



Ganpat University

॥ विद्या 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 is a sidebar with links: Monitoring, Databases, Replication, Active Tasks, Account (which is highlighted), Support, and Documentation. The main content area has a header 'Account' with 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 blue 'Enable CORS' button. A green status bar at the top right says 'CORS settings updated.'

CORS settings updated.

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

IBM Cloudant

Log Out IBMID-6930008XII

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

Key name	Date created	Controlled by
Service credentials-1tK	2024-03-26 8:00 AM	Cloudant

```
{
  "apikey": "j0EQt0KhvzKYVArJ0cIXB1-TWtA99bnJHy8QfhkjuuuCB",
  "host": "ed643cff-2d52-4e5c-b33d-71ad7ff9e885-bluemix.cloudantnosqldb.appprovisioning.cloud",
  "iam_apikey_description": "Auto-generated for key crn:v1:bluemix:public:cloudantnosqldb:au-syd:a/e30ea4cae95048a29af53e3839877643:389e4253-d555-47c7-9c4f-f8ab0bd379ad:resource-key:c6374f13-54af-45b0-9bd5-669933d24f69",
  "iam_apikey_id": "ApiKey-c561599f-c49d-45d1-8bf2-7a5648958da4",
  "iam_apikey_name": "Service credentials-1tK",
  "iam_golo_crn": "crn:v1:bluemix:public::serviceRole:Manager",
  "iam_serviceid_crn": "crn:v1:bluemix:public:iam-identity:a/e30ea4cae95048a29af53e3839877643::serviceid:ServiceId-cbb89c30-5dec-4271-baed-c601bnaad2db8",
  "password": "531f952337579bc0b3ef8cfea1abc",
  "port": 443,
  "url": "https://apikey-v2-26aj3ozdvpr2f1cmuwbxky11mhiuibctlpomomvyrjj:531f952562337579bc0b3ef8cfea14bc@ed643cff-2d52-40c5-b33d-71ad7ff9e885-bluemix.cloudantnosqldb.appprovisioning.cloud",
  "username": "apikey-v2-26aj3ozdvpr2f1cmuwbxky11mhiuibctlpomomvyrjj"
}
```

✓ 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.

The screenshot shows the 'CORS' tab in the IBM Cloudant account settings. It explains what CORS is and states that it is currently enabled. Under 'Origin Domains', it says 'Databases will accept requests from these domains:' and provides two options: 'All domains (*)' and 'Restrict to specific domains'. The 'Restrict to specific domains' option is selected, and the URL 'http://localhost:3000/' is entered into the input field. A blue 'Add Domain' button is located to the right of the input field.

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

The screenshot shows a dark-themed browser window with the title 'Cloudant CORS Example'. At the top, there is a button labeled 'Fetch Tushar's Databases'. Below the button, the text 'Fetch error: Failed to fetch' is displayed in white, indicating that the request failed due to CORS restrictions.

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

The screenshot shows the IBM Cloudant Databases interface. On the left is a sidebar with links: Monitoring, Databases (selected), Replication, Active Tasks, Account, Support, and Documentation. The main area is titled "Databases" and lists five databases: "exam", "imbdta_tushar", "novel-tushar", "tk", and "tushar_practical_10". Each database row includes columns for Name, Size, # of Docs, Partitioned, and Actions (edit, lock, delete). At the bottom, it says "Showing 1-5 of 5 databases" and "Databases per page 20".

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

The screenshot shows the IBM Cloudant Permissions interface for the "exam" database. The sidebar shows "exam" selected. The main area displays "Cloudant users and API keys with permissions on exam." It lists a user "ed643cff-2d52-40c5-b33d-71ad7ff9e885-bluemix" with permissions: _admin (checked), _reader (checked), _writer (checked), and _replicator (checked). Below this is a section for "Unauthenticated connections". At the bottom, there's a "Grant database permissions to:" 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 screenshot shows the IBM Cloudant interface. On the left, there's a sidebar with options like Monitoring, Databases, Replication, Active Tasks, Account, Support, and Documentation. The main area is titled 'exam' and shows 'All Documents'. A message at the top right says 'Database permissions have been saved.' Below this, there's a table for 'Permissions' with columns for '_id', '_rev', '_admin', '_reader', '_writer', and '_replicator'. One row is selected, highlighted with a green border. At the bottom, there's a section for 'API keys' with fields for 'Key' and 'Password', and a note about expiration.

