EJERCICIO 1 - NODEJS CON CLOUDANT

Paso 1. npm init -y

Paso 2. Instalación del SDK
Empiece con su propio proyecto de Node.js y defina este trabajo como su dependencia. Utilice el gestor de paquetes npm desde la línea de mandatos para instalar el SDK:

```
npm install --save @cloudant/cloudant
```

Tenga en cuenta que el archivo package.json ahora muestra el paquete Cloudant.

Paso 3. Inicialización del SDK
Después de inicializar el SDK en la app, puede utilizar IBM Cloudant para almacenar datos. Para inicializar la conexión, especifique sus credenciales y proporcione una función de devolución de llamada para que se ejecute cuando todo esté listo.

Cargue la biblioteca de cliente añadiendo la siguiente definición de require en el archivo server.js.

```
var Cloudant = require('@cloudant/cloudant');
```

Inicialice la biblioteca de cliente suministrando las credenciales. Utilice el plug-in iamauth para crear un cliente de base de datos con una clave de API de IAM.

```
var cloudant = new Cloudant({ url: 'https://examples.cloudant.com', plugins: { iamauth:
{ iamApiKey: 'xxxxxxxxxx' } } });
```

iamApiKey : Está en el json que se copiaron con las credenciales.
Url: Es el endpoint del cloudant.

Liste las bases de datos añadiendo el código siguiente al archivo server.js.

```
cloudant.db.list(function(err, body) {
body.forEach(function(db) {
 console.log(db);
 });
});
```

Cloudant en mayúsculas es el paquete que se carga utilizando require (). cloudant en minúsculas es la conexión autenticada con el servicio IBM Cloudant.

CRUD:

Estas operaciones básicas muestran las acciones para crear, leer, actualizar y suprimir sus documentos utilizando el cliente inicializado.

Creación de un documento

```
var createDocument = function(callback) {
  console.log("Creating document 'mydoc'");
  // especificar el ID del documento para que pueda actualizarlo y suprimirlo posteriormente
  db.insert({ _id: 'mydoc', a: 1, b: 'two' }, function(err, data) {
    console.log('Error:', err);
    console.log('Data:', data);
    callback(err, data);
  });
};
```

Lectura de un documento

```
var readDocument = function(callback) {
  console.log("Reading document 'mydoc'");
  db.get('mydoc', function(err, data) {
    console.log('Error:', err);
    console.log('Data:', data);
    // conservar una copia del documento para que conozca su señal de revisión
    doc = data;
    callback(err, data);
  });
};
```

Actualización de un documento

```
var updateDocument = function(callback) {
  console.log("Updating document 'mydoc'");
  // realizar un cambio en el documento utilizando la copia que se ha conservado al leerlo de nuevo
  doc.c = true;
  db.insert(doc, function(err, data) {
    console.log('Error:', err);
    console.log('Data:', data);
    // conservar la revisión de la actualización para que podamos suprimirla
    doc._rev = data.rev;
    callback(err, data);
  });
};
```

Supresión de un documento

```
var deleteDocument = function(callback) {
  console.log("Deleting document 'mydoc'");
  // proporcionar el ID y revisión que se debe suprimir
  db.destroy(doc._id, doc._rev, function(err, data) {
    console.log('Error:', err);
    console.log('Data:', data);
    callback(err, data);
  });
};
```

1. Arreglar el código para que utilice variables de entorno, expresiones lambda, lets, const, etc.

2. Buscar la forma de ejecutar estas operaciones como cada uno desee.