Phase-4

Data Backup and Recovery with IBM Cloud Object Storage

PHASE-4 PROJECT SUBMISSION

College Name : RR Institute of Technology

Group Members:

- **Name :** Abhinav Singh
  CAN ID Number : CAN_33714690
  Work : Recovery System Development
- **Name :** Hemashree S
  CAN ID Number : CAN_33795007
- **Name :** Chandu Shree R N
  CAN ID Number : CAN_33556996
- **Name :** Chandrashekar H
  CAN ID Number : CAN_33561505

GitHub Link : https://github.com/Abhirudra1/Data-Backup-and-Recovery-System

Video Link : https://drive.google.com/file/d/1r2gHsJWJ2-CvcoUtY-c8pXPebgGbI67R/view?usp=drive_link

---

1. Overview of Backup and Recovery Implementation:

In this phase, we focus on deploying the **Data Backup and Recovery System** using **IBM Cloud Object Storage**. The goal is to ensure data durability, accessibility, and recovery in case of disasters or data corruption. The implementation involves configuring secure storage buckets, automating backups, and developing user-friendly interfaces for seamless recovery and interaction.

2. Configuring IBM Cloud Object Storage:

2.1 Steps to Set Up Object Storage

1. **Create and Configure Buckets**:
   - **Step 1**: Log in to the **IBM Cloud Dashboard** using your credentials.
   - **Step 2**: Navigate to the **Object Storage** section and create a new instance if it doesn't already exist.

- **Step 3**: Create a storage bucket named databackupandstoragesystem with the following settings:
  - **Storage Class**: Standard (for frequently accessed data).
  - **Region**: Dallas (us-south).
  - **Resource Group**: Default.

2. **Apply Access Control and Encryption Settings**:
   - **Access Control**:
     - Used **IBM Cloud IAM** to assign roles to users and service IDs.
     - Restricted public access to ensure the bucket remains private.
   - **Encryption**:
     - Enabled **Server-Side Encryption (SSE)** using IBM-managed keys to encrypt all objects stored in the bucket.

2.2 Integrating Backup Scripts

To automate backups, we used the **IBM COS SDK for Node.js** to interact with IBM Cloud Object Storage programmatically.

1. **Install the SDK**:

```
npm install ibm-cos-sdk
```

2. **Sample Code for File Upload**:

```
const AWS = require('ibm-cos-sdk');
const cos = new AWS.S3({
  endpoint: 's3.us-south.cloud-object-storage.appdomain.cloud',
  apiKeyId: 'your-api-key',
  ibmAuthEndpoint: 'https://iam.cloud.ibm.com/identity/token',
  serviceInstanceId: 'your-service-instance-id',
});
const uploadFile = async (filePath, bucketName, objectName) => {
  try {
    await cos.putObject({
      Bucket: bucketName,
      Key: objectName,
      Body: fs.readFileSync(filePath),
    }).promise();
    console.log(`File uploaded successfully: ${objectName}`);
  } catch (error) {
    console.error(`Error uploading file: ${error}`);
  }
};
// Usage
uploadFile('local_data.csv', 'databackupandstoragesystem', 'backup_data/local_data.csv');
```

3. **Scheduling Backups**:
    - Used **cron jobs** (Linux) to schedule periodic backups (e.g., daily at 2 AM):

      *0 2 * * * node /path/to/backup_script.js*

3. Recovery System Development:

3.1 REST API for Recovery

We developed a **REST API** using **Express.js** to handle data retrieval from IBM Cloud Object Storage.

```
app.get('/restore/:filename', async (req, res) => {
  try {
    const filename = req.params.filename;
    const params = {
      Bucket: 'databackupandstoragesystem',
      Key: filename,
    };
    const data = await cos.getObject(params).promise();
    res.setHeader('Content-Disposition', `attachment; filename=${filename}`);
    res.send(data.Body);
  } catch (error) {
    console.error(error);
    res.status(500).send('Error restoring file.');
  }
});
```

3.2 Testing the API:

- Tested the /restore/:filename endpoint using **Postman** to ensure files could be retrieved successfully.