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

The screenshot shows a code editor with an 'EXPLORER' view on the left showing a project structure with files like 'index.js', 'index.html', and 'package.json'. The main editor window shows a JavaScript file with code that uses the 'fetch' API to interact with a Cloudant database. The URL, API key, and password used in the code are highlighted with a green box.

```

public2 > index.html > html > body > script > addEventListener('click') callback
1  <html lang="en">
2    <head>
3      <meta name="viewport" content="width=device-width, initial-scale=1.0">
4      <title>Cloudant API Example</title>
5    </head>
6    <body>
7      <h1>Cloudant API Example</h1>
8      <button id="fetchData">Fetch Databases</button>
9      <pre id="result"></pre>
10
11
12
13
14  document.getElementById('fetchData').addEventListener('click', function() {
15    const apiKey = 'apikey-868b7b88ecae47299b2744276011aa31';
16    const apiPassword = 'edfe811c3ab991576d35cd29f8838c9b8adda124';
17    const url = 'https://ed643cff-2d52-40c5-b33d-71ad7ff9e885-bluemix.cloudantnosqldb.appdomain.cloud/exam/_all_docs?include_docs=true';
18
19    const headers = new Headers();
20    headers.append('Authorization', 'Basic ' + btoa(apiKey + ':' + apiPassword));
21
22    fetch(url, { method: 'GET', headers: headers })
23      .then(response => response.json())
24      .then(data => {
25        document.getElementById('result').textContent = JSON.stringify(data, null, 2);
26      })
27      .catch(error => {
28        document.getElementById('result').textContent = 'Fetch error: ' + error.message;
29      });
30  });
31

```

✓ 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": "5e662e043321da432febab3d35b69672",
      "key": "5e662e043321da432febab3d35b69672",
      "value": {
        "rev": "1-214dd45ced07e8ba5984d981d891e0426"
      },
      "doc": {
        "_id": "5e662e043321da432febab3d35b69672",
        "_rev": "1-214dd45ced07e8ba5984d981d891e0426",
        "title": "harry potter",
        "author": "jk rowlling",
        "genre": "magic",
        "year": 1960
      }
    },
    {
      "id": "7cde5188e47fc9ee86c619bb2e09ed6b",
      "key": "7cde5188e47fc9ee86c619bb2e09ed6b",
      "value": {
        "rev": "1-423b5657329de5f12f6079ba8886b062"
      },
      "doc": {
        "_id": "7cde5188e47fc9ee86c619bb2e09ed6b",
        "_rev": "1-423b5657329de5f12f6079ba8886b062",
        "title": "To Kill a Mockingbird",
        "author": "Harper Lee",
        "genre": "Fiction",
        "year": 1960
      }
    }
  ]
}
```

➤ **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-868b7b88ecae47299b2744276011aa31	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" 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. The main area is titled "Databases" and lists existing databases: exam, imbddata_tushar, novel-tushar, tk, and tushar_practical_10. A modal window titled "Create Database" is open on the right. 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 modal 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 includes "Monitoring", "Databases" (selected), "Replication", "Active Tasks", "Account", "Support", and "Documentation". The main content area shows "Cloudant users and API keys with permissions on test_db". It lists two entries: "ed643cff-2d52-40c5-b33d-71ad7ff9e885-bluemix" and "Unauthenticated connections". Below this is a "Grant database permissions to:" input field containing "apikey-868b7b88ecae47299b2744276011aa31" 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. The sidebar and main content area are similar to the previous screenshot, but the "Grant database permissions to:" input field now contains "apikey-868b7b88ecae47299b2744276011aa31" and the "Grant Permissions" button is highlighted with a green border. A green arrow points to the "Grant Permissions" button. The note at the bottom about API keys and the "Generate API Key" button are also present.

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

The screenshot displays two overlapping interface windows from the IBM Cloudant dashboard.

Top Window: Replication

- Left Sidebar:** Monitoring, Databases, **Replication** (highlighted with a green box), Active Tasks, Account, Support, Documentation.
- Header:** Replication, Polling Interval (5 minutes), Refresh, Bell icon.
- Content:** Replicator DB Activity, replicate Activity. A message states: "Replications must have a replication document to display in the following table." Below is a table header with columns: Source, Target, Start Time, Type, State, Actions. A message in the table body says: "There is no replicator-db activity or history to display." A "New Replication" button is located at the bottom right of the table area, with a green arrow pointing to it.

Middle Window: Job Configuration

- Left Sidebar:** Monitoring, Databases, **Replication** (highlighted with a green box), Active Tasks, Account, Support, Documentation.
- Header:** Job Configuration, Bell icon.
- Source Section:** Type: Local database, Name: exam, Authentication: Cloudant Username or API Key, Value: apikey-868b7b88ecae47299b2744276011aa31. A note below says: "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 Section:** Type: Existing local database, Name: test_db, Authentication: Cloudant Username or API Key, Value: apikey-868b7b88ecae47299b2744276011aa31. A note below says: "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."

Bottom Window: Target (partially visible)

- Left Sidebar:** Monitoring, Databases, **Replication**, Active Tasks, Account, Support, Documentation.
- Header:** Target, Bell icon.
- Content:** Type: Existing local database, Name: test_db, Authentication: Cloudant Username or API Key, Value: apikey-868b7b88ecae47299b2744276011aa31. A note below says: "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."

Bottom Window: Options (partially visible)

- Left Sidebar:** Monitoring, Databases, **Replication**, Active Tasks, Account, Support, Documentation.
- Header:** Options, Bell icon.
- Content:** Replication type: One time, Replication document: Custom ID (optional). Buttons at the bottom include: Clear, Start Replication (highlighted with a green box and a green arrow pointing to it).

The screenshot shows the IBM Cloudant Replication interface. On the left sidebar, there are links for Monitoring, Databases, Replication (which is selected), Active Tasks, Account, Support, and Documentation. The main area is titled 'Replication' and shows a table of replication activities. A green box highlights a row where the Source is 'https://ed643cff-2d52-40c5-b33d-71ad7ff9e885-bluemix.cloudant.com/exam' and the Target is 'https://ed643cff-2d52-40c5-b33d-71ad7ff9e885-bluemix.cloudant.com/test_db'. The status bar at the bottom indicates 'Showing replications 1 - 1 Max replications displayed: 100'.

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

The screenshot shows the 'New Document' dialog for the 'exam' database. The 'Create Document' button is highlighted with a green arrow. The JSON code for the new document is shown in the editor: {_id: "8bd2e44d5790c6a14ee92307f4ac3d7", name: "James Bond"}. The status bar at the bottom indicates 'Showing documents 1 - 1 Max documents displayed: 100'.

✓ 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(replicatedDoc => {
            console.log(`Replicated document ${doc._id} from ${sourceUrl} 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 +=
`\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>
```

