



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.

The screenshot shows the IBM Cloudant dashboard with the 'Account' tab selected. On the left, there's a sidebar with links like Monitoring, Databases, Replication, Active Tasks, Account (which is highlighted), Support, and Documentation. The main content area has a header 'Account' and tabs for Capacity, Announcements, CORS (which is underlined in blue), and Settings. A message states: 'Cross-Origin Resource Sharing (CORS) lets you connect to remote servers directly from the browser. Using CORS, you can host browser-based apps on static pages, and load data directly from Cloudant.' Below this, it says 'CORS is currently disabled.' and features a prominent blue 'Enable CORS' button. A green status bar at the top right indicates 'CORS settings updated.'

The screenshot shows the IBM Cloudant CORS settings page. On the left, there's a sidebar with navigation links: Monitoring, Databases, Replication, Active Tasks, Account (which is selected), Support, and Documentation. Below the sidebar is the IBM Cloudant logo and a log out link.

The main content area has a header "Account" and tabs for Capacity, Announcements, CORS (which is selected), and Settings. A success message "CORS settings updated." is displayed at the top right. Below it, a note explains what CORS is and states that it is currently enabled. There is a "Disable CORS" button.

The "Origin Domains" section allows selecting domains from which requests will be accepted. The "All domains (*)" option is selected, and there is also a "Restrict to specific domains" option.

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

The screenshot shows the IBM Cloud Service credentials page for a service named "Cloudant-dj". The "Service credentials" tab is selected. It displays a single credential entry for "Service credentials-1tk". The details show the key name, date created (2024-03-26 8:00 AM), and the service it's controlled by (Cloudant). The credential itself is a JSON object containing various parameters like host, port, URL, and user information.

✓ 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.

Cloudant CORS Example

```

Fetch Tushar's Databases
[ "exam",
  "jndbdata_tushar",
  "novel-tushar",
  "tk",
  "tushar_practical_10"
]

```

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

Account

Capacity Announcements CORS Settings

Cross-Origin Resource Sharing (CORS) lets you connect to remote servers directly from the browser. Using CORS, you can host browser-based apps on static pages, and load data directly from Cloudant.

CORS is currently enabled.

[Disable CORS](#)

Origin Domains

Databases will accept requests from these domains:

All domains (*)
 Restrict to specific domains

[Add Domain](#)

IBM Cloudant

Log Out IBMid-693000SXSI

Now if we try to get it with another domain it will get error:
 (Unauthorized error due to restricted IP access)

Cloudant CORS Example

Fetch Tushar's Databases

Fetch error: Failed to fetch

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

The screenshot shows the IBM Cloudant interface. On the left, there's a sidebar with icons for Monitoring, Databases (which is selected), Replication, Active Tasks, Account, Support, and Documentation. Below the sidebar is the IBM Cloudant logo. The main area is titled 'Databases' and lists five databases: 'exam', 'imdbdata_tushar', 'novel-tushar', 'tk', and 'tushar_practical_10'. Each database row has columns for Name, Size, # of Docs, Partitioned, and Actions (with icons for edit, lock, and delete). A cursor is hovering over the 'exam' database. At the bottom, there's a status bar with 'Showing 1–5 of 5 databases', 'Databases per page 20', and navigation arrows.

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

The screenshot shows the 'Permissions' section for the 'exam' database. The sidebar on the left is identical to the previous screenshot. The main area shows 'Cloudant users and API keys with permissions on exam.' It lists a single user 'ed643cff-2d52-40c5-b33d-71ad7ff9e885-bluemix' with checkboxes for '_admin', '_reader', '_writer', and '_replicator' all checked. Below this is a section for 'Unauthenticated connections' with empty checkboxes. At the bottom, there's a 'Grant database permissions to:' input field with 'Username or API Key' and a 'Grant Permissions' button. A callout box points to the 'Generate API Key' button, which is highlighted with a green arrow. The status bar at the bottom is identical to the previous screenshot.

The screenshot shows the IBM Cloudant interface. On the left sidebar, 'Databases' is selected. The main view displays the 'exam' database with its document count (All Documents: 1). A green arrow points to the 'apikey-868b7b88ecae47299b274427601aa31' row in the 'Permissions' table, which has checkboxes for '_admin', '_reader', and '_writer' roles. A green box highlights the '_reader' and '_writer' checkboxes. A success message at the top right says 'Database permissions have been saved.' Below the table, there's a note about API keys and a 'Generate API Key' button. A message at the bottom indicates the API key will expire in 4:50.

		_admin	_reader	_writer	_replicator
ed643cff-2d52-40c5-b33d-71ad7ff9e885-bluemix	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>
apikey-868b7b88ecae47299b274427601aa31	<input checked="" type="checkbox"/>	<input checked="" 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-868b7b88ecae47299b274427601aa31
Password: edfe811c3ab991576d35cd29f8838c9b8adda124

This message will expire in 4:50. Please make a note of the API key password; for security reasons, the password cannot be shown again.

