Ganpat University
॥ विद्यया समाजोत्कर्षः ॥
Institute of Computer Technology

Name: Tushar Panchal

En.No: 21162101014

Sub: CS (Cloud Security)

Branch: CBA

Batch:71

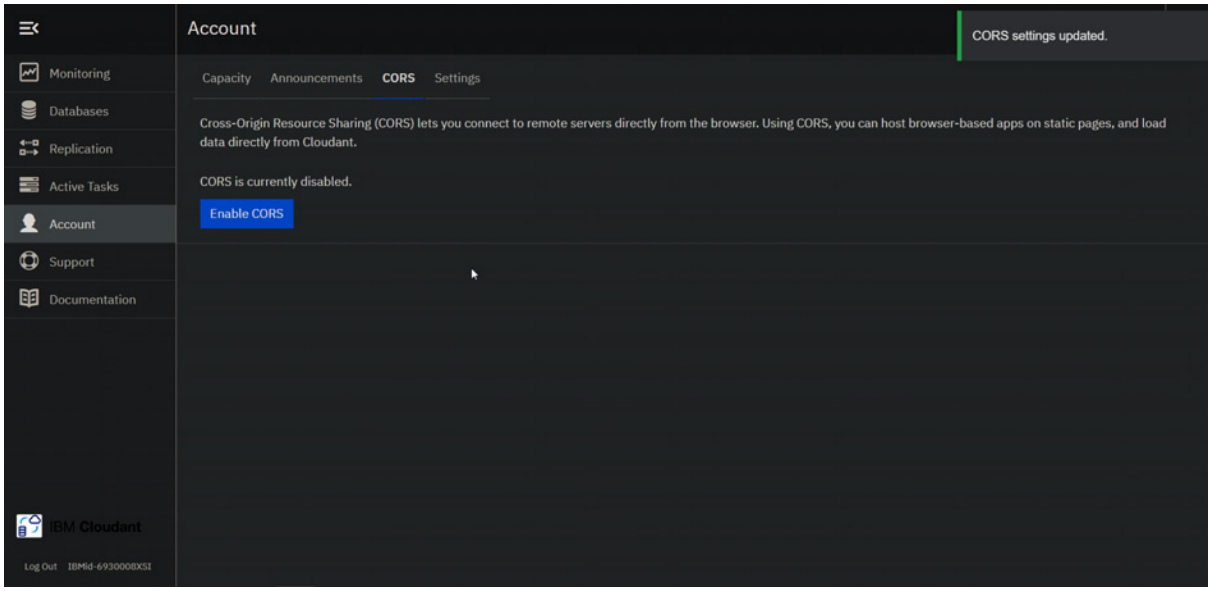
----------PRACTICAL 03----------

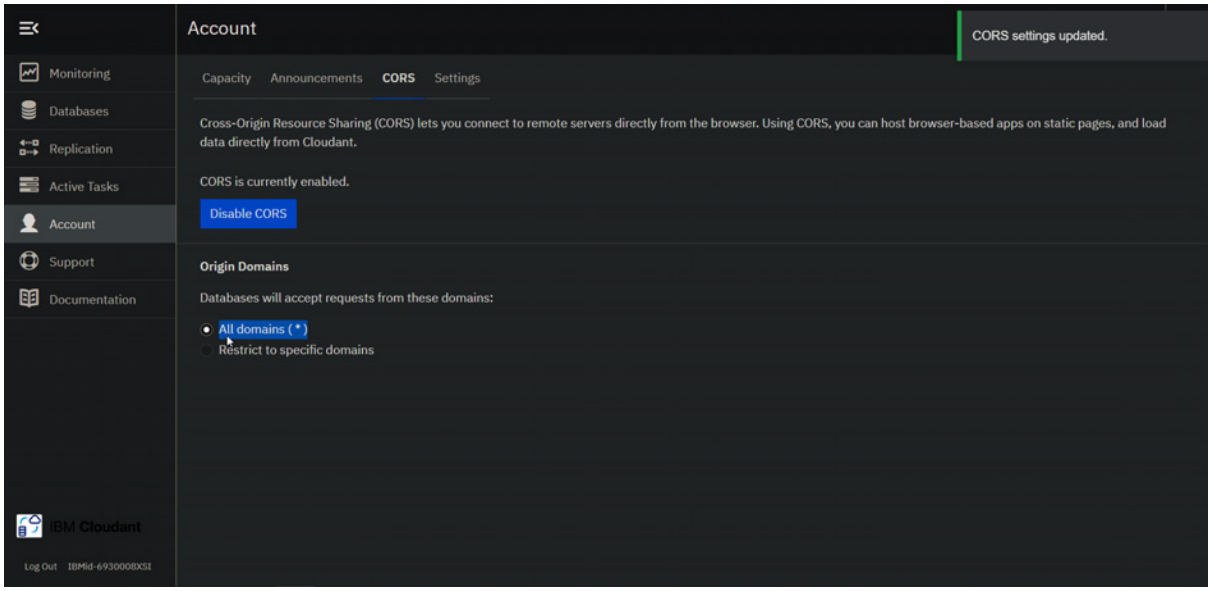
❖ Scenario :

A healthcare organization uses IBM Cloud services to store and manage sensitive patient medical records. The organization aims to ensure that the data is confidential, its integrity is maintained, and it is available whenever needed.

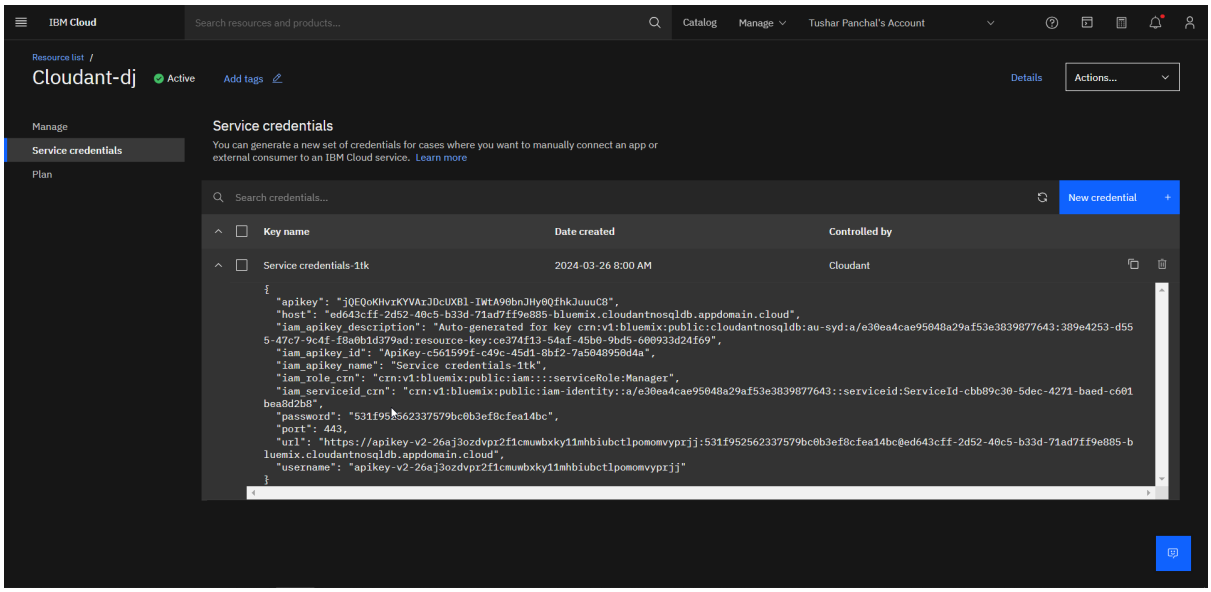
Show the implementation for both Cloudant and object storage to meet the objective

» **Step 1 :** Visit Cloudant dashboard and enable CORS settings.





» **Step 2 :** Copy the Cloudant service credentials and using API, fetch all databases' list.

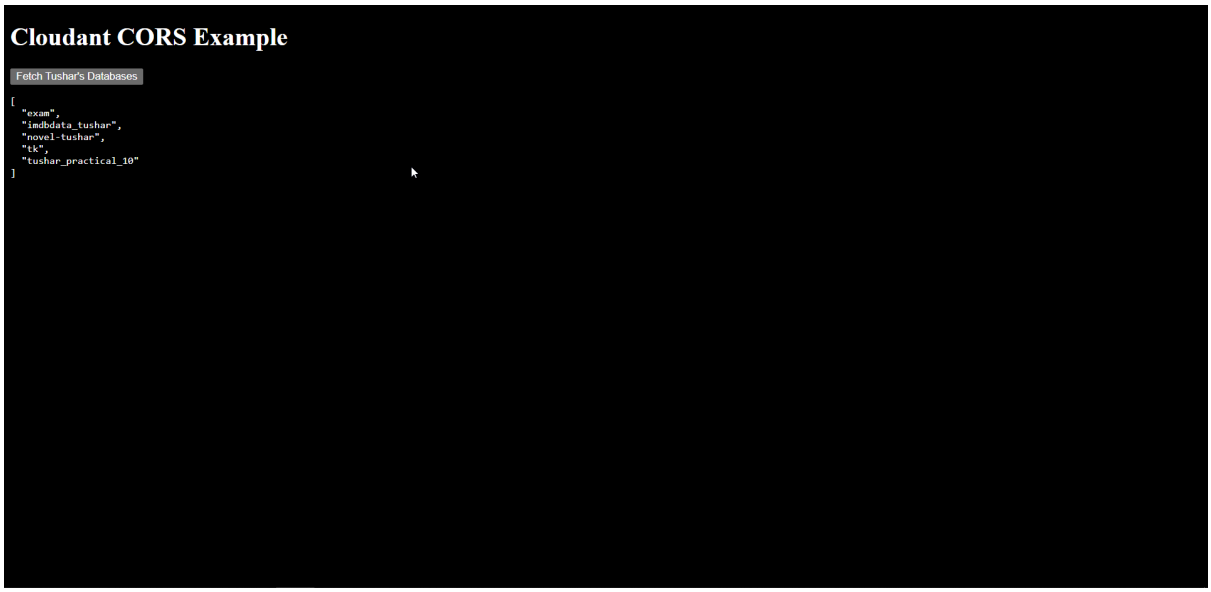


✓ **Source Code :**

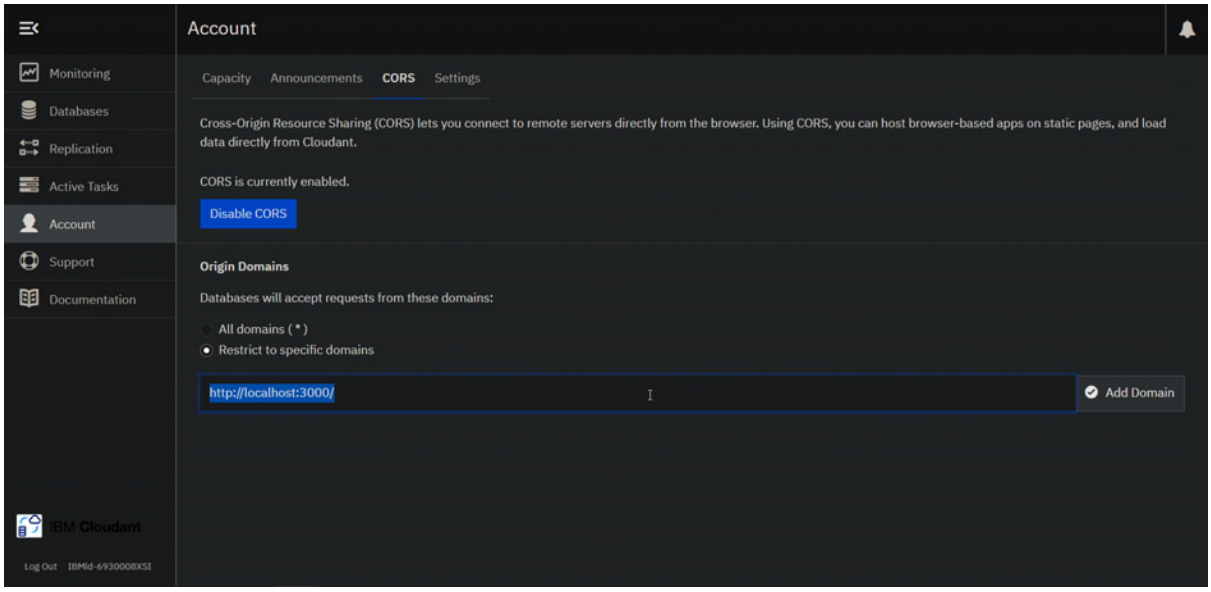
```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Cloudant CORS Example</title>
</head>
<body>
    <h1>Cloudant CORS Example</h1>
    <button id="fetchDatabases">Fetch Tushar's Databases</button>
    <pre id="result"></pre>
```

```
    <script>
        document.getElementById('fetchDatabases').addEventListener('click',
function() {
            const url = 'https://ed643cff-2d52-40c5-b33d-71ad7ff9e885-
bluemix.cloudantnosqldb.appdomain.cloud/_all_dbs';
            const username = 'apikey-v2-
26aj3ozdvpr2f1cmuwbxky11mhbiubctlpomomvyprjj';
            const password = '531f952562337579bc0b3ef8cfea14bc';
            const headers = new Headers();
            headers.append('Authorization', 'Basic ' + btoa(username + ':' +
password));

            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>
```

» **Step 3 :** Try running API through nodejs.



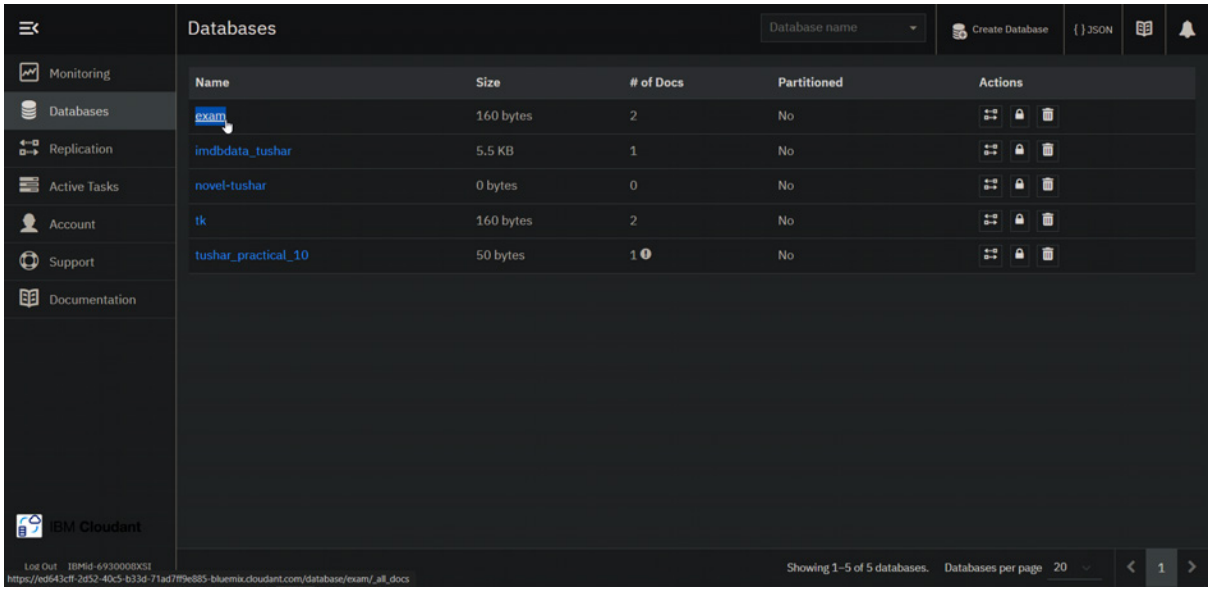
» **Step 4 :** Also if the restricted IP address is specified in CORS, the access can be secured.