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

- File Path:** index.html
- Code Editor Area:**

```

<!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-868b7088ecae47299b2744276011aa31';
    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);
            });
        });
})
```
- Explorer Bar:** Shows the project structure with files like index.js, package-lock.json, and package.json.
- Bottom Status Bar:** Displays information such as "Ln 19, Col 1", "Spaces: 4", "UTF-8", "CRLF", "HTML", "Colorize 0 variables", and "Colorize".

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:

```

{
  "_id": "8bdc2e44d5790c6a14ee92307f4ac3d7",
  "_rev": "6-d211d5967f5a6d6702debdd9fba57447",
  "value": {
    "_rev": "6-d211d5967f5a6d6702debdd9fba57447"
  },
  "name": "James Bond"
}

```

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.

Cloud Object Storage

Instances / Cloud Object Storage-tushar

Create Credentials

Name: CS_PRACTICAL_3_MANAGER

Role: Manager

Select Service ID (Optional): Auto Generated

Include HMAC Credential: On

Add

Cloud Object Storage-tushar

Create Credentials

Name: CS_PRACTICAL_3_READER

Role: Reader

Select Service ID (Optional): Auto Generated

Include HMAC Credential: On

Add

Cloud Object Storage-tushar

Create Credentials

Name: CS_PRACTICAL_3_WRITER

Role: Writer

Select Service ID (Optional): Auto Generated

Include HMAC Credential: On

Add

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

The screenshot shows the IBM Cloud Endpoints interface. On the left sidebar, 'Endpoints' is selected. In the main area, under 'Public' endpoints for 'us-geo', the URL 's3.us.cloud-object-storage.appdomain.cloud' is highlighted and has a copy icon with a tooltip 'Copied!' over it. Other options include 's3.private.us.cloud-object-storage.appdomain.cloud' and 's3.direct.us.cloud-object-storage.appdomain.cloud'. Below this, there are sections for 'Private' and 'Direct' endpoints, and a 'Legacy Endpoints' section.

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

The screenshot shows the Postman application interface. On the left sidebar, 'My Workspace' is selected. In the main area, a 'PUT New Request' is being configured. The URL field contains 's3.us.cloud-object-storage.appdomain.cloud/tkbucket'. The 'Params' tab shows a single parameter 'Key' with value 'Value'. The 'Response' pane displays an illustration of an astronaut with the text 'Click Send to get a response'. At the bottom, a status message reads 'Access is denied because IBM cloud security prevents unauthorized access'.

Access is denied because IBM cloud security prevents unauthorized access

The screenshot shows the Postman interface with a collection named 'My Workspace'. A new request is being made to the URL `s3.us.cloud-object-storage.appdomain.cloud/tkbucket`. The 'Scripts' tab contains the following code:

```

var template = '
<style type="text/css">
.tftable {font-size:14px;color:#333333;
width:100%;border-width: 1px;
border-color: #87ceeb;border-collapse:
collapse:1}

```

The response details show a 403 Forbidden status with the message "Anonymous bucket creation not allowed." The resource path is `/tkbucket/` and the request ID is `a43970c8-4115-4967-934f-a7abc4d3316a`.

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

The screenshot shows the IBM Cloud UI for Cloud Object Storage. A service credential named "CS_PRACTICAL_3_READER" is selected. The credential details page shows the following JSON configuration:

```

{
  "apikey": "6ed4CaL79xArqHTCr6hMMeIeOaEfFd9_Qxs3AYys1ldBK",
  "cos_hmac_keys": {
    "access_key_id": "c5b38811318d74142aa128ff8fb68c9a5",
    "secret_access_key": "dbff1848048284ea1e77782fc6494bbaaf1511a6aafb2bfe99"
  },
  "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/e30ea4cae9508a829a5f53e3839877643:63fe2f49-6e97-4c54-968a:c0355f92f55a:re-a...b0...5038d313-f0d7-4142-aed2-0f88186e9a5",
  "iam_apikey_name": "Cloud Object Storage - tushar"
}

