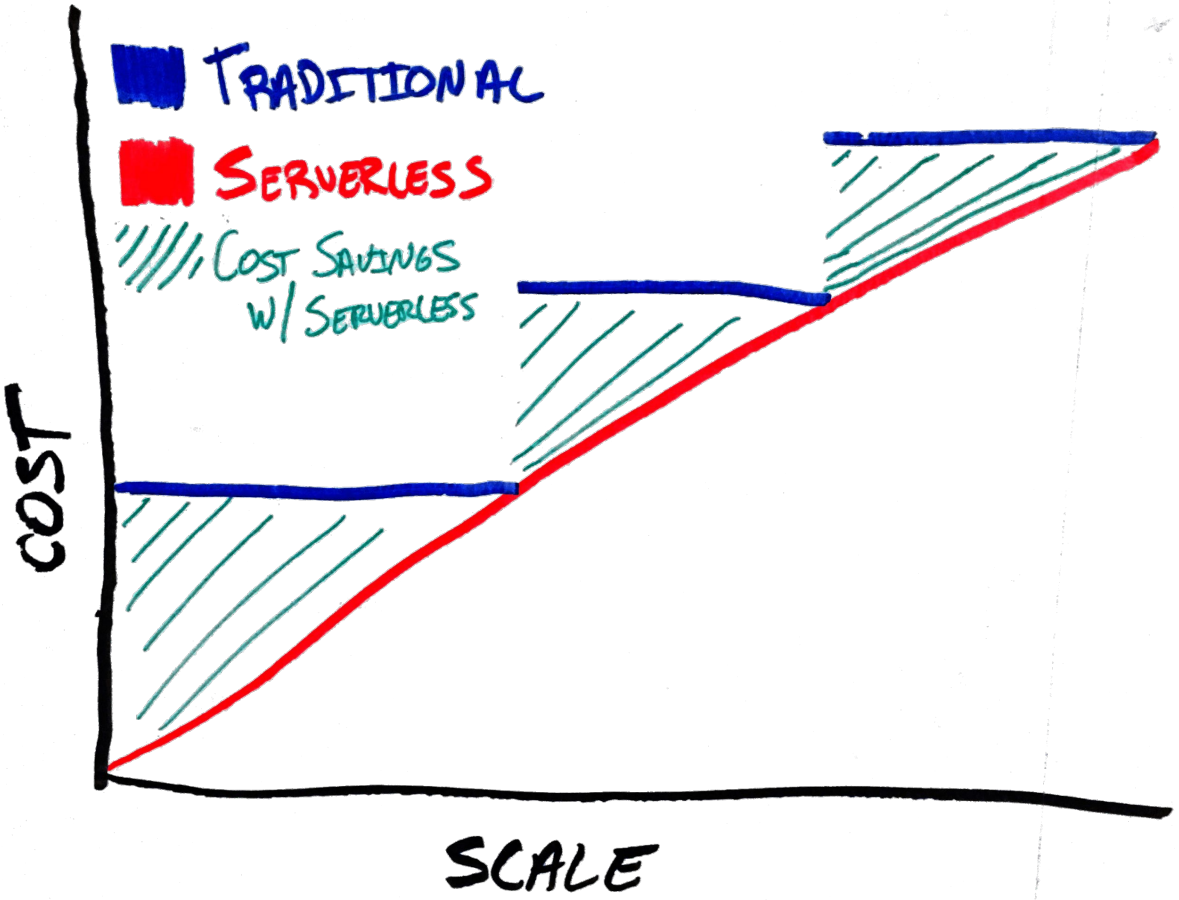
Ej:

- 512 MB de memoria
- 3 millones de veces
- 1000 ms cada vez

= 18,74 USD

Precios

Memoria (MB)	Segundos de la capa gratuita al mes	Precio por 100 ms (USD)
128	3 200 000	0,000000208
192	2 133 333	0,000000313
256	1 600 000	0,000000417
320	1 280 000	0,000000521
384	1 066 667	0,000000625
448	914 286	0,000000729
512	800 000	0,000000834



