



**Ganpat  
University**

॥ विद्यया समाजोत्कर्षः ॥

**Institute of  
Computer  
Technology**

**Name: Tushar Panchal**

**En.No: 21162101014**

**Sub: EADC (Enterprise Application Development for Cloud)**

**Branch: CBA**

**Batch:61**

## **PRACTICAL 10**

### **❖ Question :**

**You are a developer tasked with deploying an application that requires connectivity to a Cloudant database on IBM Cloud. The application is developed using a serverless architecture and needs to be deployed on IBM Cloud Code Engine.**

Steps to be accomplished to deploy the application based on Cloudant database connectivity on IBM Cloud Code Engine:

#### **Practical 10.1: Prepare Your Application**

Ensure your application is configured to connect to the Cloudant database. This may involve providing the necessary credentials and endpoint information in your application code or configuration files.

#### **Practical 10.2: Set Up IBM Cloudant Database**

If you haven't already, create a Cloudant database instance on IBM Cloud. Take note of the credentials and endpoint URL for this database as you will need them to configure the connection in your application.

#### **Practical 10.3: Create IBM Cloud Code Engine Project**

Log in to your IBM Cloud account and navigate to the IBM Cloud Code Engine dashboard. Create a new project to organize your application resources.

#### **Practical 10.4: Deploy Application**

Use the IBM Cloud Code Engine CLI or web interface to deploy your application

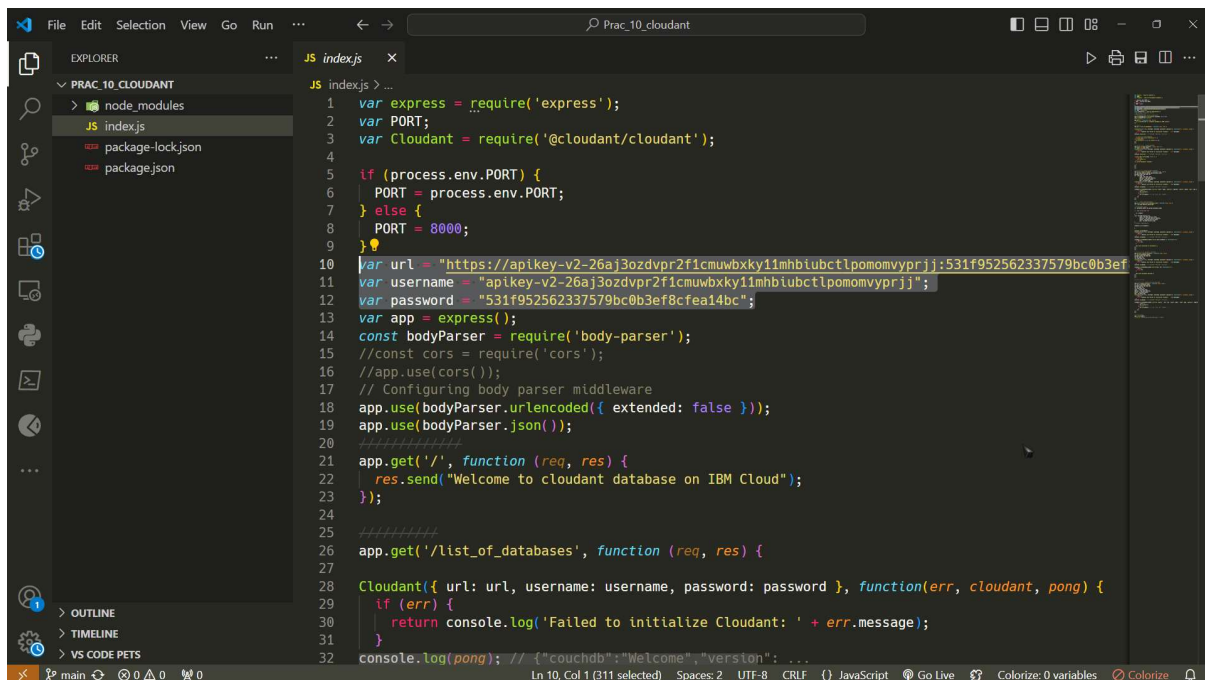
## Practical 10.5: Test Connectivity

Test the connectivity between your application and the Cloudant database to ensure that data can be read from and written to the database as expected.

**TASK:** Create a sample HTML form to collect data from the user for registration including fields like name, phone number, email address, city, country, pincode. Integrate it with node js application to collect the data from the form and update the same data on cloudant database.

## » Practical 10.1 : Prepare your application.

First make sure you enter cloudant service credentials in code:

