

**NAME: KSHITIJ GUPTA**  
**Enrolment Number: 21162101007**  
**Sub: CS**

**Practical – 3[Batch-71]**

In the first code we are trying to access the cloudant service using the service credentials that are created when we create the cloudant service.

```
<!DOCTYPE html>

<html lang="en">

<head>

    <meta charset="UTF-8">

    <meta name="viewport" content="width=device-width, initial-scale=1.0">

    <title>Cloudant CORS Example</title>

</head>

<body>

    <h1>Cloudant CORS Example</h1>

    <button id="fetchDatabases">Fetch Databases</button>

    <pre id="result"></pre>

    <script>

        document.getElementById('fetchDatabases').addEventListener('click',
        function() {

            const url = 'https://3447fb1b-02ae-4331-923a-607d107471ea-
            bluemix.cloudantnosqldb.appdomain.cloud/_all_dbs';

            const username = 'apikey-v2-
            197dnkn3t48agl1wuzpj91l7lo4dkifrzhim8wjf5ykg';

            const password = '0f75c4be5fda84f99a0d4c582ef21b89';

            const headers = new Headers();

            headers.append('Authorization', 'Basic ' + btoa(username + ':' + password));
        });
    </script>
</body>

```

```
fetch(url, { method: 'GET', headers: headers })

.then(response => response.json())

.then(data => {

    document.getElementById('result').textContent = JSON.stringify(data,
null, 2);

})

.catch(error => {

    document.getElementById('result').textContent = 'Fetch error: ' +
error.message;

});

</script>

</body>

</html>
```

In this script we can see that once the ‘Fetch Databases’ button is clicked it triggers a function, this function makes ‘GET’ request the Cloudant URL, but before that we have Authorization in which username and password are sent with encoding.

The url, username and password in the code are taken from the Service Credentials in cloudant.

The screenshot shows the IBM Cloud Service Details dashboard for a service named 'Cloudant-bb'. The 'Service credentials' tab is selected. A single credential entry is listed:

Key name	Date created	Controlled by
Service credentials-1	2024-05-16 8:28 PM	Cloudant

The credential details are as follows:

```
{ "apikey": "deYNGz1FBKlcet-olw40uy1-MADQwAxPol4jfTdu7Ag", "host": "f446d07d-a9bb-48d4-a164-03a29a05a628-bluemix.cloudantnosqldb.appdomain.cloud", "iam_apikey_description": "Auto-generated for key crn:v1:bluemix:public:cloudantnosqldb:eu-gb:a/c2eb444f0d6a4054aa5a13839c82f2af:290501ae-d89c-43a9-011f-fc7a/f88ba01b:resource-key:3213a67d-c854-4d1c-89c7-5a8500ee2ae16", "iam_apikey_id": "ApiKey-87871b02-e175-49b3-91e8-0603aca97288", "iam_apikey_name": "Service credentials-1", "iam_role_crn": "crn:v1:bluemix:public:iam::::serviceRole:Manager", "iam_serviceid_crn": "crn:v1:bluemix:public:iam-identity:a/c2eb444f0d6a4054aa5a13839c82f2af::serviceid:ServiceId-0c4da01c-0206-4ba2-a359-dd63e1425523", "password": "fb2d3f81e9fcdac4aa086dfd66dbdc7", "port": 443, "url": "https://apikey-v2-hwgelvh4xtrijtqerxlgg7odppa0ujpo92pne7phy5r", "username": "apikey-v2-hwgelvh4xtrijtqerxlgg7odppa0ujpo92pne7phy5r" }
```

**Now if the username and password are correct then once we run the app.js file we will get the databases.**

The screenshot shows a code editor interface with a dark theme. On the left is the Explorer sidebar showing a project structure with files like index.html, APPJS, package-lock.json, and package.json. The main workspace shows the content of the APPJS file:

```
var express = require('express');
var app = express();
app.listen(8001);
app.use(express.static(__dirname + '/Public'));
console.log('Server running at http://localhost:8001');
```

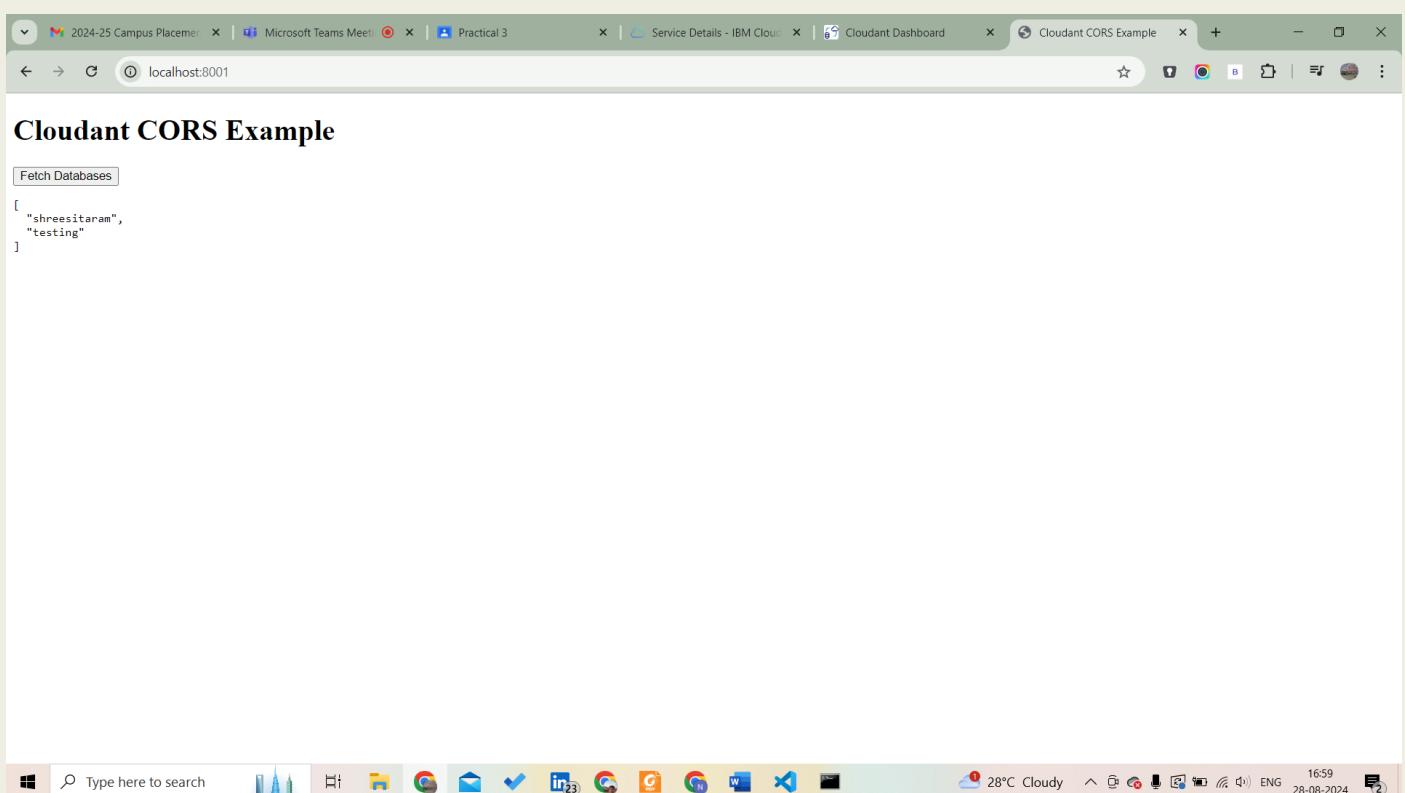
Below the editor is a terminal window titled 'powershell'. It displays the following text:

```
PS D:\SEM-7\CS\PR3> * History restored
Warning: PowerShell detected that you might be using a screen reader and has disabled PSReadLine for compatibility purposes. If you want to re-enable it, run 'Import-Module PSReadLine'.
```

At the bottom of the terminal, the command 'PS D:\SEM-7\CS\PR3>' is shown again.