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

Cloudant API Example

Fetch Databases

```
{
  "total_rows": 2,
  "offset": 0,
  "rows": [
    {
      "id": "5e662e943321da432febab3d35b69672",
      "key": "5e662e943321da432febab3d35b69672",
      "value": {
        "rev": "1-214d45ced07e8ba5984d981d891e0426"
      }
    },
    {
      "id": "5e662e943321da432febab3d35b69672",
      "key": "5e662e943321da432febab3d35b69672",
      "value": {
        "rev": "1-214d45ced07e8ba5984d981d891e0426"
      }
    }
  ]
}
```

» **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).

	_admin	_reader	_writer	_replicator
ed643cff-2d52-40c5-b33d-71ad7ff9e885-bluemix	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>
apikey-868b7b88cae47299b2744276011aa31	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>

Cloudant API Example

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

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

The screenshot shows the IBM Cloudant interface. On the left, there's a sidebar with options: Monitoring, Databases (selected), Replication, Active Tasks, Account, Support, and Documentation. Below the sidebar is the IBM Cloudant logo and log out information. The main area is titled "Databases" and shows a list of existing databases: exam, imbddata_tushar, novel-tushar, tk, and tushar_practical_10. To the right of the list is a "Create Database" dialog box. It has a "Database name" field containing "testDB", a "Partitioning" section with a radio button for "Non-partitioned - recommended for most workloads" (which is selected), and a "Which should I choose?" link. At the bottom of the dialog are "Cancel" and "Create" buttons, with a green arrow pointing to the "Create" button.

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

The screenshot shows the "test_db" database details page. The sidebar on the left has "Permissions" selected. The main area displays a table of "Cloudant users and API keys with permissions on test_db". It includes columns for "_admin", "_reader", "_writer", and "_replicator". A row for "ed643cff-2d52-40c5-b33d-71ad7ff9e885-bluemix" has all checkboxes checked. Another row for "Unauthenticated connections" has all checkboxes unchecked. Below the table, there's a "Grant database permissions to:" input field containing "apikey-868b7b88ecae47299b274427601aa31" and a "Grant Permissions" button. A green arrow points to the "Grant Permissions" button. At the bottom, there's a note about API keys and a "Generate API Key" button.

Make sure to edit permissions:

The screenshot shows the "test_db" database details page again, but this time the permissions for the "apikey-868b7b88ecae47299b274427601aa31" user have been modified. In the "Permissions" table, the checkboxes for "_writer" and "_replicator" are now checked for this user, while the others remain unchecked. A green box highlights the checked checkboxes for the user. The status message "Database permissions have been saved." is visible at the top right.

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

The screenshot shows the IBM Cloudant interface with two main panels: 'Replication' and 'Job Configuration'.

Replication Panel:

- Left sidebar: Monitoring, Databases, **Replication** (highlighted with a green box), Active Tasks, Account, Support, Documentation.
- Header: Replicator DB Activity, Polling Interval (5 minutes), Refresh, Bell icon.
- Table: 'Replications must have a replication document to display in the following table.' Headers: Source, Target, Start Time, Type, State, Actions. Subtext: 'There is no replicator-db activity or history to display.'
- Buttons: Filter replications, New Replication (highlighted with a blue box and a green arrow).

Job Configuration Panel:

- Left sidebar: Monitoring, Databases, **Replication** (highlighted with a green box), Active Tasks, Account, Support, Documentation.
- Header: Job Configuration, Log Out, IBMID-6930008XSI.
- Source Configuration (highlighted with a green box):

Type:	Local database
Name:	exam
Authentication:	Cloudant Username or API Key
.....	
For security purposes, the IBM Cloudant team recommends that you use IAM API keys or IBM Cloudant legacy authentication API keys rather than account-level credentials for replication jobs.	
- Target Configuration (highlighted with a green box):

Type:	Existing local database
Name:	test_db
Authentication:	Cloudant Username or API Key
.....	
For security purposes, the IBM Cloudant team recommends that you use IAM API keys or IBM Cloudant legacy authentication API keys rather than account-level credentials for replication jobs.	
- Target Configuration (highlighted with a green box):

Type:	Existing local database
Name:	test_db
Authentication:	Cloudant Username or API Key
.....	
For security purposes, the IBM Cloudant team recommends that you use IAM API keys or IBM Cloudant legacy authentication API keys rather than account-level credentials for replication jobs.	
- Options:

Replication type:	One time
Replication document:	Custom ID (optional)
Clear	
Start Replication (highlighted with a blue box and a green arrow)	

The screenshot shows the IBM Cloudant Replication interface. On the left sidebar, under the 'Replication' section, there is a link to 'Replicator DB Activity'. The main area is titled 'Replication' and shows a table of replication tasks. A single task is listed, with its source and target URLs highlighted by a green box. The source URL is <https://ed643cff-2d52-40c5-b33d-71ad7ff9e885-bluemix.cloudant.com/exam> and the target URL is https://ed643cff-2d52-40c5-b33d-71ad7ff9e885-bluemix.cloudant.com/test_db. A green banner at the top right indicates 'Replication from "exam" to "test_db" has been scheduled.' A blue button labeled 'New Replication' is visible.