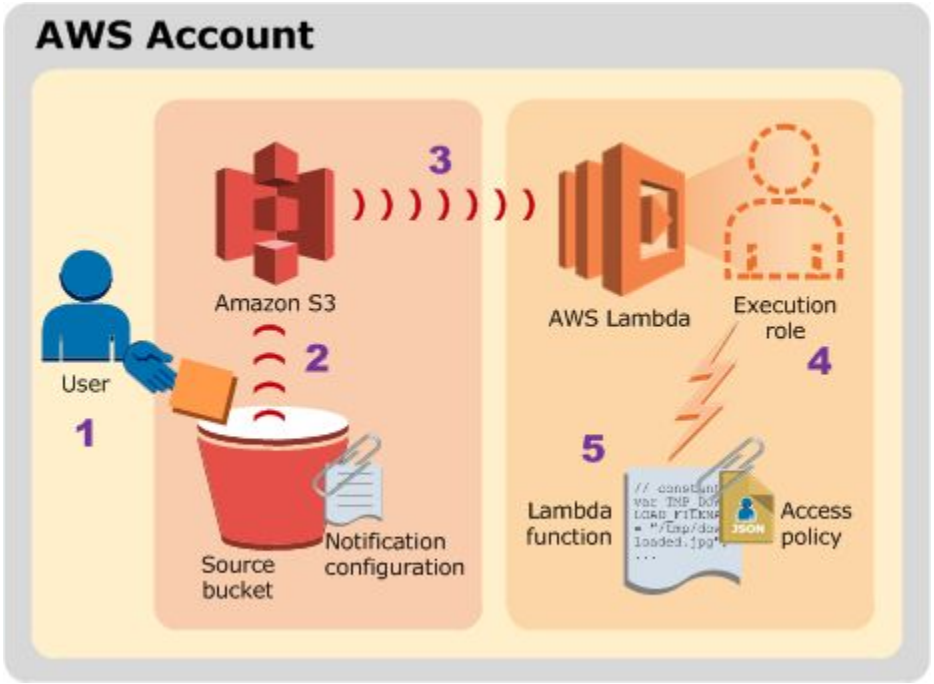
https://www.trek10.com/blog/serverless-framework-for-processes-projects-and-scale/

Plataformas Soportadas

Conectores

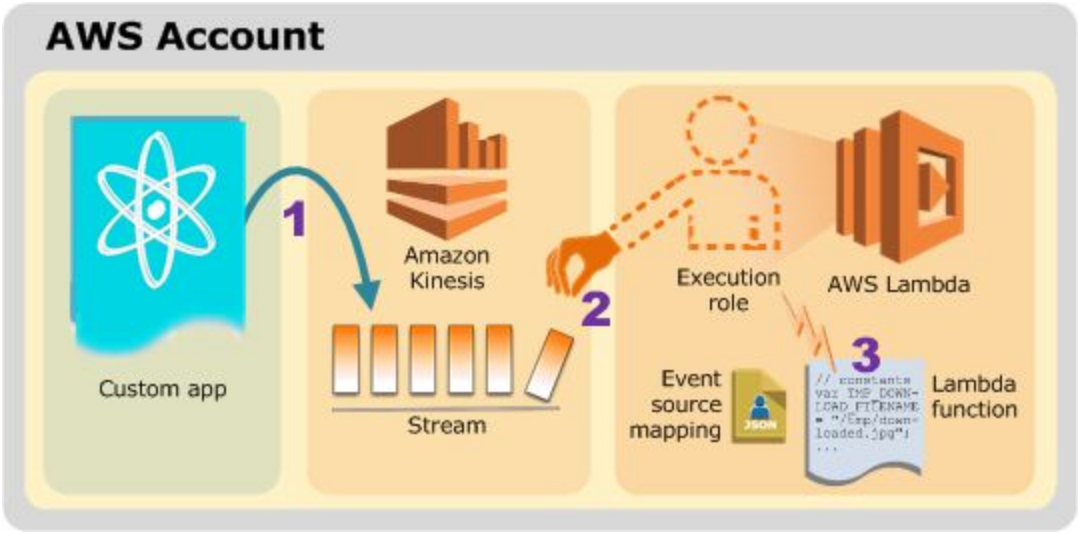
- S3
- Kinesis
- DynamoDB
- SNS
- API Gateway
- CloudWatch
- ....