The screenshot shows the Visual Studio Code interface. The Explorer sidebar on the left lists a project structure with files like APP.JS, index.html, package-lock.json, and package.json. The main editor area displays the content of index.html, which includes a title, a button labeled 'Fetch Databases', and a pre-tag containing JavaScript code for fetching databases. Below the editor are tabs for PROBLEMS, OUTPUT, DEBUG CONSOLE, TERMINAL, PORTS, and COMMENTS. A terminal window is open with the command 'powershell' and the output: 'PS D:\SEM-7\CS\PR3> \* History restored'. The status bar at the bottom shows file paths, line numbers (Ln 5, Col 51), and system information (28°C Cloudy, 16:56, 28-08-2024).

```
Public > index.html > html > head > meta
  1  PE html>
  2  ang="en">
  3
  4  ta charset="UTF-8">
  5  ta name="viewport" content="width=device-width, initial-scale=1.0">
  6  tle>Cloudant CORS Example</title>
  7
  8
  9  >Cloudant CORS Example</h1>
 10 tton id="fetchDatabases">Fetch Databases</button>
 11 e id="result"></pre>
 12
 13 ript>
 14 document.getElementById('fetchDatabases').addEventListener('click', function() {
 15   const url = 'https://f446d07d-a9bb-48d4-a164-03a29a05a628-bluemix.cloudantnosqldb.appdomain.cloud/_all_dbs';
 16   const username = 'apikey-v2-hwgelvh4xtrijtqerxlgg7odppa0ujpo92pne7phy5r';
 17   const password = 'fbdd23f81e9fcdac4aa086df66dbdc7';
 18   const headers = new Headers();
 19   headers.append('Authorization', 'Basic ' + btoa(username + ':' + password));
```



As we can see we get the list of databases , now if I make changes in my password or username I will not be able to get the list.

The screenshot shows the Visual Studio Code interface. The left sidebar has a tree view with 'PR3' expanded, showing 'node\_modules', 'Public', and 'index.html'. The main editor tab is 'index.html'. The code in the editor is:

```
Public > index.html > html > body > script > addEventListener('click') callback
  2   <html lang="en">
  3     <head>
  4       <title>Cloudant CORS Example</title>
  5     </head>
  6     <body>
  7       <h1>Cloudant CORS Example</h1>
  8       <button id="fetchDatabases">Fetch Databases</button>
  9       <pre id="result"></pre>
 10
 11
 12     <script>
 13       document.getElementById('fetchDatabases').addEventListener('click', function() {
 14         const url = 'https://f446d07d-a9bb-48d4-a164-03a29a05a628-bluemix.cloudantnosqldb.appdomain.cloud/_all_db';
 15         const username = 'apikey-v2-hwgelvh4xtrijtqerxlgg7odppa0ujpo92pne7phy5r';
 16         const password = 'K@12345678';
 17         const headers = new Headers();
 18         headers.append('Authorization', 'Basic ' + btoa(username + ':' + password));
 19
 20         fetch(url, { method: 'GET', headers: headers })
 21           .then(response => response.json())
 22           .then(data => {
 23             document.getElementById('result').innerHTML = JSON.stringify(data);
 24           })
 25       });
 26     </script>
 27   </body>
 28 </html>
```

The terminal below shows PowerShell output:

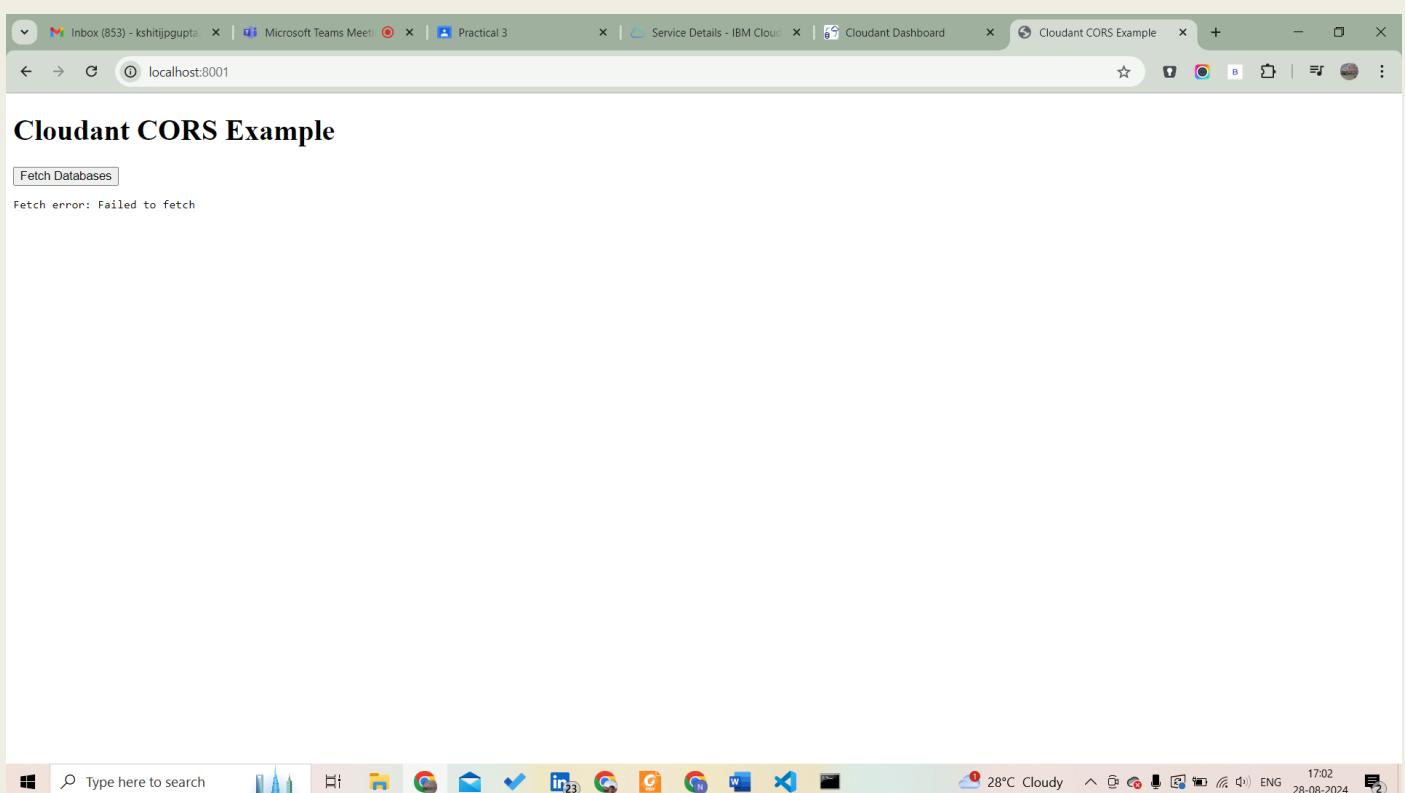
```
PS D:\SEM-7\CS\PR3> * History restored
```

Warnings in the terminal:

```
Warning: PowerShell detected that you might be using a screen reader and has disabled PSReadLine for compatibility purposes. If you want to re-enable it, run 'Import-Module PSReadLine'.
```

File status bar:

```
Ln 17, Col 13 (30 selected) Spaces: 4 UTF-8 CRLF HTML
```



**Now we will try to access a particular database by generating an API.**

**Now to generate an API first we go to the Cloudant dashboard and then in the Permissions tab in the database, there we get an option to generate an API key.**

	<code>_admin</code>	<code>_reader</code>	<code>_writer</code>	<code>_replicator</code>
f446d07d-a9bb-48d4-a164-03a29a05a628-bluemix	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>
apikey-945e9c30cb25485588f537e25b95cec7	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
Unauthenticated connections	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>

Grant database permissions to:  Username or API Key Grant Permissions

**API keys** can grant programmatic access to Cloudant databases. If you generate an API key, you can also manage that key's permissions. Generate API Key

Key: `apikey-945e9c30cb25485588f537e25b95cec7` [Copy](#)  
 Password: `7343c4a370a80faa4f4038034aeadddc56d54e91` [Copy](#)

This message will expire in 3:31. Please make a note of the API key password; for security reasons, the password cannot be shown again.

Here I have given the reader permission to the API key.