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

The screenshot shows the 'New Document' interface for the 'exam' database. The left sidebar includes a 'Replication' section. The main area shows a JSON editor with a single document entry. The document ID is '8bd2e44d5790c6a14ee92307f4ac3d7', and it contains a single field 'name' with the value 'James Bond'. A green arrow points to the 'Create Document' button at the top of the editor.

✓ 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(result => {
            console.log(`Replicated document ${doc._id} to ${targetUrl}`);
        })
        .catch(error => {
            console.error(`Error replicating document ${doc._id}: ${error}`);
        });
}

```

```

        method: 'PUT',
        headers: headers,
        body: JSON.stringify(doc)
    );
}
)
.then(response => response.json())
.then(data => {
    if (data.ok) {
        document.getElementById('result').textContent +=`  
Replication success: ${doc._id}`;
    } else {
        document.getElementById('result').textContent +=`  
Replication error: ${JSON.stringify(data)}`;
    }
})
.catch(error => {
    document.getElementById('result').textContent +=`  
Replication error: ${error.message}`;
});
}

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

```

The screenshot shows a code editor interface with the following details:

- File Structure:** The left sidebar shows a file tree with three main folders: "MAIN", "public1", and "public3". Under "public3", there are files "index.html", "index.js", "package-lock.json", and "package.json".
- Code Editor:** The main pane displays the "index.html" file content. It includes an HTML structure with a title, a button labeled "Replicate Data", and a pre-tagged "result" area. Below this is a script block containing JavaScript code for replicating documents between Cloudant databases.
- Code Block Selection:** A green rectangular selection highlights the following lines of code in the script block:

```

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';

```

Cloudant API Example

Replicate Data