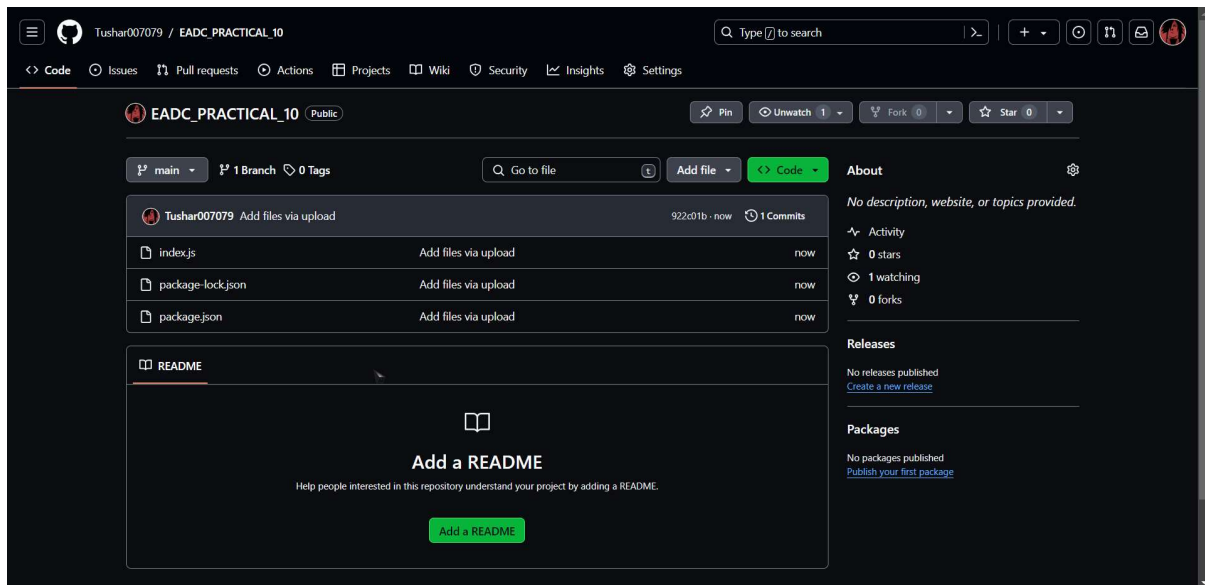


```

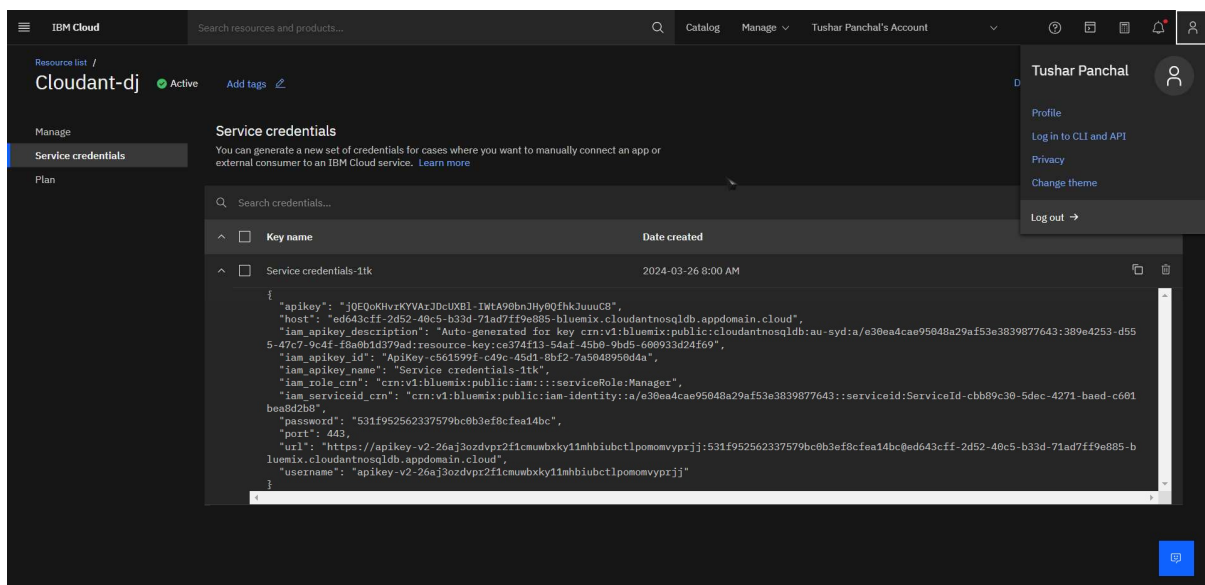
1  var express = require('express');
2  var PORT;
3  var Cloudant = require('@cloudant/cloudant');
4
5  if (process.env.PORT) {
6    PORT = process.env.PORT;
7  } else {
8    PORT = 8000;
9  }
10
11 var url = "https://apikey-v2-26aj3ozdvpr2f1cmuwbxky1mhbiubctlpomomvyprij:531f952562337579bc0b3ef";
12 var username = "apikey-v2-26aj3ozdvpr2f1cmuwbxky1mhbiubctlpomomvyprij";
13 var password = "531f952562337579bc0b3ef8cfea14bc";
14 var app = express();
15 const bodyParser = require('body-parser');
16 //const cors = require('cors');
17 //app.use(cors());
18 // Configuring body parser middleware
19 app.use(bodyParser.urlencoded({ extended: false }));
20 app.use(bodyParser.json());
21
22 app.get('/', function (req, res) {
23   res.send("Welcome to cloudant database on IBM Cloud");
24 });
25
26
27
28 app.get('/list_of_databases', function (req, res) {
29
30   Cloudant({ url: url, username: username, password: password }, function(err, cloudant, pong) {
31     if (err) {
32       return console.log('Failed to initialize Cloudant: ' + err.message);
33     }
34     console.log(pong); // {"couchdb": "Welcome", "version": "...

```

Now push it on github repository:



## ➤ Practical 10.2 : Set Up IBM Cloudant Database.



## ➤ Practical 10.3 : Create IBM Cloud Code Engine Project.

## ➤ Practical 10.4 : Deploy Application.

Now in code engine project we have create a new application.

Name it and select build container image from source and don't do any changes in other things

Now hit specify build details in that provide your github repo url:

In strategy select cloud native buildpacks

Now log in to dockehub and get access token to create registry container for output

In dockerhub click on your profile and enter into my account then in security section create new access token:

Then in target registry select dockehub and provide that access token and your username of dockerhub:

**Create registry secret**

Stores credentials to access a container registry.

Secret name: practical10

Target registry: ☒ IBM Container Registry ☒ Dockerhub ☐ Other

Registry server: <https://index.docker.io/v1/>

Username: tk007079

Access token: [masked]

Email (optional): username@email.com

Buttons: Cancel, Create

That's it now our registry secret has been added :

**Specify build details**

Learn more about image builds.