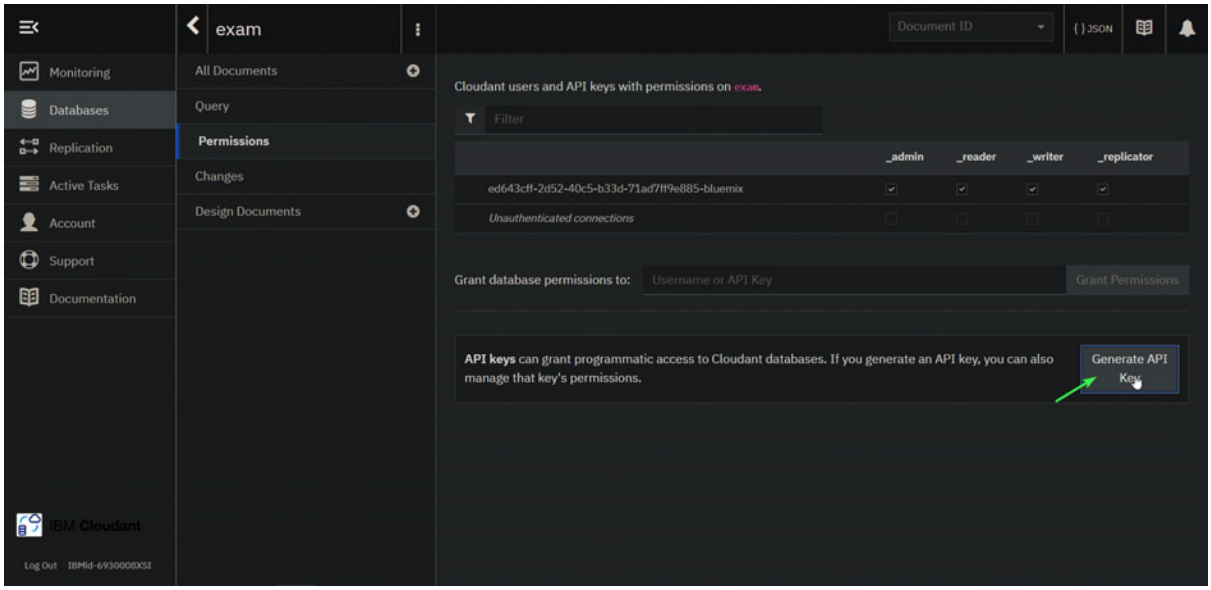
Now if we try to get it with another domain it will get error:

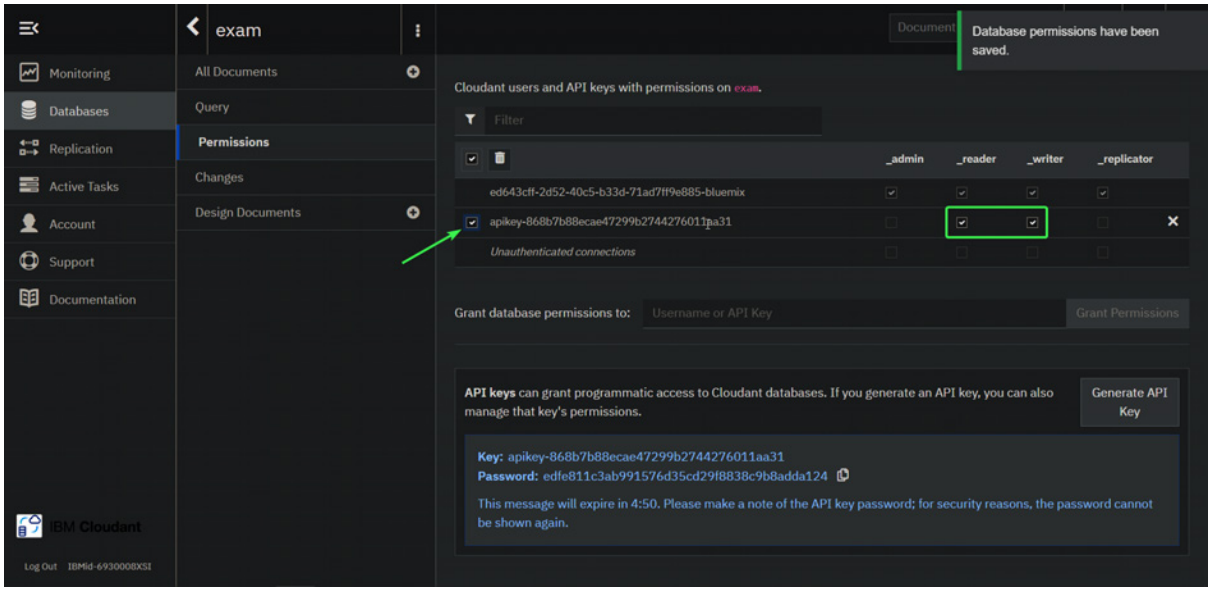
(Unauthorized error due to restricted IP access)

» **Step 5 :** Now, choose any database to check access through user generated API keys.

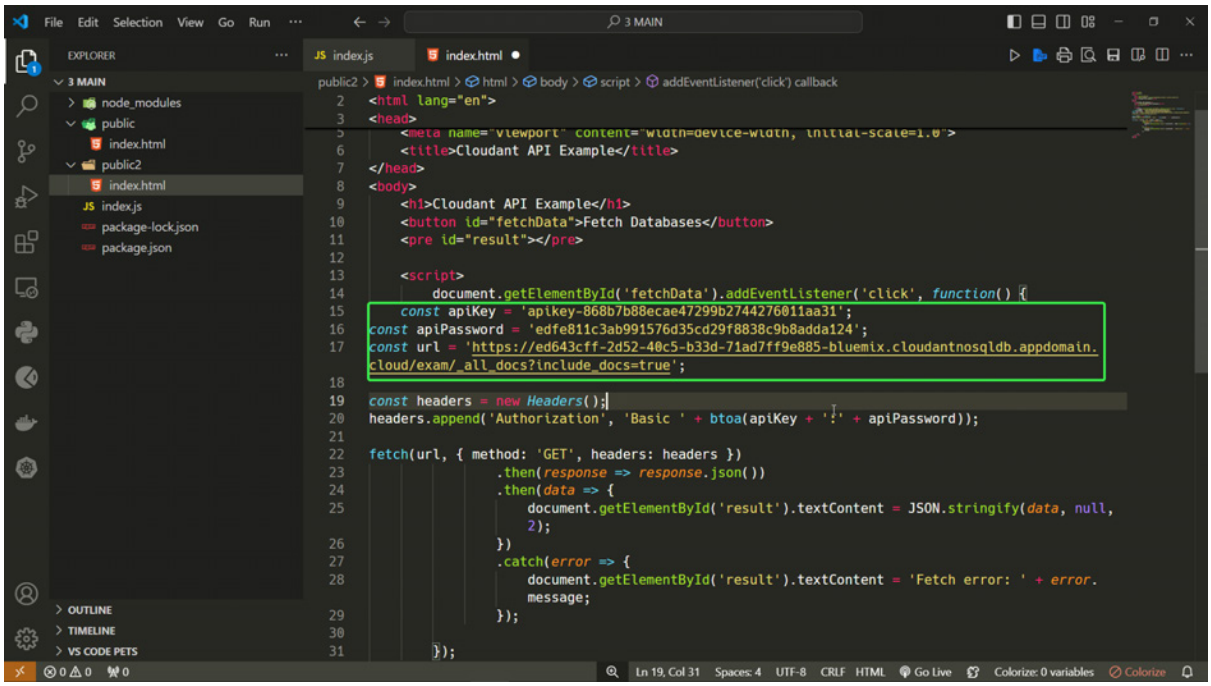


» **Step 6 :** Create new API key with necessary permissions and use with respective code.





Change this API key and password and url in new code:

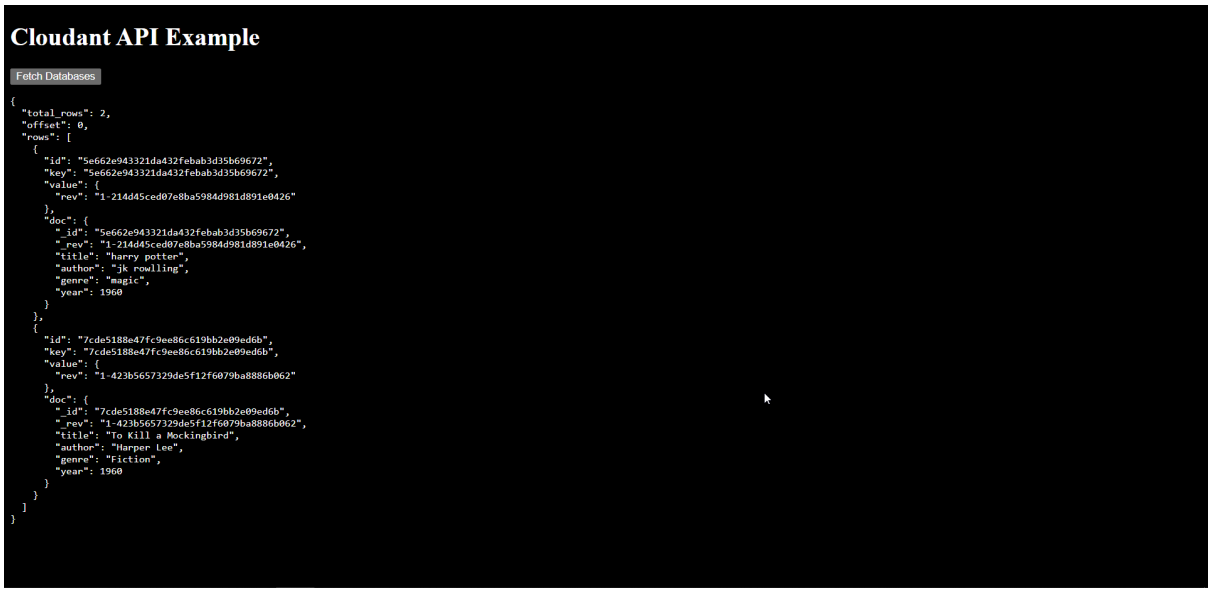


✓ Source Code :

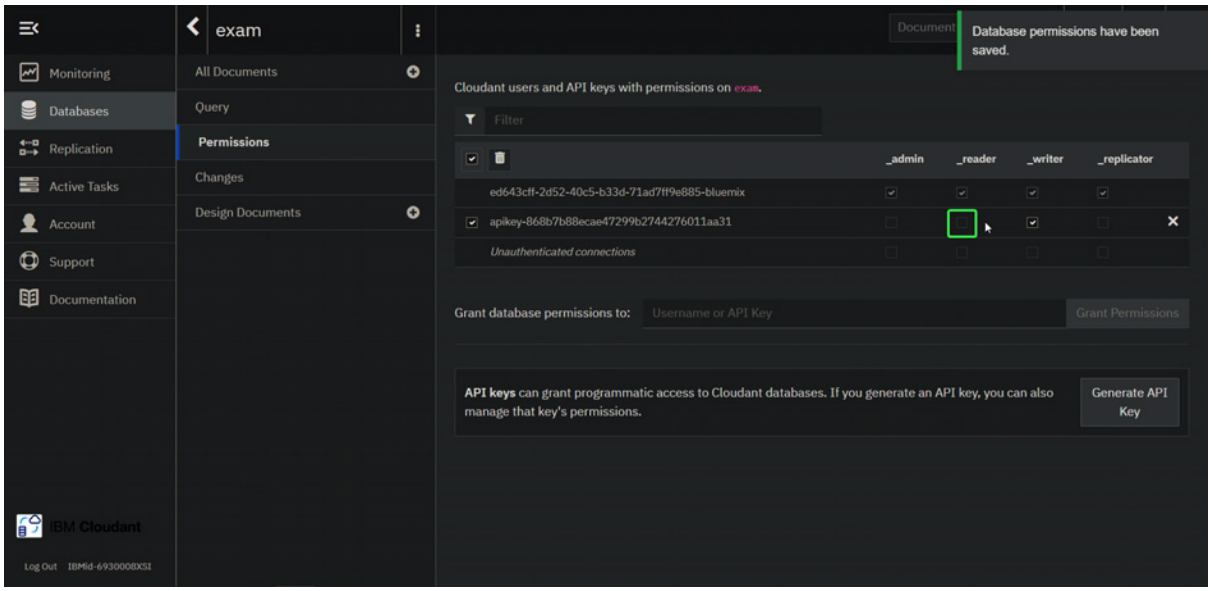
```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Cloudant API Example</title>
</head>
<body>
    <h1>Cloudant API Example</h1>
    <button id="fetchData">Fetch Databases</button>
```