```
Replication success: 8bdc2e44d5790c6a14ee92307f4ac3d7
Replication error: {"error": "conflict", "reason": "Document update conflict."}
Replication success: 7cde5188e47fc9ee86c619bb2e09ed6b
Replication error: {"error": "conflict", "reason": "Document update conflict."}
Replication success: 5e662e943321da432febab3d35b69672
```

As you can see in destination DB replication is succeeded:

The screenshot shows the IBM Cloudant interface. On the left, there's a sidebar with navigation links: Monitoring, Databases, Replication, Active Tasks, Account, Support, and Documentation. The main area is titled 'test_db' and shows 'All Documents'. A green box highlights a specific document entry.

Document ID	Options	JSON	Raw
id "8bdc2e44d5790c6a14ee92307f4ac3d7"			
rev "6-d211d5967f5a6d67e2debdd9fba57447"			
doc { "_id": "8bdc2e44d5790c6a14ee92307f4ac3d7", "_key": "8bdc2e44d5790c6a14ee92307f4ac3d7", "_value": { "rev": "6-d211d5967f5a6d67e2debdd9fba57447" }, "name": "James Bond" }			

At the bottom, it says 'Showing document 1 - 3. Documents per page: 20'.

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.

Three screenshots of the IBM Cloud Cloud Object Storage interface showing the creation of three new credentials: CS_PRACTICAL_3_MANAGER, CS_PRACTICAL_3_READER, and CS_PRACTICAL_3_WRITER.

Screenshot 1: Creating CS_PRACTICAL_3_MANAGER Credential

The 'Create Credentials' dialog is open with the following settings:

- Name: CS_PRACTICAL_3_MANAGER
- Role: Manager
- Select Service ID (Optional): Auto Generated
- Include HMAC Credential: On (switch is green)

A green arrow points from the 'On' switch to the 'Add' button at the bottom right of the dialog.

Screenshot 2: Creating CS_PRACTICAL_3_READER Credential

The 'Create Credentials' dialog is open with the following settings:

- Name: CS_PRACTICAL_3_READER
- Role: Reader
- Select Service ID (Optional): Auto Generated
- Include HMAC Credential: On (switch is green)

A green arrow points from the 'On' switch to the 'Add' button at the bottom right of the dialog.

Screenshot 3: Creating CS_PRACTICAL_3_WRITER Credential

The 'Create Credentials' dialog is open with the following settings:

- Name: CS_PRACTICAL_3_WRITER
- Role: Writer
- Select Service ID (Optional): Auto Generated
- Include HMAC Credential: On (switch is green)

A green arrow points from the 'On' switch to the 'Add' button at the bottom right of the dialog.

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

The screenshot shows the IBM Cloud Endpoints page. On the left, there's a sidebar with 'IBM Cloud' at the top, followed by 'Cloud Object Storage', 'Instances', 'Integrations', and 'Endpoints' (which is selected and highlighted in blue). Below that are 'Documentation' and 'Billing'. The main area is titled 'Endpoints' and contains a sub-section 'Endpoints are used with your credentials (Bucket name, API Key, SDK) to tell your service where to look for your bucket. Choose an endpoint URL that is located in the same region as your service or application. Learn more'. It has dropdowns for 'Select resilience' (set to 'Cross Region') and 'Select location' (set to 'United States Geo (us-geo)'). The main table has three columns: 'Public', 'Private', and 'Direct'. Under 'Public', the 'us-geo' row is highlighted with a green box and has a copy icon. A tooltip 'Copied!' is shown over the icon. Other rows include Dallas, Washington, and San Jose. The 'Private' and 'Direct' columns show corresponding URLs for each region.

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

The screenshot shows the Postman interface. The left sidebar lists 'My Workspace' with collections like '4', '10', '5', '7', '8', '9', and 'CS 3'. 'CS 3' is expanded, showing a 'GET New Request' and a 'PUT New Request'. The 'PUT' request is selected, and its URL is 's3.us.cloud-object-storage.appdomain.cloud/tkbucket'. The 'Params' tab shows a single parameter 'Key' with value 'Value'. The 'Headers' tab has 'Content-Type: application/json'. The 'Body' tab is empty. The 'Tests' and 'Settings' tabs are also visible. The 'Response' section shows a small illustration of an astronaut. At the bottom, there are buttons for 'Send', 'Save', and 'Share', along with other toolbars.

Access is denied because IBM cloud security prevents unauthorized access

The screenshot shows the Postman interface with a failed API request. The URL is `s3.us.cloud-object-storage.appdomain.cloud/tkbucket`. The response status is `403 Forbidden` with the message `Anonymous bucket creation not allowed.`