```

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 interface with a collection named 'CS 3'. A new request is being made to the URL `s3.us.cloud-object-storage.appdomain.cloud/tkbucket` using the PUT method. The 'Authorization' tab is selected, showing 'AWS Signature' as the type. The 'AccessKey' and 'SecretKey' fields contain placeholder values. Under 'Advanced configuration', the 'AWS Region' is set to 'us' and the 'Service Name' is set to 's3'. The response body shows an XML error message indicating an Access Denied error.

```

<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<Error>
<Code>AccessDenied</Code>
<Message>Access Denied</Message>
<Resource>/tkbucket</Resource>
<RequestId>7df99446-ca42-4efb-bfb1-c116cb391327</RequestId>
<httpStatusCode>403</httpStatusCode>
</Error>

```

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

The screenshot shows the 'Create Credentials' dialog in the IBM Cloud Object Storage interface. The 'Role' dropdown is set to 'Writer'. The 'Include HMAC Credential' toggle switch is turned 'On'. At the bottom right of the dialog, there is a large blue 'Add' button.

Screenshot 1: IBM Cloud Service Credentials

The screenshot shows the IBM Cloud interface for managing service credentials. The left sidebar is for Cloud Object Storage, and the main area is titled "Cloud Object Storage-tushar". The "Service credentials" tab is selected. A search bar at the top says "Search resources and products...". Below it, a table lists two entries under the "WRITER" role:

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

Screenshot 2: Postman API Request

The screenshot shows the Postman interface with a "PUT" request to "s3.us.cloud-object-storage.appdomain.cloud/tkbucket". The "Authorization" tab is selected, showing "AWS Signature" with fields for "AccessKey" and "SecretKey". The "Body" tab shows a JSON payload. The response status is "200 OK".

Screenshot 3: IBM Cloud Buckets

The screenshot shows the IBM Cloud interface for managing buckets. The left sidebar is for Cloud Object Storage, and the main area is titled "Cloud Object Storage-tushar". The "Buckets" tab is selected. A search bar at the top says "Search resources and products...". Below it, a table lists one 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 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 Cloud Object Storage interface. On the left sidebar, under 'Cloud Object Storage-tushar', the 'Instances' section is selected. In the center, the 'tkbucket' instance is shown. The 'Objects' tab is active, displaying a list of objects. A single object, '007.txt', is listed with a size of 25 bytes and last modified at 2024-08-20 12:08 PM. A green arrow points to the '007.txt' entry. At the top right, a success message says 'Upload success 007.txt' with a timestamp of '2024-08-20 12:08 PM'.

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

The screenshot shows the Postman application interface. On the left sidebar, 'My Workspace' is selected, showing various collections and environments. In the center, a 'GET CS 3 / New Request' window is open. The URL is set to '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 '(9)' headers. The 'Body' section shows the response content: 'JAMES BOND IS HERE.....'. The status bar at the bottom indicates 'Status: 200 OK'.

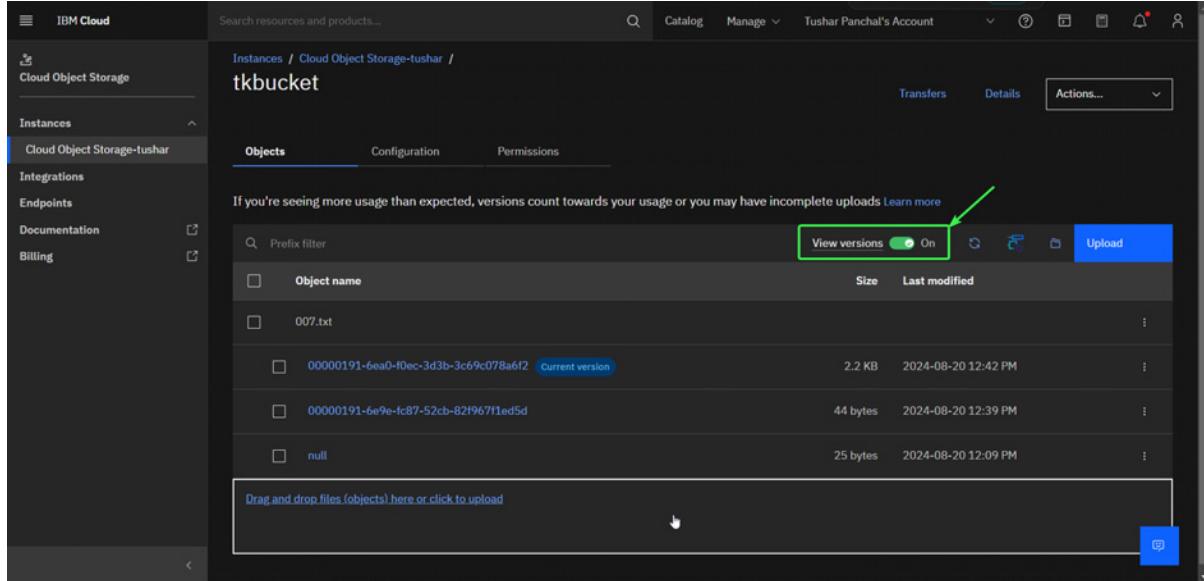
➤ **Step 19:** Now, if multiple versions of same object is needed, it can be enabled in configuration.

