

Name - Yash Lakhtariya
Enrollment number - 21162101012
Branch - CBA Batch - 61
EADC Practical 10

Aim : 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

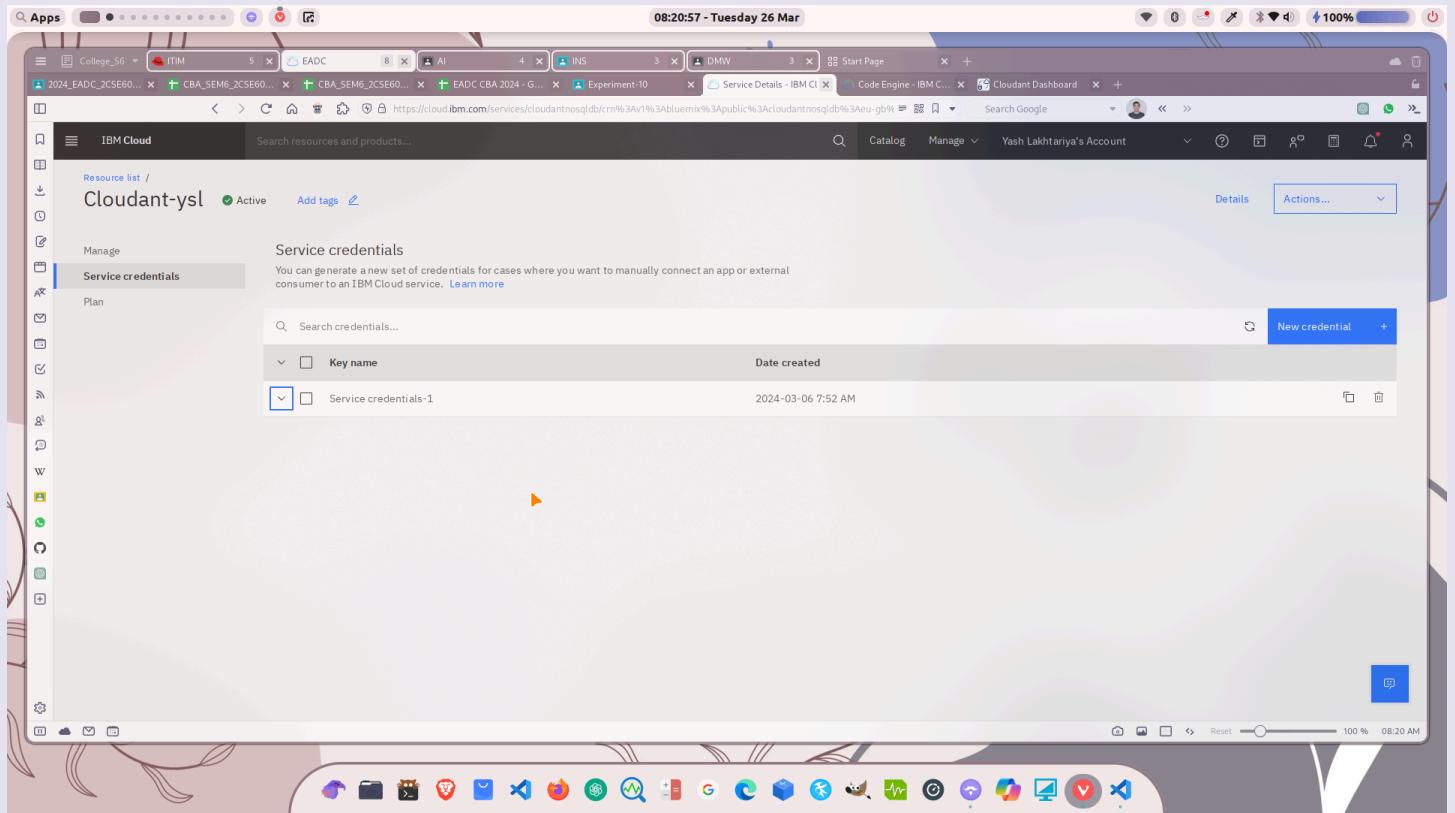
Practical 10.5: Test Connectivity

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.

Name - Yash Lakhtariya
Enrollment number - 21162101012
Branch - CBA Batch - 61
EADC Practical 10

Steps and Screenshots :

1. Create Service Credentials in Cloudant service

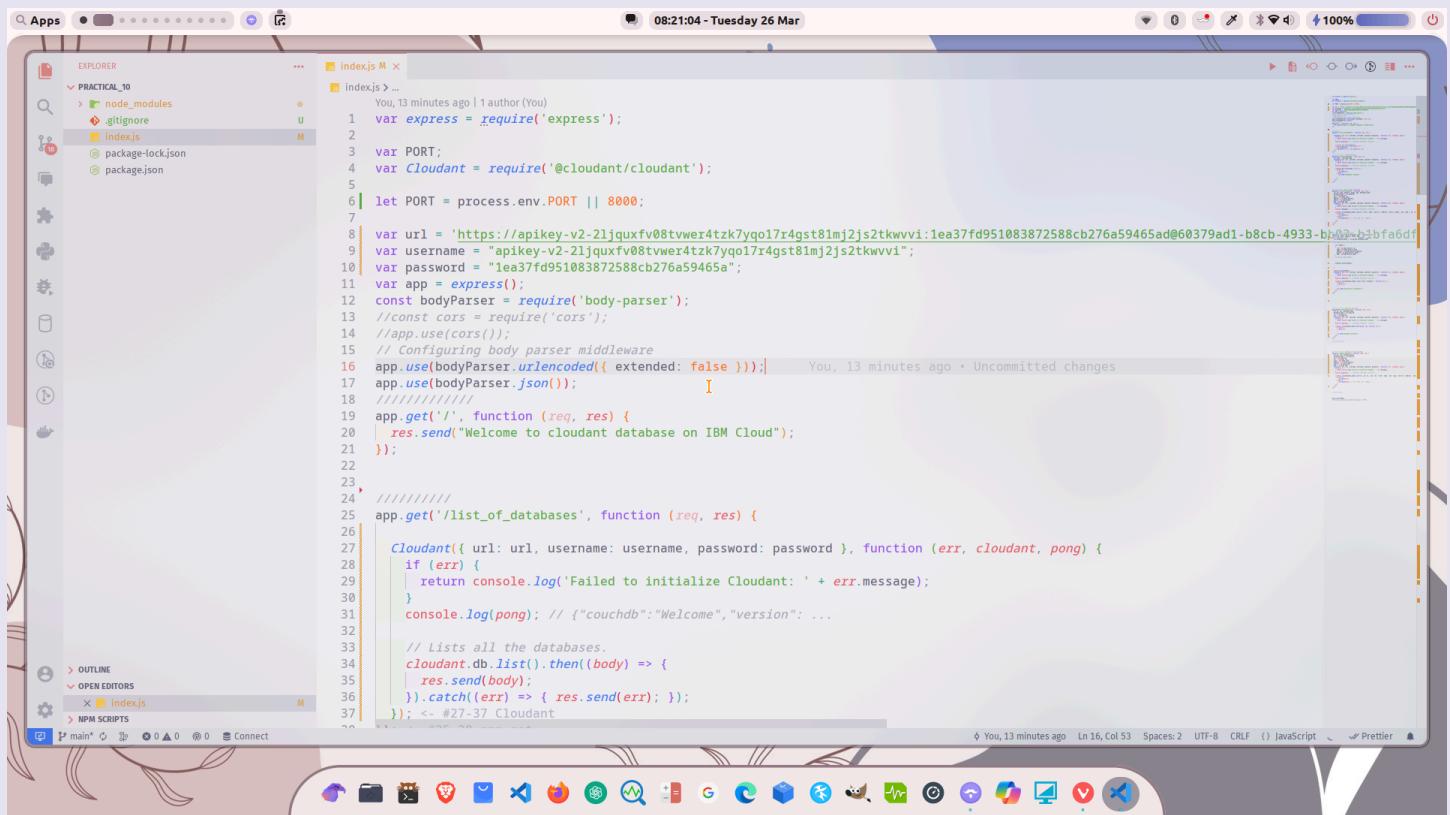


The screenshot shows the IBM Cloud Service Credentials page for the 'Cloudant-ysl' service. The page title is 'Service credentials'. A sub-header states: 'You can generate a new set of credentials for cases where you want to manually connect an app or external consumer to an IBM Cloud service.' Below this, there is a table with two columns: 'Key name' and 'Date created'. A single row is listed: 'Service credentials-1' was created on '2024-03-06 7:52 AM'. At the top right of the table, there is a blue button labeled 'New credential'.