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

The screenshot shows the IBM Cloud Object Storage service interface. A new credential named `CS_PRACTICAL_3_READER` has been created. The credential details are shown, including the access key ID and secret access key.

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

The screenshot shows the Postman application interface. On the left, the 'My Workspace' sidebar lists various collections and environments. The main area displays a 'PUT New Request' for 's3.us.cloud-object-storage.appdomain.cloud/tkbucket'. The 'Authorization' tab is selected, showing 'AWS Signature' as the type. The 'Headers' section contains 'Content-Type: application/xml'. The 'Body' section is set to 'Pretty' and shows an XML response from AWS S3 indicating an 'Access Denied' error. The status bar at the bottom indicates a 403 Forbidden error with a response time of 348 ms.

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

The screenshot shows the IBM Cloud UI for Cloud Object Storage. A modal window titled 'Create Credentials' is open. The 'Name' field is set to 'CS_PRACTICAL_3_REAL_WRITER'. The 'Role' dropdown is set to 'Writer'. A green arrow points to the 'Include HMAC Credential' toggle switch, which is turned 'On'. The 'Add' button at the bottom right of the modal is highlighted with a blue rectangle. The background shows the 'Cloud Object Storage-tushar' instance details.

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

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

Postman

PUT New Request

PUT s3.us.cloud-object-storage.appdomain.cloud/tkbucket

Params Authorization Headers (10)

AWS Signature

AccessKey

SecretKey

Advanced configuration

AWS Region: us

Service Name: s3

Session Token

Status: 200 OK Time: 1291 ms Size: 230 B

Cloud Object Storage-tushar

Buckets **Service credentials** **Instance Usage** **Plan**

tkbucket

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

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

The screenshot shows the IBM Cloud Object Storage interface. On the left, there's a sidebar with options like Cloud Object Storage, Instances, Integrations, Endpoints, Documentation, and Billing. The main area shows 'Instances / Cloud Object Storage-tushar / tkbucket'. Under 'Objects', there's a list with one item: '007.txt' (25 bytes, last modified 2024-08-20 12:08 PM). A green arrow points to the '007.txt' entry. A success message at the top right says 'Upload success 007.txt 2024-08-20 12:08 PM'.

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

The screenshot shows the Postman interface. The left sidebar has sections for Collections, Environments, APIs, Mock servers, Monitors, Flows, and History. The main area shows a 'GET New Request' for 's3.us.cloud-object-storage.appdomain.cloud/tkbucket/007.txt'. The 'Authorization' tab is selected, showing 'AWS Signature' as the auth type. The 'Headers' section lists 'Content-Type: application/json'. The 'Body' section shows the response: 'JAMES BOND IS HERE.....'. At the bottom, status information is shown: 'Status: 200 OK Time: 543 ms Size: 392 B'.

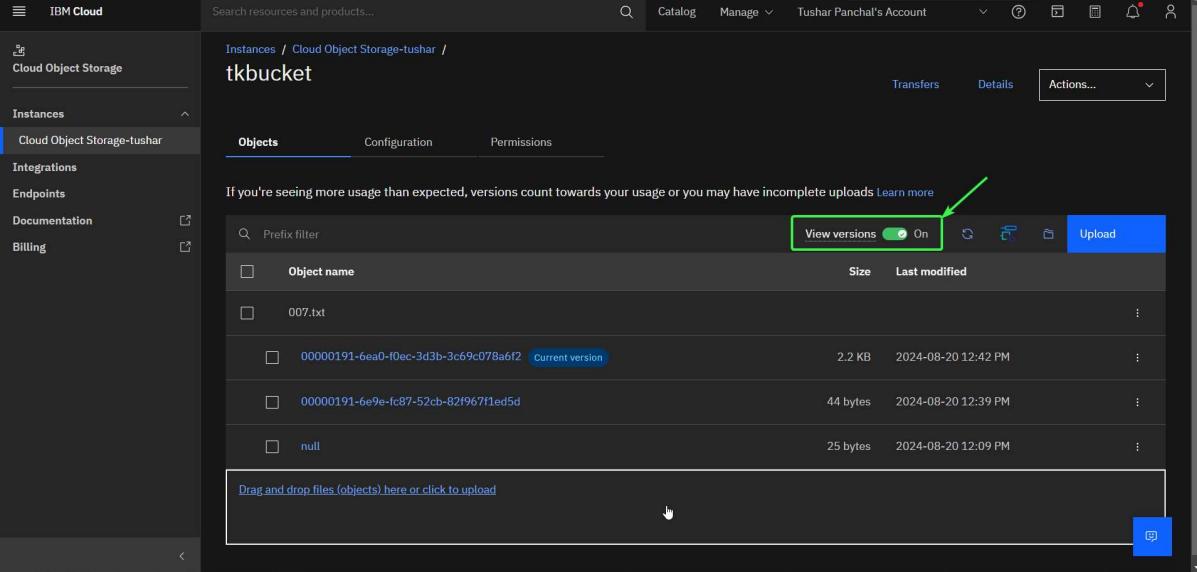
» **Step 19:** Now, if multiple versions of same object is needed, it can be enabled in configuration.

The screenshot shows the IBM Cloud Object Storage configuration interface. On the left, there's a sidebar with options like Instances, Integrations, Endpoints, Documentation, and Billing. The main area is titled 'Object versioning'. It has a sub-section 'After enabling versioning for your bucket' with a note that enabling it may increase bucket usage. Below this is the 'Object Lock' section, which is currently disabled. At the bottom, there's a 'Transfer preferences' section with 'Default transfer type' set to 'Not set' for both upload and download. A 'Save' button is visible at the top right.

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

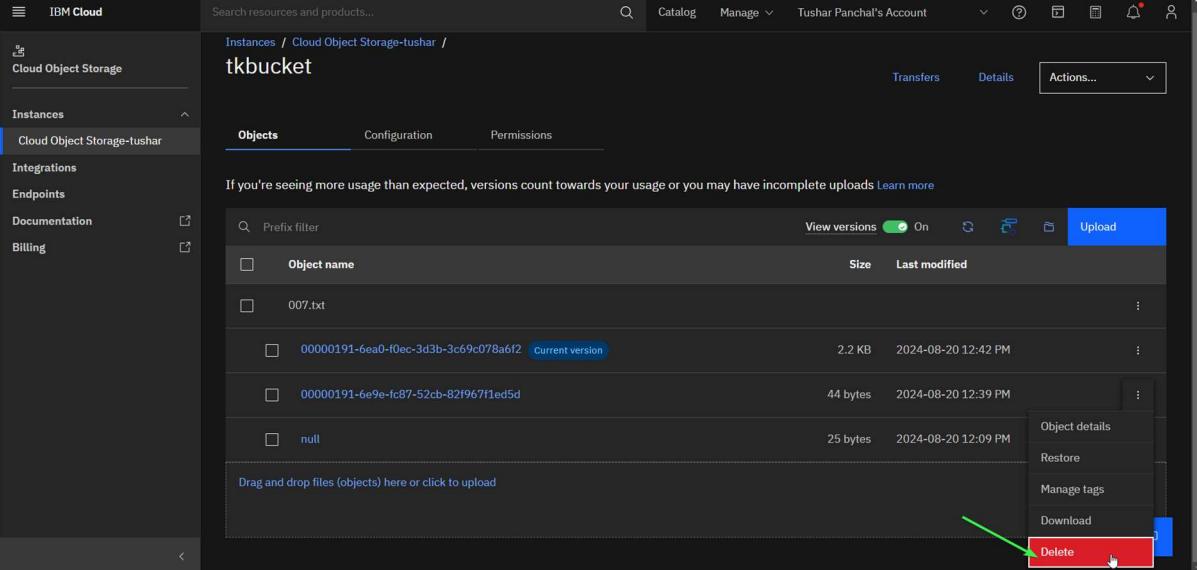
