
# Using HTTP API to create and query Cloudant databases

Estimated time needed: **30** minutes

## Objectives

After completing this lab you will be able to use HTTP API with curl, the command-line tool, to:

- Create, query and drop databases
- Perform basic commands such as inserting, updating and deleting documents
- Create an index to optimize query time

## Prerequisites

In order to complete this lab, you will need to create an instance of Cloudant on IBM Cloud. If you haven't yet created one, you can create one by referring to the Create an Instance of IBM Cloudant lab.

Before starting this lab, it'll also be helpful to have an understanding of basic operations with Cloudant. If you're unfamiliar with Cloudant, feel free to take a look at the Working with Databases in Cloudant lab!

# About This SN Labs Cloud IDE

This Skills Network Labs Cloud IDE provides a hands-on environment for course and project related labs. It utilizes Theia, an open-source IDE (Integrated Development Environment) platform, that can be run on desktop or on the cloud. To complete this lab, we will be using the Cloud IDE based on Theia and Cassandra running in a Docker container.

## Important Notice about this lab environment

Please be aware that sessions for this lab environment are not persisted. Every time you connect to this lab, a new environment is created for you. Any data you may have saved in the earlier session would get lost. Plan to complete these labs in a single session, to avoid losing your data.

Note: While working on this lab, you may be prompted to login when ever your session expires. Use your credentials to authentiate. This may happen when you step out or leave your Cloudant session unattented.

# Exercise 1 - Getting the environment ready

Before you start working with the HTTP API to access the Cloudant, you need to collect the below details of your Cloudant instance.

1. Your Cloudant username
2. Your Cloudant password
3. Your Cloudant url

You can get all these details from the Service Credentials.