Source Strategy Output

Registry server: <https://index.docker.io/v1/>

Registry secret: practical10

Namespace: tk007079

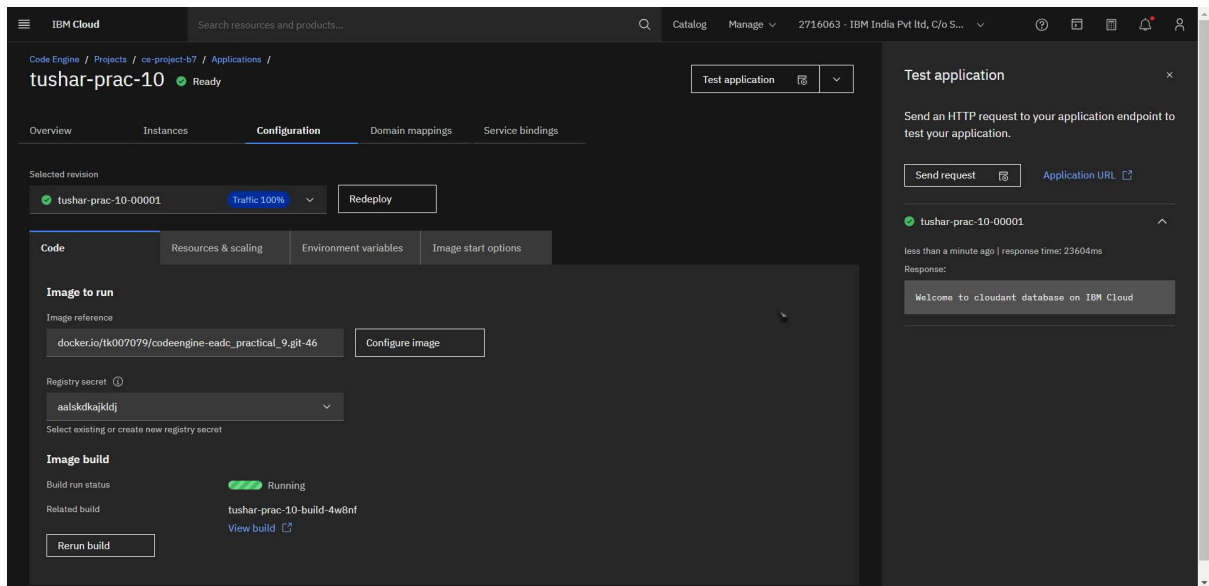
Repository (image name): codeengine-eadc\_practical\_10-git-13

Tag: [empty]

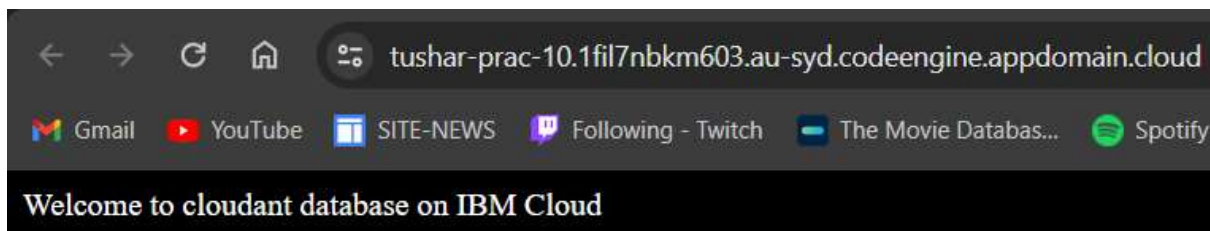
Buttons: Previous, Done

Now hit create to create application.

Now test application & send request then access application url.

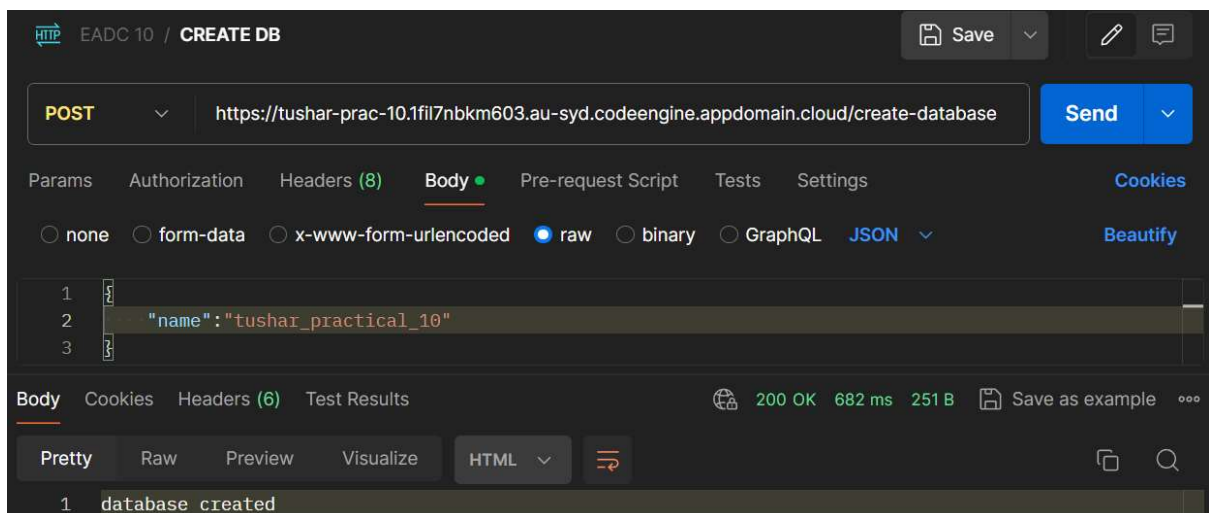


» As you can see our application url is working :



## » Practical 10.5 : Test Connectivity.

Now to test our application I send post request to create a DB.