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

    <script>
        document.getElementById('fetchData').addEventListener('click',
function() {
    const apiKey = 'apikey-868b7b88ecae47299b2744276011aa31';
const apiPassword = 'edfe811c3ab991576d35cd29f8838c9b8adda124';
const url = 'https://ed643cff-2d52-40c5-b33d-71ad7ff9e885-
bluemix.cloudantnosqldb.appdomain.cloud/exam/_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>
```

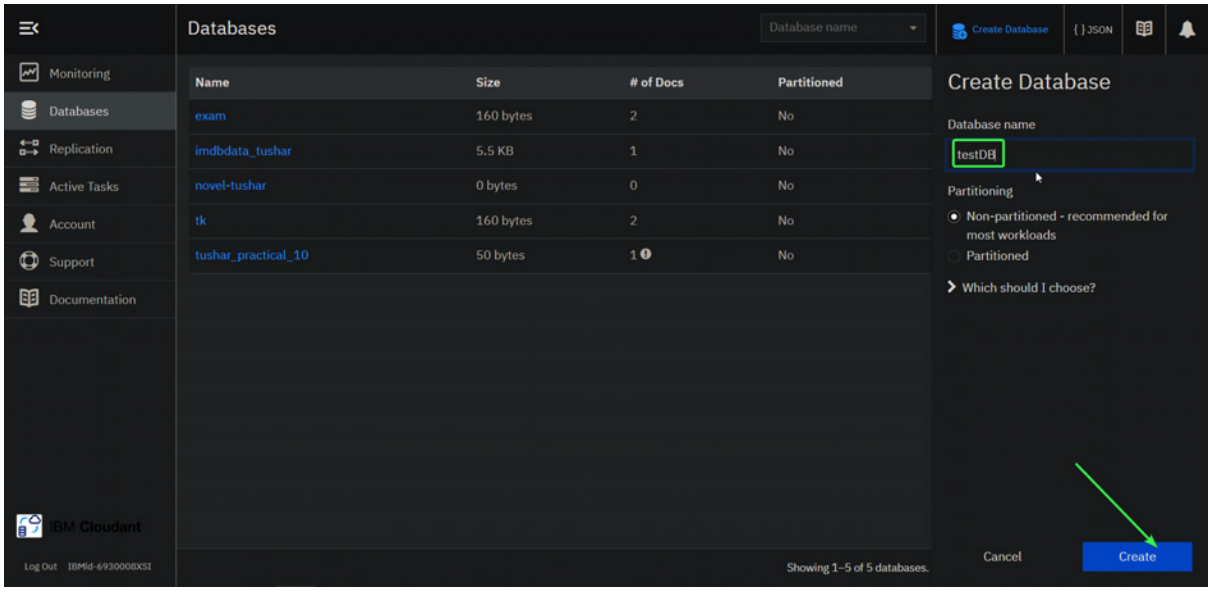
Try running API through nodejs



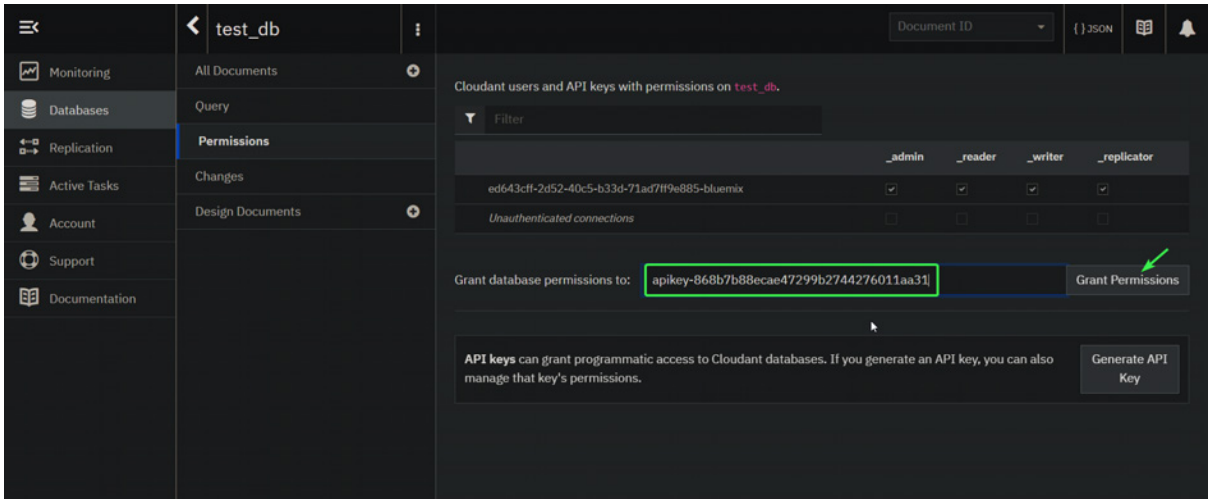
- **Step 7 :** But if access is restricted through api user, then it will not be allowed to do operations (like here no read access => can't list databases).



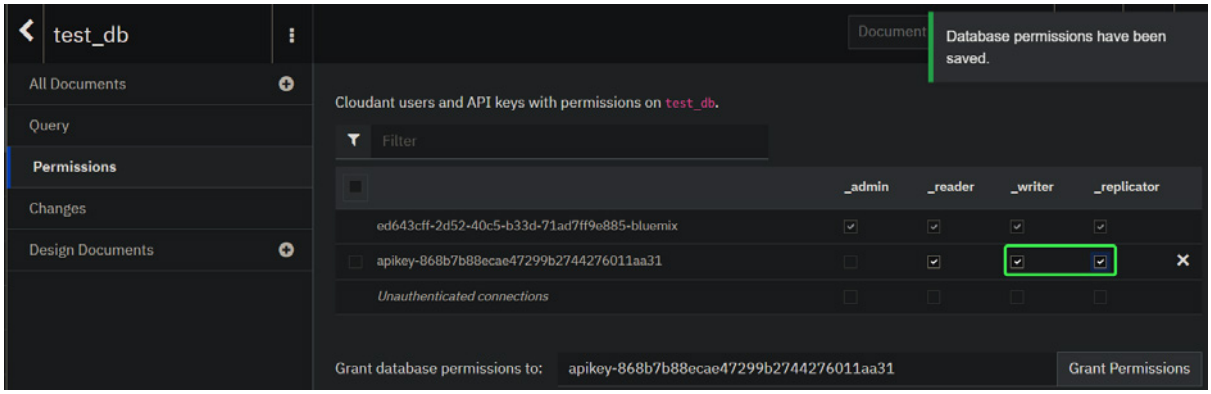
» **Step 8 :** Now create new database to check replication security.



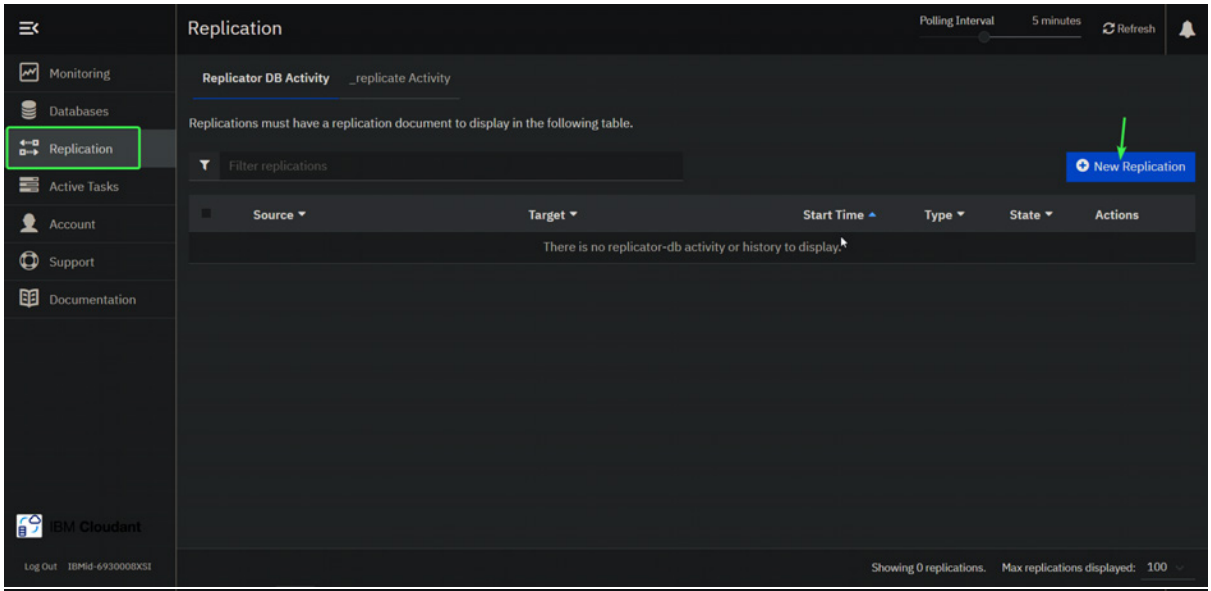
» **Step 9 :** Add previously generated API key user with necessary permissions to this new destination db.

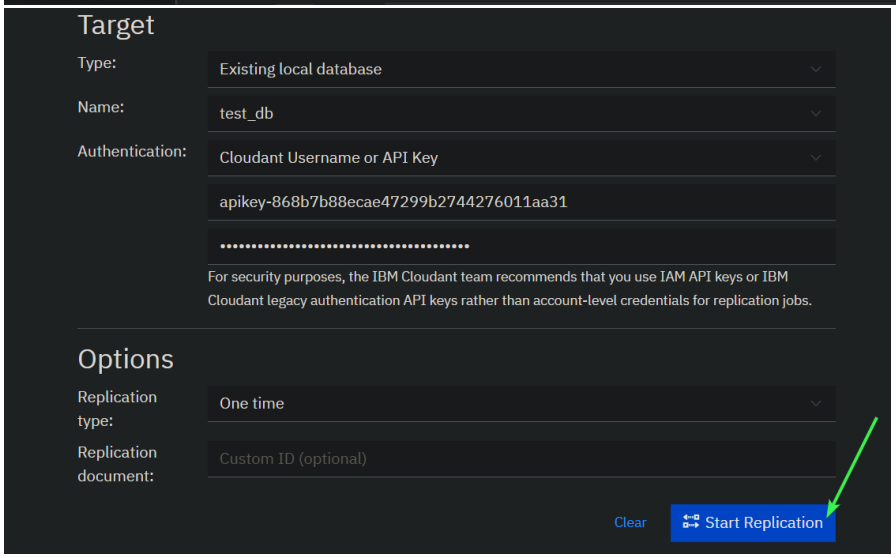


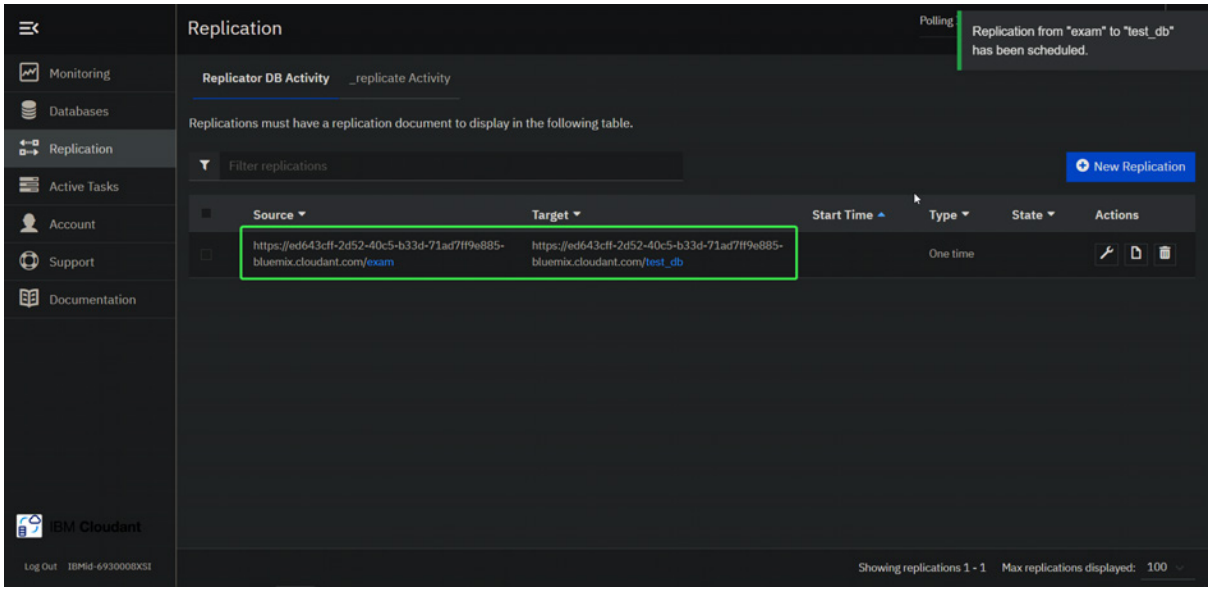
Make sure to edit permissions:



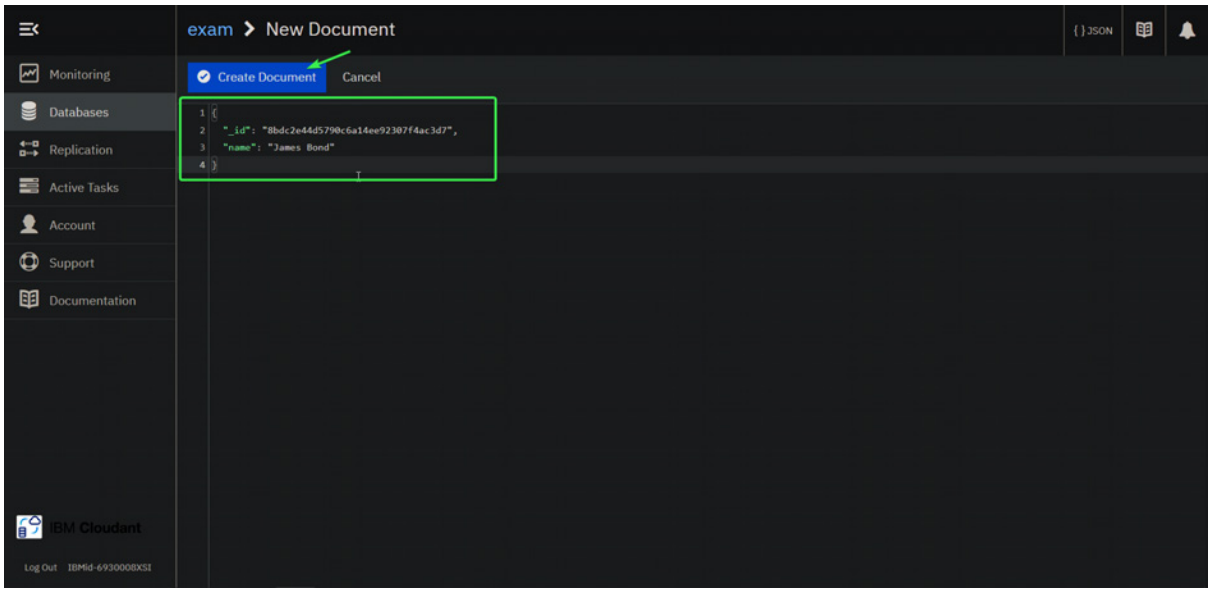
» **Step 10 :** Perform new replication using API key user selecting both source and destination databases.







» **Step 11 :** Now, check the same access through nodejs code by creating a new record in source db.

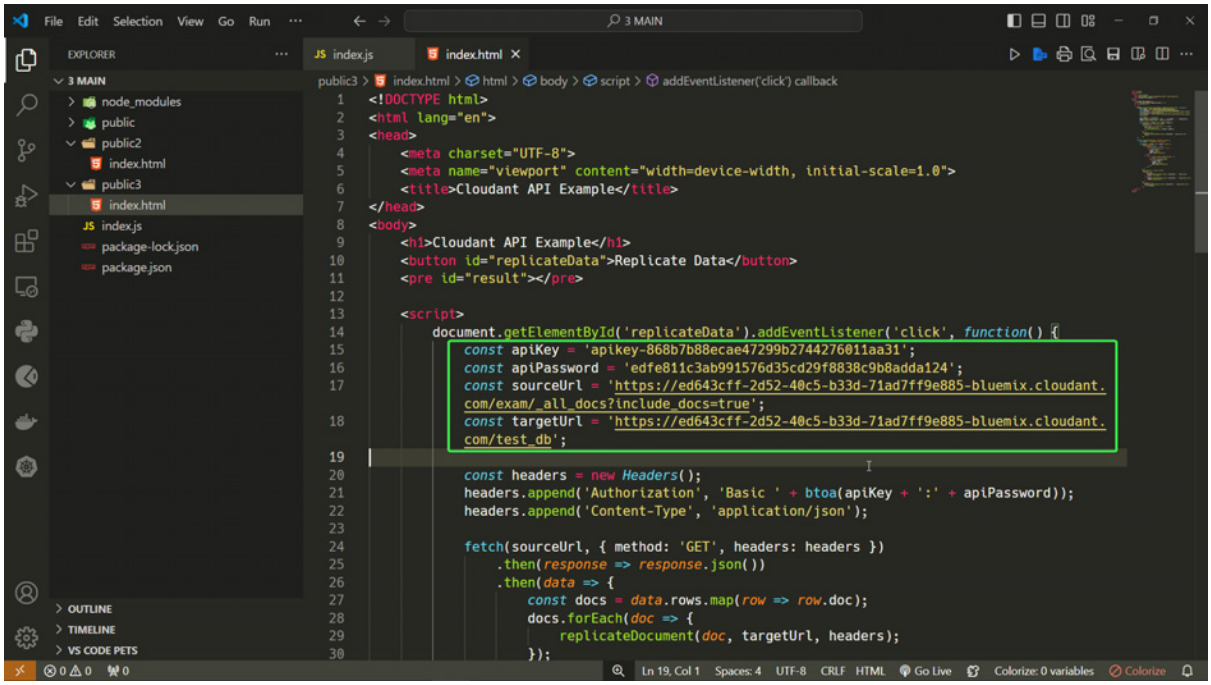


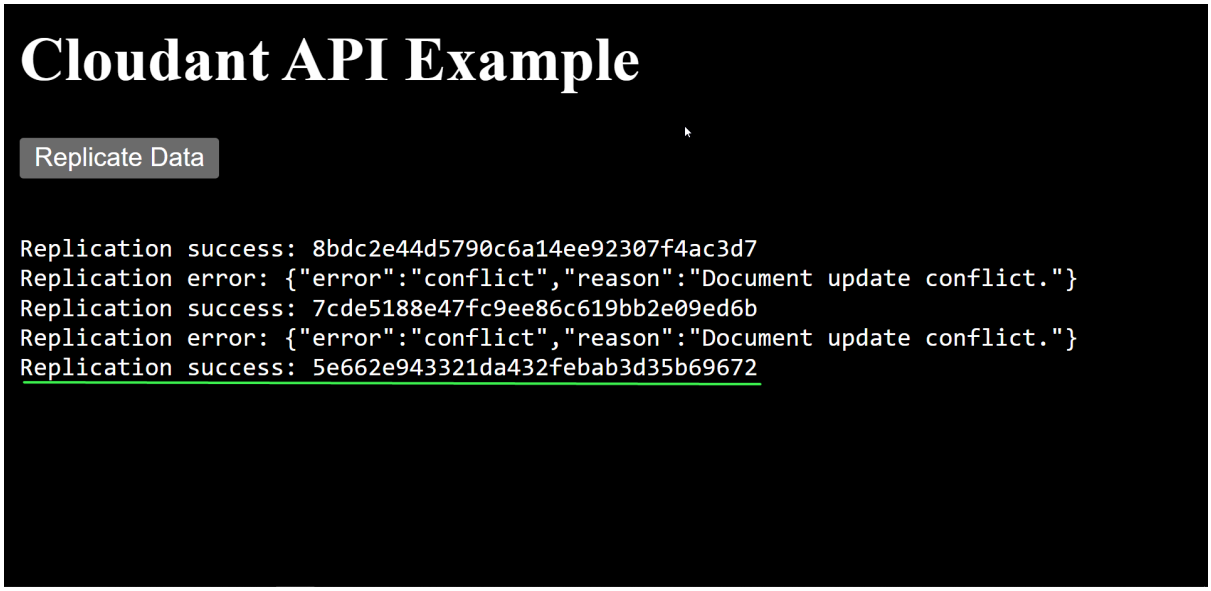
✓ **Source Code :**