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>Cloudant API Example</title>
</head>
<body>
  <h1>Cloudant API Example</h1>
  <button id="fetchData">Fetch Databases</button>
  <pre id="result"></pre>
<script>
  document.getElementById('fetchData').addEventListener('click', function() {
    const apiKey = 'a945e9c30cb25485588f537e25b95cec7';
    const apiPassword = '7343c4a370a80faa4f4038034aeadddc56d54e91';
  })
</script>
```

```

const url = 'https://f446d07d-a9bb-48d4-a164-03a29a05a628-
bluemix.cloudantnosqldb.appdomain.cloud/shreesitaram/_all_docs?include_docs=
true';

const headers = new Headers();

headers.append('Authorization', 'Basic ' + btoa(apiKey + ':' + apiPassword));

fetch(url, { method: 'GET', headers: headers })

  .then(response => response.json())

  .then(data => {

    document.getElementById('result').textContent = JSON.stringify(data,
null, 2);

  })

  .catch(error => {

    document.getElementById('result').textContent = 'Fetch error: ' +
error.message;

  });

});

</script>

</body>

</html>

```

Here the URL is the same hostname we had earlier but this time along with the database name we want to access. Also this time the authentication is done using the API key and the password we generated is also encoded using `btoa()` function. As we can see that after we run the code we are able to see the documents in the database. It is possible because we gave the reader access.

The screenshot shows a browser window with four tabs open:

- Inbox (853) - kshitijgupta21@...
- Service Details - IBM Cloud
- Cloudant Dashboard - database
- Cloudant API Example

The Cloudant API Example tab displays the following JSON response:

```
{ "total_rows": 1, "offset": 0, "rows": [ { "id": "0009368cb10a7ec3980b62b4efea8d02", "key": "0009368cb10a7ec3980b62b4efea8d02", "value": { "rev": "1-527cd95b185108a303994f0c7769d" }, "doc": { "_id": "0009368cb10a7ec3980b62b4efea8d02", "_rev": "1-527cd95b185108a303994f0c7769d", "name": "Kshitij Gupta" } } ] }
```

The browser's taskbar at the bottom shows various pinned icons and the system status bar indicating 27°C, Light rain, ENG, and the date 28-08-2024.

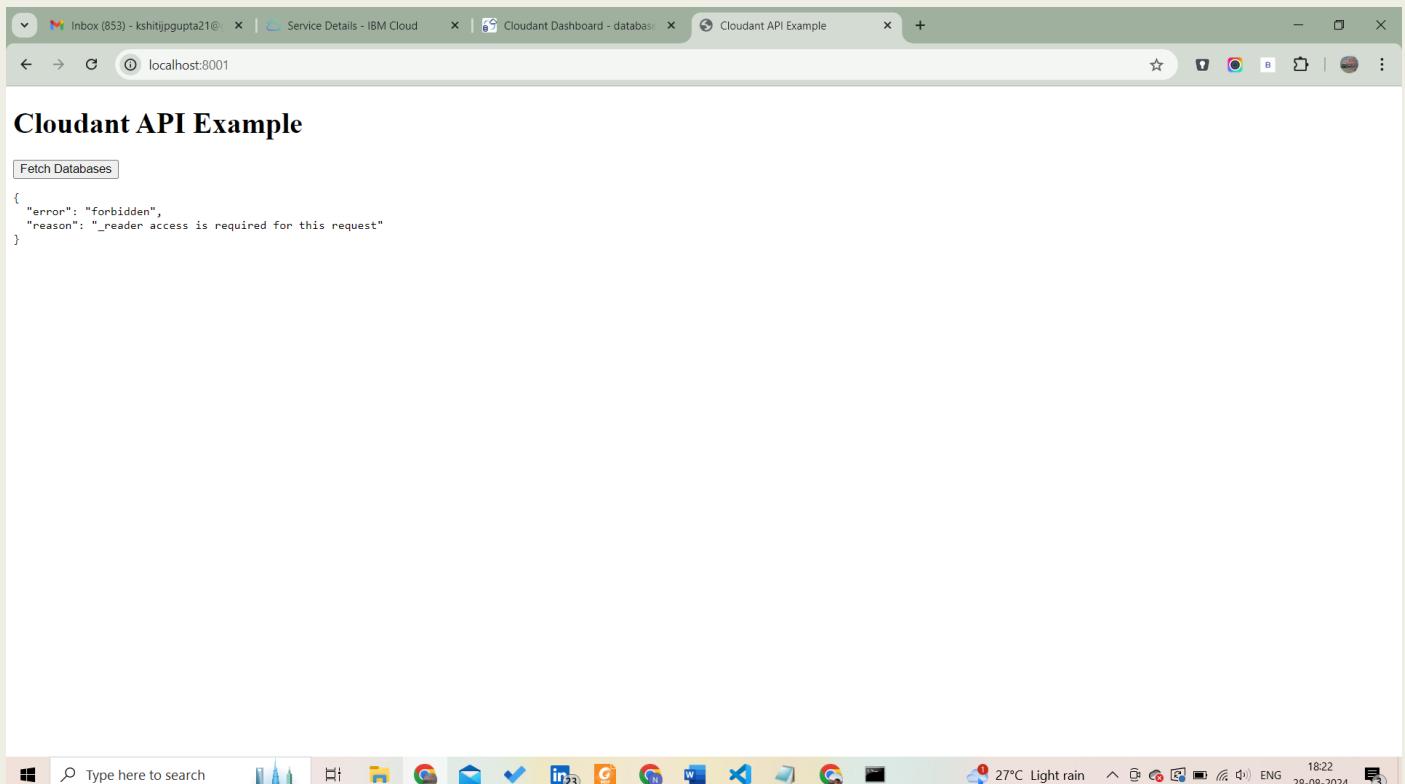
**Now if I change the permission and do not give the reader access. We will not be able to fetch the documents.**

The screenshot shows the Cloudant Dashboard for the database 'shreesitaram'. The left sidebar includes options like All Documents, Query, Permissions (which is selected), Changes, and Design Documents. The main area displays the database permissions table:

	_admin	_reader	_writer	_replicator
f446d07d-a9bb-48d4-a164-03a29a05a628-bluemix	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>
apikey-df7eb2f1932b41ac8eea4181fa0825ca	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>
Unauthenticated connections	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>

A message in the top right corner says "Database permissions have been saved." Below the table, there is a "Grant database permissions to:" input field and a "Grant Permissions" button. A note about API keys is present, along with a "Generate API Key" button. A message box at the bottom contains an API key and password, noting they will expire in 4:06.

The browser's taskbar at the bottom shows various pinned icons and the system status bar indicating 27°C, Light rain, ENG, and the date 28-08-2024.

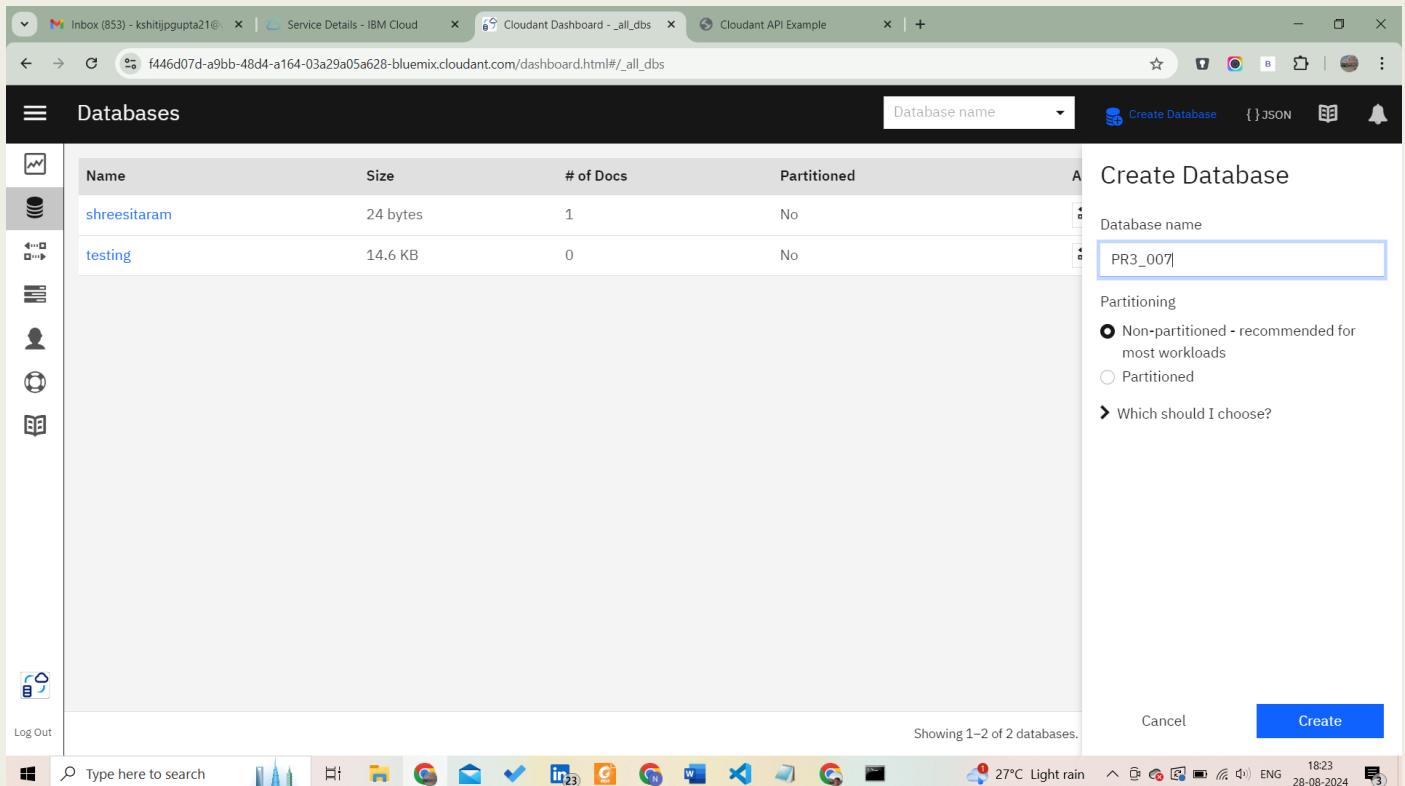