Key name	Date created
Service credentials-1	2024-03-06 7:52 AM

Name - Yash Lakhtariya
Enrollment number - 21162101012
Branch - CBA Batch - 61
EADC Practical 10

2. Copy and paste the credentials in the nodejs code



```
You, 13 minutes ago | 1 author (You)
1 var express = require('express');
2
3 var PORT;
4 var Cloudant = require('@cloudant/cloudant');
5
6 let PORT = process.env.PORT || 8000;
7
8 var url = 'https://apikey-v2-21jquxfv08tvwer4tzh7yqo17r4gst81mj2js2tkwvvi:lea37fd951083872588cb276a59465ad@60379ad1-b8cb-4933-bb03-b1bfa6df8a7e-bluemix.cloudantnosqldb.appdomain.cloud';
9 var username = "apikey-v2-21jquxfv08tvwer4tzh7yqo17r4gst81mj2js2tkwvvi";
10 var password = "lea37fd951083872588cb276a59465ad";
11 var app = express();
12 const bodyParser = require('body-parser');
13 //const cors = require('cors');
14 //app.use(cors());
15 // Configuring body parser middleware
16 app.use(bodyParser.urlencoded({ extended: false }));
17 app.use(bodyParser.json());
18 /////////////
19 app.get('/', function (req, res) {
20   res.send("Welcome to cloudant database on IBM Cloud");
21 });
22
23,
24,
25 app.get('/list_of_databases', function (req, res) {
26
27   Cloudant({ url: url, username: username, password: password }, function (err, cloudant, pong) {
28     if (err) {
29       return console.log('Failed to initialize Cloudant: ' + err.message);
30     }
31     console.log(pong); // {"couchdb": "Welcome", "version": ...
32
33     // Lists all the databases.
34     cloudant.db.list().then(body) => {
35       res.send(body);
36     }).catch(err) => { res.send(err); });
37   }); <- #27-37 Cloudant
You, 13 minutes ago Ln 16, Col 53 Spaces: 2 UTF-8 CRLF () JavaScript ✓ Prettier
```

Code :

```
var express = require('express');

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

var PORT = process.env.PORT || 8000;

var url =
'https://apikey-v2-21jquxfv08tvwer4tzh7yqo17r4gst81mj2js2tkwvvi:lea37fd951083872588cb276a59465ad@60379ad1-b8cb-4933-bb03-b1bfa6df8a7e-bluemix.cloudantnosqldb.appdomain.cloud';

var username = "apikey-v2-21jquxfv08tvwer4tzh7yqo17r4gst81mj2js2tkwvvi";

var password = "lea37fd951083872588cb276a59465ad";
```

Name - Yash Lakhtariya
Enrollment number - 21162101012
Branch - CBA Batch - 61
EADC Practical 10

```
var app = express();
const bodyParser = require('body-parser');
//const cors = require('cors');
//app.use(cors());
// Configuring body parser middleware
app.use(bodyParser.urlencoded({ extended: false }));
app.use(bodyParser.json());
///////////

app.get('/', function (req, res) {
  res.send("Welcome to cloudant database on IBM Cloud");
});

/////////
app.get('/list_of_databases', function (req, res) {
  Cloudant({ url: url, username: username, password: password }, function (err, cloudant, pong) {
    if (err) {
      return console.log('Failed to initialize Cloudant: ' + err.message);
    }
    console.log(pong); // {"couchdb": "Welcome", "version": ...
    // Lists all the databases.
    cloudant.db.list().then((body) => {
      res.send(body);
    }).catch((err) => { res.send(err); });
  });
});

/////////// create database
```

Name - Yash Lakhtariya
Enrollment number - 21162101012
Branch - CBA Batch - 61
EADC Practical 10

```
app.post('/create-database', (req, res) => {
  var name = req.body.name;
  Cloudant({ url: url, username: username, password: password }, function
  (err, cloudant, pong) {
    if (err) {
      return console.log('Failed to initialize Cloudant: ' + err.message);
    }
    console.log(pong); // {"couchdb": "Welcome", "version": ...}

    cloudant.db.create(name, (err) => {
      if (err) {
        res.send(err);
      } else {
        res.send("database created")
      }
    });
  });
}) ;

/////////// insert single document
app.post('/insert-document', function (req, res) {
  var id, name, address, phone, age, database_name;
  database_name = req.body.db;
  id = req.body.id,
  name = req.body.name;
  address = req.body.address;
  phone = req.body.phone;
  age = req.body.age;
  Cloudant({ url: url, username: username, password: password }, function
  (err, cloudant, pong) {
    if (err) {
```

Name - Yash Lakhtariya

Enrollment number - 21162101012

Branch - CBA Batch - 61

EADC Practical 10

```
        return console.log('Failed to initialize Cloudant: ' + err.message);
    }
    console.log(pong); // {"couchdb": "Welcome", "version": ...
}

cloudant.use(database_name).insert({ "name": name, "address": address,
"phone": phone, "age": age }, id, (err, data) => {
    if (err) {
        res.send(err);
    } else {
        res.send(data); // { ok: true, id: 'rabbit', ...
    }
});
});

////// insert bulk documents
app.post("/insert-bulk/:database_name", function (req, res) {
const database_name = req.params.database_name;
const students = [];

for (let i = 0; i < 3; i++) {
    const student = {
        _id: req.body.docs[i].id,
        name: req.body.docs[i].name,
        address: req.body.docs[i].address,
        phone: req.body.docs[i].phone,
        age: req.body.docs[i].age,
    };

    students.push(student);
}

students.push(student);
}
}
```

Name - Yash Lakhtariya

Enrollment number - 21162101012

Branch - CBA Batch - 61

EADC Practical 10

```
Cloudant({ url: url, username: username, password: password }, function (err, cloudant, pong) {
  if (err) {
    return console.log("Failed to initialize Cloudant: " + err.message);
  }

  cloudant.use(database_name).bulk({ docs: students }, function (err) {
    if (err) {
      throw err;
    }

    res.send("Inserted all documents");
  });
});

/////////////// delete a document
app.delete('/delete-document', function (req, res) {
  var id, rev, database_name;
  database_name = req.body.db;
  id = req.body.id;
  rev = req.body.rev;
  Cloudant({ url: url, username: username, password: password }, function (err, cloudant, pong) {
    if (err) {
      return console.log('Failed to initialize Cloudant: ' + err.message);
    }
    console.log(pong); // {"couchdb": "Welcome", "version": ...

    cloudant.use(database_name).destroy(id, rev, function (err) {
      if (err) {
        throw err;
      }
    });
  });
});
```

Name - Yash Lakhtariya
Enrollment number - 21162101012
Branch - CBA Batch - 61
EADC Practical 10

```
}

    res.send('document deleted');

}) ;

}) ;

}) ;

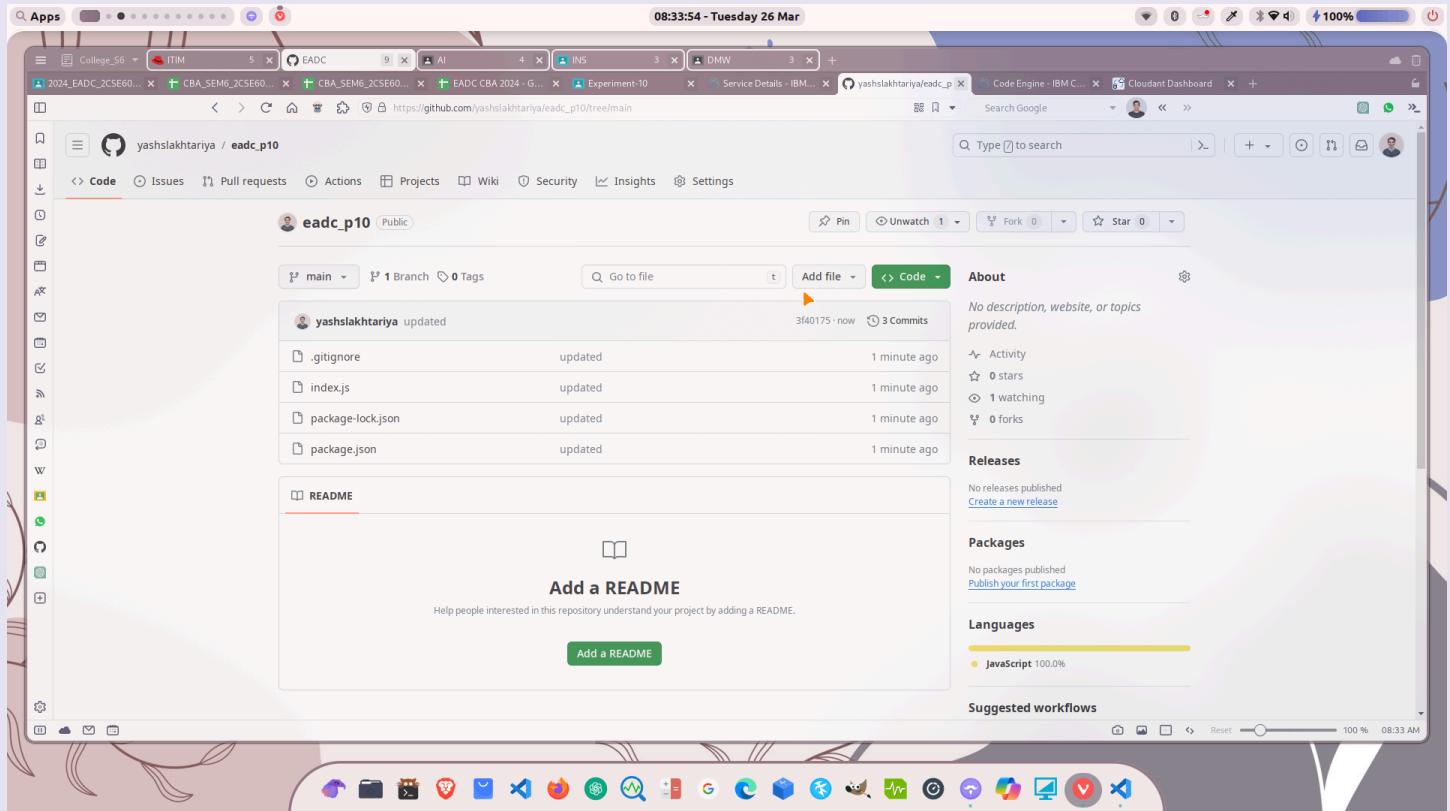
/////////////////// update existing document
app.put('/update-document', function (req, res) {
  var id, rev, database_name;
  database_name = req.body.db;
  id = req.body.id;
  rev = req.body.rev;
  name = req.body.name;
  address = req.body.address;
  phone = req.body.phone;
  age = req.body.age;
  Cloudant({ url: url, username: username, password: password }, function
  (err, cloudant, pong) {
    if (err) {
      return console.log('Failed to initialize Cloudant: ' + err.message);
    }
    console.log(pong); // {"couchdb": "Welcome", "version": ...

    cloudant.use(database_name).insert({ _id: id, _rev: rev, "name": name,
  "age": age, "address": address, "phone": phone }, (err, data) => {
      if (err) {
        res.send(err);
      } else {
        res.send(data); // { ok: true, id: 'rabbit', ...
      }
    });
  });
}) ;
```