```
<!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>
```

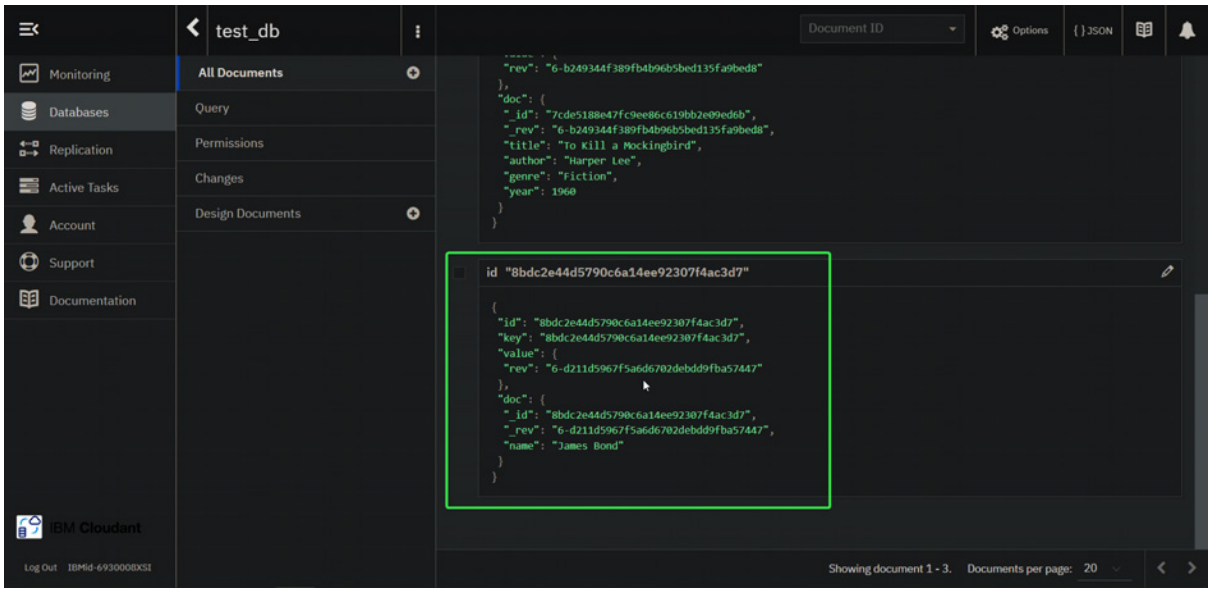
```
    <script>
        document.getElementById('replicateData').addEventListener('click',
function() {
            const apiKey = 'apikey-868b7b88ecae47299b2744276011aa31';
            const apiPassword =
'edfe811c3ab991576d35cd29f8838c9b8adda124';
            const sourceUrl = 'https://ed643cff-2d52-40c5-b33d-
71ad7ff9e885-bluemix.cloudant.com/exam/_all_docs?include_docs=true';
            const targetUrl = 'https://ed643cff-2d52-40c5-b33d-
71ad7ff9e885-bluemix.cloudant.com/test_db';