```
{ "error": "forbidden",
  "reason": "_reader access is required for this request"
}
```

Now we will try to replicate a database using the API key we generated.

For this we will first create a new database in which we will replicate the data. We need to have a database already created else this code will generate errors.

To create a new database in cloudant dashboard we will click on create database.



Databases

Name	Size	# of Docs	Partitioned
shreesitaram	24 bytes	1	No
testing	14.6 KB	0	No

Create Database

Database name: PR3\_007

Partitioning:

Non-partitioned - recommended for most workloads

Partitioned

Which should I choose?

Create

**Once this is created we copy the API key we previously created in this and grant access to write only.**

	_admin	_reader	_writer	_replicator
f446d07d-a9bb-48d4-a164-03a29a05a628-bluemix	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>
apikey-df7eb2f1932b41ac8eea4181fa0825ca	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
Unauthenticated connections	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>

	_admin	_reader	_writer	_replicator
f446d07d-a9bb-48d4-a164-03a29a05a628-bluemix	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>
apikey-df7eb2f1932b41ac8eea4181fa0825ca	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>
Unauthenticated connections	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>

**Also we need to make sure that in the source database we also need to give the permission of the replicator. This permission is compulsory as without this it will not allow replication.**

	_admin	_reader	_writer	_replicator
f446d07d-a9bb-48d4-a164-03a29a05a628-bluemix	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>
apikey-df7eb2f1932b41ac8eea4181fa0825ca	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>
Unauthenticated connections	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>

```

<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Cloudant API Example</title>
</head>
<body>
    <h1>Cloudant API Example</h1>
    <button id="replicateData">Replicate Data</button>
    <pre id="result"></pre>
<script>
    document.getElementById('replicateData').addEventListener('click', function () {
        const apiKey = 'apikey-df7eb2f1932b41ac8eea4181fa0825ca';
    })
</script>

```

```
const apiPassword = 'fe4d3890b2fdbbe23db5782e38e8da0cb55fdff3f';

const sourceUrl = 'https://f446d07d-a9bb-48d4-a164-03a29a05a628-
bluemix.cloudantnosqldb.appdomain.cloud/shreesitaram/_all_docs?include_docs=
true';

const targetUrl = 'https://f446d07d-a9bb-48d4-a164-03a29a05a628-
bluemix.cloudantnosqldb.appdomain.cloud/pr3_007';

const headers = new Headers();

headers.append('Authorization', 'Basic ' + btoa(apiKey + ':' + apiPassword));
headers.append('Content-Type', 'application/json');

fetch(sourceUrl, { method: 'GET', headers: headers })

.then(response => response.json())

.then(data => {

  const docs = data.rows.map(row => row.doc);

  docs.forEach(doc => {

    replicateDocument(doc, targetUrl, headers);

  });

})

.catch(error => {

  document.getElementById('result').textContent = 'Replication error: ' +
error;

});

}

function replicateDocument(doc, targetUrl, headers) {

  const docUrl = `${targetUrl}/${doc._id}`;

  fetch(docUrl, { method: 'GET', headers: headers })
```

```
.then(response => {

  if (response.status === 404) {

    // Document does not exist, create it

    return fetch(targetUrl, {

      method: 'POST',

      headers: headers,

      body: JSON.stringify(doc)

    });

  } else {

    return response.json().then(existingDoc => {

      // Document exists, update it

      doc._rev = existingDoc._rev;

      return fetch(docUrl, {

        method: 'PUT',

        headers: headers,

        body: JSON.stringify(doc)

      );

    });

  }

}

.then(response => response.json())

.then(data => {

  if (data.ok) {

    document.getElementById('result').textContent += '\nReplication
success: ${doc._id}';

  } else {

    document.getElementById('result').textContent += '\nReplication
error: ${JSON.stringify(data)}';

  }

})
```

```

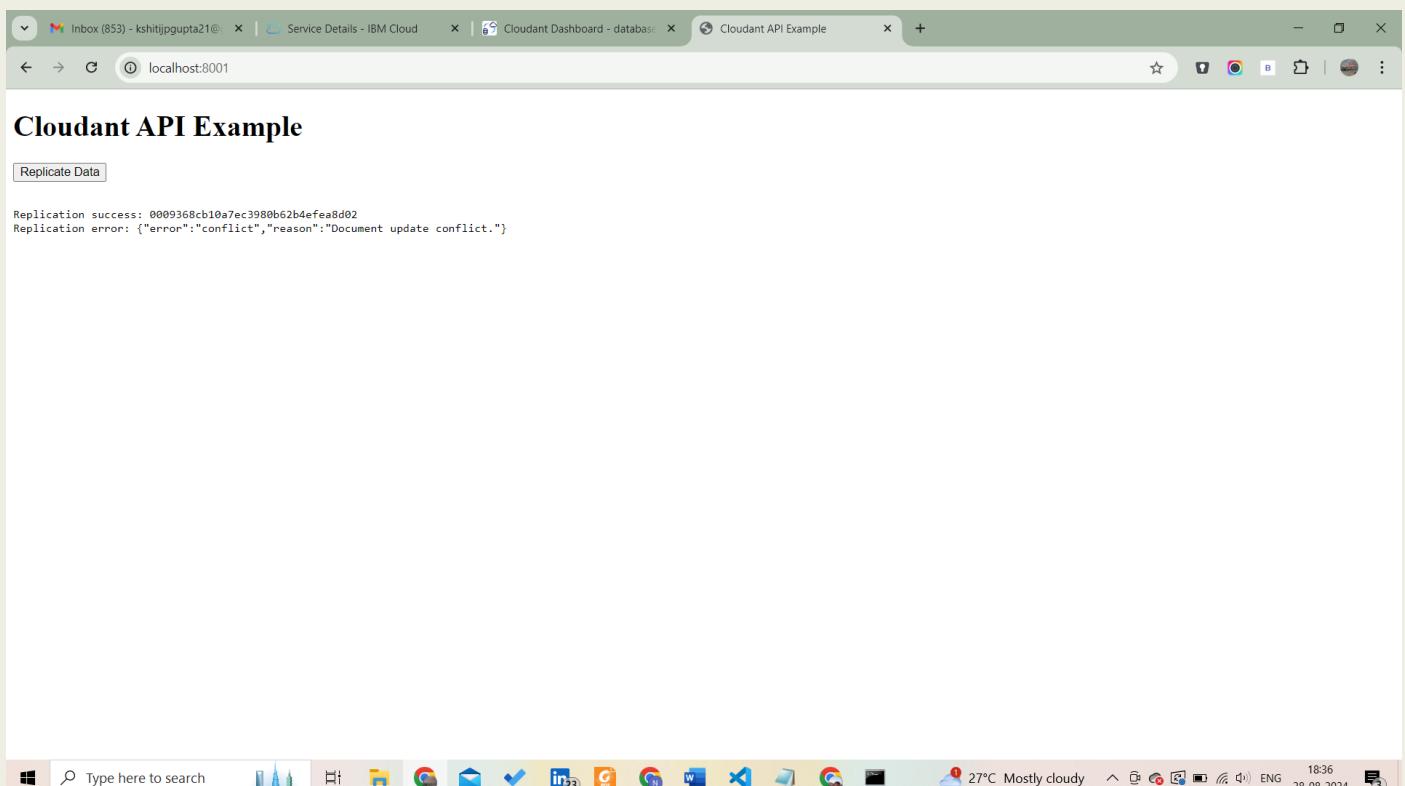
        }
    })
    .catch(error => {
        document.getElementById('result').textContent += `\nReplication error: ${error.message}`;
    });
}