Name - Yash Lakhtariya
Enrollment number - 21162101012
Branch - CBA Batch - 61
EADC Practical 10

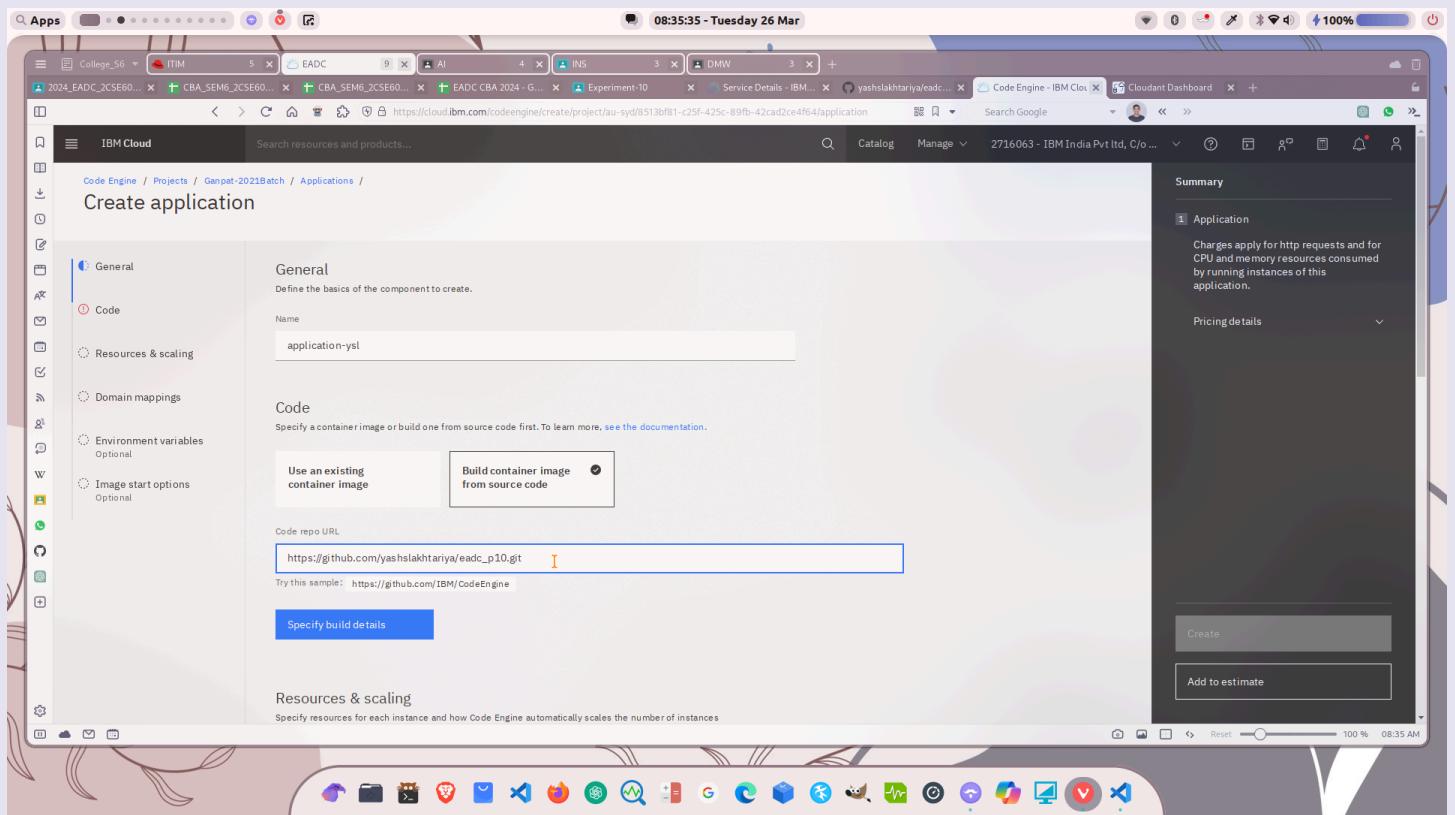
```
});
```

3. Create git repository of the application



Name - Yash Lakhtariya
Enrollment number - 21162101012
Branch - CBA Batch - 61
EADC Practical 10

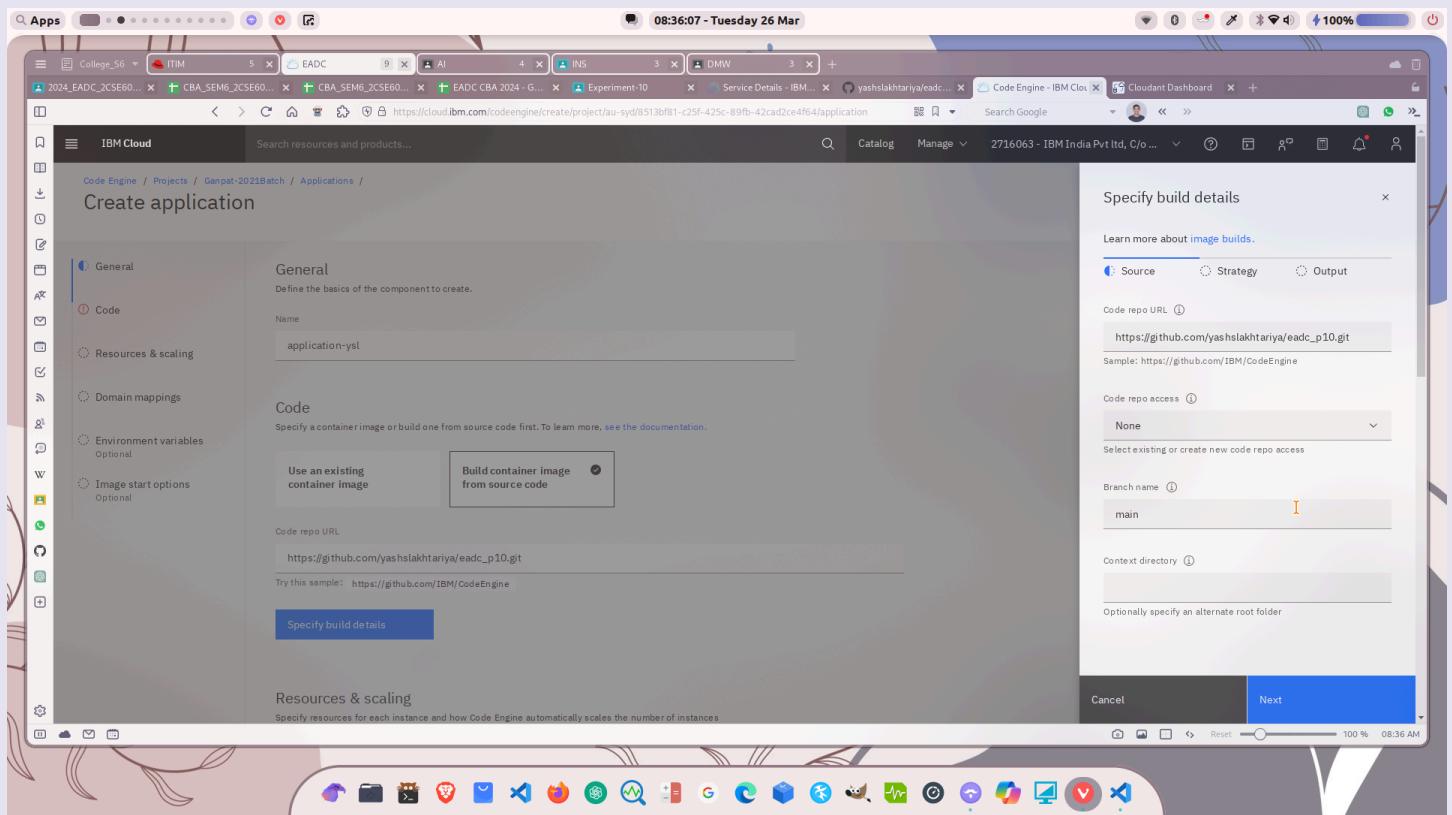
4. Create new application in the existing (or new) project on Code Engine



The screenshot shows the 'Create application' page in the IBM Cloud interface. On the left, a sidebar lists options like General, Code, Resources & scaling, Domain mappings, Environment variables, and Image start options. The 'General' section is active, showing a 'Name' field with 'application-ysl'. Below it is a 'Code' section with two options: 'Use an existing container image' (selected) and 'Build container image from source code'. A 'Code repo URL' input field contains the value 'https://github.com/yashlakhtariya/eadc_p10.git'. A 'Specify build details' button is visible below the code section. To the right, a 'Summary' panel displays information about the application, including its name and a note about charges for CPU and memory resources. At the bottom right of the summary panel are 'Create' and 'Add to estimate' buttons.