The screenshot shows the IBM Cloud Object Storage interface with a bucket named 'tkbucket'. The left sidebar shows the same navigation as before. The main area lists an object named '007.txt'. To the right, an 'Upload' dialog box is open. It has sections for 'Choose upload type' (with 'Standard transfer' selected), 'Aspera high-speed transfer' (disabled), and 'Upload files (objects)'. The 'Upload files' section contains a large blue 'Drag and drop files and folders or click to upload' area, which is highlighted with a green box in the screenshot. Below this are buttons for 'Upload files' and 'Upload folders'. At the bottom, there's a summary '1 objects | 2.2 KB' and 'Cancel' and 'Upload' buttons.

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



The screenshot shows the IBM Cloud Object Storage interface. On the left, there's a sidebar with options like Instances, Integrations, Endpoints, Documentation, and Billing. The main area shows a bucket named 'tkbucket'. At the top right, there are buttons for Catalog, Manage, and account information. Below that is a toolbar with Transfer, Details, Actions..., and other icons. A green box highlights the 'View versions' button, which is currently set to 'On'. The main table lists objects: '007.txt' (current version), '00000191-6e9e-fc87-52cb-82f967f1ed5d', and 'null'. Each row includes columns for Object name, Size, and Last modified. A message at the bottom says 'Drag and drop files (objects) here or click to upload'.

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



This screenshot is similar to the previous one but shows a context menu for the 'null' object. The menu options are: Object details, Restore, Manage tags, Download, and Delete. A green box highlights the 'Delete' button, which has a red border and is being pointed to by a red arrow. The rest of the interface is identical to the previous screenshot.

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

The screenshot shows the IBM Cloud Object Storage interface. On the left, there's a sidebar with 'Cloud Object Storage' selected under 'Instances'. The main area is titled 'Objects' and shows a list of files. One file, '007.txt', is highlighted with a red box. In the top right of the object details panel, there's a red arrow pointing to the 'Delete' button.

Object Name	Size	Last modified	Action
007.txt	0 bytes	2024-08-20 12:50	Object details
00000191-6ea8-4e61-0608-acde035cc712	2.2 KB	2024-08-20 12:46	Manage tags
00000191-6ea4-ad5d-f9b1-6b613dbe68da	2.2 KB	2024-08-20 12:46	Download
00000191-6ea4-9ab8-d5e8-1c764be44a62	2.2 KB	2024-08-20 12:46	Delete
00000191-6ea0-f0ec-3d3b-3c69c078a6f2	2.2 KB	2024-08-20 12:42 PM	Download
00000191-6e9e-fc87-52cb-82f967f1ed5d	44 bytes	2024-08-20 12:39 PM	Download

This screenshot is similar to the previous one, showing the IBM Cloud Object Storage interface. The '007.txt' file is selected. However, the 'Delete marker' button in the top right of the object details panel is now highlighted with a red box, indicating it has been selected or is the focus of the action.

Object Name	Size	Last modified	Action
007.txt	0 bytes	2024-08-20 12:50	Object details
00000191-6ea8-4e61-0608-acde035cc712	2.2 KB	2024-08-20 12:46	Manage tags
00000191-6ea4-ad5d-f9b1-6b613dbe68da	2.2 KB	2024-08-20 12:46	Download
00000191-6ea4-9ab8-d5e8-1c764be44a62	2.2 KB	2024-08-20 12:46	Delete
00000191-6ea0-f0ec-3d3b-3c69c078a6f2	2.2 KB	2024-08-20 12:42 PM	Download
00000191-6e9e-fc87-52cb-82f967f1ed5d	44 bytes	2024-08-20 12:39 PM	Download

The screenshot shows the IBM Cloud interface for Cloud Object Storage. On the left, a sidebar lists 'Instances' (selected), 'Cloud Object Storage-tushar', 'Integrations', 'Endpoints', 'Documentation', and 'Billing'. The main area displays a list of objects in the 'Cloud Object Storage-tushar' bucket. A green arrow points to the 'Object details' button for the first object, '007.txt'. The table headers are 'Object name', 'Size', and 'Last modified'. The data rows show:

Object name	Size	Last modified
007.txt	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
00000191-6ea0-0ec-3d3b-3c69c078a6f2	2.2 KB	2024-08-20 12:42 PM
00000191-6e9e-fc87-52cb-821967f1ed5d	44 bytes	2024-08-20 12:39 PM
null	25 bytes	2024-08-20 12:09 PM

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](#)". There is also a message at the bottom: "Drag and drop files (objects) here or click to upload".

The screenshot shows the '007.txt' object details dialog. The sidebar on the left is identical to the previous screenshot. The main area shows the object details for '007.txt'. A green arrow points to the 'Versions' tab, which is selected. The table headers are 'Version ID', 'Archive Type', 'Size', and 'Last modified'. The data rows show:

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