</script>
</body>
</html>

```

**Here we are giving the source url and target url including the hostname along with the database name, also the generated API key and password are sent encoded with request.**

**Now if we run the code and the permissions and password are correct then we get success.**



**In the above case I gave writer access to the target database hence we were able to replicate the data , but if I don't give writer permission then it will**

**4. Now we will see Compaction, which is a process that helps manage and optimize the storage of documents within the database. When documents are deleted or updated, the previous versions of these documents are not immediately removed. Instead, they are marked for deletion and removed later during the compaction process.**

**Now this is automatically done by IBM itself; we can not do it ourselves.**

**The next thing we are doing is to take backup in text file format. We are using a package called couchbackup. So first we will install this package and then in the command prompt.**

**First I am going to take backup of a database and store it in my machine.**

\*Untitled - Notepad

```
File Edit Format View Help
curl "https://apikey-v2-hwge1vh4xtrijtqerxlgg7odppa0u1po92pne7phy5r:fb2d3f81e9fcda4aa086df66dbdc7@f446d07d-a9bb-48d4-a164-03a29a05a628-bluemix.cloudantnosqldb.appdomain.cloud" --db pr3_007 > backup.txt
```

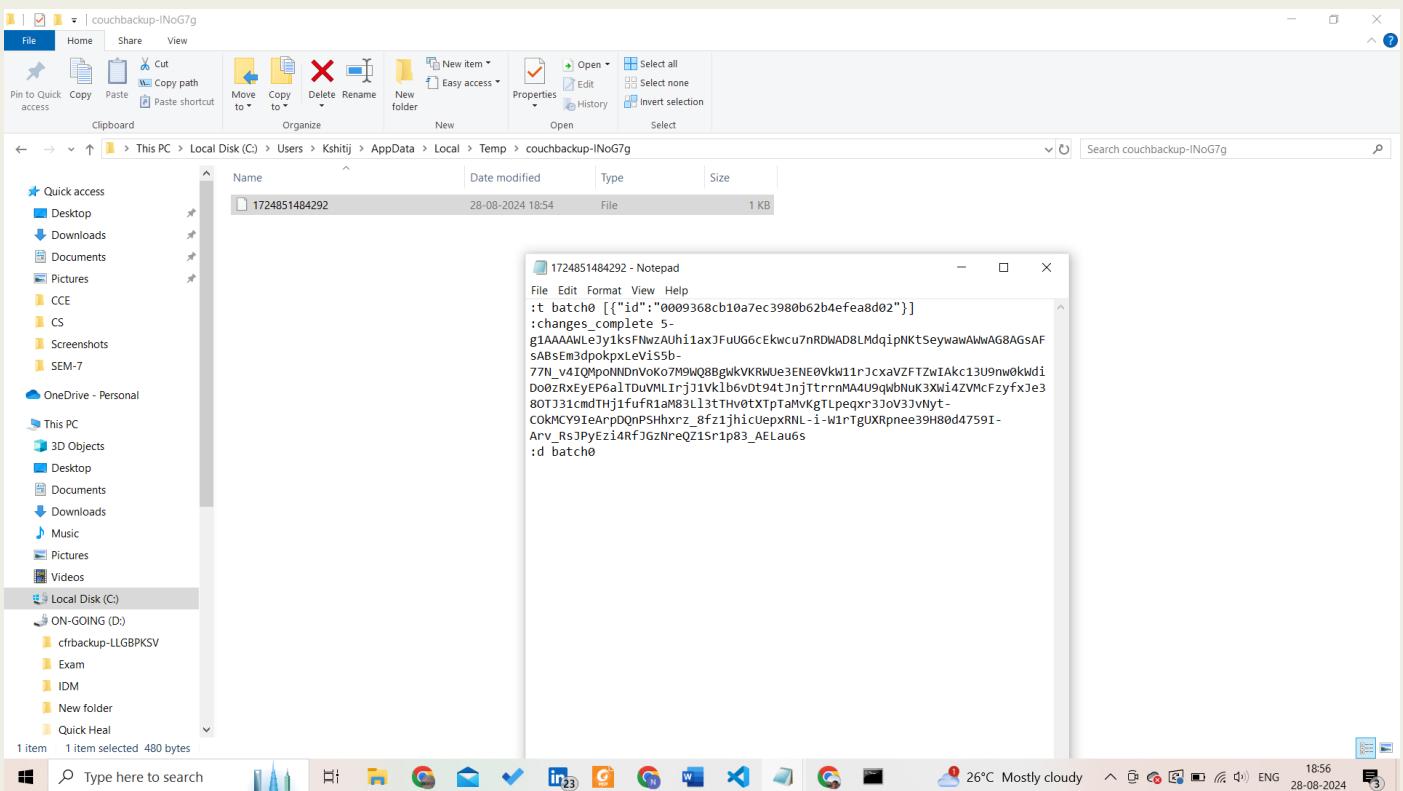
C:\Windows\system32\cmd.exe

```
D:\SEM-7\CS\PR3>couchbackup --url "https://apikey-v2-hwge1vh4xtrijtqerxlgg7odppa0u1po92pne7phy5r:fb2d3f81e9fcda4aa086df66dbdc7@f446d07d-a9bb-48d4-a164-03a29a05a628-bluemix.cloudantnosqldb.appdomain.cloud" --db pr3_007 > backup.txt
=====
Performing backup on https://****:****@f446d07d-a9bb-48d4-a164-03a29a05a628-bluemix.cloudantnosqldb.appdomain.cloud/pr3_007 using configuration:
{
  "bufferSize": 500,
  "log": "C:\\Users\\kshitij\\AppData\\Local\\Temp\\couchbackup-1NoG7g\\1724851484292",
  "mode": "full",
  "parallelism": 5,
  "requestTimeout": 120000,
  "resume": false
}
=====
couchbackup:backup Fetching all database changes... +0ms
couchbackup:backup:batch Total batches received: 1 +0ms
couchbackup:backup:batch Written batch ID: 0 Total document revisions written: 1 Time: 0.84 +276ms
couchbackup:backup Finished - Total document revisions written: 1 +25

D:\SEM-7\CS\PR3>
```

Type here to search 27°C Mostly cloudy 100% Windows (CRLF) 18:54 28-08-2024

Now if I check I will have a text file named backup.



Now from this text file I want to add this data in a database in cloudant.