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 note about protecting objects from accidental deletion or overwrites. A radio button for 'Enable' is selected, highlighted with a green border. Below this, a note says 'After enabling versioning for your bucket' and 'Enabling versioning may increase your bucket usage (bytes) as all versions of objects will be retained until deleted by the user.' At the bottom, there's a 'Transfer preferences' section with 'Edit' and 'Not set' for both upload and download types. The top right has 'Cancel' and 'Save' buttons.

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

The screenshot shows the IBM Cloud Object Storage upload interface. The left sidebar shows 'Cloud Object Storage' under 'Instances'. The main area shows a list of objects in a bucket named 'tkbucket'. One file, '007.txt', is listed with a size of 44 bytes and last modified date of 2024-08-20 12:40 PM. To the right, an 'Upload' dialog box is open. It has sections for 'Choose upload type' (Standard transfer selected), 'Aspera high-speed transfer' (disabled), and 'Upload files (objects)'. A large blue box highlights the 'Drag and drop files and folders or click to upload' area. Below it are buttons for 'Upload files' and 'Upload folders'. At the bottom, it shows '1 objects | 2.2 KB' and lists '007.txt 2.2 KB'. There are 'Cancel' and 'Upload' buttons at the very bottom.

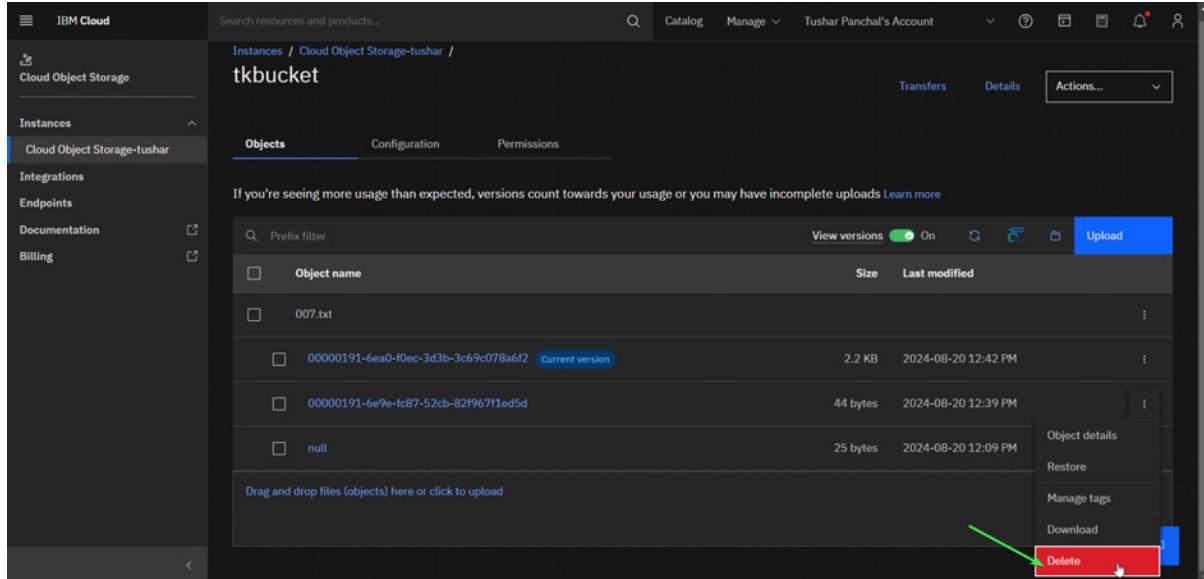
➤ **Step 21:** View Versions button can be used to show files with all versions including null (initial version).



The screenshot shows the IBM Cloud Cloud Object Storage interface. On the left, there's a sidebar with options like Instances, Integrations, Endpoints, Documentation, and Billing. The main area shows an instance named 'tkbucket'. Under 'Objects', there are three entries: '007.txt', '00000191-6ea0-f0ec-3d3b-3c69c078a6f2' (labeled 'current version'), and 'null'. Above the object list, there's a toolbar with a 'View versions' button, which is highlighted with a green box and has a blue arrow pointing to it from the top right.

Object Name	Size	Last modified
007.txt	2.2 KB	2024-08-20 12:42 PM
00000191-6ea0-f0ec-3d3b-3c69c078a6f2	2.2 KB	2024-08-20 12:42 PM
null	25 bytes	2024-08-20 12:09 PM