00000191-6ea0-0ec-3d3b-3c69c078a6f2		2.2 KB	2024-08-20 12:42 PM
00000191-6e9e-fc87-52cb-821967f1ed5d		44 bytes	2024-08-20 12:39 PM
null		25 bytes	2024-08-20 12:09 PM

At the bottom of the dialog, there are buttons for 'Download object' and 'Delete object'. Below the table, there are buttons for 'Overview', 'Versions' (selected), 'Lifecycle', and 'Retention'. The message '1–6 of 6 items' is displayed at the bottom.

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

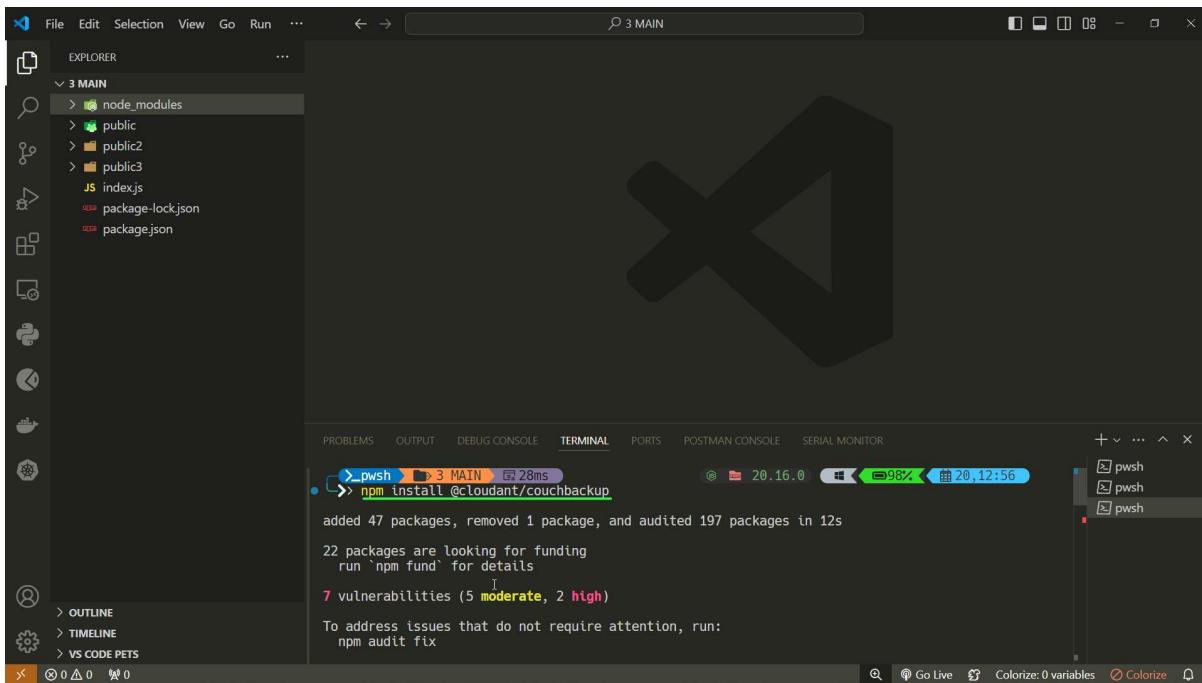
The screenshot shows the IBM Cloud Object Storage interface. On the left, there's a sidebar with options like Instances, Integrations, Endpoints, Documentation, and Billing. The main area is titled '007.txt' and has tabs for Overview, Versions, Lifecycle, and Retention. The Versions tab is selected. It displays a table of versions with columns for Version ID, Archive Type, Size, and Last modified. The first version is marked as 'Current version'. The second version, with Version ID '00000191-6ea4-ad5d-f9b1-6b613dbe68da', has a green arrow pointing to the 'Make current vers...' button in its row.

Version ID	Archive Type	Size	Last modified
00000191-6ea8-4e61-0608-acde035cc712	Current version	0 bytes	2024-08-20 12:50 PM
00000191-6ea4-ad5d-f9b1-6b613dbe68da		2.2 KB	2024-08-20 12:46 PM
00000191-6ea4-d5e8-1c764be44a62		2.2 KB	2024-08-20 12:46 PM
00000191-6ea0-f0ec-3d3b-3c69c078a6f2		2.2 KB	2024-08-20 12:46 PM
00000191-6e9e-fc87-52cb-82f967f1ed5d		44 bytes	2024-08-20 12:46 PM

This screenshot shows the same interface after the action from the previous step. The second version of the file is now marked as the 'Current version' in the table. The rest of the interface remains the same, with the sidebar and the 'Versions' tab selected.

Version ID	Archive Type	Size	Last modified
00000191-6ea8-4e61-0608-acde035cc712	Current version	0 bytes	2024-08-20 12:50 PM
00000191-6ea4-ad5d-f9b1-6b613dbe68da	Delete marker	2.2 KB	2024-08-20 12:50 PM
00000191-6ea4-d5e8-1c764be44a62		2.2 KB	2024-08-20 12:46 PM
00000191-6ea0-f0ec-3d3b-3c69c078a6f2		2.2 KB	2024-08-20 12:46 PM
00000191-6e9e-fc87-52cb-82f967f1ed5d		44 bytes	2024-08-20 12:46 PM

» **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
```

```

File Edit Selection View Go Run Terminal Help ← → 3 MAIN DB data
EXPLORER backup.txt
> node_modules
> public
> public2
> public3
> backup.txt
JS index.js
package-lock.json
package.json
1 {"name": "@cloudant/couchbackup", "version": "2.10.1", "mode": "full"}
2 [{"_id": "00cd5f90264a57b987fd3784ee2230", "_rev": "1-fdd53e61e6db5c744d6f105b1c808bb", "_revisions": [{"ids": ["fdd53e61e6db5c744d6f105b1c808bb"], "start": 1}, {"name": "Avengers Assemble"}, {"_id": "7cde518e47fc9ee86c619bb2e09ed6b", "_rev": "1-423b5657329de5f12f6079ba8886b062", "_revisions": [{"ids": ["423b5657329de5f12f6079ba8886b062"], "start": 1}, {"title": "To Kill a Mockingbird", "author": "Harper Lee", "genre": "Fiction", "year": 1960}, {"_id": "00cd5f90264a57b987fd3b784ee961b", "_rev": "1-5ee76b35afeae5df36d749770fbfe025", "_revisions": [{"ids": ["5ee76b35afeae5df36d749770fbfe025"], "start": 1}, {"name": "Harry Potter", "author": "jk rowling", "genre": "magic", "year": 1960}, {"_id": "8bdc2e44d5790c6a14ee92307f4ac3d7", "_rev": "1-a790c098aa33e688b5c3899e35852605", "_revisions": [{"ids": ["a790c098aa33e688b5c3899e35852605"], "start": 1}, {"name": "James Bond"}]}]
3

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS POSTMAN CONSOLE SERIAL MONITOR
@ 20.16.0
● couchbackup -> https://apikey-v2-26a13ozdyvr2f1cmuw0xky1mbh1ubclpomomvprj:531f952
562337579bc03ef8fea14bc@d643cff-2d52-40c5-b33d-71ad7ff9e885-bluemix.cloudantnosqldb.appdomain
in.cloud" --db exam > backup.txt
Performing backup on https://*****:@ed643cff-2d52-40c5-b33d-71ad7ff9e885-bluemix.cloudantno
sqldb.appdomain.cloud/exam using configuration:
{
  "bufferSize": 500,
  "log": "C:\Users\tusha\AppData\Local\Temp\couchbackup-lgTTQH\1724139343967",
  "mode": "full",
  "mode": "full",
  "parallelism": 5,
  "requestTimeout": 120000,
  "resume": false
}
couchbackup:backup Fetching all database changes... +0ms
couchbackup:batch Total batches received: 1 +0ms
couchbackup:batch Written batch ID: 0 Total document revisions written: 5 Time: 0.559 +186ms
couchbackup:batch Finished - Total document revisions written: 5 +2s
@ 20.16.0
● couchbackup -> 3 MAIN 2s 509ms
Ln 3, Col 1 Spaces:4 UTF-8 CRLF Plain Text Go Live Colorize 0 variables Colorize