```
couchbackup --url "https://apikey-v2-
hwgelvh4xtrijtqerxlgg7odppa0ujpo92pne7phy5r:fbfd2d3f81e9fcdac4aa086dfd66dbd
c7@f446d07d-a9bb-48d4-a164-03a29a05a628-
bluemix.cloudantnosqldb.appdomain.cloud" --db pr3_007 < backup.txt
```

```
C:\Windows\system32\cmd.exe
D:\SEM-7\CS\PR3>couchbackup --url "https://apikey-v2-hwgelvh4xtrijtqerxlgg7odppa0ujpo92pne7phy5r:fbfd2d3f81e9fcdac4aa086df
d66dbdc7@f446d07d-a9bb-48d4-a164-03a29a05a628-bluemix.cloudantnosqldb.appdomain.cloud" --db pr3_007 < backup.txt
=====
Performing backup on https://****:****@f446d07d-a9bb-48d4-a164-03a29a05a628-bluemix.cloudantnosqldb.appdomain.cloud/pr3_
007 using configuration:
{
  "bufferSize": 500,
  "log": "C:\Users\Kshitij\AppData\Local\Temp\couchbackup-Vu2x2z\1724851791367",
  "mode": "full",
  "parallelism": 5,
  "requestTimeout": 120000,
  "resume": false
}
=====
couchbackup:backup Fetching all database changes... +0ms
{"name": "@cloudant/couchbackup", "version": "2.10.2", "mode": "full"}
couchbackup:backup:batch Total batches received: 1 +0ms
[{"_id": "0009368cb10a7ec3980b62b4efea8d02", "_rev": "1-527cd95b18510b8a303994f0c7769d", "_revisions": {"ids": ["527cd95b1
8510b8a303994f0c7769d"], "start": 1}, "name": "Kshitij Gupta"}
couchbackup:backup:batch Written batch ID: 0 Total document revisions written: 1 Time: 0.906 +401ms
couchbackup:backup Finished - Total document revisions written: 1 +3s

D:\SEM-7\CS\PR3>
```

Now if we check the database we will have the same data here.

The screenshot shows the Cloudant Dashboard interface. On the left, a sidebar menu includes options like Monitoring, Databases, Replication, Active Tasks, Account, Support, and Documentation. The main area is titled 'pr3\_007' and shows the 'All Documents' list. A single document is listed with the following details:

	id	key	value
<input type="checkbox"/>	0009368cb10a7ec3980b62b4fea8...	0009368cb10a7ec3980b62b4fea8...	{ "rev": "1-527cd...303..."}

Below the table, the status bar indicates 'Showing document 1 - 1. Documents per page: 20'. The bottom taskbar shows various system icons and the date/time '28-08-2024 19:04'.

## Object storage:

Now in the instance of cloud object storage we will create 3 new credentials: Manager, Object writer and Reader. In Service Credentials click on New Credential and here we can give name and select the role.

### MANAGER:

The screenshot shows the IBM Cloud Object Storage service credentials page for the instance "Cloud Object Storage-8e". The "Service credentials" tab is selected. A modal dialog titled "Service Credential Created" is open, stating "PR3\_MANAGER was created" at "2024-08-28 10:18 PM". The main table lists several credentials, including "PR3\_MANAGER" which was just created. The table columns are "Key name" and "Date created".

Key name	Date created
editor-216ed907-0871-40c4-b578-eb70feed9312	2024-04-22 8:46 AM
WDP-Editor-007eadcpr17-donotdelete-pr-ujfjmkn7l5ewq6d	2024-04-25 9:32 AM
PR3_2	2024-08-12 9:46 AM
viewer-5208858a-e966-40b9-8def-cbed0da06a7b	2024-04-25 9:54 AM
<b>PR3_MANAGER</b>	2024-08-28 10:18 PM
admin-fb3d730b-3020-44fa-ac4d-6ea582e4c33e	2024-04-25 11:24 AM
admin-5208858a-e966-40b9-8def-cbed0da06a7b	2024-04-25 9:54 AM

### OBJECT-WRITER

The screenshot shows the IBM Cloud Object Storage service credentials page for the instance "Cloud Object Storage-8e". A modal dialog titled "Create Credentials" is open, prompting for a "Name" (set to "PR3\_OBJ\_WRITER") and a "Role" (set to "Object Writer"). The "Role" dropdown also includes "Select Service ID (Optional)" (set to "Auto Generated"). The "Include HMAC Credential" toggle is off. The "Add" button is highlighted in blue. The background shows a list of existing credentials.

## Reader:

The screenshot shows the IBM Cloud Object Storage interface. On the left, there's a sidebar with 'Instances' and 'Cloud Object Storage' selected. The main area is titled 'Cloud Object Storage-8e' and shows a 'Create Credentials' dialog. In the dialog, the 'Name' field is set to 'PR3\_READER' and the 'Role' is set to 'Reader'. Below these, there's a dropdown for 'Select Service ID (Optional)' with 'Auto Generated' selected. There's also a toggle for 'Include HMAC Credential' which is off. At the bottom of the dialog are 'Cancel' and 'Add' buttons. To the right of the dialog, a list of existing credentials is shown, each with a delete icon. The list includes entries like 'PR3\_M... 4-08-28 10:18 PM', 'editor-2 4-04-22 8:46 AM', 'WDP-Ed... 4-04-25 9:32 AM', 'PR3\_2 4-08-12 9:46 AM', 'viewer-1 4-04-25 9:54 AM', 'admin-fb3d730b-3020-44fa-ac4d-6ea582e4c33e 2024-04-25 11:24 AM', and 'WDP-Editor-pr1821162101007-donotdelete-pr-2x8ehs4bzucp0 2024-04-25 12:22 PM'. The status bar at the bottom shows the date as 28-08-2024.

Now in the Endpoints we can copy the public URL and now go to postman and paste it without any Authorization.

The screenshot shows the 'Endpoints' section of the IBM Cloud Object Storage interface. The sidebar has 'Endpoints' selected. The main area is titled 'Endpoints' and contains a table with three columns: 'Public', 'Private', and 'Direct'. The table has four rows corresponding to the regions: 'us-geo', 'Dallas', 'Washington', and 'San Jose'. Each row shows a public endpoint URL under the 'Public' column. The status bar at the bottom shows the date as 28-08-2024.

Now here we tried to create a bucket but as we did not give any authorization hence we could not create the bucket and Now we can try the access key an ID of Reader Role as authorization.

The screenshot shows the IBM Cloud interface for managing service credentials. The left sidebar has 'Cloud Object Storage' selected under 'Instances'. The main area shows 'Service credentials' for 'Cloud Object Storage-2n'. A 'Reader' key is listed, with its details expanded. The 'Auth Type' dropdown is set to 'AWS Signature'. The JSON key details show the access key ID and secret access key, both of which are highlighted with a red box.

## In authorization we selected AWS Signature

The screenshot shows the Postman interface with the 'Authorization' tab selected. The 'Auth Type' dropdown is set to 'AWS Signature', which is highlighted with a red box. The request body shows XML error responses for unauthorized access attempts.

Now in Reader we only have access to read hence we can not create a bucket

The screenshot shows the Postman interface with the 'Test Results' tab selected. It displays an XML error response indicating an 'Access Denied' error for a bucket creation attempt. The XML content is as follows:

```

<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<Error>
<Code>AccessDenied</Code>
<Message>Access Denied</Message>
<Resource>/bucket-dhzuvil/</Resource>
<RequestId>ee2aceca-199d-45ee-b2b4-57076102b7ba</RequestId>
<httpStatusCode>403</httpStatusCode>
</Error>

```

Similarly we can take the access keys from the Object writer and try to create a bucket. Here also we will be denied the access.

The screenshot shows the Cloud Object Storage Endpoints page. A specific endpoint named "Object Writer" is selected and expanded. The expanded view shows the following JSON configuration:

```
{
  "apikey": "ufd9ogcX87xhiCaDfjvKmJaKOyB191gcio3aaQjANQxm",
  "cos_hmac_keys": {
    "access_key_id": "13624b282382499hba9h553d7c6faaaa",
    "secret_access_key": "7236daa30090d7ae80bed4d8579922a4ff83d7dd3c3897c9"
  },
  "endpoints": "https://control.cloud-object-storage.cloud.ibm.com/v2/endpoints",
  "iam_apikey_description": "Auto-generated for key crn:v1:bluemix:public:cloud-object-storage:global:a/e169921866da4484a0d8a384a35e69a5:e01ecbb4-758d-4c11-9ccb-6fe28ef533da:re...",
  "iam_apikey_id": "3624b28-2382-499b-be9b-553d7ccfaaaa"
}
```

The screenshot shows a Postman request result with a status of 403 Forbidden. The error message is displayed as XML:

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<Error>
  <Code>AccessDenied</Code>
  <Message>Access Denied</Message>
  <Resource>/bucket-dhruvi/</Resource>
  <RequestId>e9f8a432-e25a-48b5-911a-e21f884ac36f</RequestId>
  <httpStatusCode>403</httpStatusCode>
```