## Now inserting one document in database:

The screenshot shows a REST client interface with the following details:

- Method:** POST
- URL:** `https://tushar-prac-10.1fil7nbkm603.au-syd.codeengine.appdomain.cloud/insert-document`
- Body (raw):**

```
{
  "db": "tushar_practical_10",
  "id": "1",
  "name": "Tushar",
  "address": "Los Angeles",
  "phone": "7777777",
  "age": "69"
}
```
- Status:** 200 OK, 18.76 s, 305 B
- Response Body (JSON):**

```
{
  "ok": true,
  "id": "1",
  "rev": "1-13d7f78be24c5c07655f772b6d317fba"
}
```

Now we can also see in cloudant dashboard that the document is created:

The screenshot shows the Cloudant dashboard for the database `tushar_practical_10`. The document is listed in the `All Documents` view.

id	key	value
1	1	{ "rev": "1-13d7f78be24c5c07655f772b6d317fba" }

At the bottom of the dashboard, it shows: "Showing document 1 - 1. Documents per page: 20".



## Now I'm updating a document that already exist:

The screenshot shows a REST client interface with the following details:

- Method:** PUT
- URL:** `https://tushar-prac-10.1fil7nbkm603.au-syd.codeengine.appdomain.cloud/update-document/`
- Body Type:** raw (selected), JSON (available)
- Body Content:**

```

1 {
2   "db": "tushar_practical_10",
3   "_id": "1",
4   "rev": "1-13d7f78be24c5c07655f772b6d317fba",
5   "name": "Tushar_2.0",
6   "address": "Los Angeles_2.0",
7   "phone": "7777777_2.0",
8   "age": "69"
9 }

```
- Response:** 200 OK, 19.44 s, 336 B
- Response Body:**

```

1 {
2   "ok": true,
3   "id": "3d2b85a8a864d47d97f5f07adc982317",
4   "rev": "1-4503f5d264985ff5c3b4adf13b6d5c5d"
5 }

```

Now we can also see in cloudant dashboard that the document is updated..

The screenshot shows the Cloudant dashboard interface with the following details:

- Database:** tushar\_practical\_10
- Document ID:** 3d2b85a8a864d47d97f5f07adc982317
- Actions:** Save Changes, Upload Attachment, Clone Document, Delete
- Document Content:**

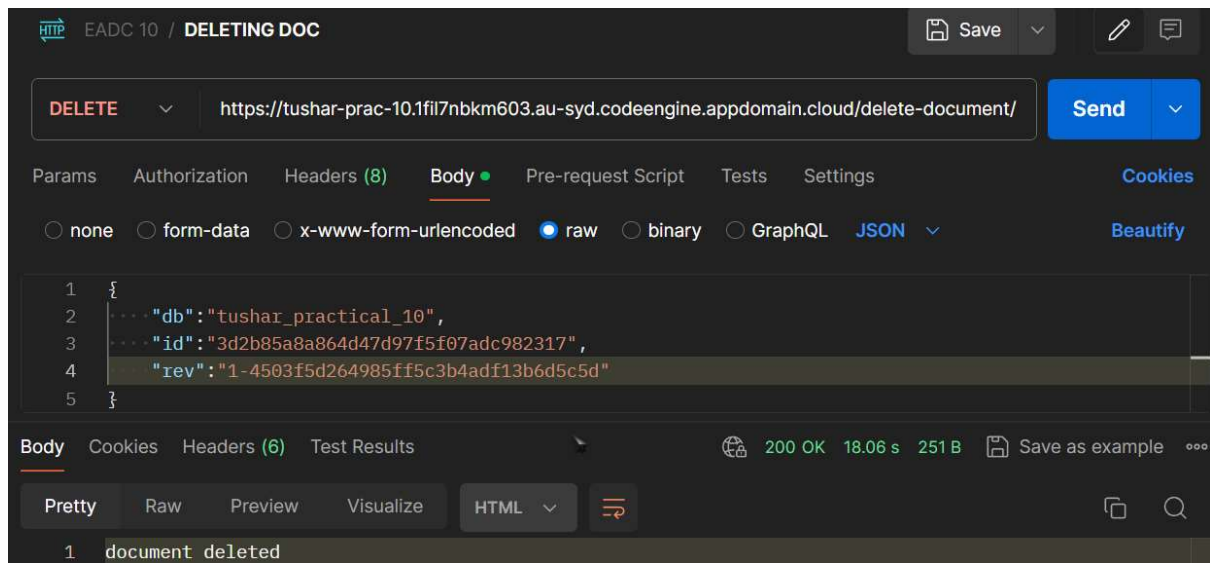
```

1 {
2   "_id": "3d2b85a8a864d47d97f5f07adc982317",
3   "_rev": "1-4503f5d264985ff5c3b4adf13b6d5c5d",
4   "name": "Tushar_2.0",
5   "age": "69",
6   "phone": "7777777_2.0"
7 }

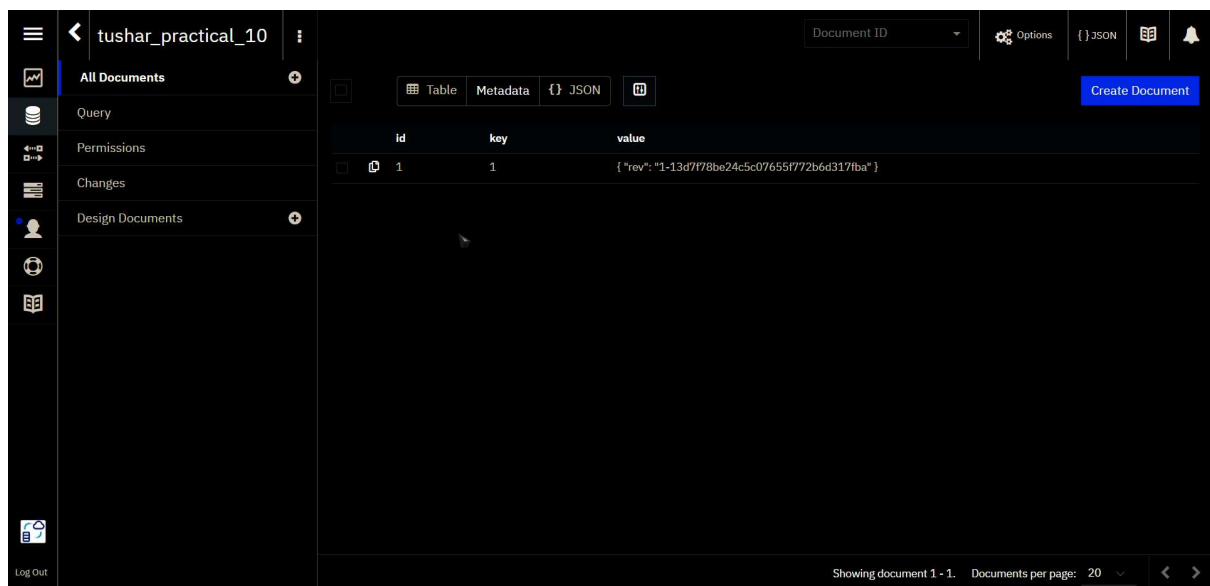
```