```

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

The screenshot shows the Apache CouchDB Futon interface. On the left, there's a sidebar with icons for All Documents, Query, Permissions, Changes, Design Documents, and Log Out. The main area has a title bar 'backup_db'. Below the title bar, there are tabs for Document ID, Options, JSON, and XML. A blue button labeled 'Create Document' is visible. The central part of the screen displays a large cloud icon and the message 'No Documents Found'. At the bottom, there's a footer bar with the text 'Showing 0 documents. Documents per page: 20' and navigation arrows.

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
```

```
C:\Users\tusha>cd "C:\Users\tusha\Documents\SEM 7\CS\CODES\3 MAIN"
C:\Users\tusha\Documents\SEM 7\CS\CODES\3 MAIN>couchrestore --url "https://apikey-v2-26aj3ozdvr2f1cmuwbxky1lmhbiubctlpomovvprij:531f952562337579bc0b3ef8cf
ea14bc@ed643cff-2d52-40c5-b33d-71ad7ff9e885-bluemix.cloudantnosqldb.appdomain.cloud" --db backup_db < backup.txt
=====
Performing restore on https://*****-*****@ed643cff-2d52-40c5-b33d-71ad7ff9e885-bluemix.cloudantnosqldb.appdomain.cloud/backup_db using configuration:
{
  "bufferSize": 500,
  "parallelism": 5,
  "requestTimeout": 120000
}
=====
couchbackup:restore:batch Restored batch ID: 0 Total document revisions restored: 5 Time: 0.197 +0ms
couchbackup:restore finished { total: 5 } +0ms
C:\Users\tusha\Documents\SEM 7\CS\CODES\3 MAIN>
```

	_id	name	author	genre	title
<input type="checkbox"/>	00cd55f90264a57b987fd3...	Avengers Assemble			
<input type="checkbox"/>	00cd55f90264a57b987fd3...	Henry Cavil			
<input type="checkbox"/>	5e662e943321da432febab...	jk rowling	magic	harry potter	
<input type="checkbox"/>	7cde5188e47fc9ee86c619...	Harper Lee	Fiction	To Kill a Mockingbird	
<input type="checkbox"/>	8bcd2e44d5790c6a14ee92...	James Bond			

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