4. User Interface Development:

4.1 Building a Recovery Dashboard

We developed a **React.js** frontend to provide a user-friendly interface for managing backups and recoveries.

1. **Key Features**:
    - **File Upload**: Users can upload files to the backup system.
    - **File List**: Displays all uploaded files with options to download or delete them.
    - **File Download**: Allows users to download files from the backup system.
    - **File Deletion**: Enables users to permanently delete files from the backup system.
2. **Sample Code for File List**:

```
const [files, setFiles] = useState([]);
const fetchFiles = async () => {
  try {
    const response = await axios.get('http://localhost:5000/files');
    setFiles(response.data);
  } catch (error) {
    console.error('Error fetching files:', error);   }};
```

```
useEffect(() => {
  fetchFiles();
}, []);
```

5. IBM Cloud Platform Features and Considerations:

1. **Scalability**:
    - IBM Cloud Object Storage supports automatic scaling to handle large datasets.
    - Best Practice: Monitor usage and plan for proactive scaling during peak loads.
2. **Security**:
    - Enabled **IAM policies** and **encryption** to protect sensitive data.
    - Best Practice: Use least privilege policies and enforce MFA for admin roles.
3. **Monitoring**:
    - Used **IBM Cloud Monitoring** to track API calls and storage usage.
    - Best Practice: Set thresholds for critical metrics and use dashboards for trend analysis.
4. **Cost Efficiency**:
    - Optimized storage costs by selecting the appropriate storage class (Standard for frequently accessed data).
    - Best Practice: Analyze access patterns and configure automated transitions to lower-cost classes.

6. Conclusion:

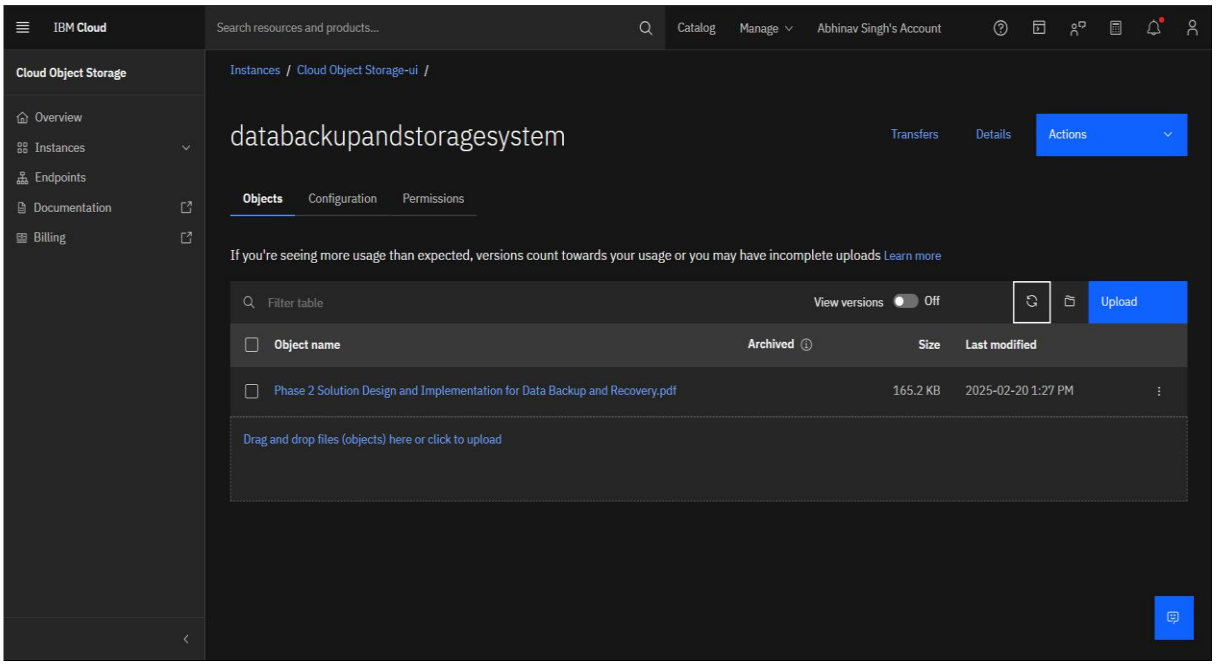
The Data Backup and Recovery System has been successfully deployed using IBM Cloud Object Storage, Express.js, and React.js. The system ensures data protection and easy retrieval through automation and user-friendly interfaces. By leveraging IBM Cloud's scalable and secure platform, the project is ready for deployment in production environments.