Lambda Functions Flow



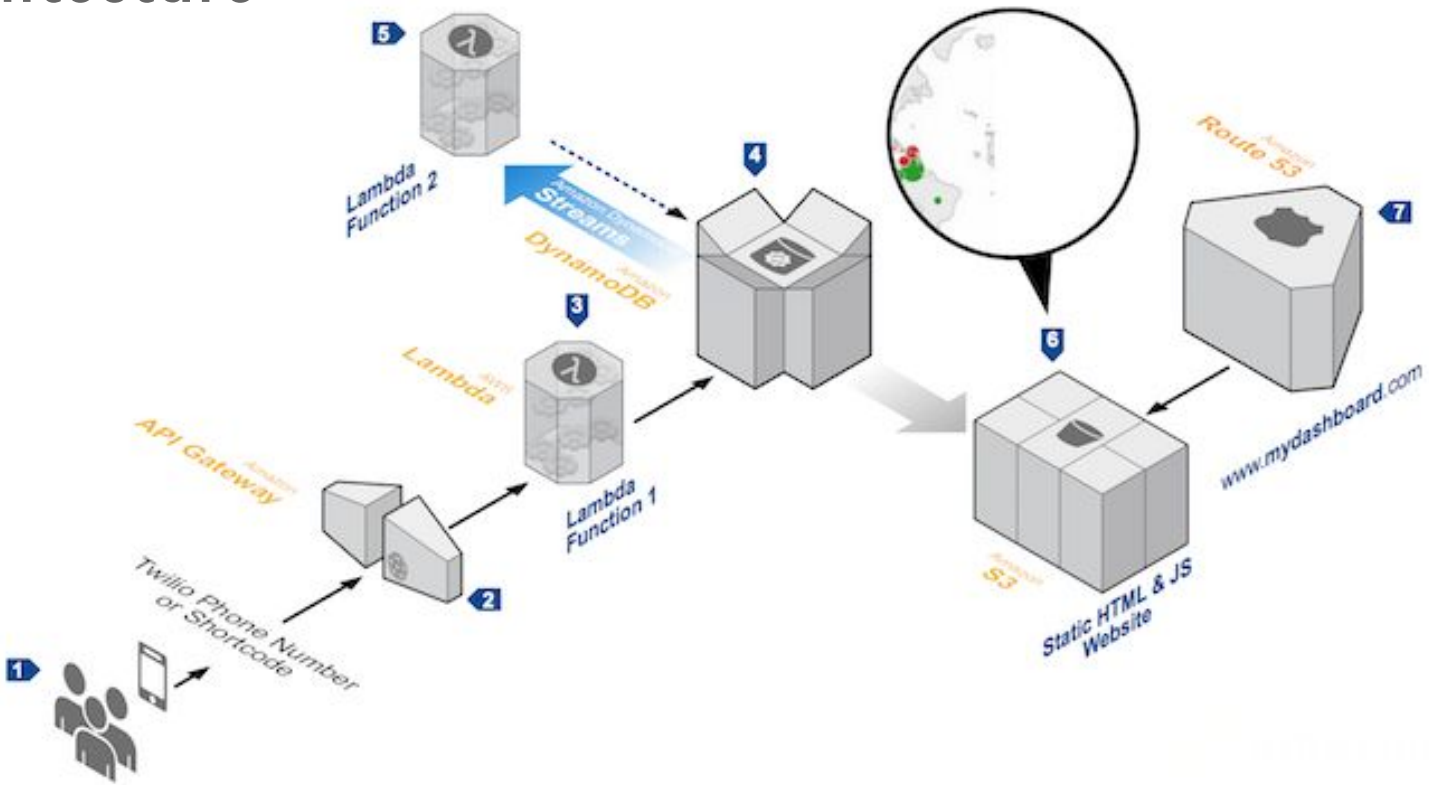
http://www.allthingsdistributed.com/2016/06/aws-lambda-serverless-reference-architectures.html

Lambda Functions Flow



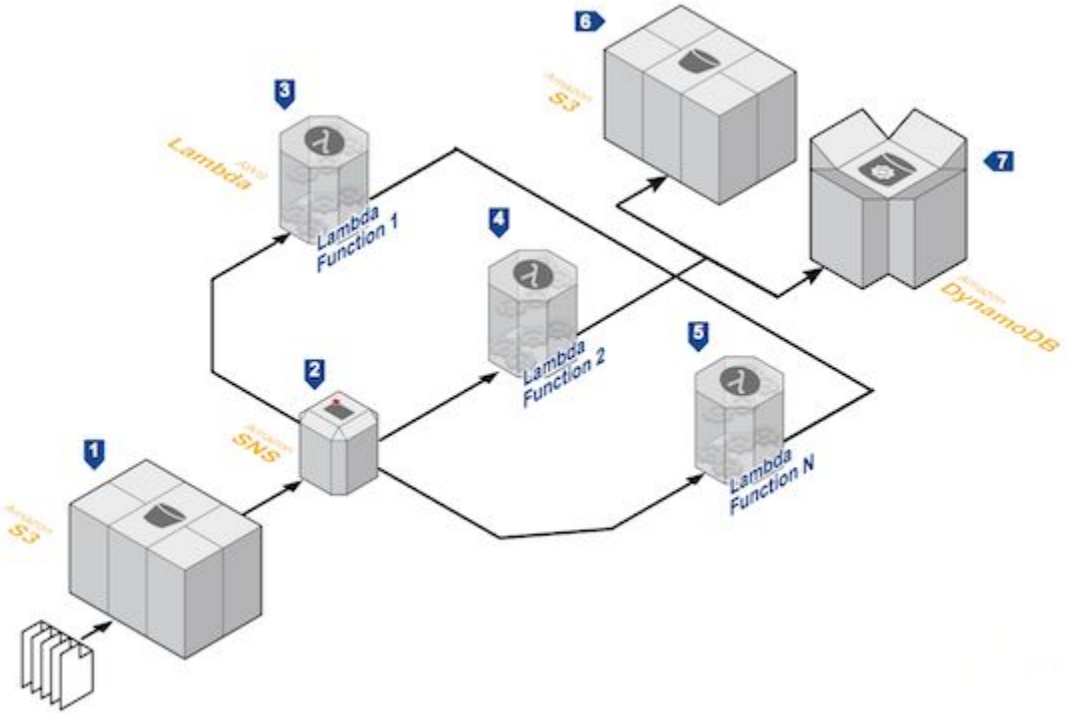
http://docs.aws.amazon.com/lambda/latest/dg/lambda-introduction.html