            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>
```



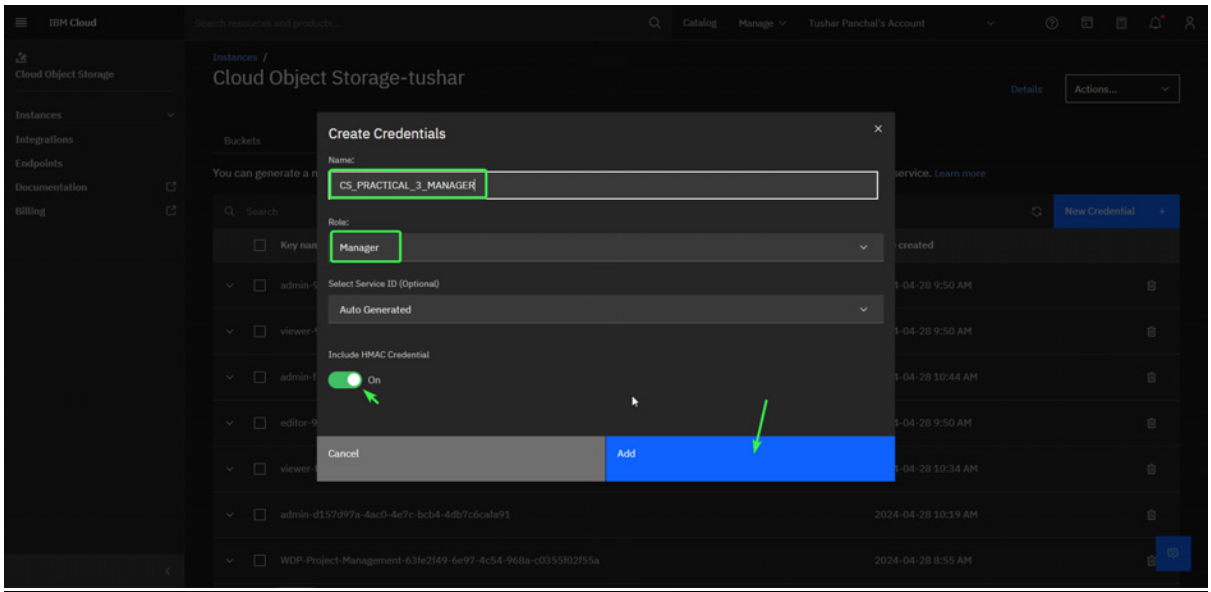


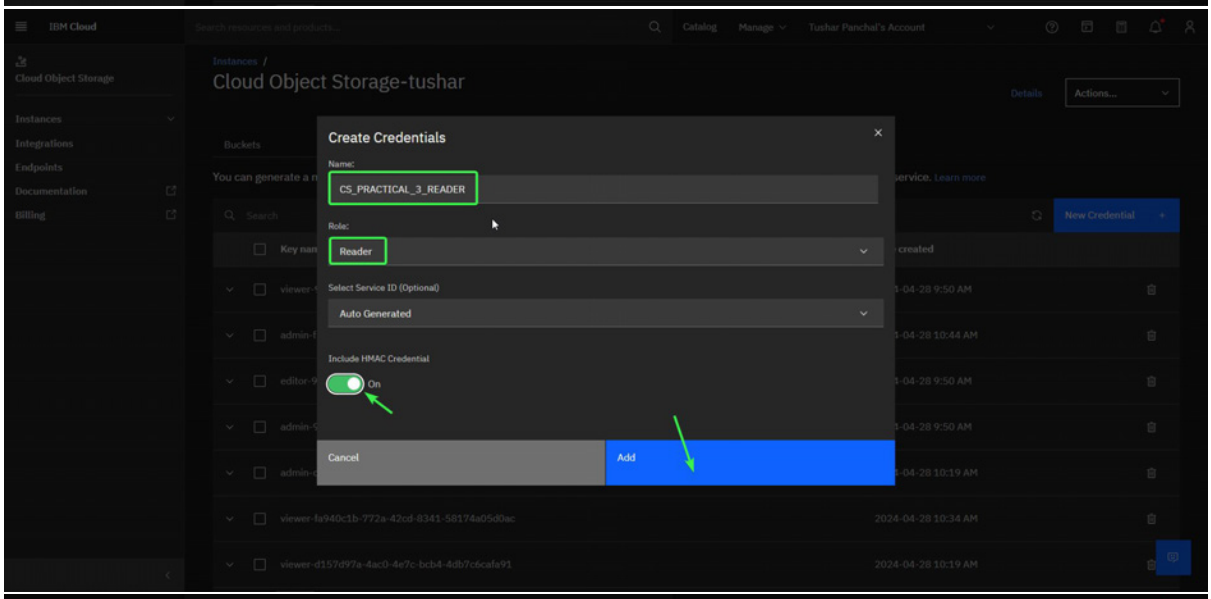
As you can see in destination DB replication is succeeded:

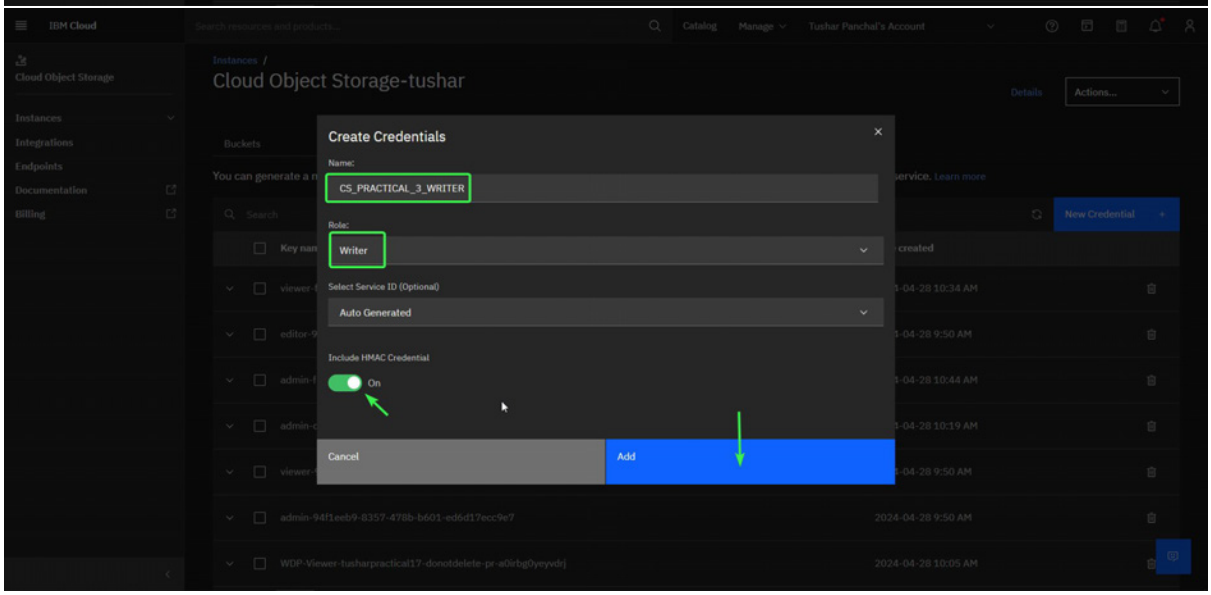


reader access for source and writer-replicator access for destination is needed as seen here for successful replication.

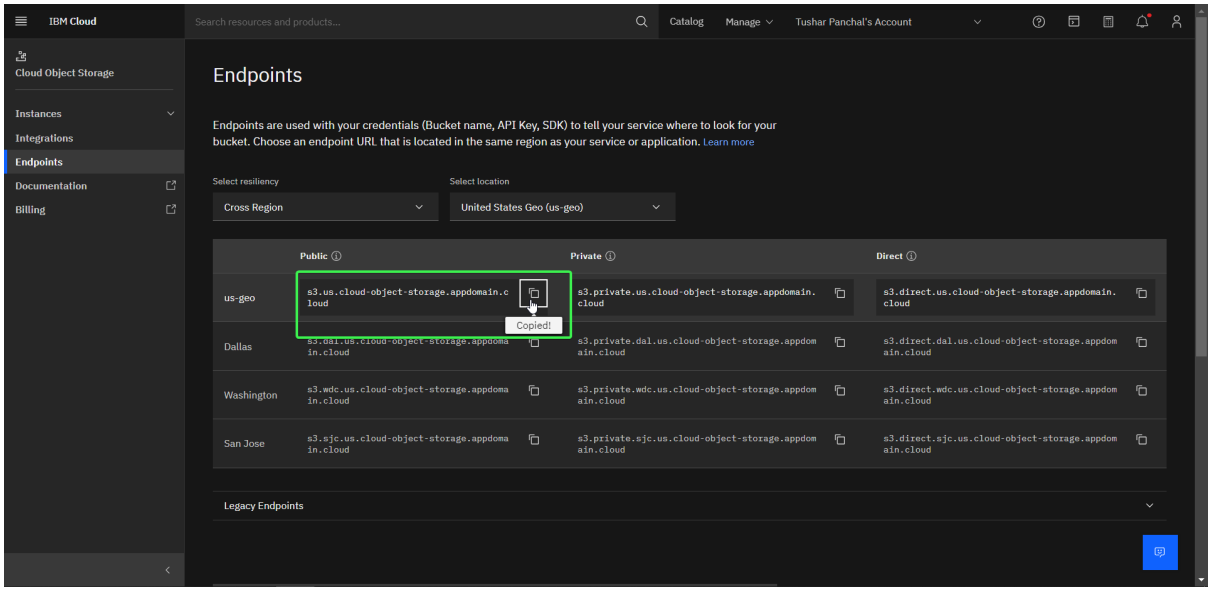
- **Step 12 :** Now, for object storage part, create new roles as specified with HMAC credentials in existing instance : Manager, Reader, Object Writer.


IBM Cloud
Cloud Object Storage
Instances
Integrations
Endpoints
Documentation
Billing
Cloud Object Storage-tushar
Create Credentials
Name:
CS_PRACTICAL_3_MANAGER
Role:
Manager
Select Service ID (Optional)
Auto Generated
Include HMAC Credential
On
Cancel
Add

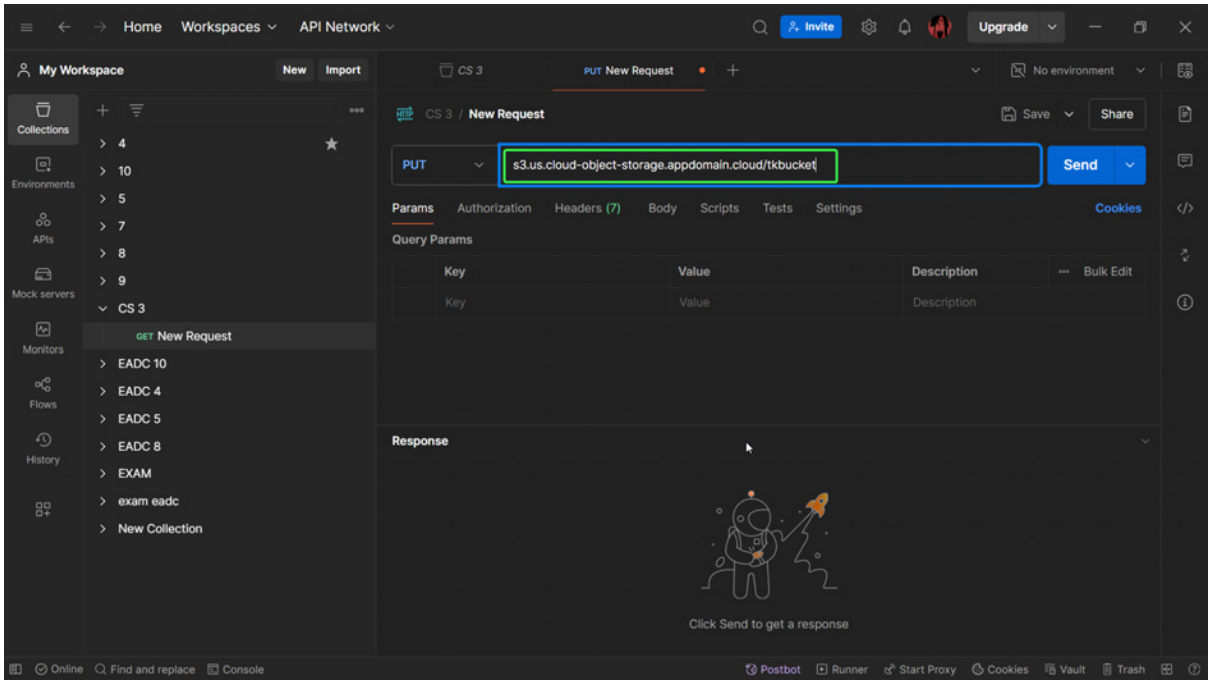

IBM Cloud
Cloud Object Storage
Instances
Integrations
Endpoints
Documentation
Billing
Cloud Object Storage-tushar
Create Credentials
Name:
CS_PRACTICAL_3_READER
Role:
Reader
Select Service ID (Optional)
Auto Generated
Include HMAC Credential
On
Cancel
Add


IBM Cloud
Cloud Object Storage
Instances
Integrations
Endpoints
Documentation
Billing
Cloud Object Storage-tushar
Create Credentials
Name:
CS_PRACTICAL_3_WRITER
Role:
Writer
Select Service ID (Optional)
Auto Generated
Include HMAC Credential
On
Cancel
Add

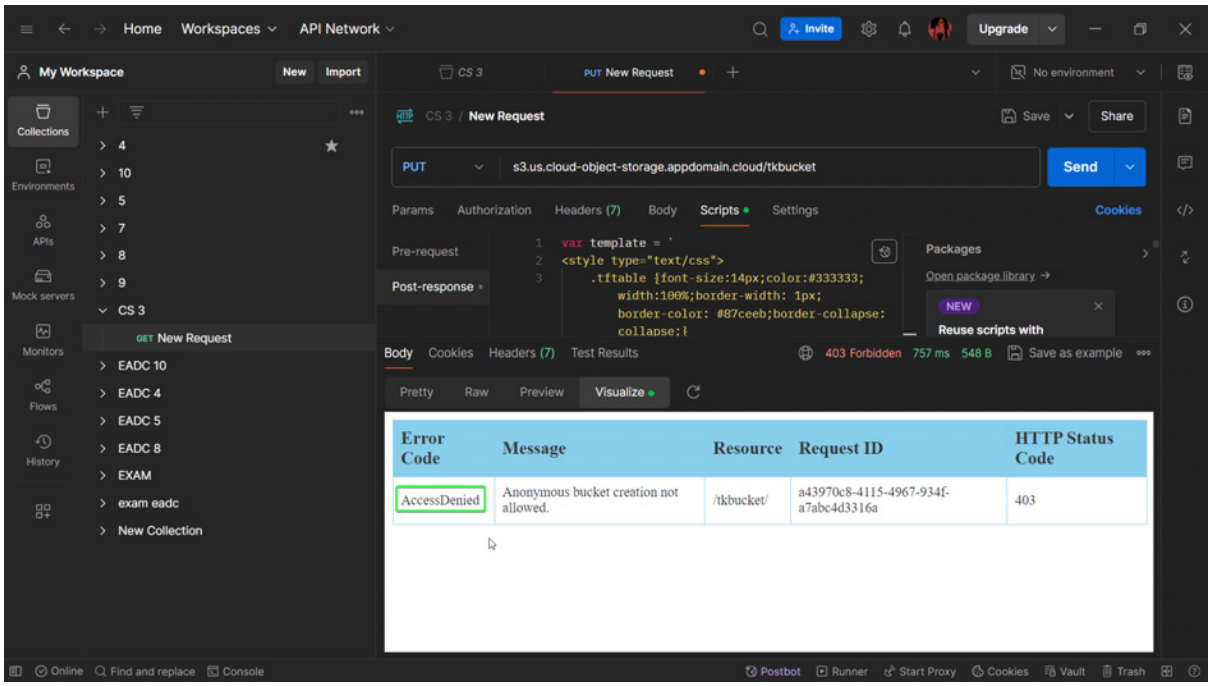
Step 13 : Copy the location URL to access through third party apps like Postman.



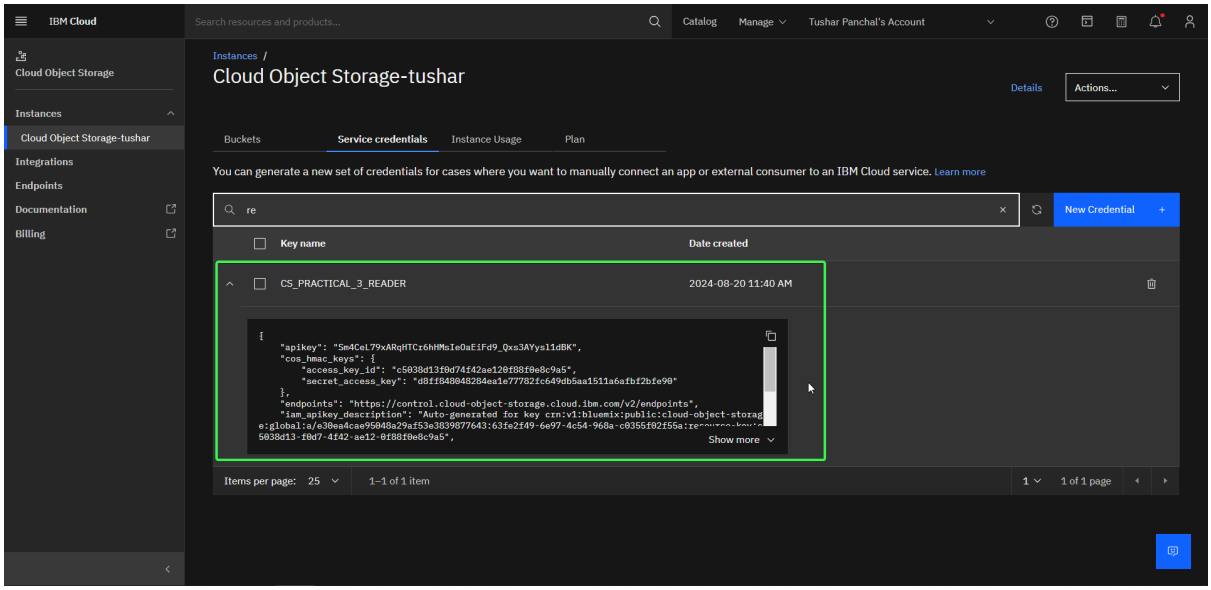
Step 14 : Try creating new bucket through post request on copied URL/bucketname.



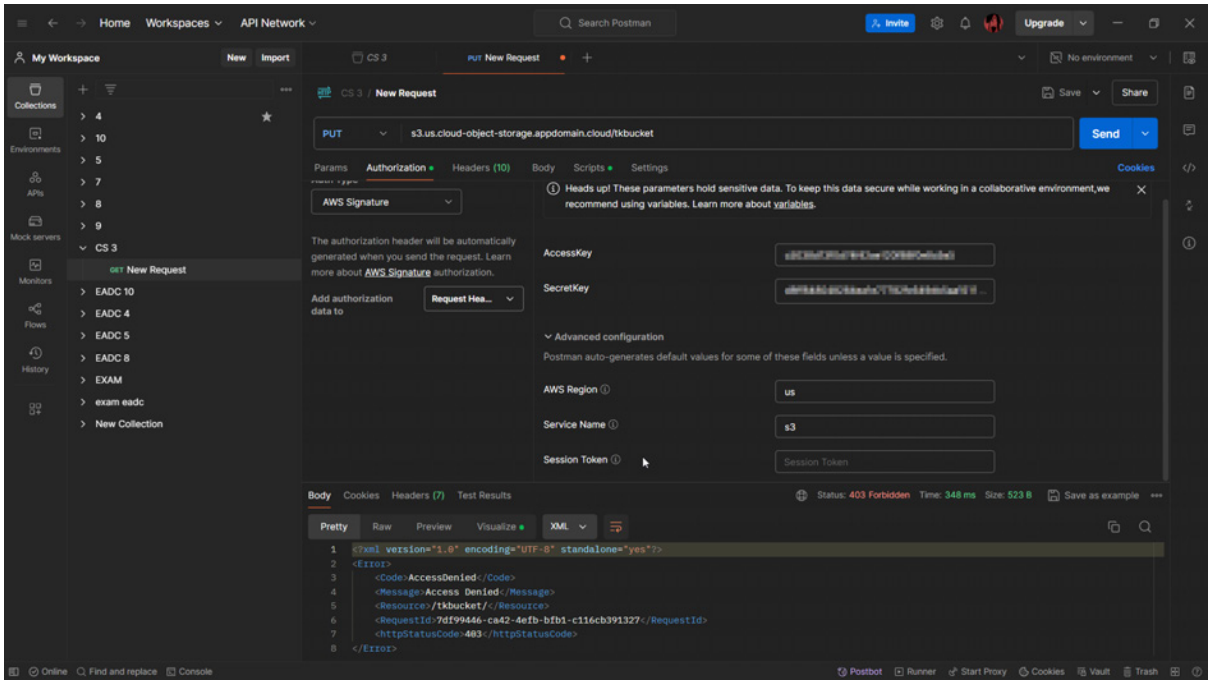
Access is denied because IBM cloud security prevents unauthorized access



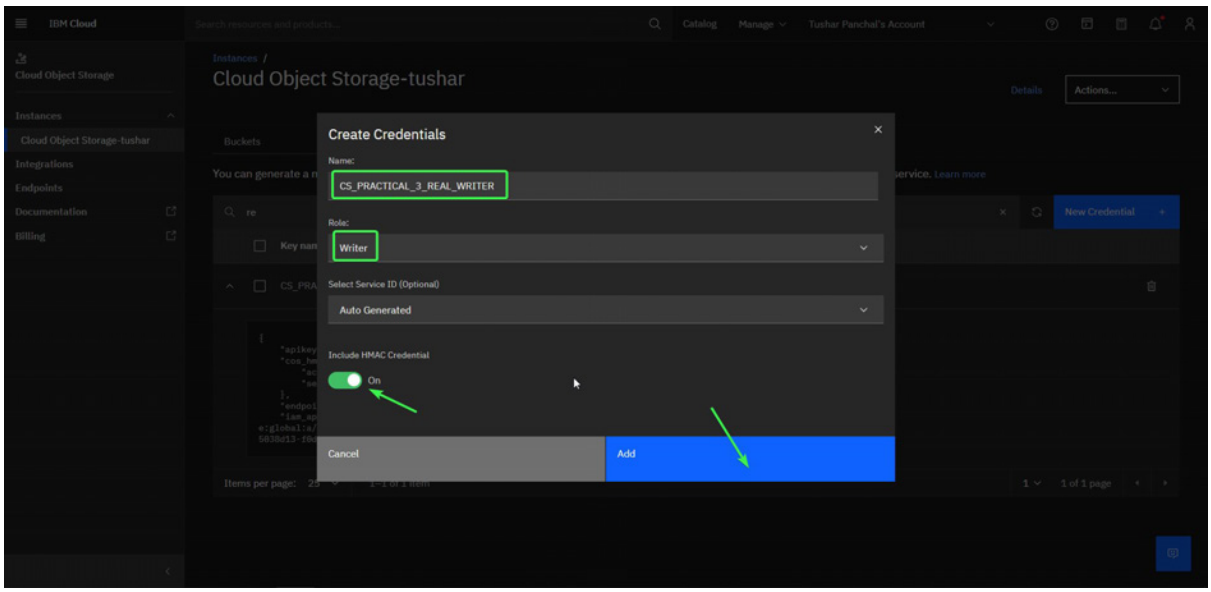
Step 15 : It should also fail for only reader role.

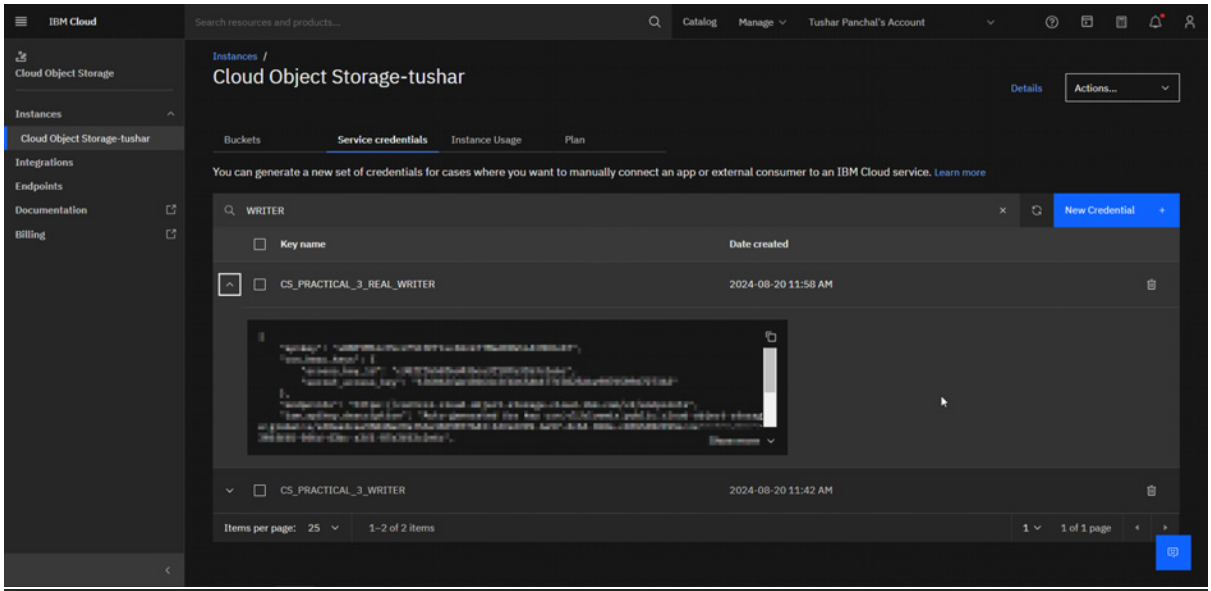


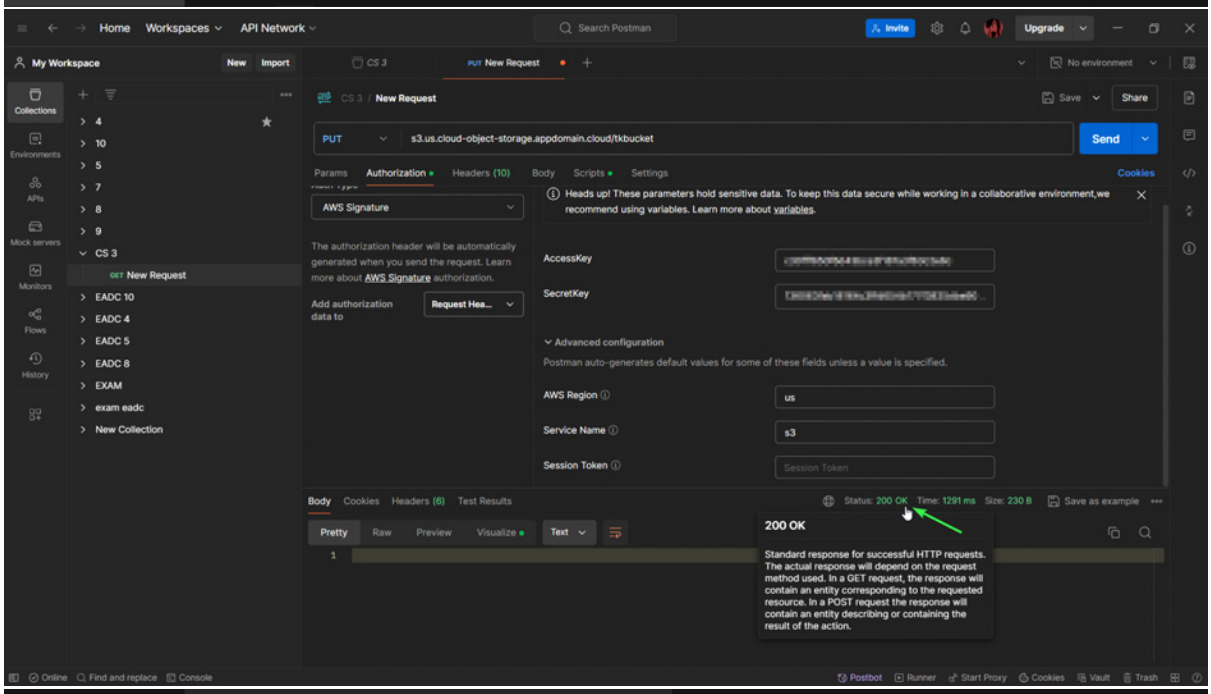
Authorization should be of type AWS Signature for integration of apikey and user password for IBM Cloud Object Storage service

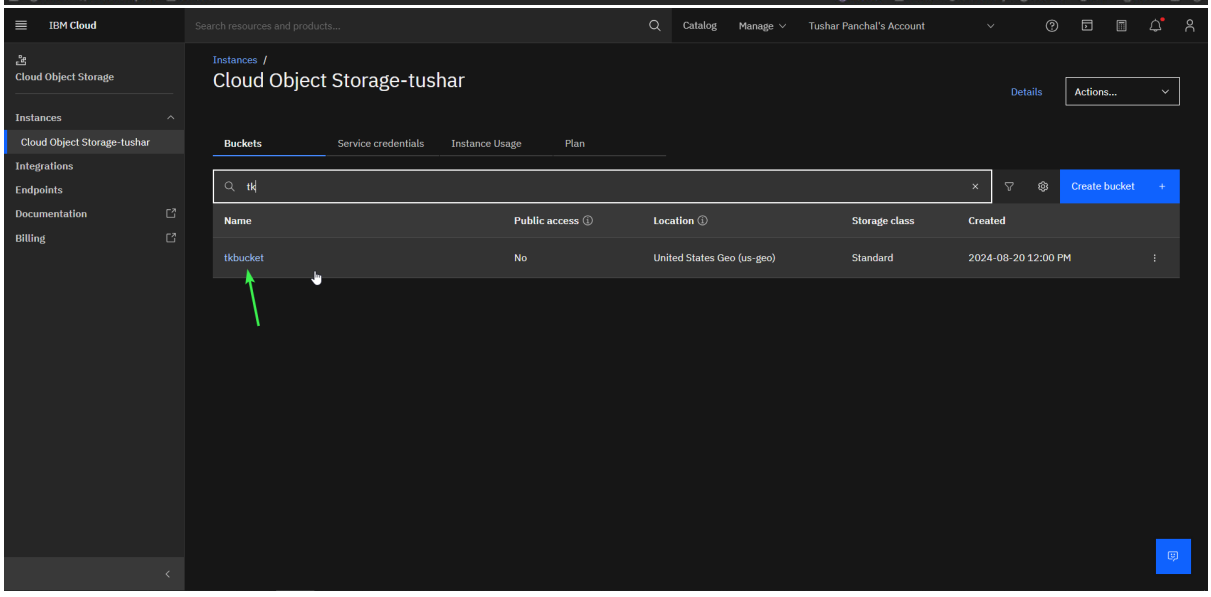


Step 16 : Create Writer role and check, the new bucket can be successfully created.



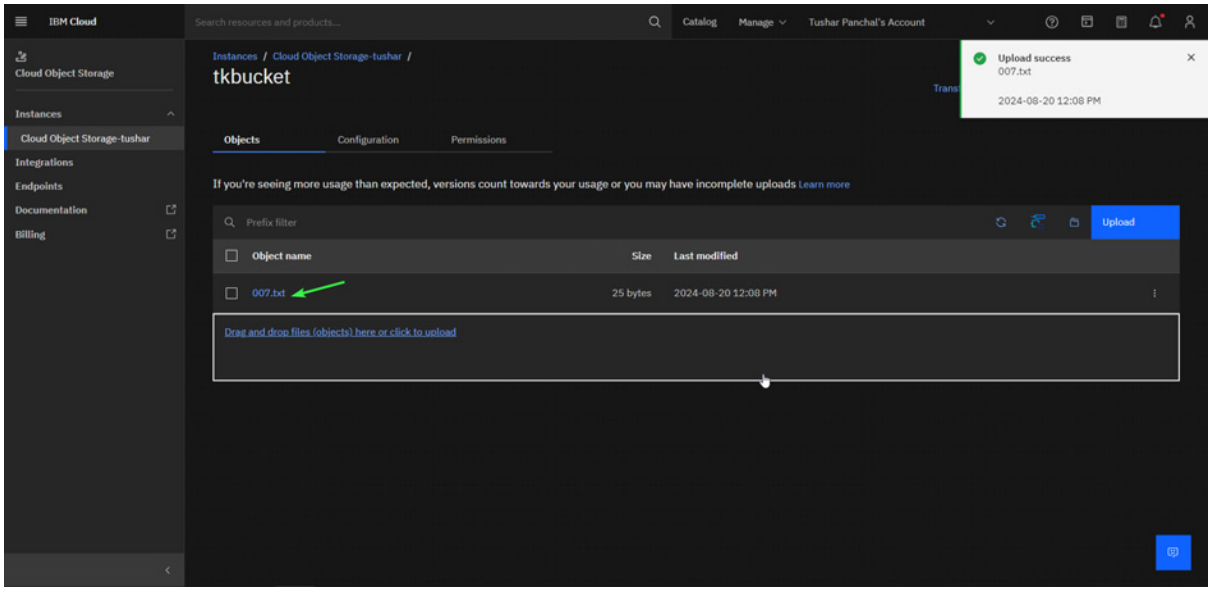

IBM Cloud
Cloud Object Storage
Instances
Cloud Object Storage-tushar
Integrations
Endpoints
Documentation
Billing
Cloud Object Storage-tushar
Buckets
Service credentials
Instance Usage
Plan
You can generate a new set of credentials for cases where you want to manually connect an app or external consumer to an IBM Cloud service. Learn more
WRITER
New Credential
Key name
Date created
CS_PRACTICAL_3_REAL_WRITER
2024-08-20 11:58 AM
CS_PRACTICAL_3_WRITER