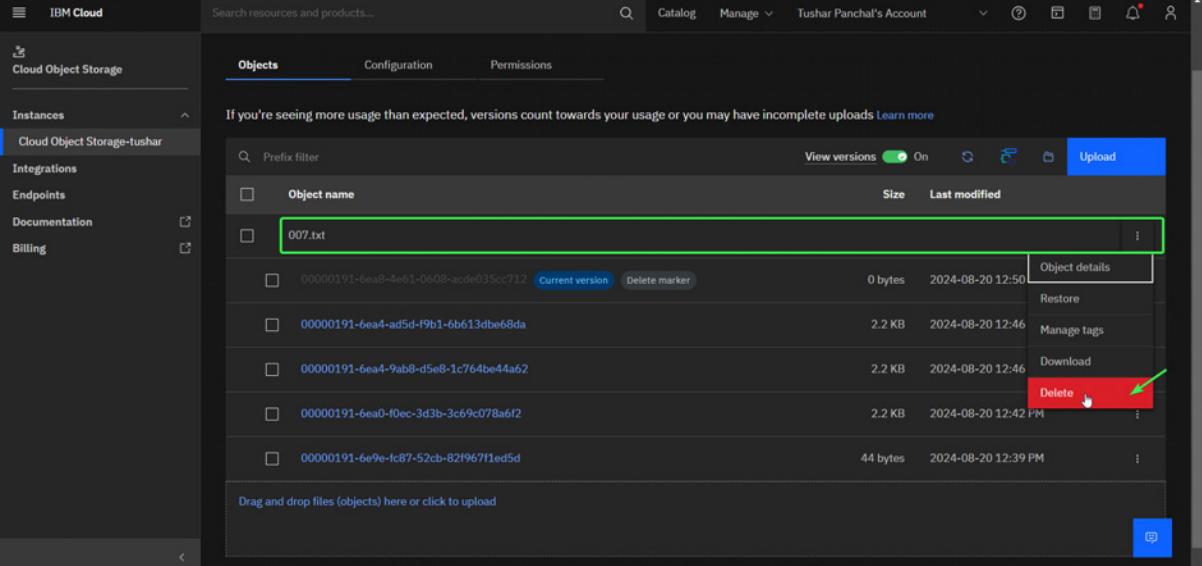
➤ **Step 22:** Try deleting the versions and changing default one.



This screenshot is similar to the previous one but shows a context menu opened over the '00000191-6ea0-f0ec-3d3b-3c69c078a6f2' object. The menu includes options like 'Object details', 'Restore', 'Manage tags', 'Download', and 'Delete'. The 'Delete' option is highlighted with a red box and has a green arrow pointing to it from the bottom right.

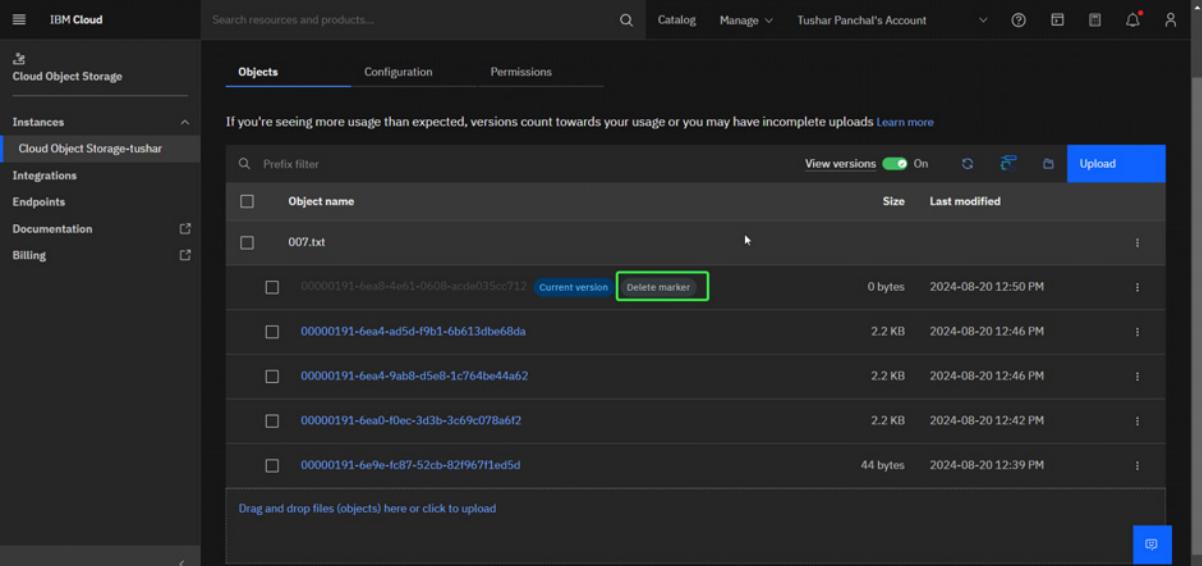
Object Name	Size	Last modified
007.txt	2.2 KB	2024-08-20 12:42 PM
00000191-6ea0-f0ec-3d3b-3c69c078a6f2	2.2 KB	2024-08-20 12:42 PM
null	25 bytes	2024-08-20 12:09 PM

➤ **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. The left sidebar is collapsed. The main area has a dark header with 'IBM Cloud' and a search bar. Below the header are tabs: 'Objects' (which is active), '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 this is a table with columns: 'Object name', 'Size', and 'Last modified'. The first row, '007.txt', is highlighted with a green border. The 'Delete' button in the row for '007.txt' is highlighted with a red box and a green arrow pointing to it. Other rows show various file versions with their respective sizes and last modified dates.