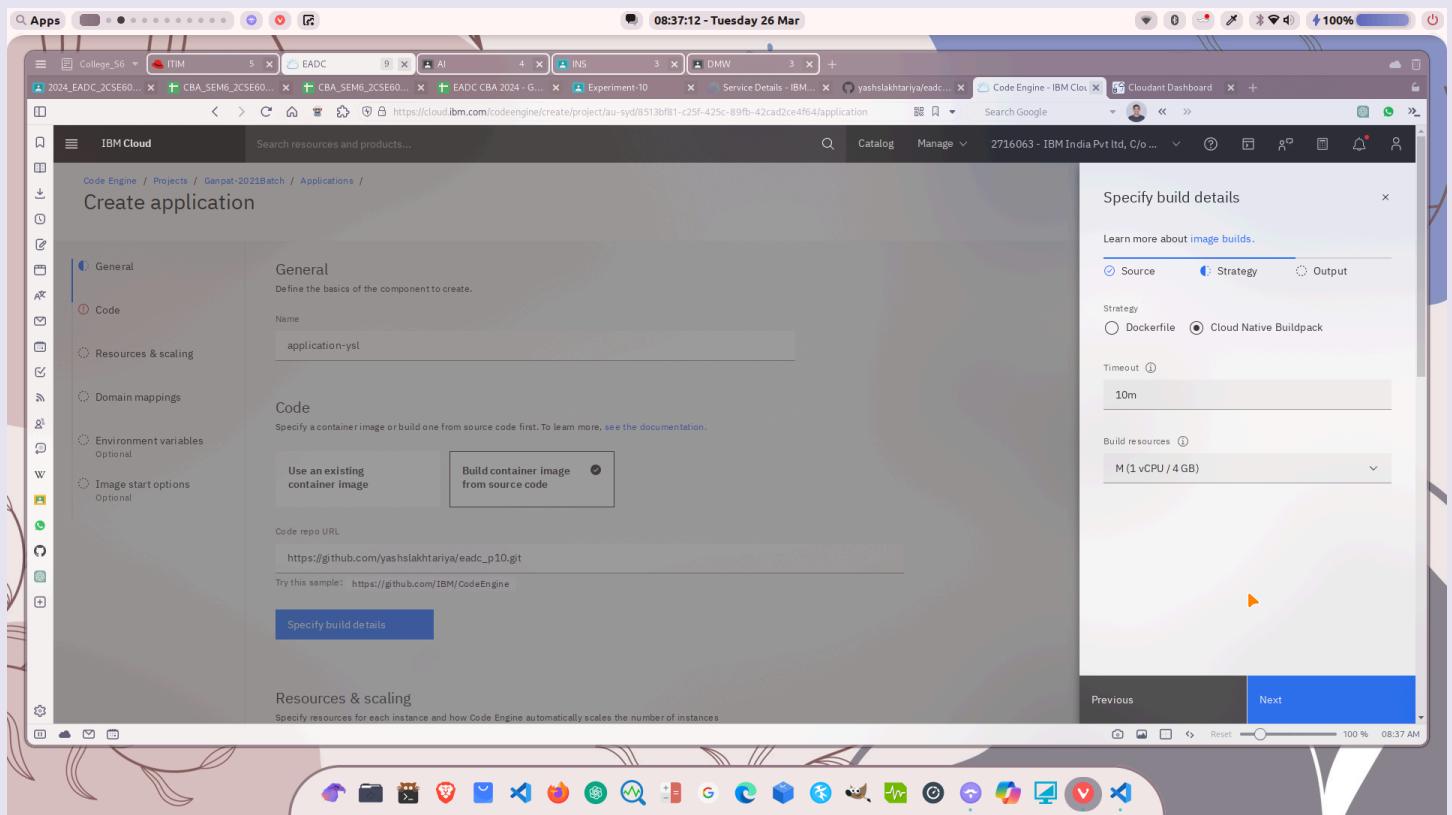
Name - Yash Lakhtariya
Enrollment number - 21162101012
Branch - CBA Batch - 61
EADC Practical 10

5. Specify build details, source as git repo and main branch



Name - Yash Lakhtariya
Enrollment number - 21162101012
Branch - CBA Batch - 61
EADC Practical 10

6. Use Cloud Native Buildpack as strategy as we don't have dockerfile



The screenshot shows the IBM Cloud interface for creating a new application. On the left, a sidebar lists options like General, Code, Resources & scaling, Domain mappings, Environment variables, and Image start options. The main panel is titled 'Create application' and has a 'General' section where the name 'application-ysl' is specified. Below it is a 'Code' section with two options: 'Use an existing container image' (disabled) and 'Build container image from source code' (selected). A 'Code repo URL' field contains the URL 'https://github.com/yashlakhtariya/eadc_p10.git'. To the right, a 'Specify build details' panel is open, showing the 'Strategy' tab selected (Cloud Native Buildpack). Other tabs include 'Source' and 'Output'. It also shows a 'Timeout' of '10m' and a 'Build resources' of 'M (1 vCPU / 4 GB)'. At the bottom, there are 'Previous' and 'Next' buttons.

Name - Yash Lakhtariya
Enrollment number - 21162101012
Branch - CBA Batch - 61
EADC Practical 10

7. Create new access token if not existing in docker hub

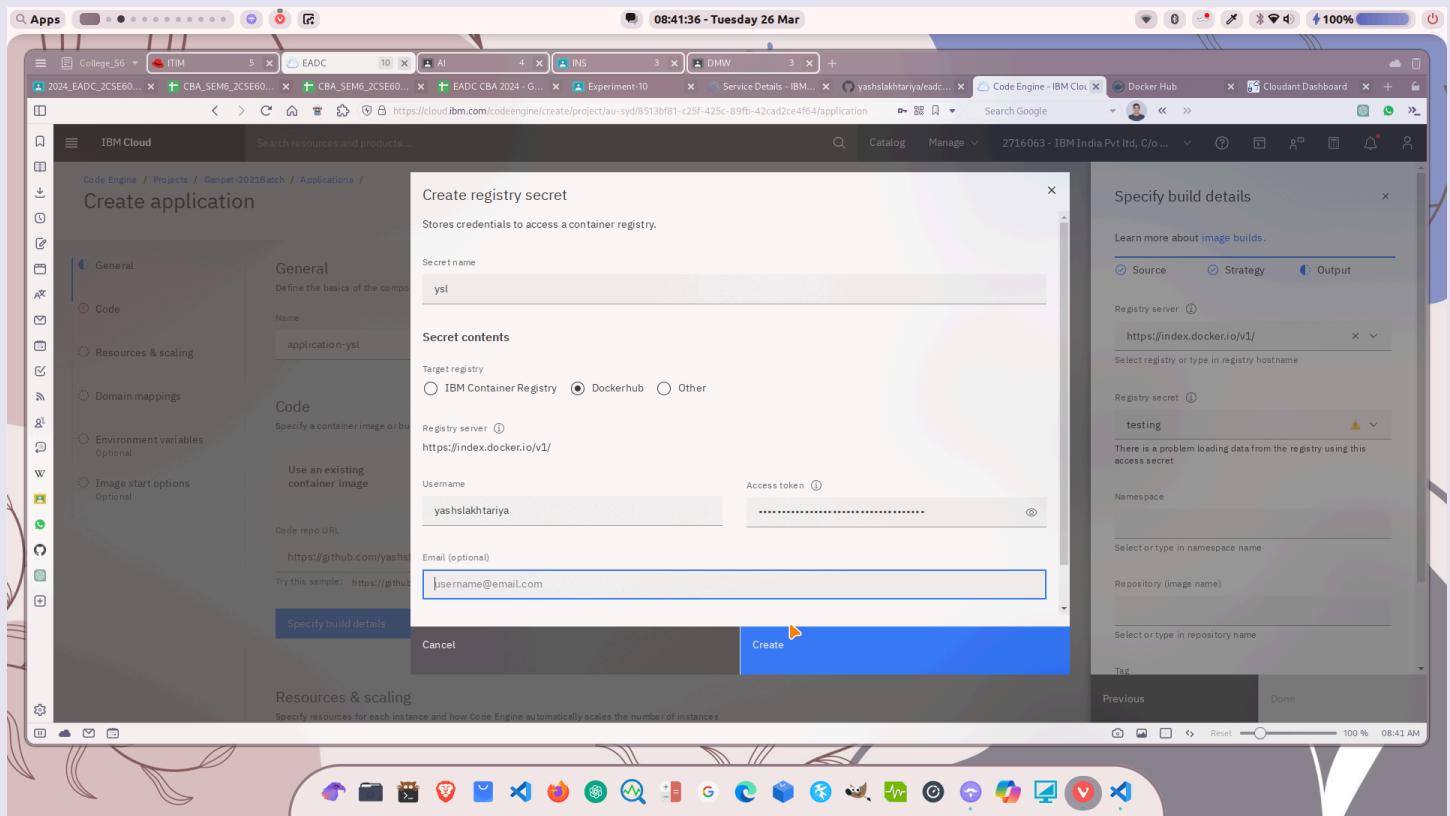
The screenshot shows the Docker Hub security settings page for the user 'yashlakhtariya'. On the left, a sidebar lists account settings like General, Security (selected), Default Privacy, Notifications, Convert Account, and Deactivate Account. The main area is titled 'Access Tokens' and displays a table of existing tokens. A blue button at the top right says 'New Access Token'. The table columns are Description, Source, Scope, Last Used, and Created. The tokens listed are all auto-generated by Docker Desktop and have scopes 'Read, Write, Delete'. The last token was created on October 11, 2023, at 10:36:27.