Web Applications Serverless Reference Architecture



http://www.allthingsdistributed.com/2016/06/aws-lambda-serverless-reference-architectures.html

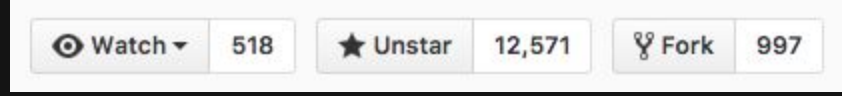
Real-time File Processing Serverless Reference Architecture



http://www.allthingsdistributed.com/2016/06/aws-lambda-serverless-reference-architectures.html

DEMO

http://tinyurl.com/arqui-demo



SERVERLESS

serverless.com

serverless.yml

```yaml
service: serverless-landingpage

provider:
  name: aws
  runtime: nodejs4.3
  region: us-east-2
  stage: dev
  iamRoleStatements:
    - Effect: Allow
      Action:
        - dynamodb:DescribeTable
        - dynamodb:Query
        - dynamodb:Scan
        - dynamodb:GetItem
        - dynamodb:PutItem
      Resource: "arn:aws:dynamodb:us-east-2:*:*"
functions:
  create:
    handler: handler.create
    events:
      - http: POST email
resources:
  Resources:
    EmailsDynamoDbTable:
      Type: 'AWS::DynamoDB::Table'
      DeletionPolicy: Retain
      Properties:
        AttributeDefinitions:
          -
            AttributeName: id
            AttributeType: S
        KeySchema:
          -
            AttributeName: id
            KeyType: HASH
        ProvisionedThroughput:
          ReadCapacityUnits: 1
```

handler.js

```js
'use strict';

module.exports.create = (event, context, callback) => {
  create(event, (error, result) => {
    const response = {
      statusCode: 200,
      body: JSON.stringify(result),
    };

    context.succeed(response);
  });
};

const AWS = require('aws-sdk');
const dynamoDb = new AWS.DynamoDB.DocumentClient();
const uuid = require('uuid');

function create (event, callback) {
  const email = JSON.parse(event.body);

  email.id = uuid.v1();
  email.createdAt = new Date().getTime();
  email.updatedAt = new Date().getTime();

  const params = {
    TableName: 'emails',
    Item: email
  };

  return dynamoDb.put(params, (error, result) => {
    if (error) {
      callback(error);
    }
    callback(error, params.Item);
  });
};
```