Object name	Size	Last modified
007.txt	0 bytes	2024-08-20 12:50
00000191-6ea8-4e61-0608-acde035cc712	2.2 KB	2024-08-20 12:46
00000191-6ea4-ad5d-f9b1-6b613dbe68da	2.2 KB	2024-08-20 12:46
00000191-6ea4-9ab8-d5e8-1c764be44a62	2.2 KB	2024-08-20 12:46
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



This screenshot is identical to the one above, showing the IBM Cloud Object Storage interface. The 'Objects' tab is active, and the file '007.txt' is selected. The 'Delete' button in its row is highlighted with a red box and a green arrow pointing to it. The rest of the table and interface elements are the same as in the previous screenshot.

Object name	Size	Last modified
007.txt	0 bytes	2024-08-20 12:50
00000191-6ea8-4e61-0608-acde035cc712	2.2 KB	2024-08-20 12:46
00000191-6ea4-ad5d-f9b1-6b613dbe68da	2.2 KB	2024-08-20 12:46
00000191-6ea4-9ab8-d5e8-1c764be44a62	2.2 KB	2024-08-20 12:46
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

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

Object name	Size	Last modified	Actions
007.txt	2.2 KB	2024-08-20 12:46 PM	View details
00000191-6ea4-9ab8-d5e8-1c764be44a62	2.2 KB	2024-08-20 12:46 PM	View details
00000191-6ea4-3e3e-f73f-b6a4e1af35b7	2.2 KB	2024-08-20 12:45 PM	View details
00000191-6ea0-10ec-3d3b-3c69c078a6f2	2.2 KB	2024-08-20 12:42 PM	View details
00000191-6e9e-fc87-52cb-821967f1ed5d	44 bytes	2024-08-20 12:39 PM	View details
null	25 bytes	2024-08-20 12:09 PM	View details

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

007.txt

Download object [↓](#) Delete object [✖](#)

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
00000191-6ea0-10ec-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

Items per page: 25 [▼](#) 1–6 of 6 items [1](#) [▼](#) 1 of 1 page [◀](#) [▶](#)

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

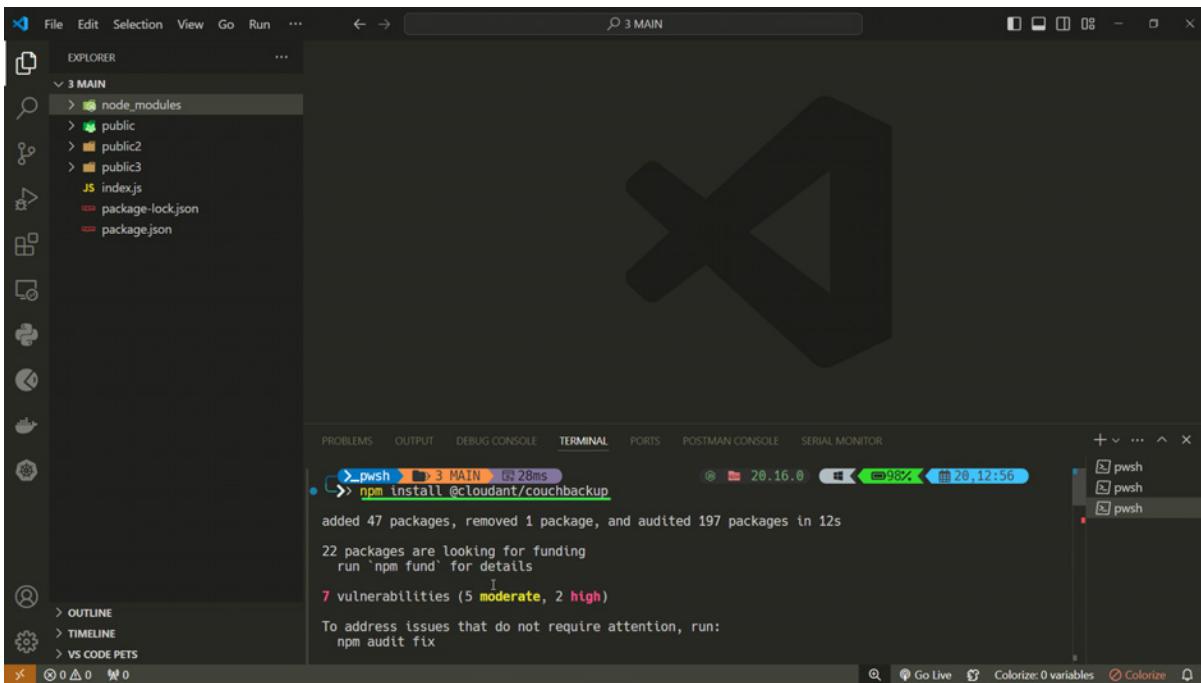
The screenshot shows the IBM Cloud Object Storage interface. On the left, there's a sidebar with 'Cloud Object Storage' selected. In the center, a file named '007.txt' is displayed. The 'Versions' tab is active. A table lists several versions of the file. The second version, which has a size of 2.2 KB and was modified on 2024-08-20 12:46 PM, has a green box around its 'Archive Type' column and a green arrow pointing to the 'Make current vers...' button in the table header.