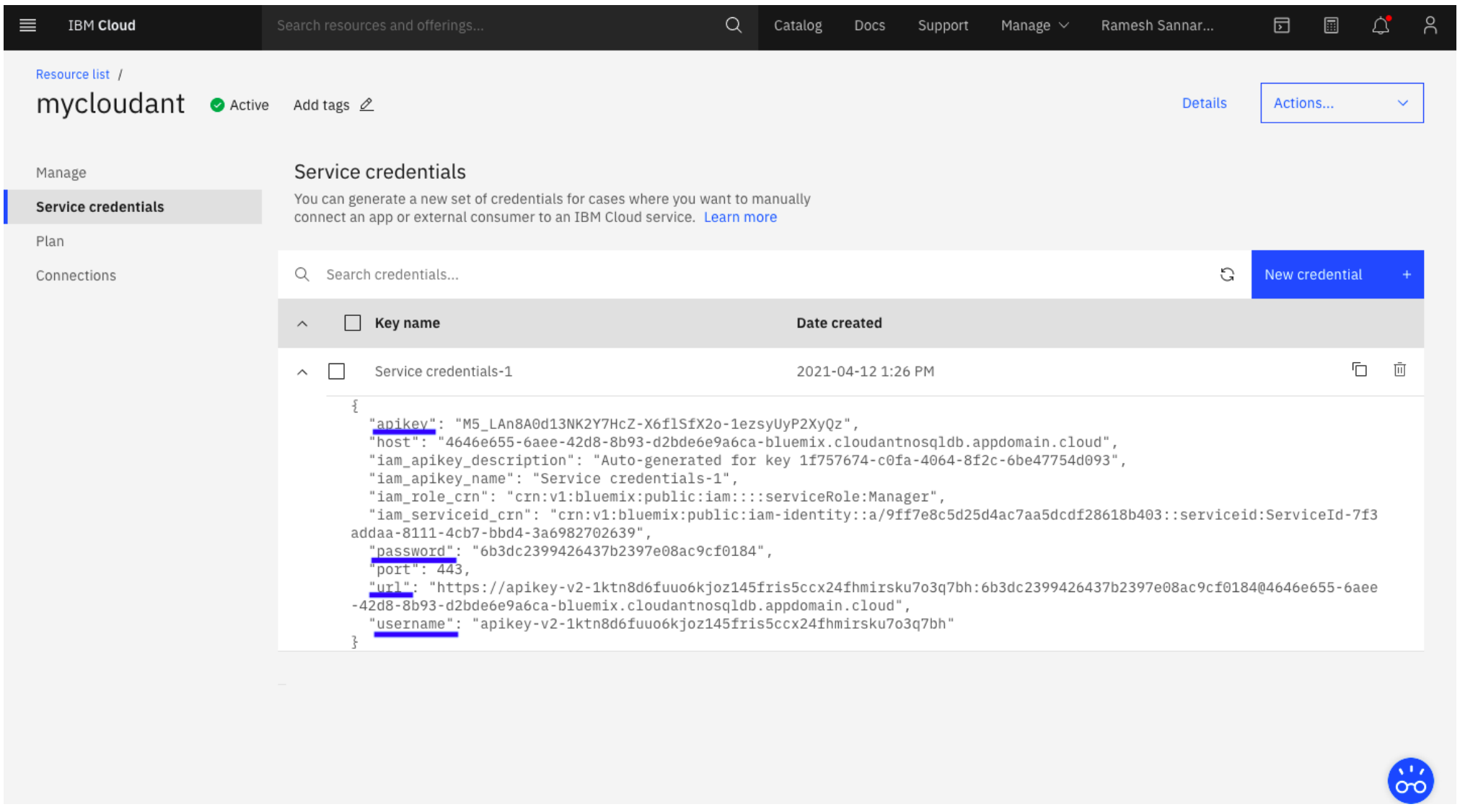
▼ Click here for Hint

Step 1: Go to cloud.ibm.com/resources.

Step 2: Under Services click on your instance of Cloudant.

Step 3: On the Cloudant instance page, click on Service Credentials.

Step 5: Click on the chevron of your Service Credentials. You should see your Service Credentials




Step 5: Copy the url without the double quotes on either side.

Note: If you do not see the password field in your credentials, it is because you have not selected "IAM and legacy credentials" in Exercise 2, Step 5 of the lab Sign Up for an IBM Cloudant Instance. You can fix it by deleting your current Cloudant instance and following the instructions from the beginning to create a new instance of Cloudant.

End of hint.

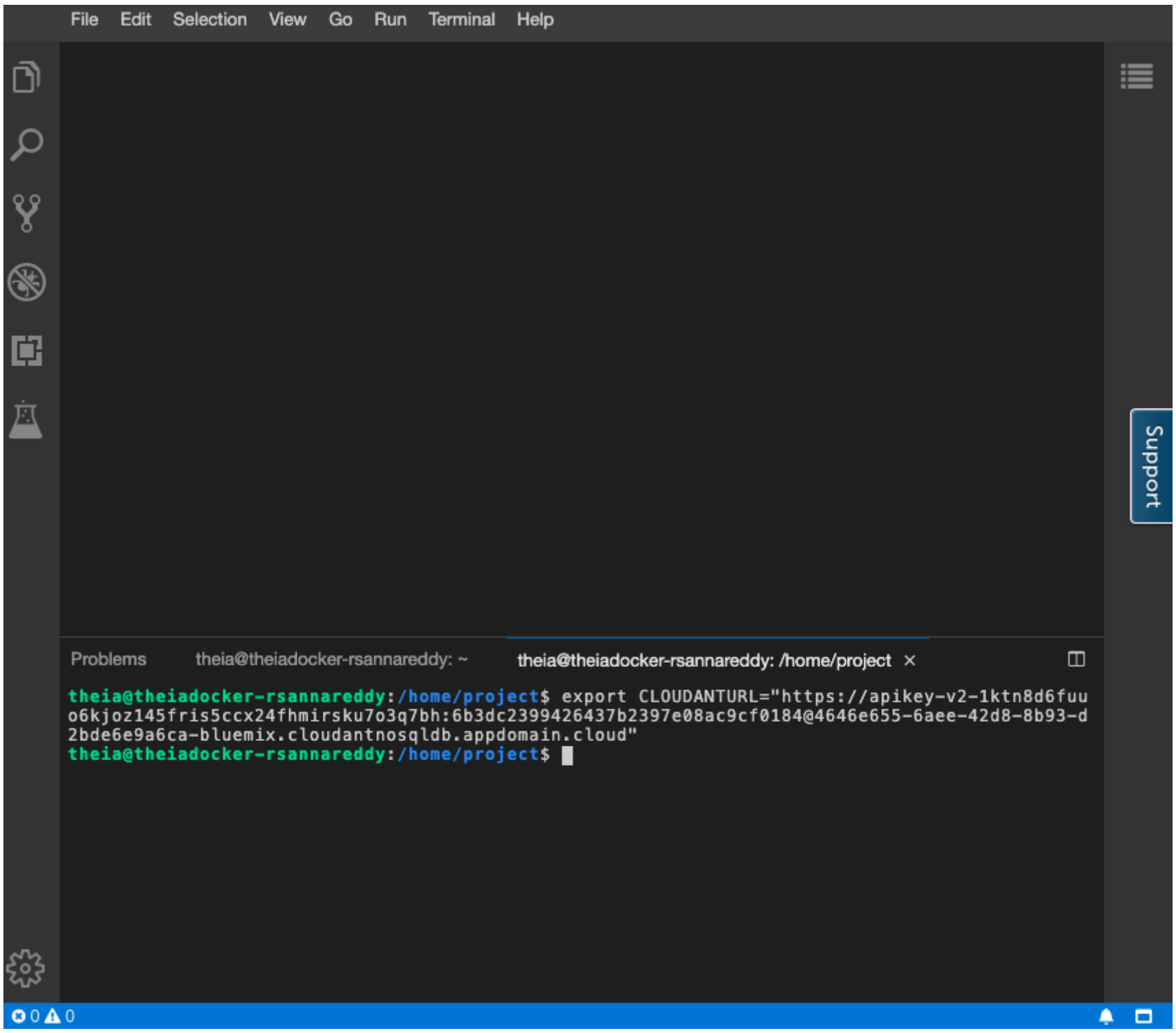
Replace the value of CLOUDANTURL given below with yours.