Description	Source	Scope	Last Used	Created
Generated by Docker De...	AUTO-GENERATED	Read, Write, Delete	Oct 01, 2023 20:31:13	Sep 29, 2023 08:42:58
Generated by Docker De...	AUTO-GENERATED	Read, Write, Delete	Sep 27, 2023 08:12:48	Sep 26, 2023 22:11:28
Generated by Docker De...	AUTO-GENERATED	Read, Write, Delete	Never	Oct 18, 2023 11:49:40
Generated by Docker De...	AUTO-GENERATED	Read, Write, Delete	Never	Oct 11, 2023 10:36:27

The screenshot shows the Docker Hub security settings page with a modal window open for generating a new access token. The modal has fields for 'ACCESS TOKEN DESCRIPTION' (set to 'ysl') and 'ACCESS PERMISSIONS' (set to 'Read, Write, Delete'). It contains instructions: 'To use the access token from your Docker CLI client:' followed by two steps: '1. Run docker login -u yashlakhtariya' and '2. At the password prompt, enter the personal access token.' A warning message states: 'WARNING: This access token will only be displayed once. It will not be stored and cannot be retrieved. Please be sure to save it now.' At the bottom of the modal are 'Copy' and 'Close' buttons. In the background, the main security page shows a table of existing access tokens, with a blue 'New Access Token' button visible at the top right.

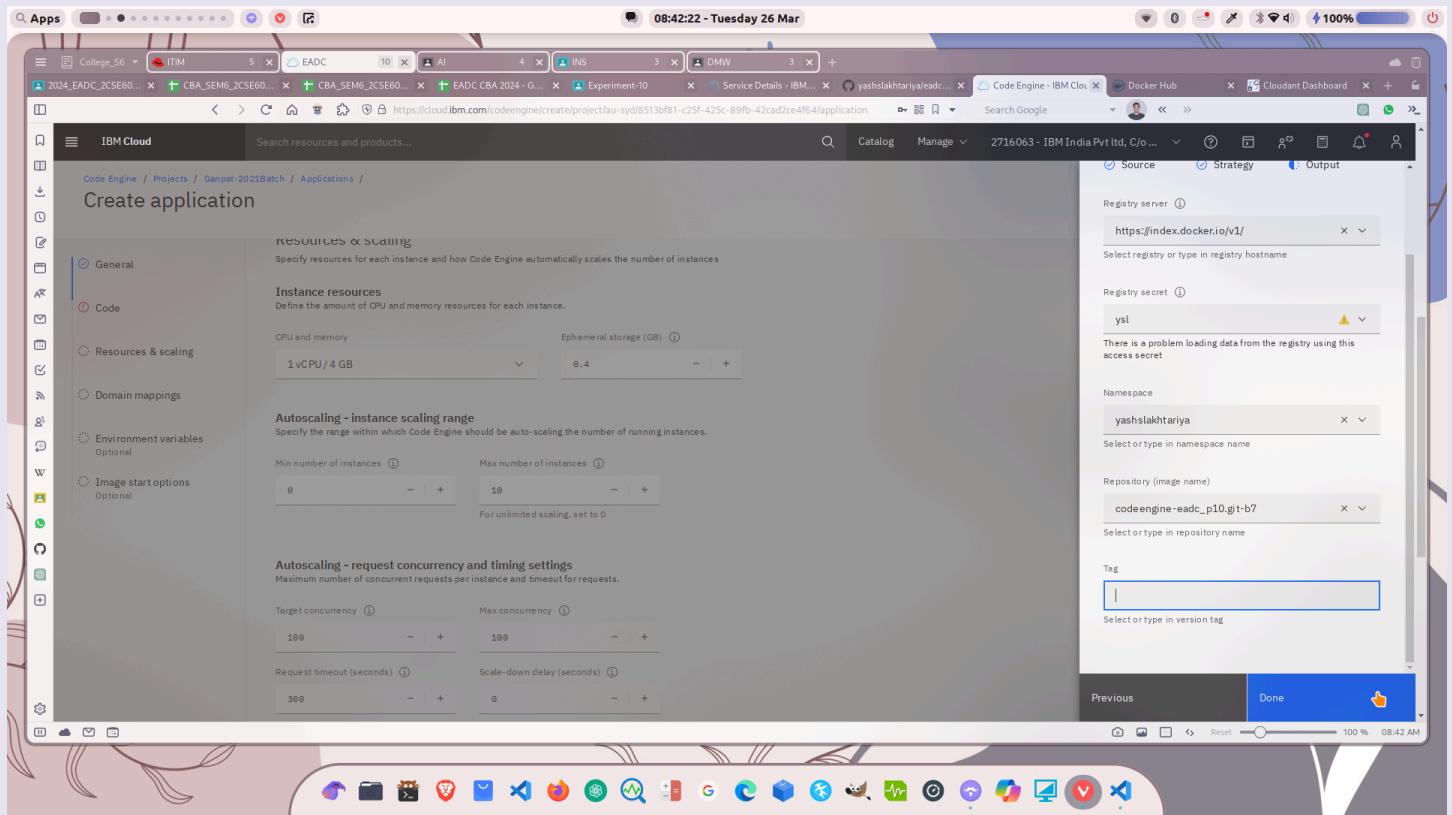
Name - Yash Lakhtariya
Enrollment number - 21162101012
Branch - CBA Batch - 61
EADC Practical 10

8. Create new registry secret if not existing specifying the username and access token



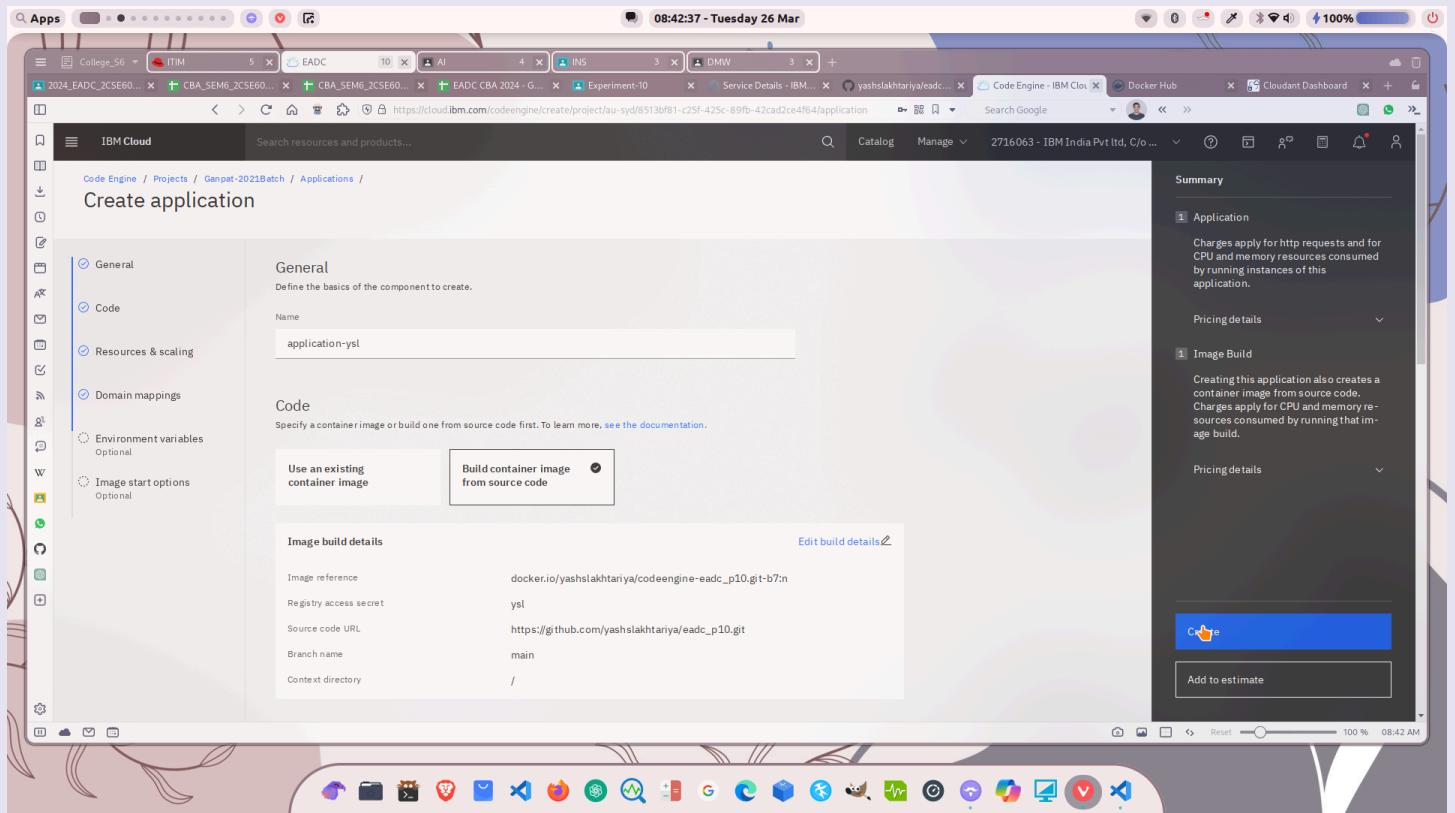
Name - Yash Lakhtariya
Enrollment number - 21162101012
Branch - CBA Batch - 61
EADC Practical 10

9. Use the registry secret in output tab



Name - Yash Lakhtariya
Enrollment number - 21162101012
Branch - CBA Batch - 61
EADC Practical 10