## Delete a one document in database:



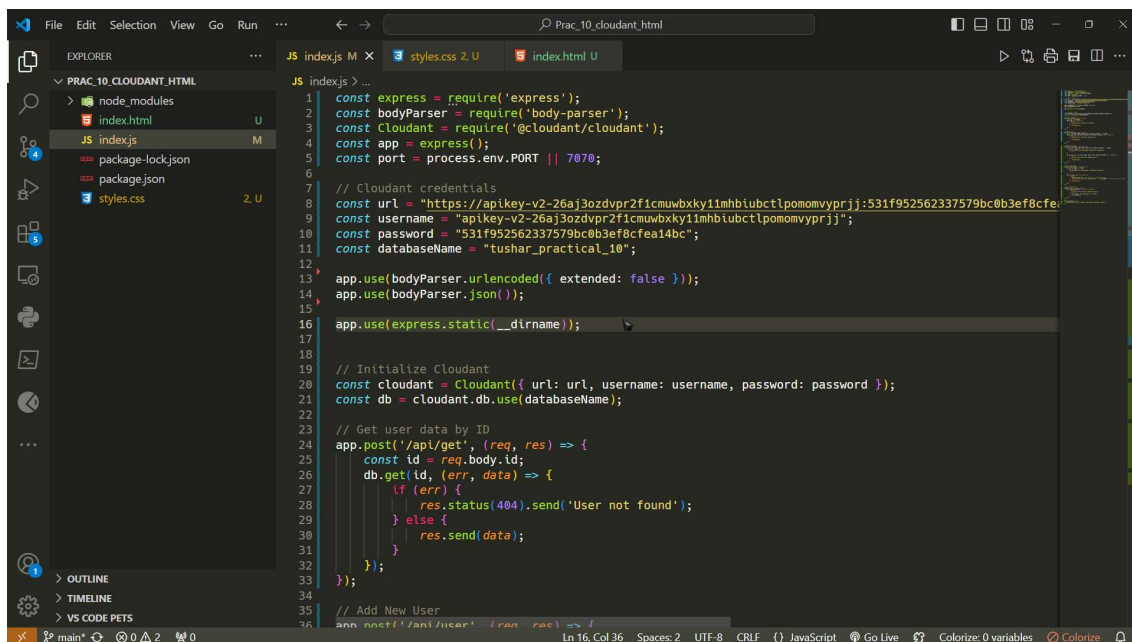
Now we can also see in cloudant dashboard that the document is deleted..



» **TASK: Create a sample HTML form to collect data from the user for registration including fields like name, phone number, email address, city, country, pincode. Integrate it with node js application to collect the data from the form and update the same data on cloudant database.**

For this create a html form and api :