DO NOT use the values given below. They are all expired credentials.

After replacing with your credentials, copy and paste the below command on the terminal.

This url contains your Cloudant username and password. DO NOT MAKE IT PUBLIC OR SHARE IT WITH ANYONE.

```
export CLOUDANTURL="https://apikey-v2-1ktn8d6fuuo6kjoz145fris5ccx24fhmirsku7o3q7bh:6b3dc2399426437b2397e08ac9cf0184@4646e655-6aee-42d8-8b93-d2bde6e9a6ca-bluemix.cloudantnosqldb.appdomain.cloud"
```




Test if you can reach your Cloudant server from the lab environment.

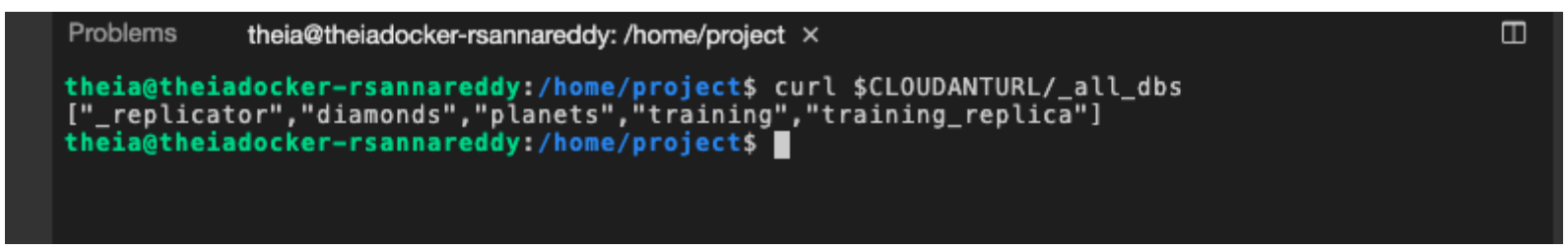
Run the below command on the terminal.

```
curl $CLOUDANTURL
```

If you get an output similiar to the one below, it means you are able to reach your Cloudant server.


Test your credentials.

```
curl $CLOUDANTURL/_all_dbs
```




If you get a list of your databases, it indicates that your credentials are correct. Now we can proceed with the rest of the lab.

# Exercise 2 - Create a database

Run the below command on the terminal to create a database named animals.

```
curl -X PUT $CLOUDANTURL/animals
```

A response of `{"ok":true}` indicates that the database is successfully created.

Verify by listing all databases.

```
curl $CLOUDANTURL/_all_dbs
```

# Exercise 3 - Drop a database

Run the below command to delete the database named animals.

```
curl -X DELETE $CLOUDANTURL/animals
```

A response of `{"ok":true}` indicates that the database is successfully deleted.

Verify that the database is deleted by listing all databases.

```
curl $CLOUDANTURL/_all_dbs
```

# Exercise 4 - Insert a document

> *Create a database named planets.*

▼ Click here for Hint

> use curl command with PUT option and send the database name in the url.

▼ Click here for Solution

```
curl -X PUT $CLOUDANTURL/planets
```

Run the below command to insert a document in the planets database with _id of "1".

```
curl -X PUT $CLOUDANTURL/planets/"1" -d '{
    "name" : "Mercury" ,
    "position_from_sum" :1
     }'
```

If you see `{"ok":true}` in the response, it means the document is successfully inserted.