7. Further Enhancements:

1. **Automated Backup Scheduling**:
    - Implement an automated scheduling system that backs up data at regular intervals (e.g., daily, weekly).

2. **Backup Encryption**:
   - Encrypt backup files before uploading them to IBM Cloud Object Storage for enhanced security.
3. **Versioning for Backup Files**:
   - Enable versioning on the IBM Cloud Object Storage bucket to retain multiple versions of backup files.
4. **Backup Health Monitoring**:
   - Set up logging or monitoring to track backup status, errors, or success.
5. **Data Integrity Check**:
   - Implement a checksum or hash comparison to ensure the integrity of backup files before restoring them.
6. **Recovery Time Objective (RTO) and Recovery Point Objective (RPO)**:
   - Fine-tune backup intervals and ensure minimal downtime during recovery to meet RTO and RPO requirements.

Screenshots



databackupandstoragesystem

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

Object name	Archived	Size	Last modified
Phase 2 Solution Design and Implementation for Data Backup and Recovery.pdf		165.2 KB	2025-02-20 1:27 PM

Fig. - Bucket for Storing the objects/data

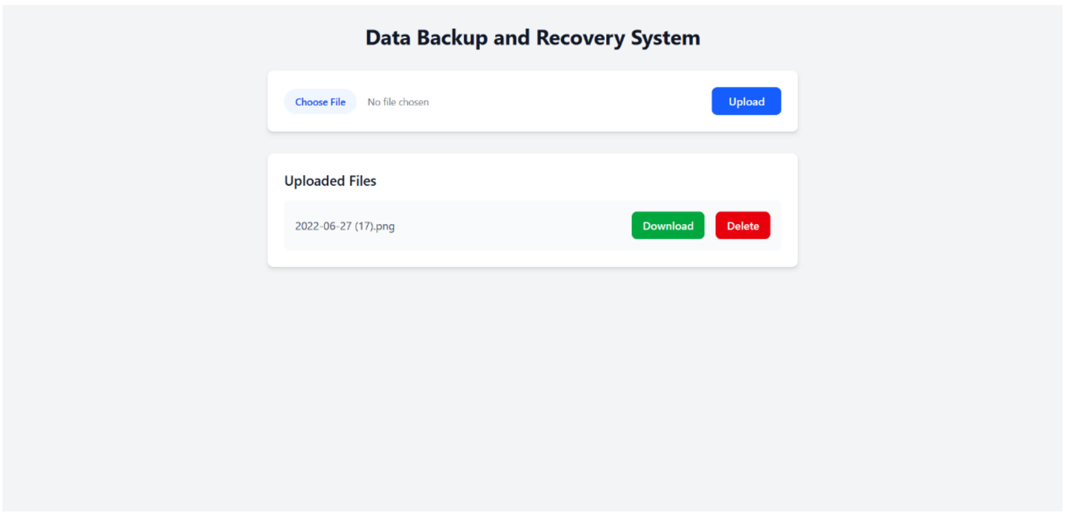


Fig. User Interface for Data Backup and Recovery System