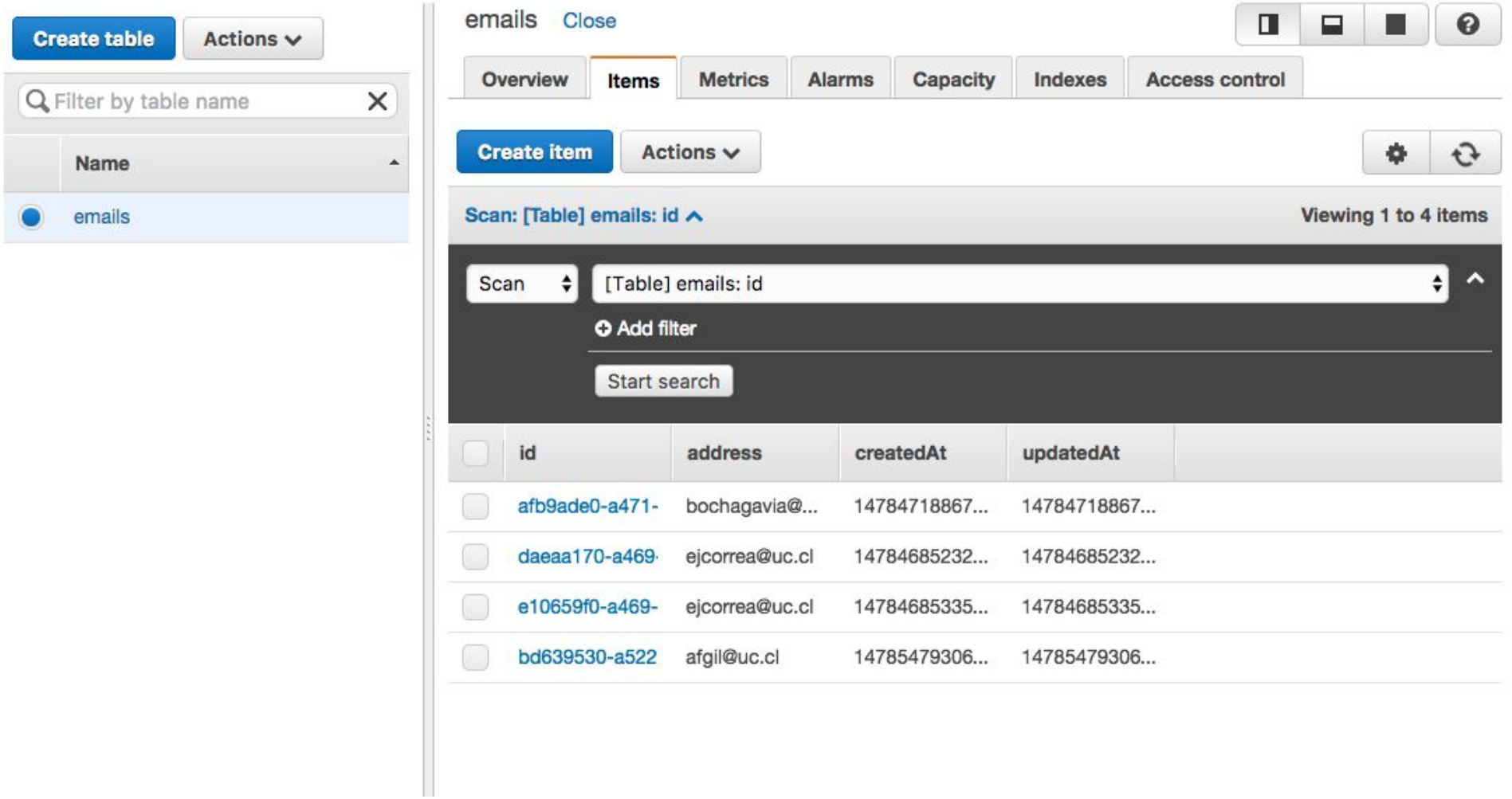
$ serverless deploy

```
> serverless-landingpage@0.1.0 deploy-lambda /Users/isseu/workspace/demo-serverless
> serverless deploy

Serverless: Creating Stack...
Serverless: Checking Stack create progress...
.....
Serverless: Stack create finished...
Serverless: Deprecation Notice: Starting with the next update, we will drop support for Lambda to implicitly create LogGroups.
Please remove your log groups and set "provider.cfLogs: true", for CloudFormation to explicitly create them for you.
Serverless: Packaging service...
Serverless: Uploading CloudFormation file to S3...
Serverless: Uploading service .zip file to S3...
Serverless: Updating Stack...
Serverless: Checking Stack update progress...
.............................
Serverless: Stack update finished...

Service Information
service: serverless-landingpage
stage: dev
region: us-east-2
api keys:
  None
endpoints:
  POST - https://5j8oukea12.execute-api.us-east-2.amazonaws.com/dev/email
functions:
  serverless-landingpage-dev-create: arn:aws:lambda:us-east-2:383295167917:function:serverless-landingpage-dev-create

$
```


Create table
Actions
Filter by table name
Name
emails
emails
Close
Overview
Items
Metrics
Alarms
Capacity
Indexes
Access control
Create item
Actions
Scan: [Table] emails: id
Viewing 1 to 4 items
Scan
[Table] emails: id
Add filter
Start search
id
address
createdAt
updatedAt
afb9ade0-a471-
bochagavia@...
14784718867...
14784718867...
daeaa170-a469-
ejcorrea@uc.cl
14784685232...
14784685232...
e10659f0-a469-
ejcorrea@uc.cl
14784685335...
14784685335...
bd639530-a522
afgil@uc.cl
14785479306...
14785479306...

RETURN;