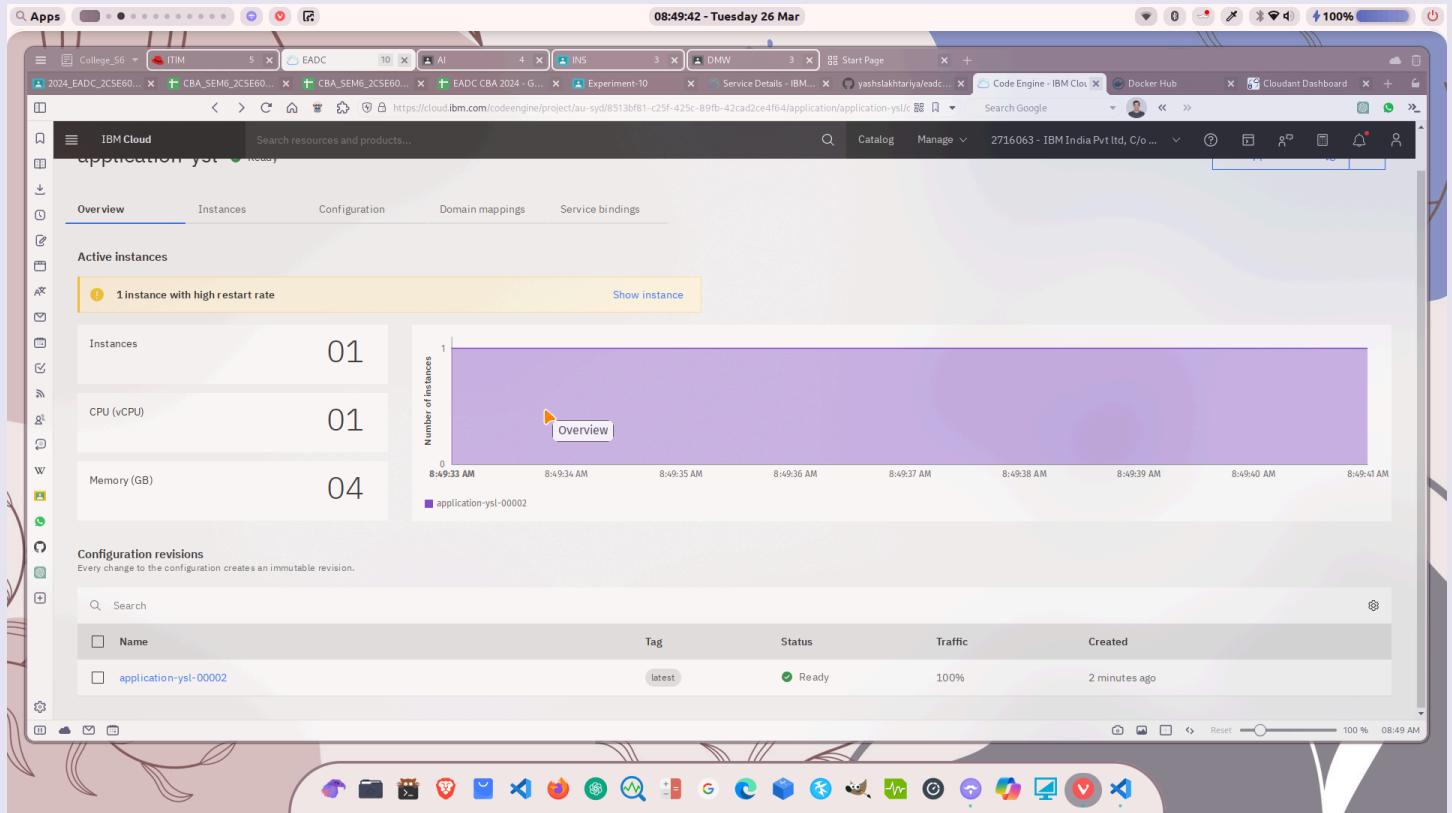
10. Create the application



The screenshot shows the IBM Cloud interface for creating a new application. The left sidebar lists categories: General, Code, Resources & scaling, Domain mappings, Environment variables (Optional), and Image start options (Optional). The 'Code' section is selected. The main panel has a 'General' tab where the name 'application-ysl' is entered. Below it is a 'Code' section with two options: 'Use an existing container image' and 'Build container image from source code', with the latter being selected. Under 'Image build details', the image reference is set to 'docker.io/yashlakhtariya/codeengine-eadc_p10.git-b7n', the registry access secret is 'ysl', the source code URL is 'https://github.com/yashlakhtariya/eadc_p10.git', the branch name is 'main', and the context directory is '/'. On the right side, there are sections for 'Application' (Summary) and 'Image Build' (Summary), both with detailed descriptions and pricing information. A large blue 'Create' button is prominently displayed at the bottom right.

Name - Yash Lakhtariya
Enrollment number - 21162101012
Branch - CBA Batch - 61
EADC Practical 10

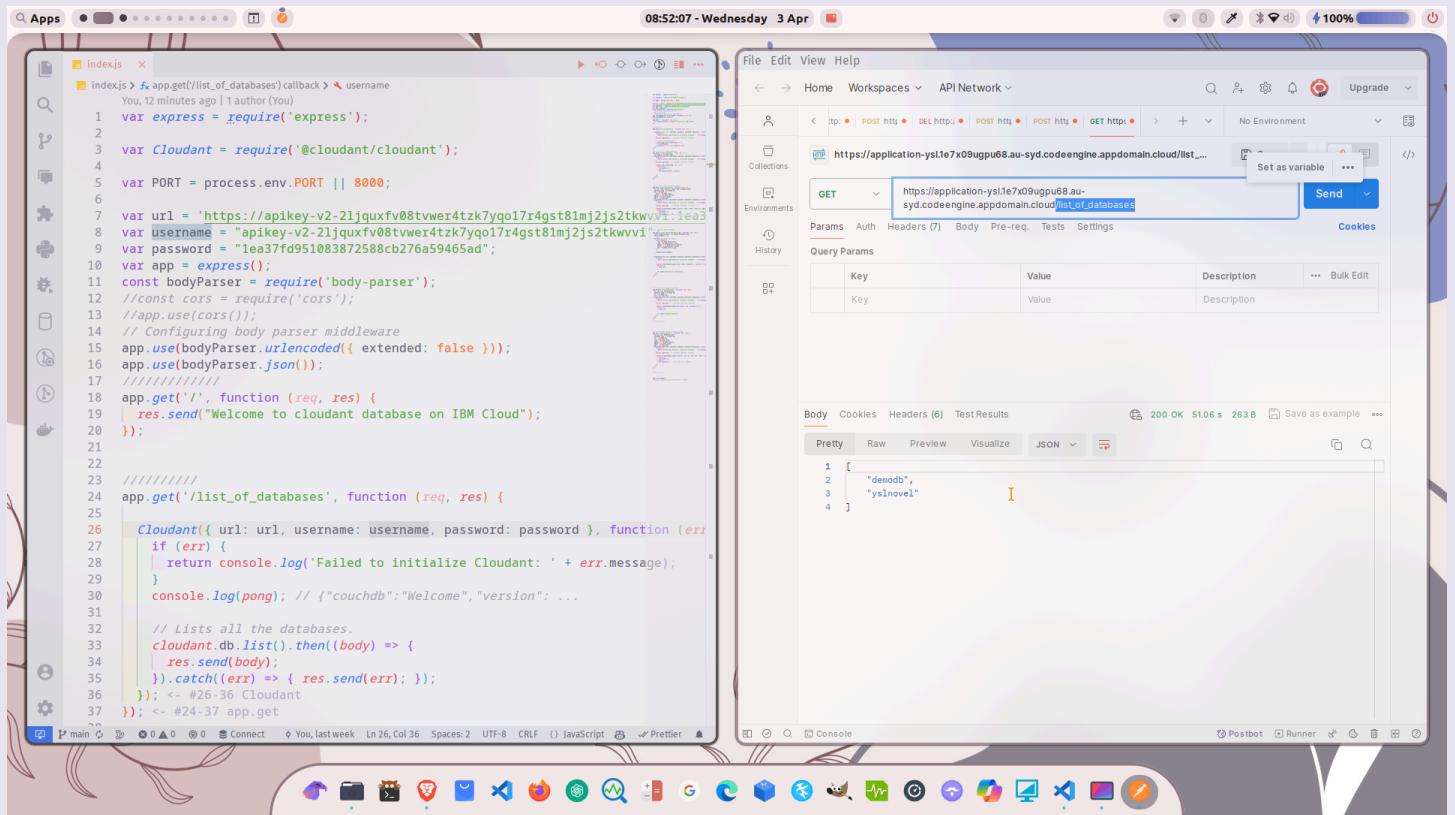
11. Wait for instance to be ready



The screenshot shows the IBM Cloud Code Engine Overview page. At the top, there is a navigation bar with tabs for Overview, Instances, Configuration, Domain mappings, and Service bindings. The Overview tab is selected. Below the navigation bar, there is a section titled "Active instances" which displays a message: "1 instance with high restart rate" and a "Show instance" button. On the left, there is a sidebar with icons for different services like Cloud Functions, Cloud Run, Cloud Build, and Cloud Deployment. The main content area shows resource usage statistics: Instances (01), CPU (vCPU) (01), and Memory (GB) (04). To the right, there is a chart titled "Number of instances" showing a single purple bar at level 1 from 8:49:33 AM to 8:49:41 AM. A legend indicates the bar represents "application-ysl-00002". Below the chart, there is a section titled "Configuration revisions" with a table showing a single revision named "application-ysl-00002". The revision details are: Tag (latest), Status (Ready), Traffic (100%), and Created (2 minutes ago). The bottom of the screen shows a Mac OS X dock with various application icons.