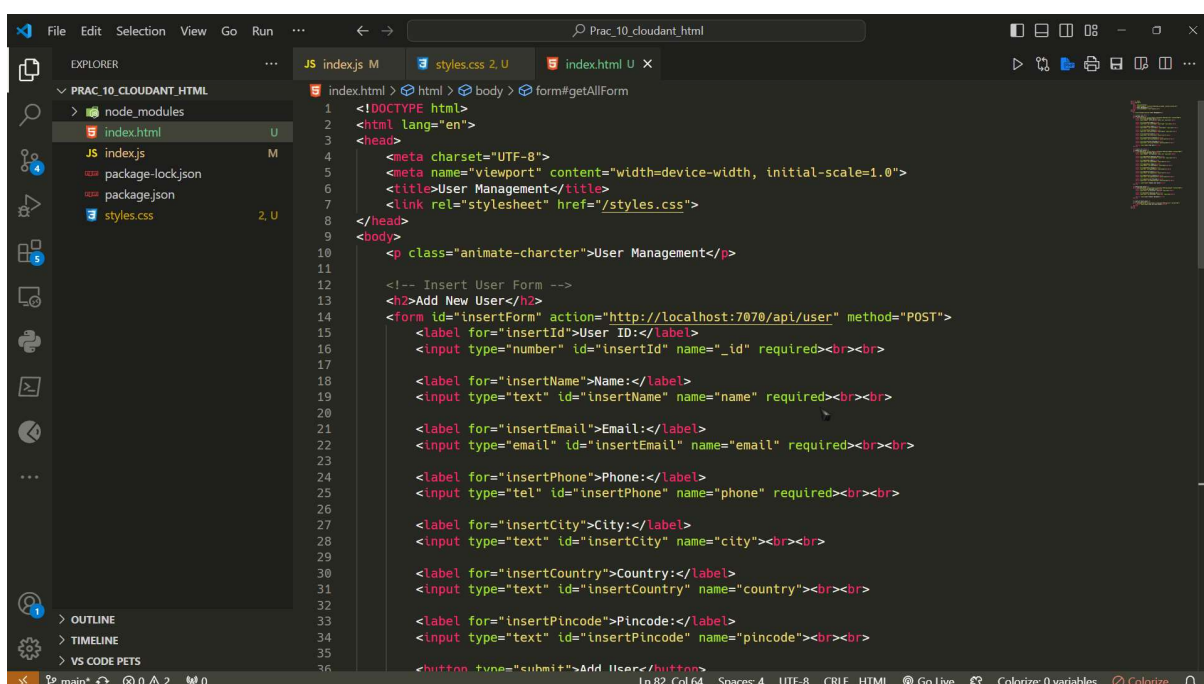
In first we deploy using this endpoint **http://localhost:7070**



```

1  const express = require('express');
2  const bodyParser = require('body-parser');
3  const Cloudant = require('@cloudant/cloudant');
4  const app = express();
5  const port = process.env.PORT || 7070;
6
7  // Cloudant credentials
8  const url = "https://apikey-v2-26aj3ozdvpr2f1cmuwbxky11mhbiubctlpomomvyprjj:531f952562337579bc0b3ef8cfe";
9  const username = "apikey-v2-26aj3ozdvpr2f1cmuwbxky11mhbiubctlpomomvyprjj";
10 const password = "531f952562337579bc0b3ef8cfe14bc";
11 const dbName = "tushar_practical_10";
12
13 app.use(bodyParser.urlencoded({ extended: false }));
14 app.use(bodyParser.json());
15
16 app.use(express.static(__dirname));
17
18 // Initialize Cloudant
19 const cloudant = Cloudant({ url: url, username: username, password: password });
20 const db = cloudant.db.use(databaseName);
21
22 // Get user data by ID
23 app.post('/api/get', (req, res) => {
24   const id = req.body.id;
25   db.get(id, (err, data) => {
26     if (err) {
27       res.status(404).send('User not found');
28     } else {
29       res.send(data);
30     }
31   });
32 });
33
34 // Add New User
35 app.post('/api/user', (req, res) => {