Note: You can also create a document using POST. When you use POST to create a document, you do not have to mention the id. Cloudant will use an auto generated id for the document.

Verify by listing the document with the _id "1".

```
curl -X GET $CLOUDANTURL/planets/1
```

Make a note of the value for the '_rev' key. You will need it later.

# Exercise 5 - Update a document

To update a document you need its revision id.

Replace the "_rev" below with the one you noted earlier.

```
curl -X PUT $CLOUDANTURL/planets/1 -d '{
    "name" : "Mercury" ,
    "position_from_sum" :1,
    "revolution_time":"88 days",
    "_rev":"1-3fb3ccfe80573e1ae334f0cfa7304f6c"
    }'
```

Verify by listing the document with the _id "1".

```
curl -X GET $CLOUDANTURL/planets/1
```

Make a note of the changed value for the '_rev' key. You will need it later.

Let us add another key "rotation_time" with a value of "59 days" to the existing document.

Run the below command to update the document in the planets database with _id of "1".

REPLACE the value of "_rev" with what you have copied in the previous step.

```
curl -X PUT $CLOUDANTURL/planets/1 -d '{
        "name": "Mercury",
        "position_from_sum": 1,
        "revolution_time": "88 days",
        "rotation_time": "59 days",
        "_rev": "1-3fb3ccfe80573e1ae334f0cfa7304f6c"
}'
```

If you see `{"ok":true}` in the response, it means the document is successfully updated.

Verify by listing the document with the _id "1".

```
curl -X GET $CLOUDANTURL/planets/1
```

Make a note of the '_rev' key. You will need it in the next step.

# Exercise 6 - Delete a document

To delete a document you need its revision id.

REPLACE the value of "_rev" with what you have noted earlier and run the command.

```
curl -X DELETE $CLOUDANTURL/planets/1?rev=2-de9fdd2d971e377c5db2d6425cb38ff1
```

If you see `{"ok":true}` in the response, it means the document is successfully deleted.

Verify by listing the document with the _id "1".

```
curl -X GET $CLOUDANTURL/planets/1
```

# Exercise 7 - Query a database

For this exercise, we'll be using a database called **diamonds**. If you don't have this database, you can create and populate it with the following code:

```
curl -X DELETE $CLOUDANTURL/diamonds
curl -X PUT $CLOUDANTURL/diamonds


curl -X PUT $CLOUDANTURL/diamonds/1 -d '{
    "carat": 0.31,
    "cut": "Ideal",
    "color": "J",
    "clarity": "SI2",
    "depth": 62.2,
    "table": 54,
    "price": 339
  }'

curl -X PUT $CLOUDANTURL/diamonds/2 -d '{
    "carat": 0.2,
    "cut": "Premium",
    "color": "E",
    "clarity": "SI2",
    "depth": 60.2,
    "table": 62,
    "price": 351

  }'

curl -X PUT $CLOUDANTURL/diamonds/3 -d '{
    "carat": 0.32,
    "cut": "Premium",
    "color": "E",
    "clarity": "I1",
    "depth": 60.9,
    "table": 58,
    "price": 342

  }'


curl -X PUT $CLOUDANTURL/diamonds/4 -d '{
    "carat": 0.3,
    "cut": "Good",
    "color": "J",
    "clarity": "SI1",
    "depth": 63.4,
    "table": 54,
    "price": 349

  }'

curl -X PUT $CLOUDANTURL/diamonds/5 -d '{
    "carat": 0.3,
    "cut": "Good",
    "color": "J",
    "clarity": "SI1",
    "depth": 63.8,
    "table": 56,
    "price": 347

  }'

curl -X PUT $CLOUDANTURL/diamonds/6 -d '{
    "carat": 0.3,
    "cut": "Very Good",
    "color": "J",
    "clarity": "SI1",
    "depth": 62.7,
    "table": 59,
    "price": 349
  }'

curl -X PUT $CLOUDANTURL/diamonds/7 -d '{
    "carat": 0.3,
```

```
    "cut": "Good",
    "color": "I",
    "clarity": "SI2",
    "depth": 63.3,
    "table": 56,
    "price": 343
  }'

curl -X PUT $CLOUDANTURL/diamonds/8 -d '{
    "carat": 0.23,
    "cut": "Very Good",
    "color": "E",
    "clarity": "VS2",
    "depth": 63.8,
    "table": 55,
    "price": 339
  }'

curl -X PUT $CLOUDANTURL/diamonds/9 -d '{
    "carat": 0.23,
    "cut": "Very Good",
    "color": "H",
    "clarity": "VS1",
    "depth": 61,
    "table": 57,
    "price": 323
  }'

curl -X PUT $CLOUDANTURL/diamonds/10 -d '{
    "carat": 0.31,
    "cut": "Very Good",
    "color": "J",
    "clarity": "SI1",
    "depth": 59.4,
    "table": 62,
    "price": 346
  }'
```

Now you can start querying the diamond database.