Name - Yash Lakhtariya
Enrollment number - 21162101012
Branch - CBA Batch - 61
EADC Practical 10

12. Now test the application using POSTMAN requests



Request : GET
URL : {appdomain}/list_databases

For listing all databases

**Name - Yash Lakhtariya
Enrollment number - 21162101012
Branch - CBA Batch - 61
EADC Practical 10**

13. For creating the new database on Cloudant

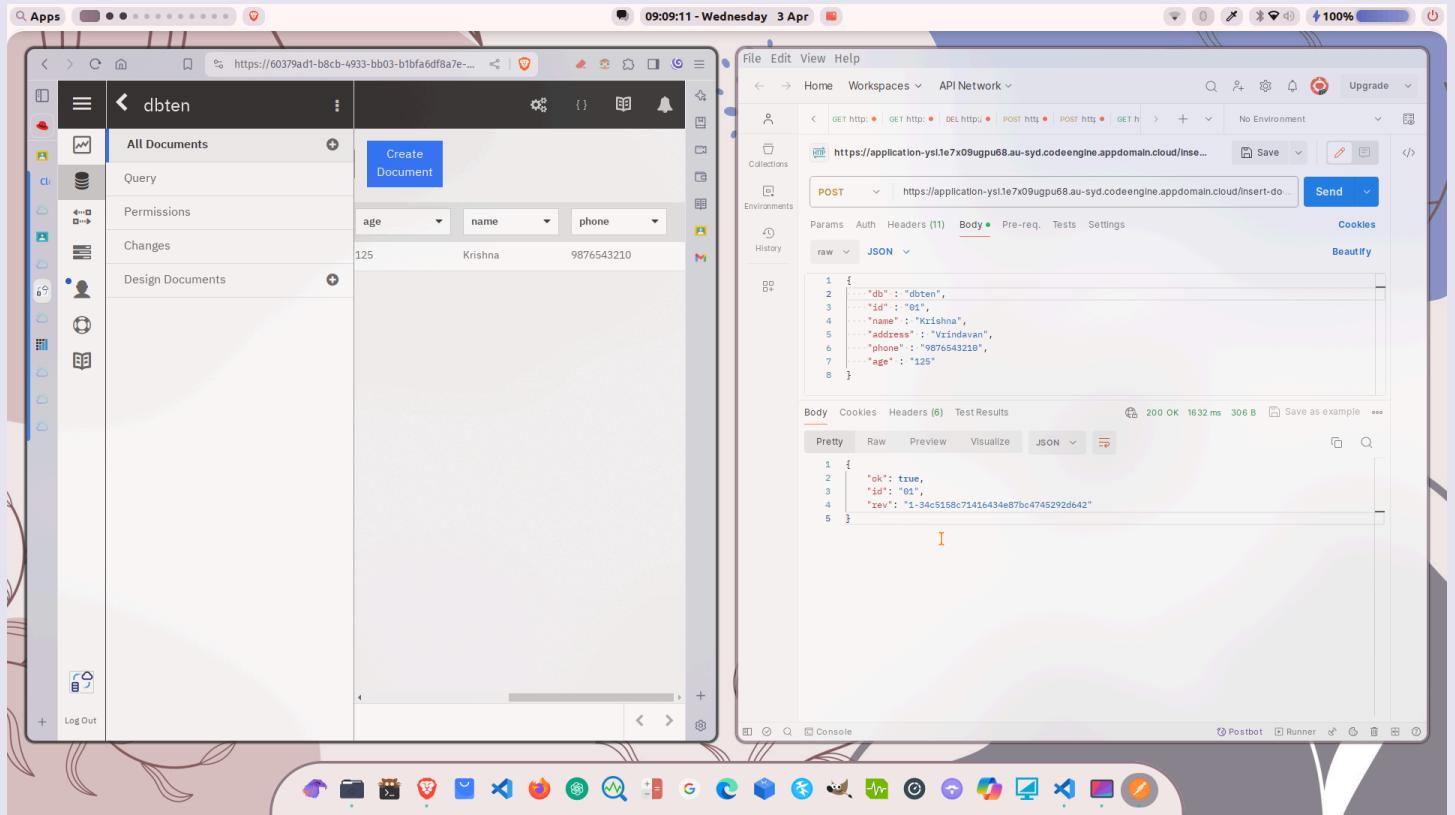
The screenshot shows a developer's environment with two main windows. On the left is a code editor displaying index.js, which contains logic for handling database requests using Cloudant. The code includes functions for listing databases and creating new ones. On the right is a browser-based tool for testing API endpoints, showing a POST request to https://application-ysl.1e7x09ugpu68.au-syd.codeengine.appdomain.cloud/create-db. The response is a JSON object indicating a successful creation of a database named 'dbten'. The bottom of the screen features a toolbar with various icons for file operations, navigation, and system controls.

```
index.js
index.js > fx app.get('/list_of_databases') callback > username
20  );
21
22
23 ///////////////
24 app.get('/list_of_databases', function (req, res) {
25
26   Cloudant({ url: url, username: username, password: password }, function (err, client) {
27     if (err) {
28       return console.log('Failed to initialize Cloudant: ' + err.message);
29     }
30     console.log(pong); // {"couchdb": "Welcome", "version": ...}
31
32     // Lists all the databases.
33     cloudant.db.list().then((body) => {
34       res.send(body);
35     }).catch((err) => { res.send(err); });
36   });
37 }); <- #26-36 Cloudant
38 ); <- #24-37 app.get
39
40 ///////////////// create database
41 app.post('/create-database', (req, res) => {
42   var name = req.body.name;
43   Cloudant({ url: url, username: username, password: password }, function (err, client) {
44     if (err) {
45       return console.log('Failed to initialize Cloudant: ' + err.message);
46     }
47     console.log(pong); // {"couchdb": "Welcome", "version": ...}
48
49     cloudant.db.create(name, (err) => {
50       if (err) {
51         res.send(err);
52       } else {
53         res.send("database created")
54       }
55     });
56   });
57 }); <- #48-55 cloudant.db.create
58 ); <- #42-56 Cloudant
59 )); <- #40-57 app.post
```

Request : POST
URL : {appdomain}/create-database

Name - Yash Lakhtariya
Enrollment number - 21162101012
Branch - CBA Batch - 61
EADC Practical 10

14. For adding the single document to the database



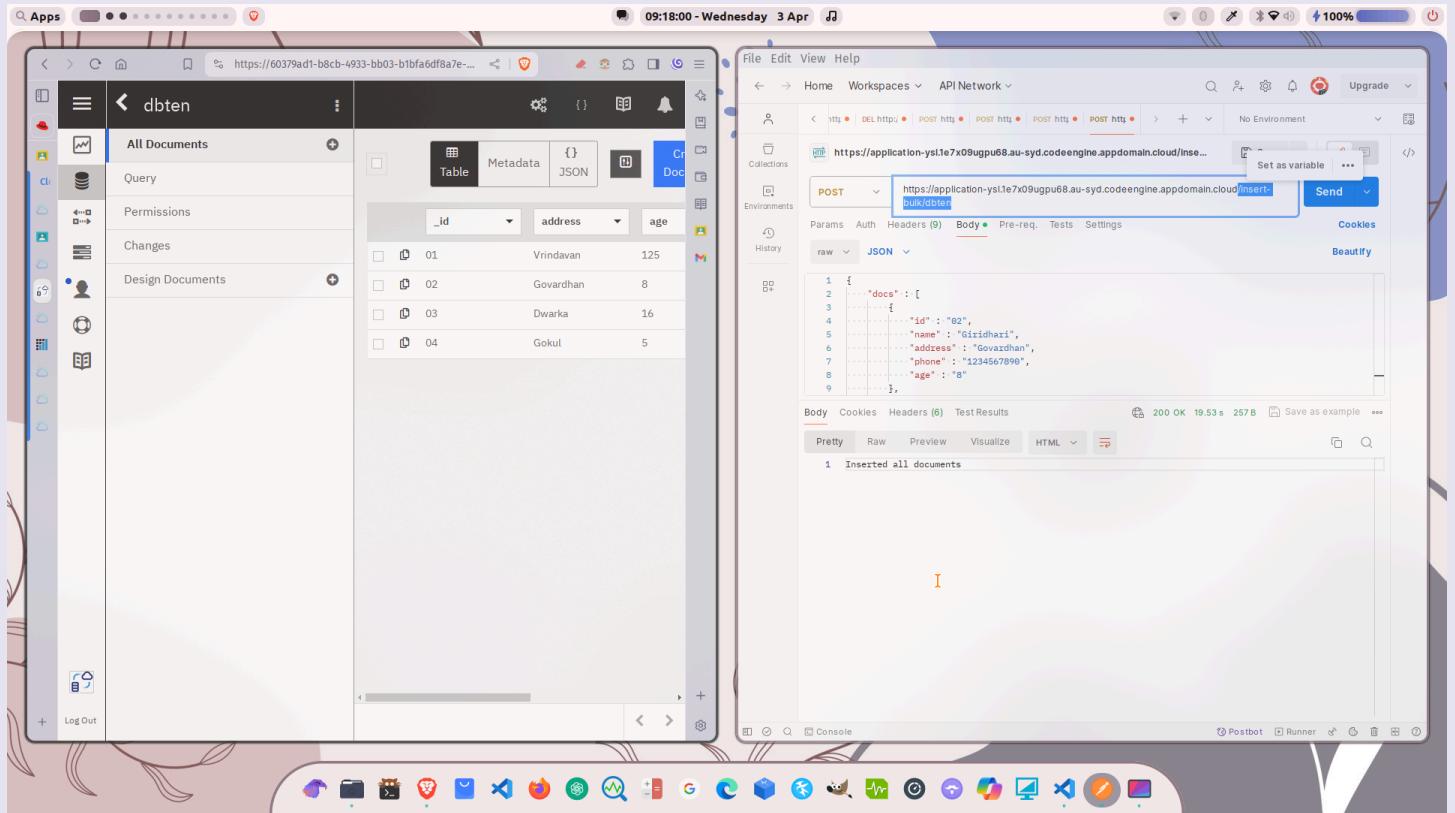
Request : POST

URL : {appdomain}/insert-document

Body : required fields in json format

Name - Yash Lakhtariya
Enrollment number - 21162101012
Branch - CBA Batch - 61
EADC Practical 10

15. For adding bulk documents, like say 3 in this case



The screenshot displays two windows side-by-side. On the left is a MongoDB Compass interface showing a database named 'dbten' with a collection named 'All Documents'. It lists four documents with IDs 01, 02, 03, and 04, each having fields 'address' and 'age'. On the right is a Postman application window showing a POST request to 'https://application-ysl1e7x09ugpu68.au-syd.codeengine.appdomain.cloud/_insert-bulk/docs'. The request body is a JSON object with a 'docs' key containing an array of three documents. The response shows a status of 200 OK with a message 'Inserted all documents'.