Now we will try for the Manager role in postman to create a bucket

The screenshot shows the Cloud Object Storage Endpoints page. A specific endpoint named "Manager" is selected and expanded. The expanded view shows the following JSON configuration:

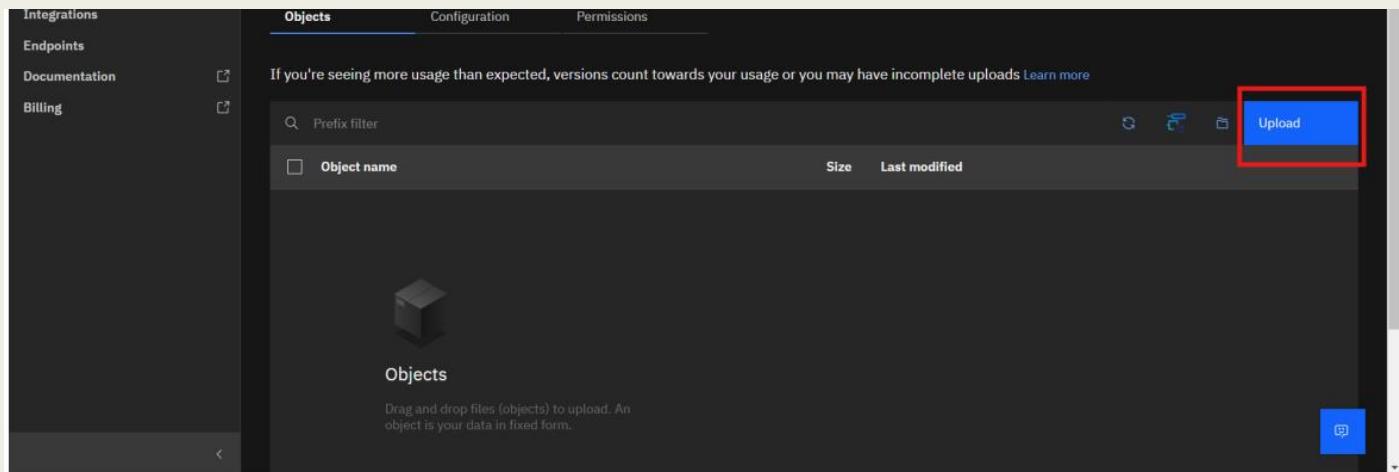
```
{
  "apikey": "So2SSYVoUYJUIfOIBACZpt79cQwhEEni6pGJcM49zzdV",
  "cos_hmac_keys": {
    "access_key_id": "1183940457f34df0af0cd3d7047514d5",
    "secret_access_key": "27ae33946a662f61476004e4c537fa3cd29acdc7a7c467a"
  },
  "endpoints": "https://control.cloud-object-storage.cloud.ibm.com/v2/endpoints",
  "iam_apikey_description": "Auto-generated for key crn:v1:bluemix:public:cloud-object-storage:global:a/e169921866da4484a0d8a384a35e69a5:e01ecbb4-758d-4c11-9ccb-6fe28ef533da:re...",
  "iam_apikey_id": "183940457f34df0af0cd3d7047514d5"
}
```

Here as we can see that the bucket is created

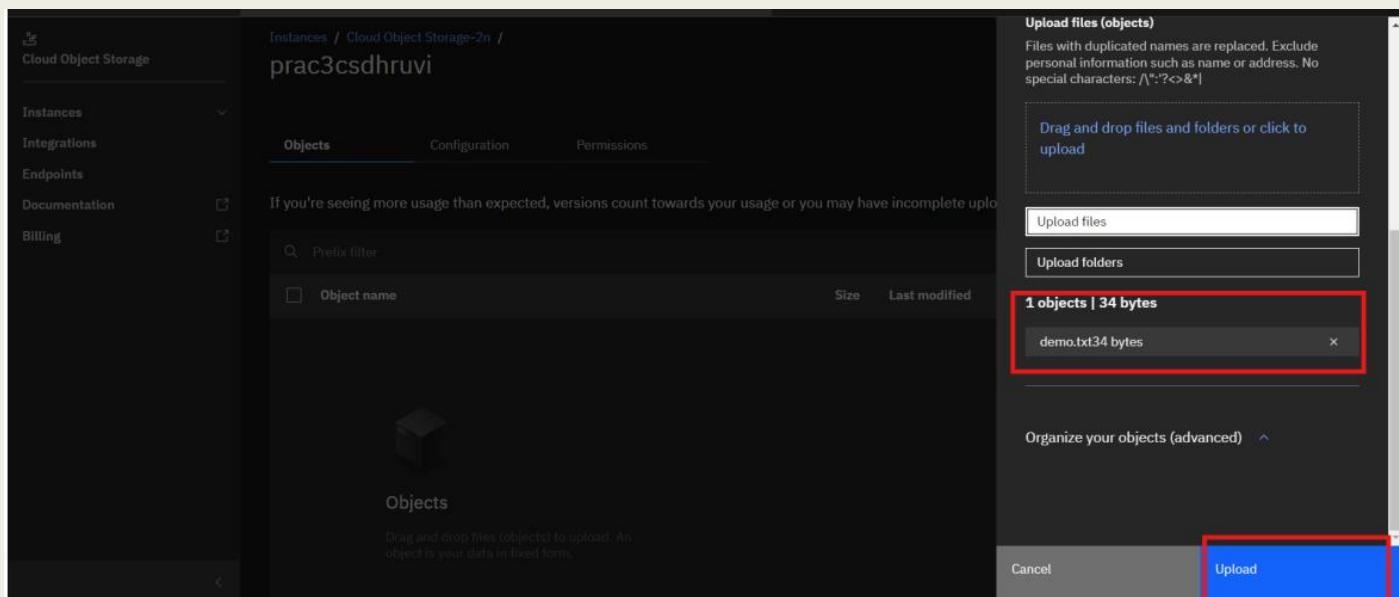
The screenshot shows a Postman request result with a status of 200 OK. The request headers section shows the following configuration:

- AccessKey: 1183940457f34df0af0cd3d7047514d5
- SecretKey: 27ae33946a662f61476004e4c537fa3cd29acdc7a7c467a
- AWS Region: us
- Service Name: s3