```

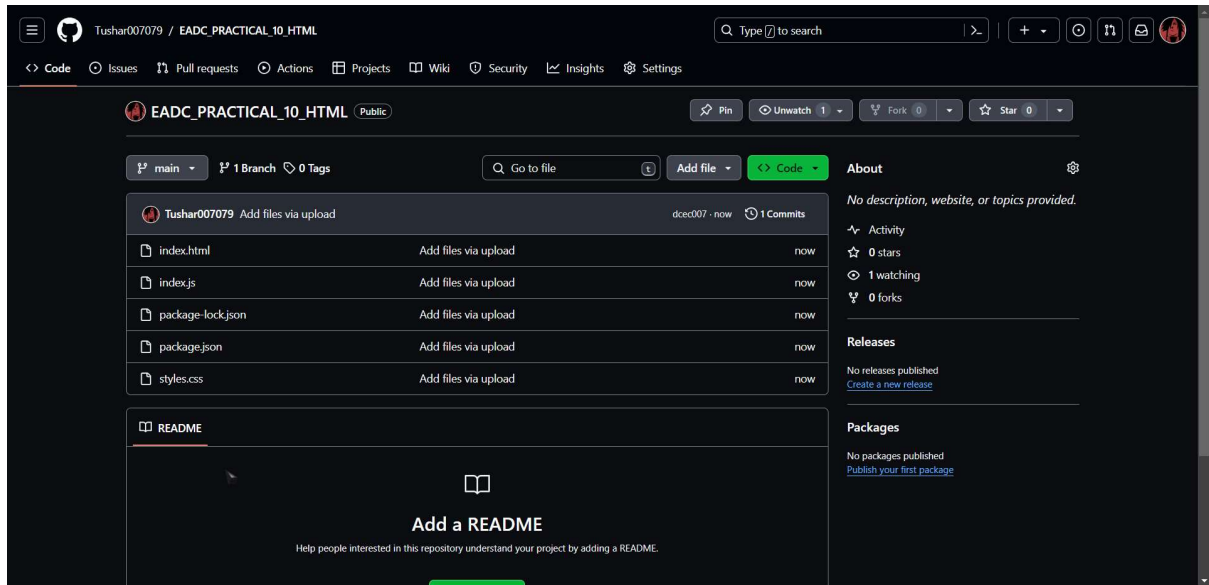


```

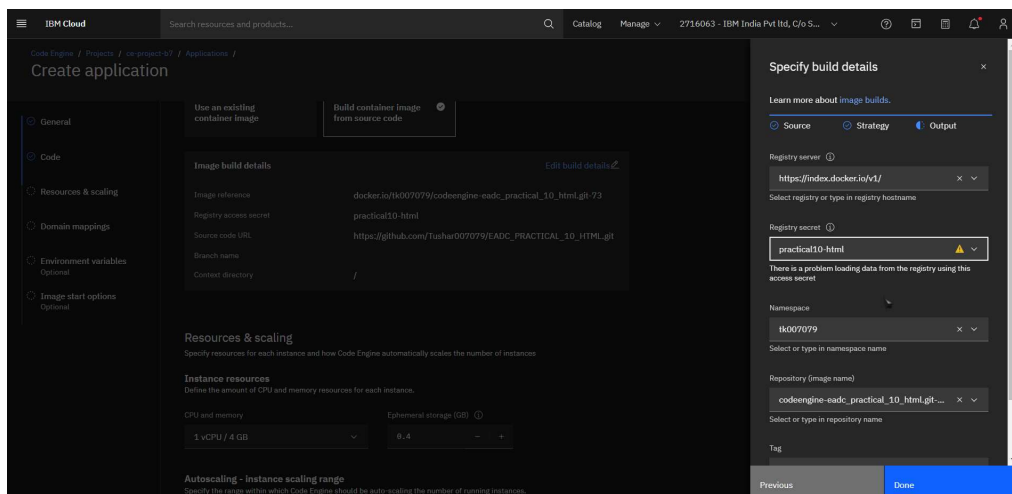
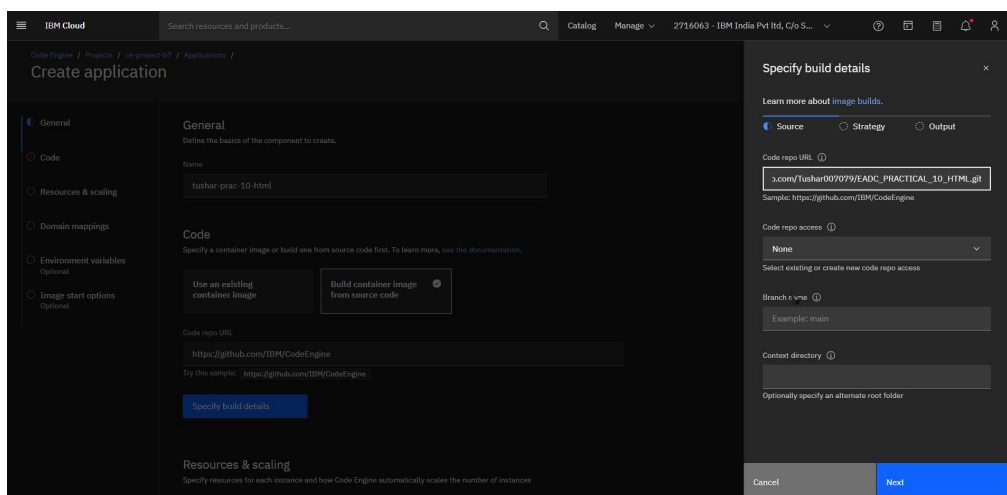
1  <!DOCTYPE html>
2  <html lang="en">
3  <head>
4    <meta charset="UTF-8">
5    <meta name="viewport" content="width=device-width, initial-scale=1.0">
6    <title>User Management</title>
7    <link rel="stylesheet" href="/styles.css">
8  </head>
9  <body>
10   <p class="animate-character">User Management</p>
11
12   <!-- Insert User Form -->
13   <h2>Add New User</h2>
14   <form id="insertForm" action="http://localhost:7070/api/user" method="POST">
15     <label for="insertId">User ID:</label>
16     <input type="number" id="insertId" name="_id" required><br><br>
17
18     <label for="insertName">Name:</label>
19     <input type="text" id="insertName" name="name" required><br><br>
20
21     <label for="insertEmail">Email:</label>
22     <input type="email" id="insertEmail" name="email" required><br><br>
23
24     <label for="insertPhone">Phone:</label>
25     <input type="tel" id="insertPhone" name="phone" required><br><br>
26
27     <label for="insertCity">City:</label>
28     <input type="text" id="insertCity" name="city"><br><br>
29
30     <label for="insertCountry">Country:</label>
31     <input type="text" id="insertCountry" name="country"><br><br>
32
33     <label for="insertPincode">Pincode:</label>
34     <input type="text" id="insertPincode" name="pincode"><br><br>
35
36     <button type="submit">Add User</button>

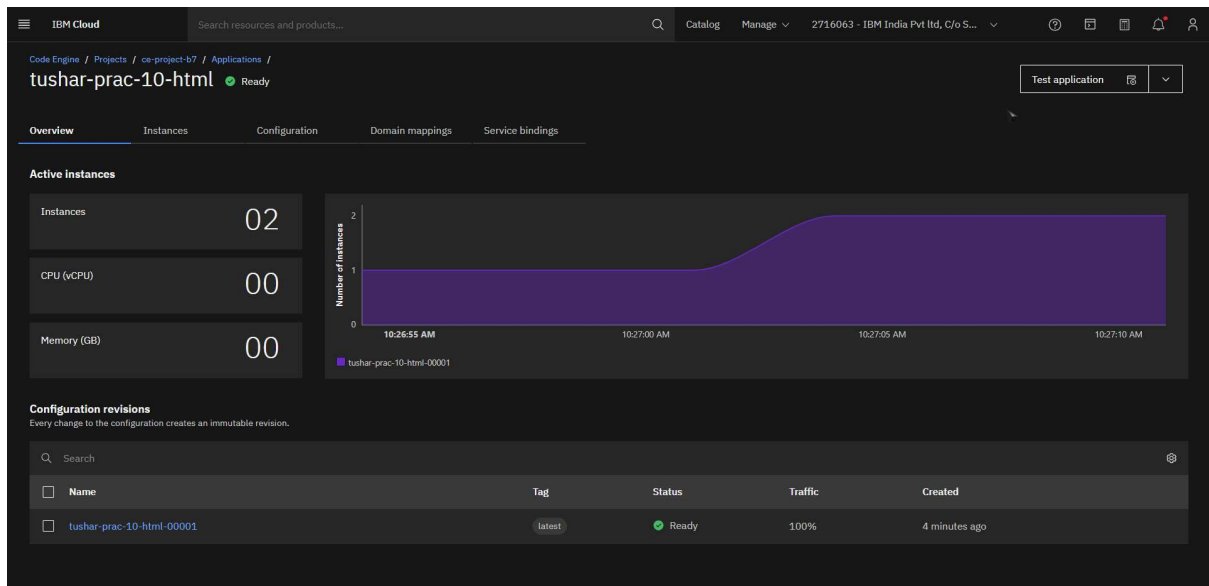
```

Then push it on github repository:



Now deploy it on code engine:





We can see our application in the URL.