2024-08-20 11:42 AM
Items per page: 25
1–2 of 2 items
1 of 1 page


Home
Workspaces
API Network
Search Postman
Invite
Upgrade
My Workspace
New
Import
CS 3
New Request
Save
Share
PUT
s3.us.cloud-object-storage.appdomain.cloud/tkbucket
Send
Params
Authorization
Headers (10)
Body
Scripts
Settings
Cookies
AWS Signature
Heads up! These parameters hold sensitive data. To keep this data secure while working in a collaborative environment,we recommend using variables. Learn more about variables.
The authorization header will be automatically generated when you send the request. Learn more about AWS Signature authorization.
AccessKey
SecretKey
Add authorization data to
Request Hea...
Advanced configuration
Postman auto-generates default values for some of these fields unless a value is specified.
AWS Region
us
Service Name
s3
Session Token
Body
Cookies
Headers (6)
Test Results
Status: 200 OK
Time: 1291 ms
Size: 230 B
Save as example
Pretty
Raw
Preview
Visualize
Text
200 OK
Standard response for successful HTTP requests. The actual response will depend on the request method used. In a GET request, the response will contain an entity corresponding to the requested resource. In a POST request the response will contain an entity describing or containing the result of the action.
Collections
Environments
APIs
Mock servers
Monitors
Flows
History
CS 3
New Request
EADC 10
EADC 4
EADC 5
EADC 8
EXAM
exam eadc
New Collection

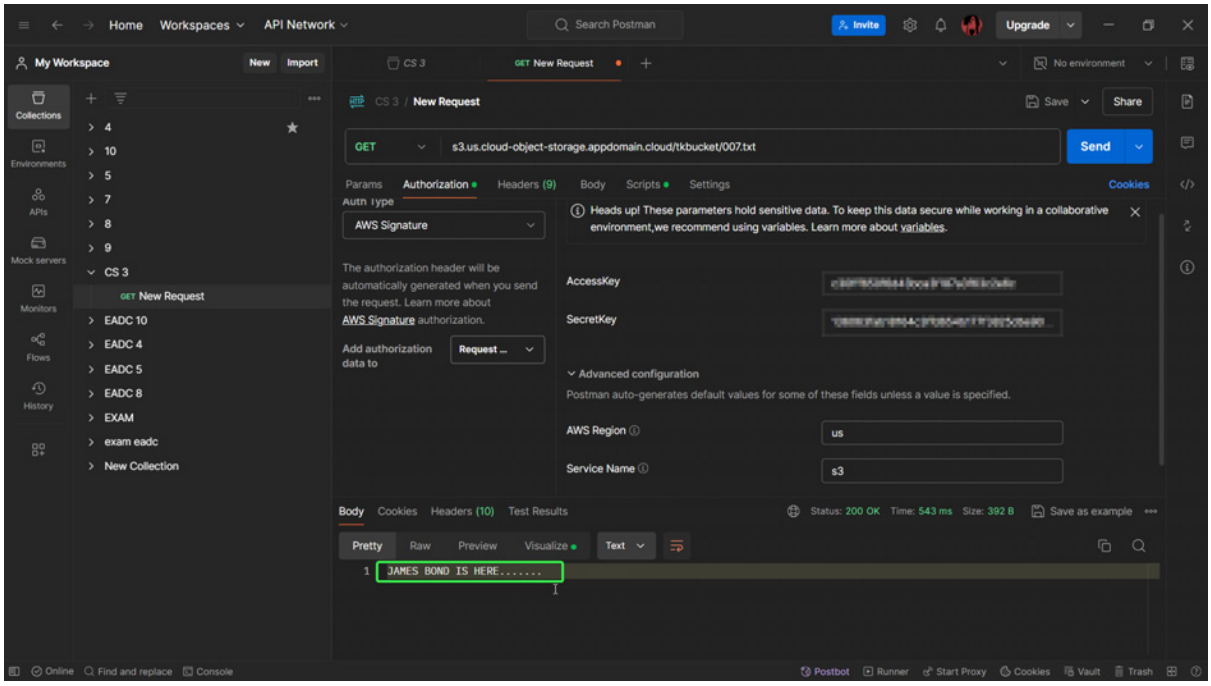

IBM Cloud
Cloud Object Storage
Instances
Cloud Object Storage-tushar
Integrations
Endpoints
Documentation
Billing
Cloud Object Storage-tushar
Buckets
Service credentials
Instance Usage
Plan
tk
Create bucket
Name
Public access
Location
Storage class
Created
tkbucket
No
United States Geo (us-geo)
Standard
2024-08-20 12:00 PM

As you can see above the tkbucket is created,it can only created by writer or manager role.

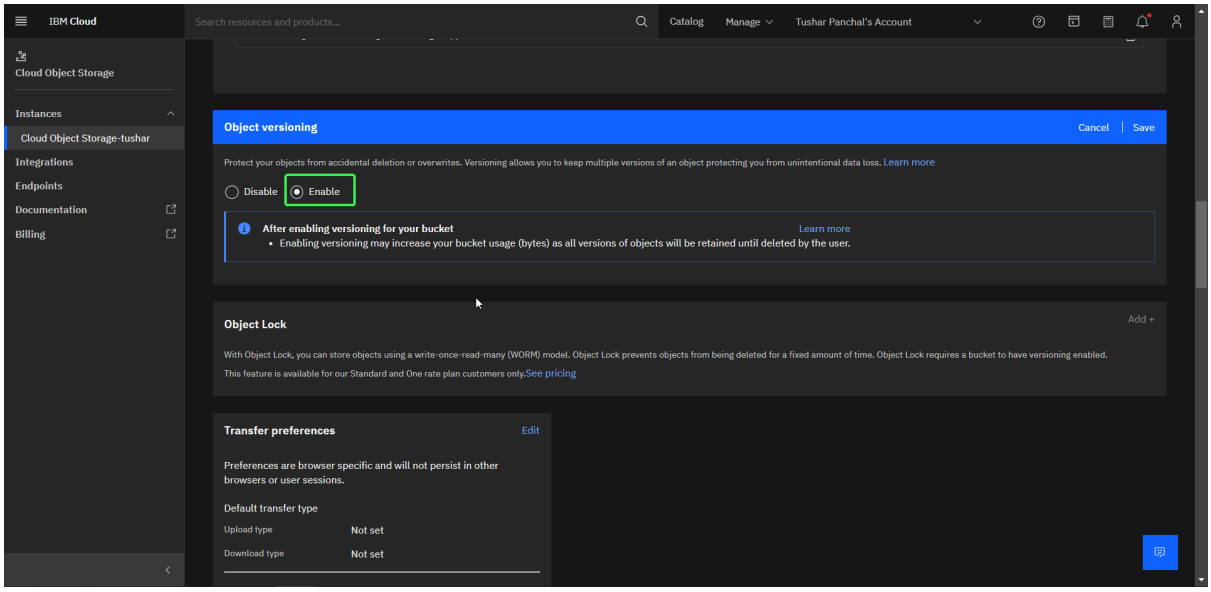
- **Step 17 :** Add any file as object in the new bucket (say text file here).



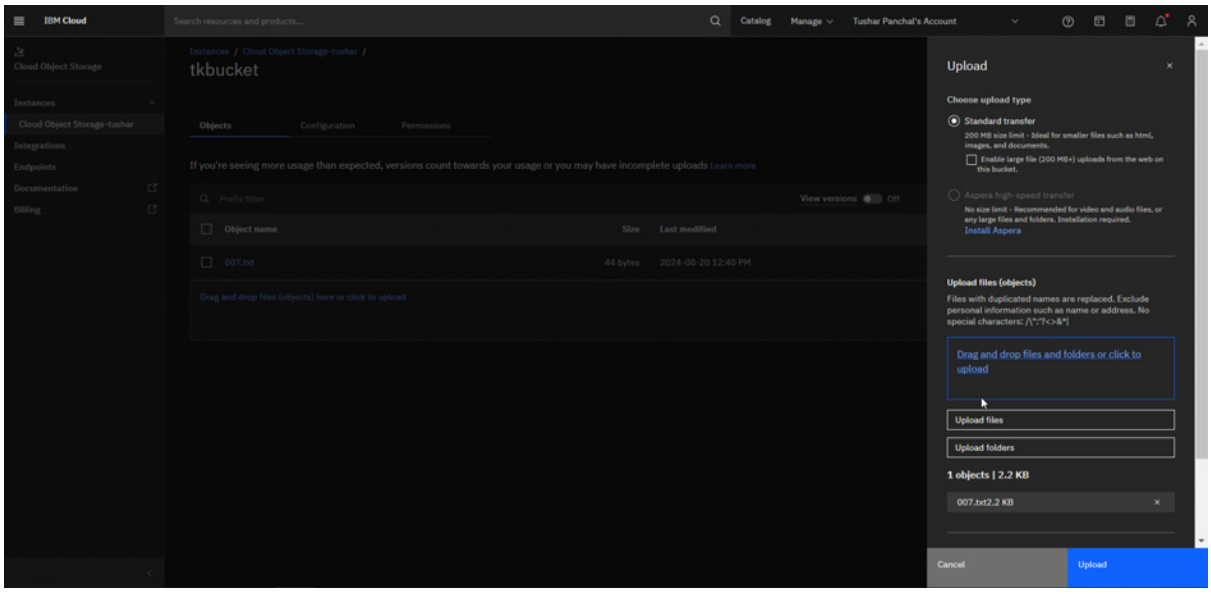
- **Step 18 :** Try getting its contents via GET request on URL/bucket/object.



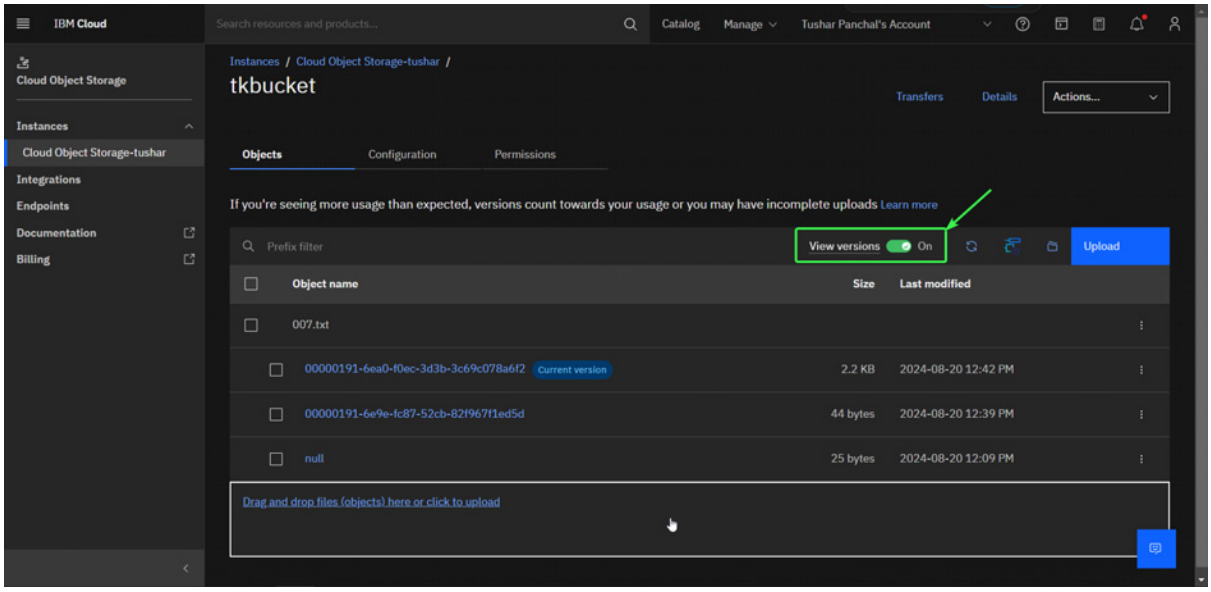
Step 19 : Now, if multiple versions of same object is needed, it can be enabled in configuration.



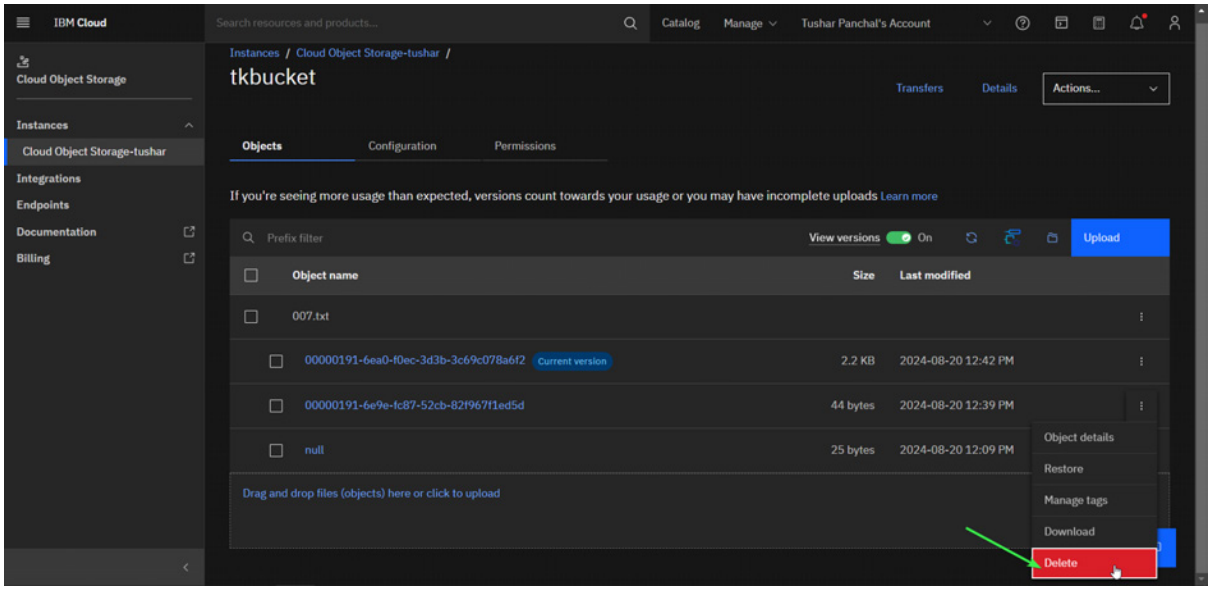
Step 20 : Try uploading changed file with same name and it should be allowed now.



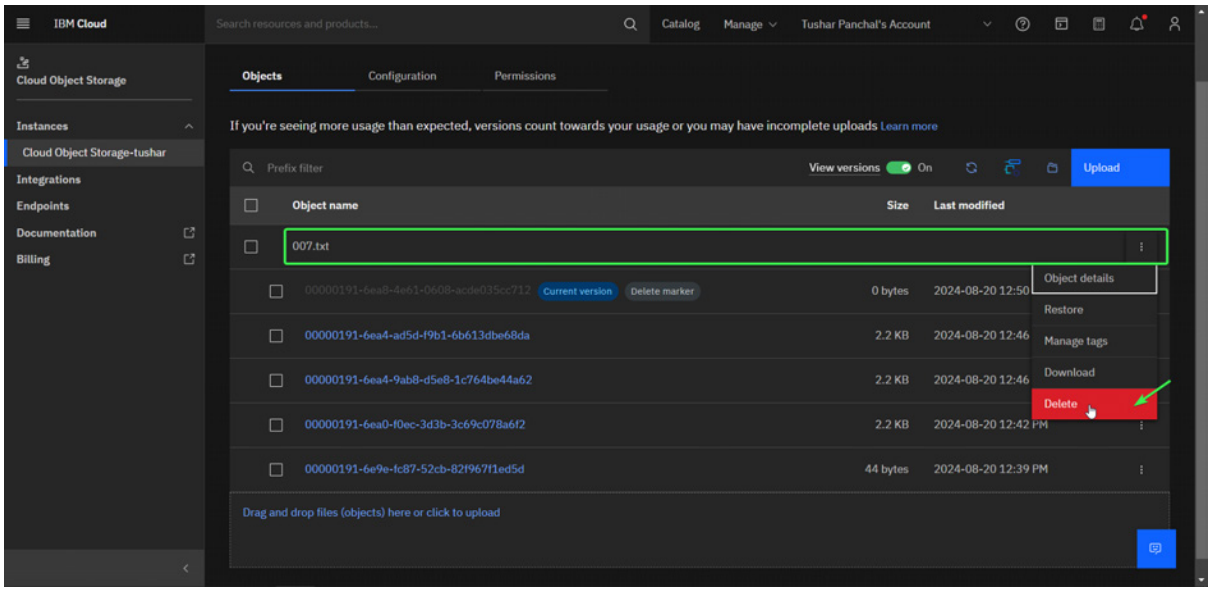
» **Step 21 :** View Versions button can be used to show files with all versions including null (initial version).

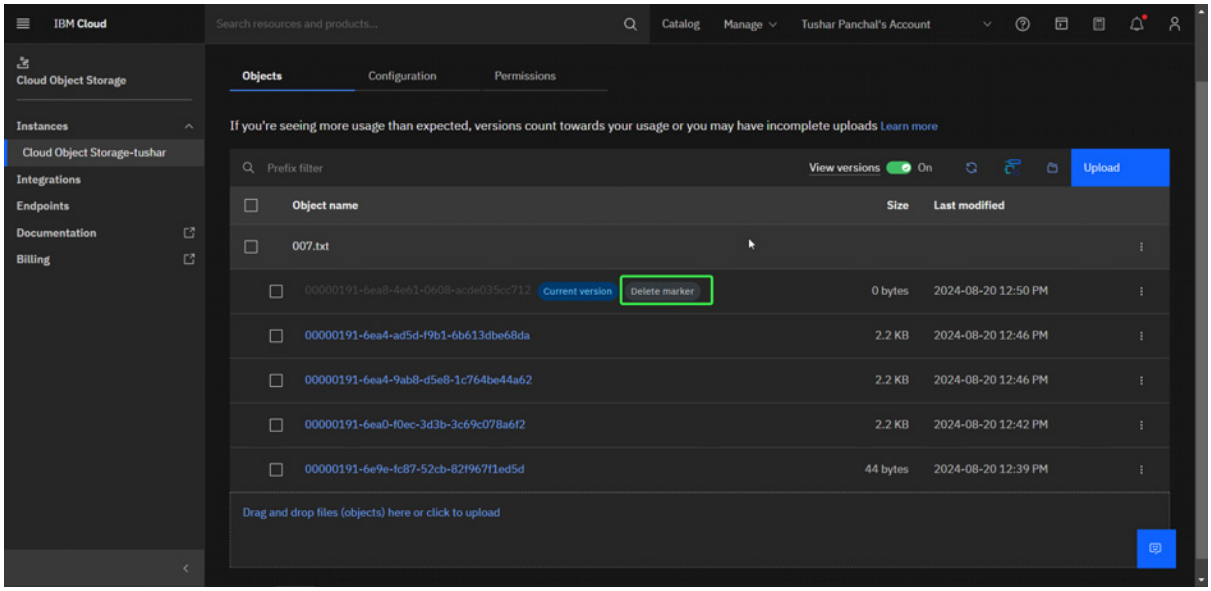


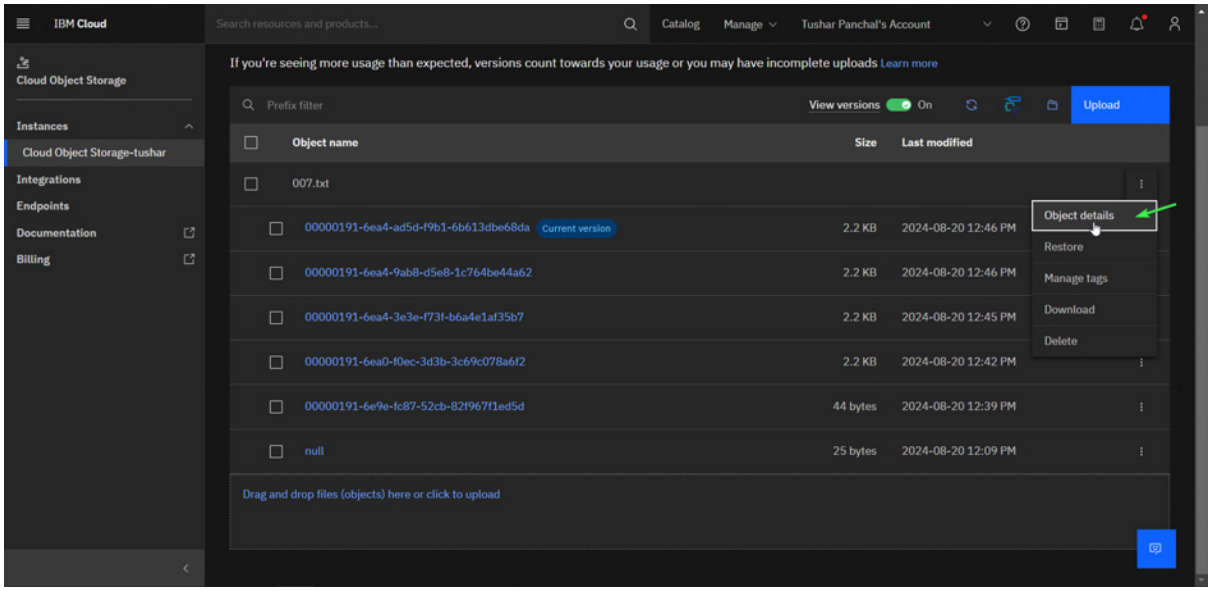
» **Step 22 :** Try deleting the versions and changing default one.