Version ID	Archive Type	Size	Last modified
00000191-6ea8-4e61-0608-acde035cc712	Current version	0 bytes	2024-08-20 12:50 PM
00000191-6ea8-ad5d-f9b1-6b613dbe68da		2.2 KB	2024-08-20 12:46 PM
00000191-6ea4-9ab8-d5e8-1c764be44a62		2.2 KB	2024-08-20 12:46 PM
00000191-6ea0-10ec-3d3b-3c69c078a6f2		2.2 KB	2024-08-20 12:42 PM
00000191-6e9e-fc87-52cb-82f967f1ed5d		44 bytes	2024-08-20 12:39 PM

This screenshot shows the same interface after the action from the previous step. The second version of the file is now highlighted with a green box around its 'Archive Type' column, indicating it is the current version. The other versions remain listed below it.

Version ID	Archive Type	Size	Last modified
00000191-6ea8-4e61-0608-acde035cc712	Current version	0 bytes	2024-08-20 12:50 PM
00000191-6ea8-ad5d-f9b1-6b613dbe68da		2.2 KB	2024-08-20 12:46 PM
00000191-6ea4-9ab8-d5e8-1c764be44a62		2.2 KB	2024-08-20 12:46 PM
00000191-6ea0-10ec-3d3b-3c69c078a6f2		2.2 KB	2024-08-20 12:42 PM
00000191-6e9e-fc87-52cb-82f967f1ed5d		44 bytes	2024-08-20 12:39 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
```

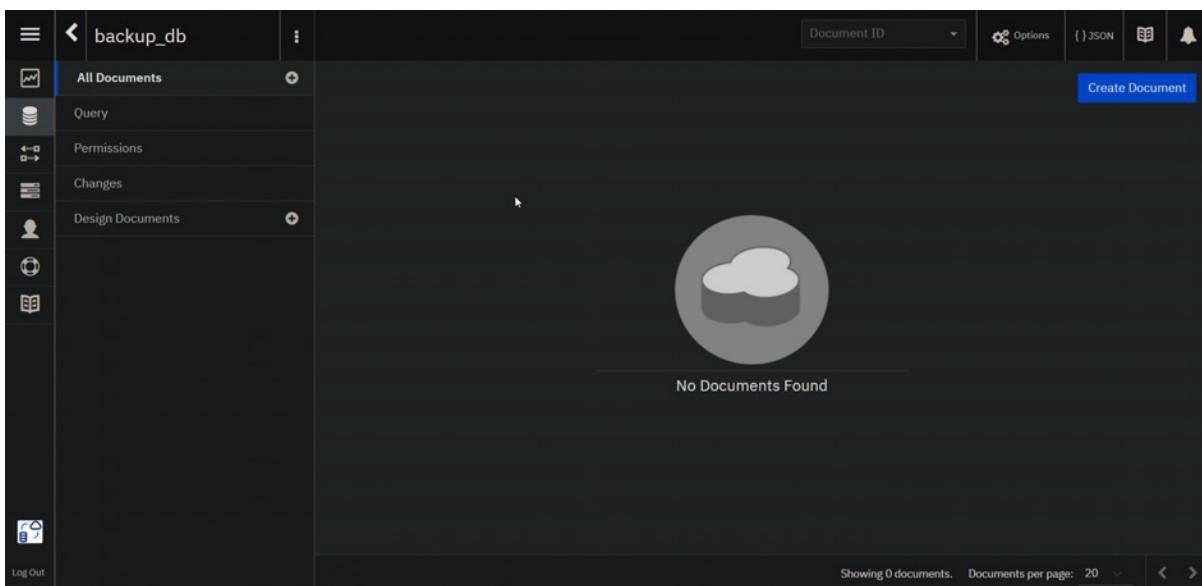
b. For global installation of couchbackup node module :

```
couchbackup --url "URL here" --db databasename > textfile
```

The screenshot shows a VS Code interface with the following details:

- EXPLORER** sidebar: Shows files like node_modules, public, public2, public3, index.js, package-lock.json, and backup.txt.
- TERMINAL** tab: Contains the command `couchbackup --url "https://apikey-v2-26aj3ozdvr2f1cmuw0xky1mbh1ubclpomovyrj1:531f952562337579bc0b3ef8cfe14bc@ed643cff-2d52-40c5-b33d-71ad7ff9e885-bluemix.cloudantnosqldb.appdomain.cloud" --db exam > backup.txt` and its output, which includes document counts and revision details.
- OUTPUT** tab: Shows logs from the couchbackup process.
- PROBLEMS** tab: Empty.
- PORTS** tab: Shows port 28138 is in use by a process.
- POSTMAN CONSOLE**: Not visible in the screenshot.
- SERIAL MONITOR**: Not visible in the screenshot.
- TERMINAL** status bar: Shows the command being run, the host (20.16.0), and the timestamp (28.13:02).
- RIGHT SIDE**: Shows a sidebar with tabs for pwsh, pwsh, and pwsh.

➤ **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
```

```
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-v2-26aj3ozdvr2f1cmuwbxkyllmhbiubctlpomovvpri:531f952562337579bc0b3ef8cf"
eal4bc@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"/>	8bdc2e44d5790c6a14ee92...	James Bond			

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