tushar-prac-10-html.1fil7nbkm603.au-syd.codeengine.appdomain.cloud

# USER MANAGEMENT

## ADD NEW USER

User ID:

Name:

Email:

Phone:

City:

After that we will get an application url :

<https://tushar-prac-10-html.1fil7nbkm603.au-syd.codeengine.appdomain.cloud>

now you have to update this endpoint on github repository

```

7 <link rel="stylesheet" href="/styles.css">
8 </head>
9 <body>
10 <p class="animate-character">User Management</p>
11
12 <!-- Insert User Form -->
13 <h2>Add New User</h2>
14 <form id="insertForm" action="https://tushar-prac-10-html-1f117bka683.au-syd.codeengine.appdomain.cloud/api/user" method="POST">
15   <label for="insertId">User ID:</label>
16   <input type="number" id="insertId" name="id" required><br>
17
18   <label for="insertName">Name:</label>
19   <input type="text" id="insertName" name="name" required><br>
20
21   <label for="insertEmail">Email:</label>
22   <input type="email" id="insertEmail" name="email" required><br>
23
24   <label for="insertPhone">Phone:</label>
25   <input type="tel" id="insertPhone" name="phone" required><br>
26
27   <label for="insertCity">City:</label>
28   <input type="text" id="insertCity" name="city"><br>
29
30   <label for="insertCountry">Country:</label>
31   <input type="text" id="insertCountry" name="country"><br>
32
33   <label for="insertPincode">Pincode:</label>
34   <input type="text" id="insertPincode" name="pincode"><br>
35
36   <button type="submit">Add User</button>
37 </form>
38

```

Then make new application and deploy this updated repository

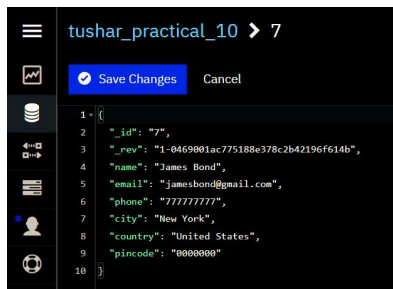
**Now I'm adding data:**

```

{
  "ok": true,
  "id": "7",
  "rev": "1-0469001ac775188e378c2b42196f614b"
}

```

Now we can also see in cloudbant dashboard that the document is created.



## Updating data of existing document:

UPDATE USER DATA

User ID:

Revision ID:

Name:

Email:

Phone:

City:

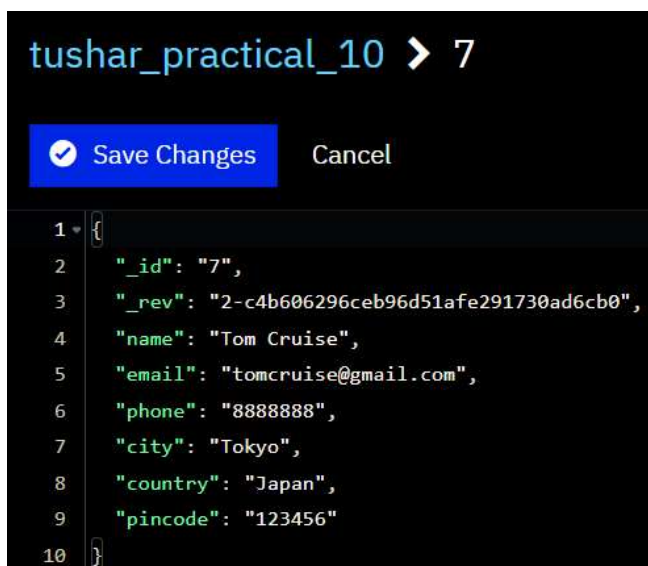
Country:

Pincode:

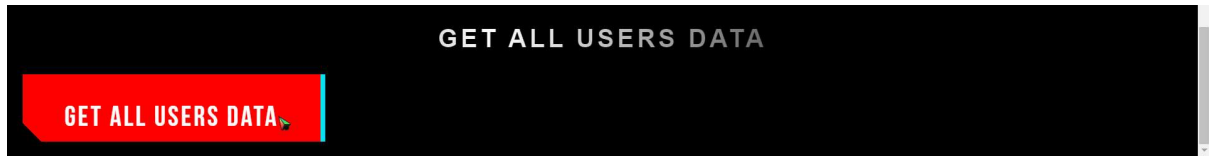
UPDATE USER DATA

DELETE USER DATA

Now we can also see in cloudant dashboard that the document is updated:



## Getting details of all existing users or documents of DB:



```
[
  {
    "_id": "1",
    "_rev": "1-13d7f78be24c5c07655f772b6d317fba",
    "name": "Tushar",
    "phone": "7777777",
    "age": "69"
  },
  {
    "_id": "7",
    "_rev": "2-c4b606296ceb96d51afe291730ad6cb0",
    "name": "Tom Cruise",
    "email": "tomcruise@gmail.com",
    "phone": "8888888",
    "city": "Tokyo",
    "country": "Japan",
    "pincode": "123456"
  }
]
```

## Now deleting existing user :

A screenshot of a dark-themed user interface for deleting a document. The title 'DELETE USER DATA' is at the top. Below it, there are two input fields: 'Document ID:' with the value '7' and 'Revision ID:' with the value '2-c4b606296ceb96d51afe291730ad6cb0'. At the bottom, there is a red button labeled 'DELETE DOCUMENT'.

You can see here my document deleted from DB

A screenshot of a document management interface. The left sidebar shows a menu with 'All Documents' selected. The main area displays a table of documents. The table has columns for '\_id', 'age', 'name', and 'phone'. The first row shows a document with \_id '1', age '69', name 'Tushar', and phone '7777777'. There is a 'Create Document' button in the top right corner.

_id	age	name	phone
1	69	Tushar	7777777