**Now we will check the versioning in the bucket we just created. We will upload a file.**

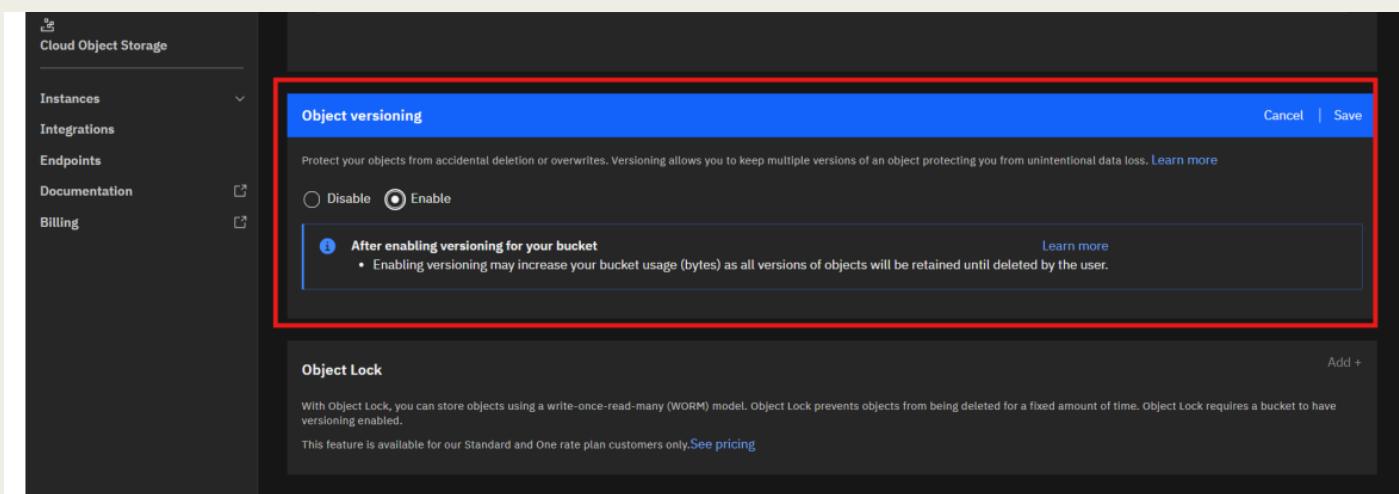


The screenshot shows the AWS Cloud Object Storage interface. On the left, there's a sidebar with 'Cloud Object Storage' selected. The main area has tabs for 'Objects', 'Configuration', and 'Permissions'. A message at the top says, 'If you're seeing more usage than expected, versions count towards your usage or you may have incomplete uploads Learn more'. Below it is a search bar labeled 'Prefix filter' and a checkbox for 'Object name'. To the right are filters for 'Size' and 'Last modified'. At the top right is a blue 'Upload' button, which is highlighted with a red box. The central part of the screen is titled 'Objects' with a sub-instruction 'Drag and drop files (objects) to upload. An object is your data in fixed form.' Below this is a large empty area for file upload.

This screenshot shows the same AWS Cloud Object Storage interface after a file has been uploaded. The 'Upload' button is still highlighted with a red box. A modal window titled 'Upload files (objects)' is open on the right. It contains instructions: 'Files with duplicated names are replaced. Exclude personal information such as name or address. No special characters: /\\,:?>&\*'. Below this is a dashed box with 'Drag and drop files and folders or click to upload'. Underneath are two buttons: 'Upload files' and 'Upload folders'. A summary box shows '1 objects | 34 bytes' with 'demo.txt 34 bytes'. At the bottom of the modal are 'Cancel' and 'Upload' buttons, with the 'Upload' button highlighted with a red box.

**Now enable Object Versioning.**



The screenshot shows the AWS Cloud Object Storage Configuration page. On the left, there's a sidebar with 'Cloud Object Storage' selected. The main area has tabs for 'Instances', 'Integrations', 'Endpoints', 'Documentation', and 'Billing'. A central box is titled 'Object versioning' with a blue header. It contains the text: 'Protect your objects from accidental deletion or overwrites. Versioning allows you to keep multiple versions of an object protecting you from unintentional data loss. [Learn more](#)'. Below this are two radio buttons: 'Disable' and 'Enable', with 'Enable' selected. A note below says: 'After enabling versioning for your bucket' followed by a bullet point: 'Enabling versioning may increase your bucket usage (bytes) as all versions of objects will be retained until deleted by the user.' At the top right of this box are 'Cancel' and 'Save' buttons, with 'Save' highlighted with a red box. Below this is a section titled 'Object Lock' with a note: 'With Object Lock, you can store objects using a write-once-read-many (WORM) model. Object Lock prevents objects from being deleted for a fixed amount of time. Object Lock requires a bucket to have versioning enabled.' and a link 'See pricing'.

**Now we can upload multiple versions of a file and once we enable the View Versions, now we can see all the versions here.**

If you're seeing more usage than expected, versions count towards your usage or you may have incomplete uploads [Learn more](#)

<input type="checkbox"/>	Object name	Size	Last modified	⋮
<input type="checkbox"/>	demo.txt			⋮
<input type="checkbox"/>	00000191-4a7e-2847-4cde-c10b42c49e48	86 bytes	2024-08-13 12:17 PM	⋮
<input type="checkbox"/>	00000191-4a7d-9ef0-aa7f-56411fc80dab	70 bytes	2024-08-13 12:17 PM	⋮
<input type="checkbox"/>	00000191-4a7c-f9a2-f6f3-6b0068436b0f	51 bytes	2024-08-13 12:16 PM	⋮

**Now if we delete any older versions we can see that it is deleted and removed from the list too.**

If you're seeing more usage than expected, versions count towards your usage or you may have incomplete uploads [Learn more](#)

<input type="checkbox"/>	Object name	Size	Last modified	⋮
<input type="checkbox"/>	demo.txt			⋮
<input type="checkbox"/>	00000191-4a7e-2847-4cde-c10b42c49e48	86 bytes	2024-08-13 12:17 PM	⋮
<input type="checkbox"/>	00000191-4a7d-9ef0-aa7f-56411fc80dab	70 bytes	2024-08-13 12:17 PM	⋮
<input type="checkbox"/>	00000191-4a7c-f9a2-f6f3-6b0068436b0f	51 bytes	2024-08-13 12:16 PM	⋮
<input type="checkbox"/>	null	34 bytes	2024-08-13 12:11 PM	⋮

Drag and drop files (objects) here or click to upload

Delete

Cloud Object Storage

- Instances
- Integrations
- Endpoints
- Documentation
- Billing

Objects

If you're seeing more usage than expected, versions count towards your usage or you may have incomplete uploads [Learn more](#)

demo.txt

Items per page: 25 1–1 of 1 item

Are you sure you want to delete the following object version?

This action will delete the object version permanently. The object version will not be recoverable.

To confirm deletion, type 'Delete'

Delete object version

Object name	Version ID	Size	Last modified
demo.txt	00000191-4a7d-9ef0-aa7f-56411fc80dab	70 bytes	2024-08-13 12:17 PM

Cancel Delete

**But now if we try to delete the current version it is deleted but not removed instead delete market is there and we can also restore it**

<input type="checkbox"/> Object name		Size	Last modified	
<input type="checkbox"/> demo.txt				
<input type="checkbox"/> 00000191-783b-36f2-95a6-a0ab6f273935	Current version	0 bytes	2024-08-22 9:27 AM	
<input type="checkbox"/> 00000191-4a7e-2847-4cde-c10b42c49e48		86 bytes	2024-08-13 12:17 PM	
<input type="checkbox"/> 00000191-4a7c-f9a2-f6f3-6b0068436b0f		51 bytes	2024-08-13 12:16 PM	
<input type="checkbox"/> null		34 bytes	2024-08-13 12:11 PM	

Now if I want to change the current version of the object we can go to the Versions

Overview		Versions	Lifecycle	Retention
<input type="checkbox"/> Search				
<input type="checkbox"/> Version ID		Archive Type	Size	Last modified
<input type="checkbox"/> 00000191-4a7e-2847-4cde-c10b42c49e48	Current version		86 bytes	2024-08-13 12:17 PM
<input type="checkbox"/> 00000191-4a7c-f9a2-f6f3-6b0068436b0f			51 bytes	2024-08-13 12:16 PM
<input type="checkbox"/> null			34 bytes	2024-08-13 12:11 PM
Items per page:	25	1–3 of 3 items	1	1 of 1 page

Here select the version you want to make current version.

Overview		Versions	Lifecycle	Retention
<input type="checkbox"/> Search				
<input type="checkbox"/> Version ID		Archive Type	Size	Last modified
<input type="checkbox"/> 00000191-4a7e-2847-4cde-c10b42c49e48	Current version		86 bytes	2024-08-13 12:17 PM
<input type="checkbox"/> 00000191-4a7c-f9a2-f6f3-6b0068436b0f			51 bytes	2024-08-13 12:16 PM
<input type="checkbox"/> null			34 bytes	2024-08-13
Items per page:	25	1–3 of 3 items	1	Manage tags Download Delete

And as we can see the current version is changed.

Search 

<input type="checkbox"/> Version ID	Archive Type	Size	Last modified	
<input type="checkbox"/> 00000191-783e-02b2-8b3e-cf932cf24386	Current version	51 bytes	2024-08-22 9:30 AM	
<input type="checkbox"/> 00000191-4a7e-2847-4cde-c10b42c49e48		86 bytes	2024-08-13 12:17 PM	
<input type="checkbox"/> 00000191-4a7c-f9a2-f6f3-6b0068436b0f		51 bytes	2024-08-13 12:16 PM	
<input type="checkbox"/> null		34 bytes	2024-08-13 12:11 PM	

Items per page: 25 ▾ 1–4 of 4 items 1 ▾ 1 of 1 page  