_id	address	age
01	Vrindavan	125
02	Gowardhan	8
03	Dwarka	16
04	Gokul	5

```
1 {  
2   "docs": [  
3     {  
4       "_id": "02",  
5       "name": "Gizidhaz1",  
6       "address": "Govardhan",  
7       "phone": "1234567890",  
8       "age": 8  
9     },  
10    {  
11      "_id": "03",  
12      "name": "Gizidhaz2",  
13      "address": "Vrindavan",  
14      "phone": "9876543210",  
15      "age": 16  
16    }  
17  ]  
18}
```

Request : POST

URL : {appdomain}/insert-bulk/:database-name

Param : database name at end of URL

Body : array of json object of required fields as value of "docs" key in root json object

Name - Yash Lakhtariya
Enrollment number - 21162101012
Branch - CBA Batch - 61
EADC Practical 10

16. For updating the data of a single document

The screenshot displays two windows side-by-side. On the left is the MongoDB Compass interface, showing a database named 'dbten' with a collection named 'All Documents'. It lists three documents with fields: age, name, and phone. The third document has an '_id' of '5', a 'name' of 'Gopal', a 'phone' of '8888999916', and an 'age' of '5'. On the right is a Postman API request window. The URL is 'https://application-ysi.te7x09ugpu68.au-syd.codeengine.appdomain.cloud/update-document'. The method is 'PUT'. The body is in JSON format, showing the document update. The response status is '200 OK' with a duration of '19.88 s' and a size of '306 B'. The response body is: { "ok": true, "id": "04", "rev": "2-856c6f4542f440d642acb78586f4e253" }.

Request : PUT
URL : {appdomain}/update-document
Body : required fields in json format

Name - Yash Lakhtariya
Enrollment number - 21162101012
Branch - CBA Batch - 61
EADC Practical 10

17. For deleting the document from database

The screenshot displays two windows side-by-side. On the left is a browser window showing the Apache CouchDB Futon interface for the 'dbten' database. It lists three documents with IDs 01, 02, and 03, addresses Vrindavan, Govardhan, and Dwarka, and ages 125, 8, and 16 respectively. On the right is a Postman application window showing a DELETE request to the same database. The JSON body of the request is:

```
1 {  
2   "db": "dbten",  
3   "id": "04",  
4   "rev": "2-056ccf4542f440d642acb78586f4e253"  
5 }
```

The response status is 200 OK, and the message is "document deleted".

Request : DELETE
URL : {appdomain}/delete-document
Body : database-name, id and rev fields in json object

Name - Yash Lakhtariya
Enrollment number - 21162101012
Branch - CBA Batch - 61
EADC Practical 10

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.

Additional code in previous index.js code :

```
app.use(express.static('public'));
app.use(express.urlencoded({ extended: true }));

///////////
app.post('/html-insert', function (req, res) {
  var name = req.body.name;
  var phone = req.body.phone;
  var email = req.body.email;
  var city = req.body.city;
  var country = req.body.country;
  var pincode = req.body.pincode;
  var database_name = 'dbten';

  Cloudant({ url: url, username: username, password: password }, function
(err, cloudant, pong) {
    if (err) {
      return console.log('Failed to initialize Cloudant: ' + err.message);
    }
    console.log(pong);

    cloudant.use(database_name).insert({ "name": name, "phone": phone,
"email": email, "city": city, "country": country, "pincode": pincode },
(err, data) => {
      if (err) {
        res.send(err);
      } else {
```

Name - Yash Lakhtariya
Enrollment number - 21162101012
Branch - CBA Batch - 61
EADC Practical 10

```
    res.send(data);  
  }  
});  
});  
});
```

HTML Code :

```
<!DOCTYPE html>  
<html>  
  
<head>  
  <style>  
    body {  
      font-family: 'Roboto', sans-serif;  
      padding: 20px;  
      background-color: #F5F5F5;  
    }  
  
    form {  
      display: flex;  
      flex-direction: column;  
      width: 300px;  
      margin: auto;  
    }  
  
    label {  
      margin: 10px 0 5px 0;  
      color: #b75969;  
    }  
  
    input[type="text"] {
```

Name - Yash Lakhtariya
Enrollment number - 21162101012
Branch - CBA Batch - 61
EADC Practical 10

```
padding: 10px;
border: none;
border-radius: 4px;
box-shadow: 0 2px 2px 0 rgba(0, 0, 0, 0.14), 0 1px 5px 0
rgba(0, 0, 0, 0.12), 0 3px 1px -2px rgba(0, 0, 0, 0.2);
}

input[type="submit"] {
margin-top: 20px;
padding: 10px;
border: none;
color: white;
background-color: #b75969;
border-radius: 4px;
cursor: pointer;
transition: background-color 0.3s ease;
}

input[type="submit"]:hover {
background-color: #8b3e4a;
}

</style>
</head>

<body>
<!DOCTYPE html>
<html>

<head>
<style>
body {
font-family: 'Roboto', sans-serif;
padding: 20px;
```

Name - Yash Lakhtariya
Enrollment number - 21162101012
Branch - CBA Batch - 61
EADC Practical 10

```
background-color: #F5F5F5;  
}  
  
form {  
    display: flex;  
    flex-direction: column;  
    width: 300px;  
    margin: auto;  
}  
  
label {  
    margin: 10px 0 5px 0;  
    color: #b75969;  
}  
  
input[type="text"] {  
    padding: 10px;  
    border: none;  
    border-radius: 4px;  
    box-shadow: 0 2px 2px 0 rgba(0, 0, 0, 0.14), 0 1px 5px 0  
    rgba(0, 0, 0, 0.12), 0 3px 1px -2px rgba(0, 0, 0, 0.2);  
}  
  
input[type="submit"] {  
    margin-top: 20px;  
    padding: 10px;  
    border: none;  
    color: white;  
    background-color: #b75969;  
    border-radius: 4px;  
    cursor: pointer;  
    transition: background-color 0.3s ease;  
}
```

Name - Yash Lakhtariya
Enrollment number - 21162101012
Branch - CBA Batch - 61
EADC Practical 10

```
        input[type="submit"]:hover {
            background-color: #8b3e4a;
        }
    </style>
</head>

<body>
    <form action="/html-insert" method="post">
        <label for="name">Name:</label>
        <input type="text" id="name" name="name">
        <label for="phone">Phone:</label>
        <input type="text" id="phone" name="phone">
        <label for="email">Email:</label>
        <input type="text" id="email" name="email">
        <label for="city">City:</label>
        <input type="text" id="city" name="city">
        <label for="country">Country:</label>
        <input type="text" id="country" name="country">
        <label for="pincode">Pincode:</label>
        <input type="text" id="pincode" name="pincode">
        <input type="submit" value="Submit">
    </form>
</body>

</html>
</body>

</html>
```

Name - Yash Lakhtariya

Enrollment number - 21162101012

Branch - CBA Batch - 61

EADC Practical 10

- After updating the code and pushing on Github, rerun build in Configuration of the Code Engine application and visit the URL

The screenshot displays a macOS desktop environment with two browser windows open. The left window contains a form for submitting personal information, including Name, Phone, Email, City, Country, and Pincode, with a 'Submit' button. The right window shows a list of databases with their respective sizes, document counts, and partitioning status. The system tray at the bottom features a variety of application icons.

Name	Size	# of Docs	Partitioned	Actions
dbten	221 bytes	3	No	[Edit] [Lock] [Delete]
demodb	23 bytes	1	No	[Edit] [Lock] [Delete]
yslnovel	386 bytes	4	No	[Edit] [Lock] [Delete]

Name - Yash Lakhtariya

Enrollment number - 21162101012

Branch - CBA Batch - 61

EADC Practical 10

- As seen the data is added (as ID isn't specified, it creates automatically)

Pretty-print

```
[{"ok":true,"id":"42118ad9ed1856891539b86f5410eefa","rev":"1-a7f970e2c620dbe2a78bdff58d7a84c3"}]
```

dbten

All Documents

Query

Permissions

Changes

Design Documents

Log Out

_id	name	phon
01	Krishna	9876543210
02	Girdhari	1234567890
03	Ranchod	1478523109
42118ad9ed1856891539b86f5410eefa	Yash	6351543210