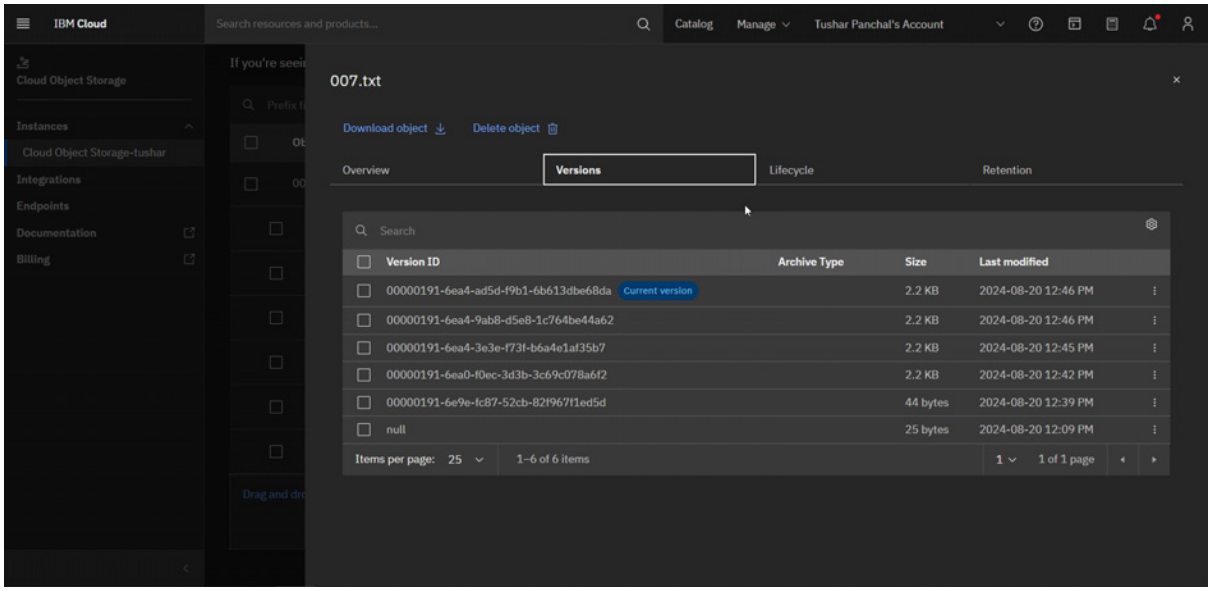


» **Step 23 :** Also try deleting the whole object, then it will not be permanently deleted, but delete marker will be set as current version.

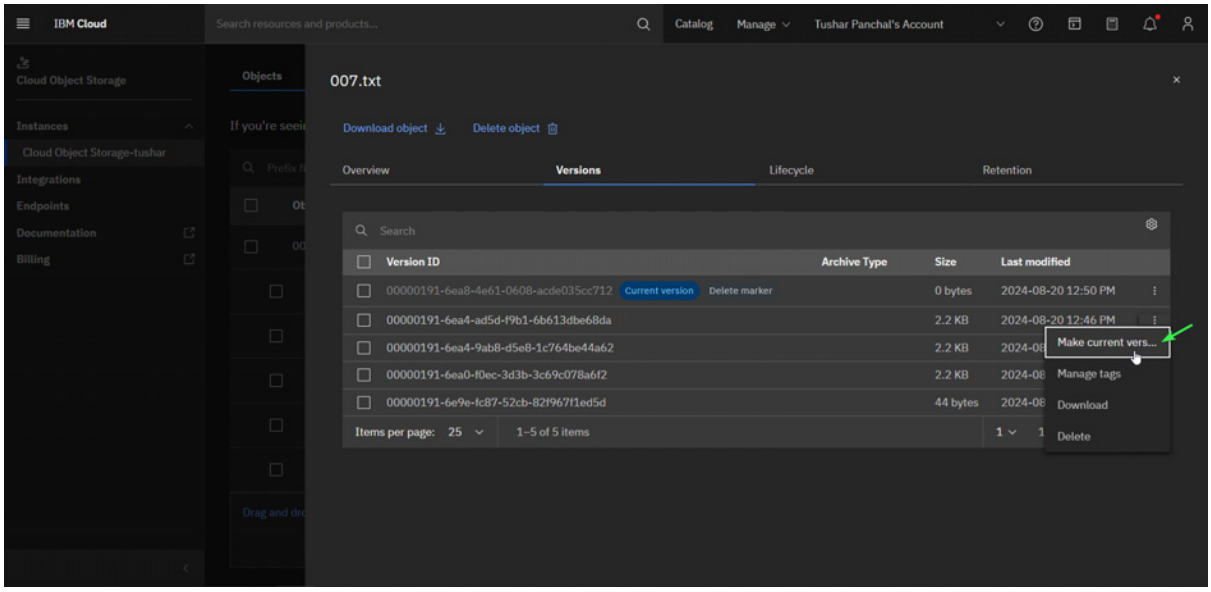


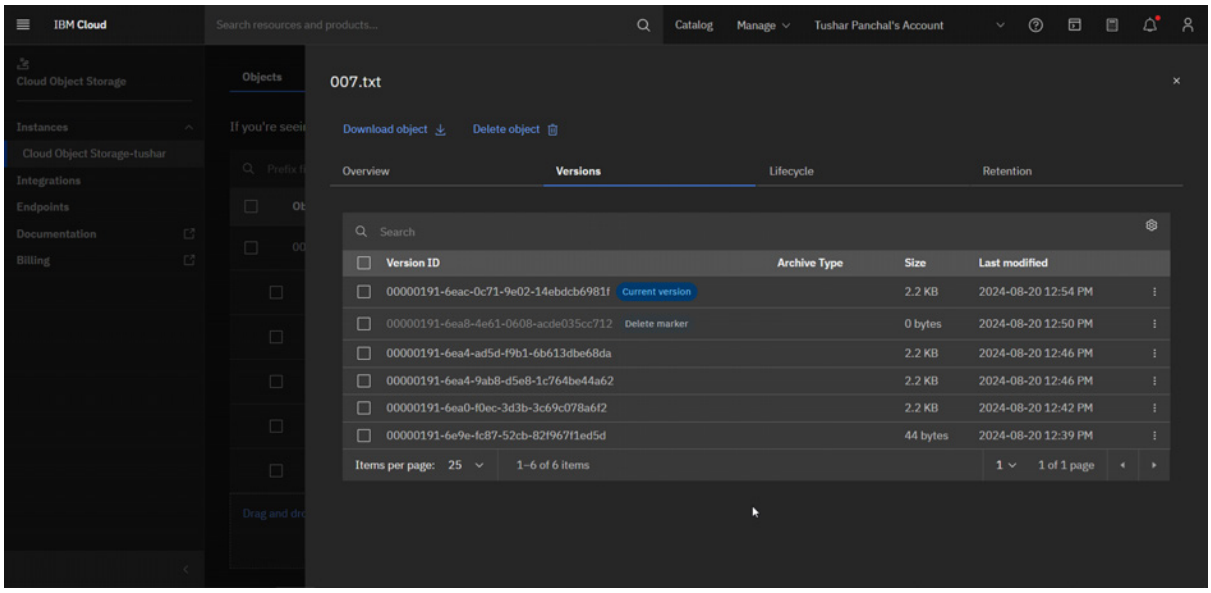



IBM Cloud
Search resources and products...
Catalog
Manage
Tushar Panchal's Account
Cloud Object Storage
Instances
Cloud Object Storage-tushar
Integrations
Endpoints
Documentation
Billing
If you're seeing more usage than expected, versions count towards your usage or you may have incomplete uploads Learn more
Prefix filter
View versions On
Upload
Object name
Size
Last modified
007.txt
00000191-6ea4-ad5d-f9b1-6b613dbe68da Current version 2.2 KB 2024-08-20 12:46 PM
00000191-6ea4-9ab8-d5e8-1c764be44a62 2.2 KB 2024-08-20 12:46 PM
00000191-6ea4-3e3e-f73f-b6a4e1af35b7 2.2 KB 2024-08-20 12:45 PM
00000191-6ea0-f0ec-3d3b-3c69c078a6f2 2.2 KB 2024-08-20 12:42 PM
00000191-6e9e-fc87-52cb-82f967f1ed5d 44 bytes 2024-08-20 12:39 PM
null 25 bytes 2024-08-20 12:09 PM
Object details
Restore
Manage tags
Download
Delete
Drag and drop files (objects) here or click to upload

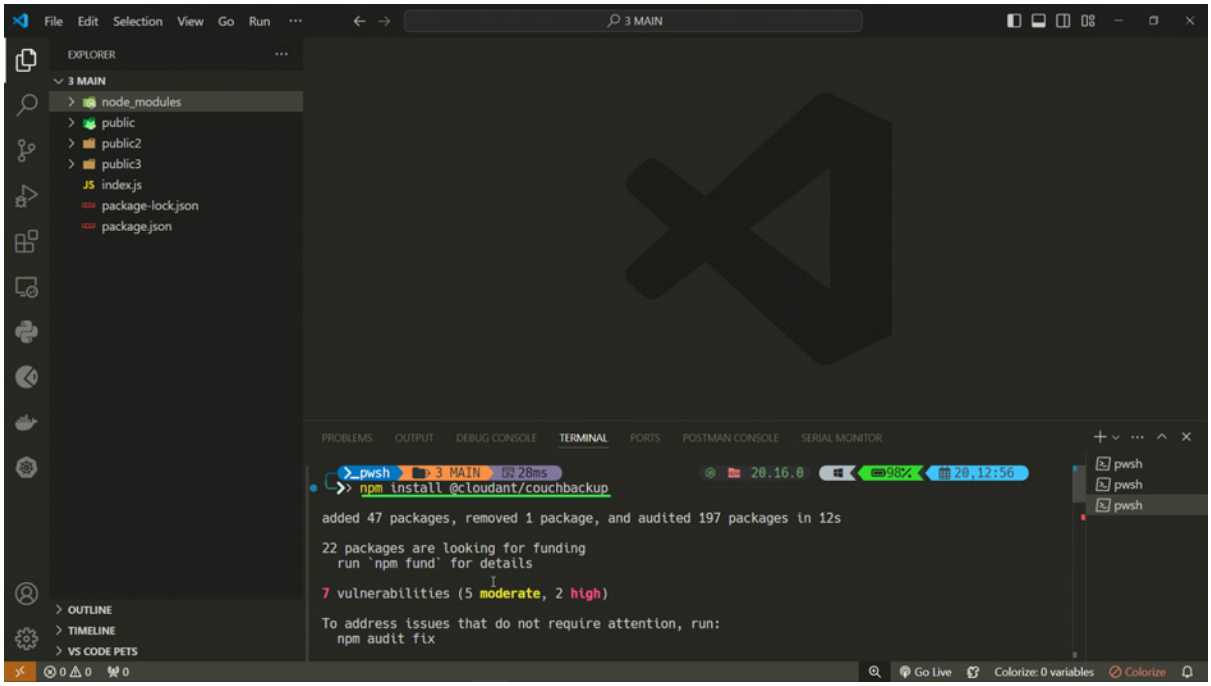

IBM Cloud
Search resources and products...
Catalog
Manage
Tushar Panchal's Account
Cloud Object Storage
Instances
Cloud Object Storage-tushar
Integrations
Endpoints
Documentation
Billing
007.txt
Download object
Delete object
Overview
Versions
Lifecycle
Retention
Search
Version ID
Archive Type
Size
Last modified
00000191-6ea4-ad5d-f9b1-6b613dbe68da Current version 2.2 KB 2024-08-20 12:46 PM
00000191-6ea4-9ab8-d5e8-1c764be44a62 2.2 KB 2024-08-20 12:46 PM
00000191-6ea4-3e3e-f73f-b6a4e1af35b7 2.2 KB 2024-08-20 12:45 PM
00000191-6ea0-f0ec-3d3b-3c69c078a6f2 2.2 KB 2024-08-20 12:42 PM
00000191-6e9e-fc87-52cb-82f967f1ed5d 44 bytes 2024-08-20 12:39 PM
null 25 bytes 2024-08-20 12:09 PM
Items per page: 25
1–6 of 6 items
1 of 1 page

» **Step 24 :** Now, change default version to previously used ones to restore object.





- **Step 25 :** Now,install node package @cloudant/couchbackup.



- **Step 26 :** Copy the main URL from service credentials of Cloudant instance and use couchbackup command (here local project node module bin.js command, couchbackup without extension command if globally installed) to backup the database to a plain file.

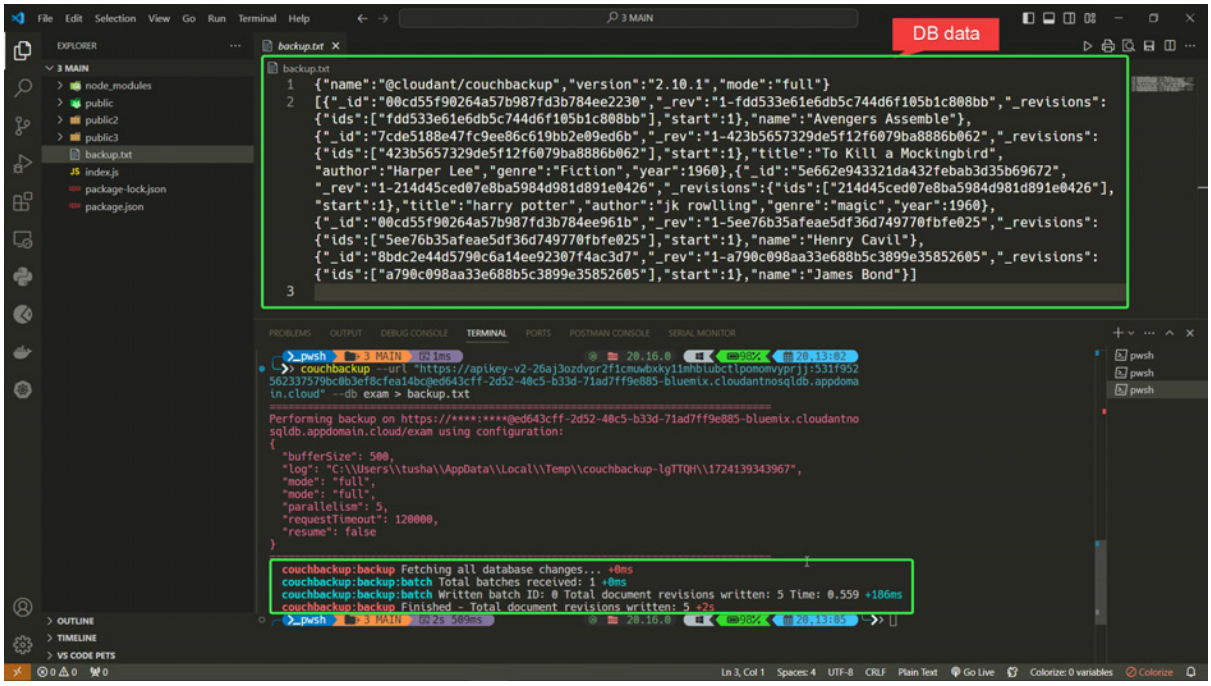
Commands :

a. For local installation inside project :

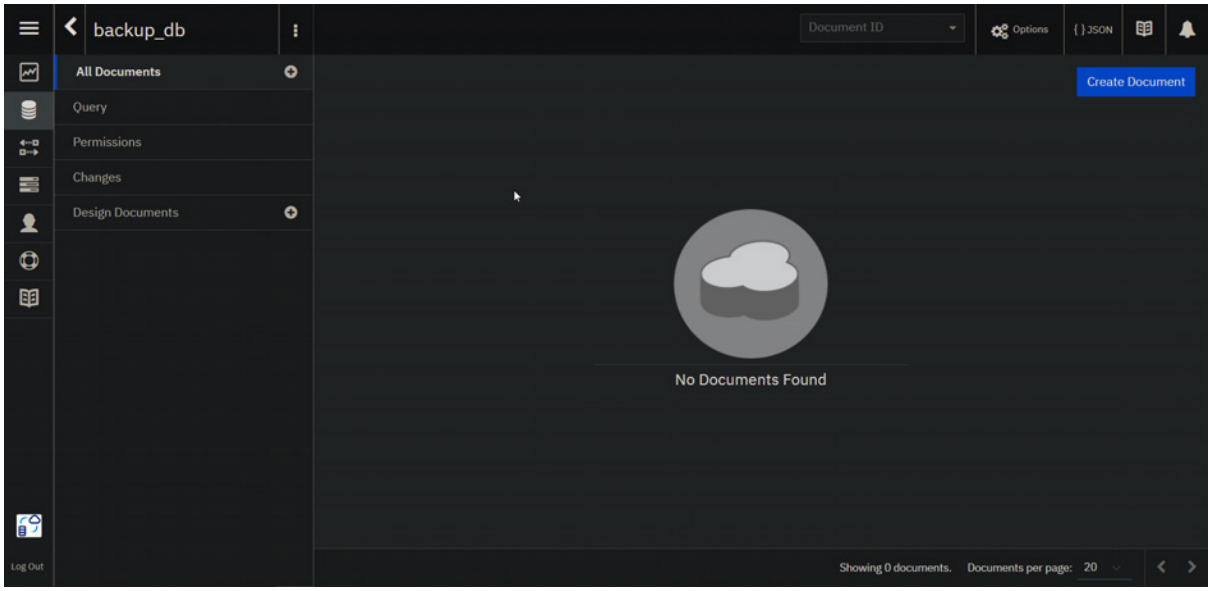
```
./path_to_project/node_modules/@cloudant/couchbackup/bin/couchbackup
.bin.js --url “URL here” --db databasename > textfile
```

b. For global installation of couchbackup node module :