Query for diamond with _id 1

```
curl -X POST $CLOUDANTURL/diamonds/_find \
-H"Content-Type: application/json" \
-d'{
    "selector":
        {
            "_id":"1"
        }
    }'
```

Query for diamonds with carat size of 0.3

```
curl -X POST $CLOUDANTURL/diamonds/_find \
-H"Content-Type: application/json" \
-d'{ "selector":
        {
            "carat":0.3
        }
    }'
```

Query for diamonds with price more than 345

```
curl -X POST $CLOUDANTURL/diamonds/_find \
-H"Content-Type: application/json" \
-d'{ "selector":
        {
            "price":
                {
                    "$gt":345
                }
        }
    }'
```

# Exercise 8 - Create an index

You would have noticed a 'Warning': "No matching index found, create an index to optimize query time." in the output of previous queries. That is Cloudant telling you that, you are running a query on a field that is not indexed.

Let us create an index on the key "price".

```
curl -X POST $CLOUDANTURL/diamonds/_index \
-H"Content-Type: application/json" \
-d'{
    "index": {
        "fields": ["price"]
    }
}'
```

Now let us run the earlier query and you should not see the warning, and the query also runs faster.

Query for diamonds with price more than 345

```
curl -X POST $CLOUDANTURL/diamonds/_find \
-H"Content-Type: application/json" \
-d'{ "selector":
        {
            "price":
                {
                    "$gt":345
                }
        }
    }'
```

# Practice exercises

1. Problem:

> *Set the price of the diamond with id 2 to 352.*

▼ Click here for Hint

find the revision number for the document with id 2. Use it to send a PUT command using curl.

▼ Click here for Solution

Set the rev as per your database before running this command.

```
curl -X PUT $CLOUDANTURL/diamonds/2 -d '{

        "_id": "2",
        "_rev": "1-bb99032cde872d889f41bb4069e23675",
        "carat": 0.2,
        "cut": "Premium",
        "color": "E",
        "clarity": "SI2",
        "depth": 60.2,
        "table": 62,
        "price": 352

    }'
```

2. Problem:

> *Create an index on the key "carat".*

▶ Click here for Hint

▼ Click here for Solution

```
curl -X POST $CLOUDANTURL/diamonds/_index \
-H"Content-Type: application/json" \
-d'{
    "index": {
        "fields": ["carat"]
    }
}'
```

3. Problem:

> *Find all diamonds that are more than 0.3 carats.*

▼ Click here for Hint

> use curl command with the POST and use the $gt opertor in the query JSON document.

▼ Click here for Solution

```
curl -X POST $CLOUDANTURL/diamonds/_find \
-H"Content-Type: application/json" \
-d'{ "selector":
        {
            "carat":
                {
                    "$gt":0.3
                }
        }
    }'
```

4. Problem:

> *Find all diamonds that have a premium cut.*

▼ Click here for Hint

> use curl command with the POST and include the field cut in the query JSON document.

▼ Click here for Solution

```
curl -X POST $CLOUDANTURL/diamonds/_find \
-H"Content-Type: application/json" \
-d'{ "selector":
        {
            "cut":"Premium"

        }
    }'
```

5. Problem:

> *Find all diamonds that are of color 'E'.*

▼ Click here for Hint

> use curl command with the POST and include the field color in the query JSON document.

▼ Click here for Solution

```
curl -X POST $CLOUDANTURL/diamonds/_find \
-H"Content-Type: application/json" \
-d'{ "selector":
        {
            "color":"E"

        }
    }'
```

# Authors

Ramesh Sannareddy

## Other Contributors

Rav Ahuja

# Change Log

| Date (YYYY-MM-DD) | Version | Changed By | Change Description |
|---|---|---|---|
| 2021-11-01 | 0.5 | Kathy An | Updated lab instructions |
| 2021-04-28 | 0.4 | Steve Ryan | Changed IBM cloud links to markdown format |
| 2021-04-14 | 0.3 | Steve Ryan | Review pass |
| 2021-04-13 | 0.2 | Ramesh Sannareddy | Made updates to the lab |
| 2021-03-24 | 0.1 | Ramesh Sannareddy | Created initial version of the lab |