```
couchbackup --url “URL here” --db databasename > textfile
```



» **Step 27 :** Now, create new empty database and load data from backup file to it.



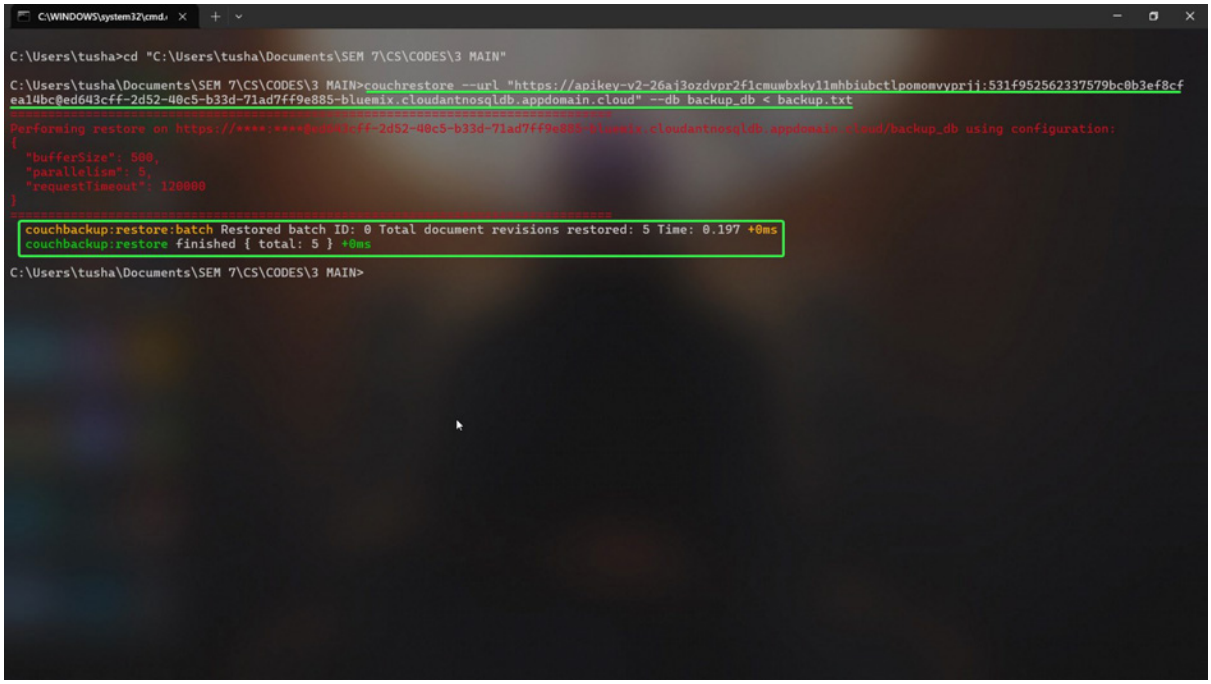
Commands :

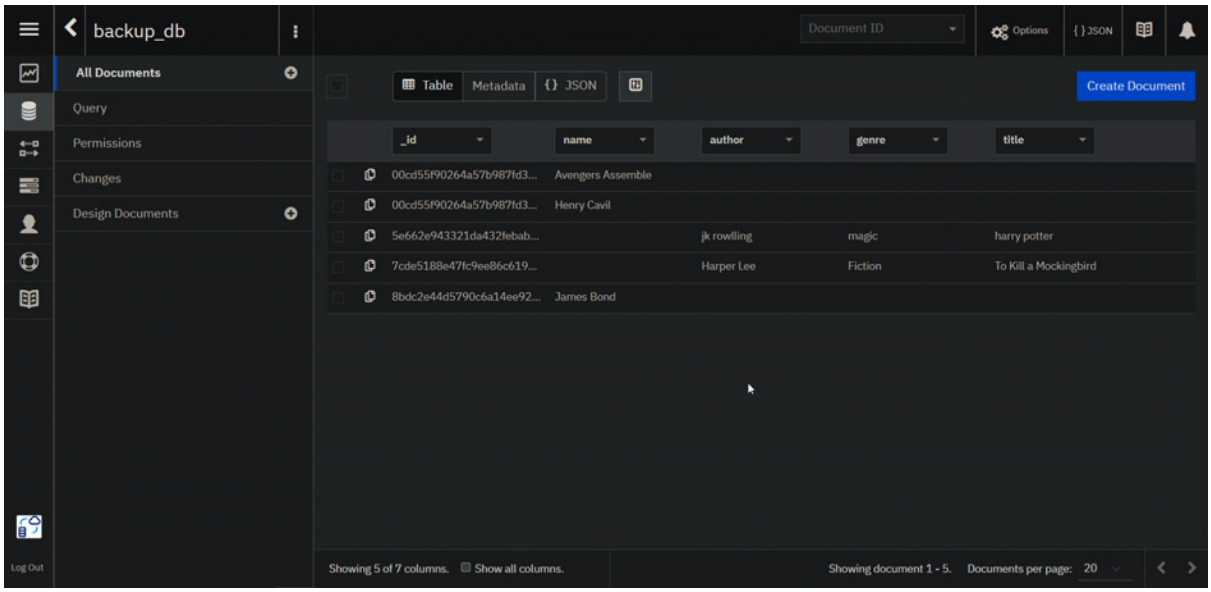
a. For local installation :

```
./path_to_project/node_modules/@cloudant/couchbackup/bin/couchbackup.bin.js --url “URL here” --db databasename > textfile
```

b. For global installation of couchbackup node module :

```
couchbackup --url “URL here” --db databasename < textfile
```





As you can see in above our backup is restored in backup_db database.