Phase 3

Project Title: *Data Backup and Recovery System Using Cloud Object Storage*

1. Select Tools/Platforms Suitable for the Project Type

When building a MongoDB backup and recovery system using IBM Cloud, selecting appropriate tools and platforms is crucial for achieving efficiency, reliability, and scalability.

i. IBM Cloud Object Storage

- **Purpose:** Used for securely storing MongoDB backup files in the cloud.
- **Key Features:**
  - **High Durability and Availability:** Ensures data remains available even in case of hardware failures.
  - **Multi-Region Accessibility:** Allows backup access from multiple geographic locations.
  - **Storage Classes:** Options for Standard, Vault, and Cold Vault storage to optimize cost based on access frequency.
- **Role in Project:** Primary storage location for backup archives.

ii. MongoDB

- **Purpose:** Database system for storing structured and semi-structured data.
- **Key Features:**
  - **mongodump:** Tool for creating backups of MongoDB collections and databases.
  - **mongorestore:** Tool for restoring data from backups.
- **Role in Project:** Source of data for backup and the target for recovery operations.

iii. IBM Cloud CLI (Command Line Interface)

- **Purpose:** Command-line interface to interact with IBM Cloud services.
- **Key Features:**
  - Ability to manage buckets and perform file uploads/downloads.
  - Simplified authentication using API keys.
- **Role in Project:** Used for automating interactions with IBM Cloud Object Storage for backup uploads and recovery downloads.

iv. PowerShell Scripting

- **Purpose:** Automates backup and recovery operations.
- **Key Features:**
  - Automates scheduling via Windows Task Scheduler.
- **Role in Project:** Powers the automation of MongoDB backup creation, compression, upload, download, and restoration.

2. Develop the Product Using Cloud Tools and Services

This section outlines how cloud tools and services were integrated to build the MongoDB Backup and Recovery system.

Step 1: Provisioning Resources

- **IBM Cloud Object Storage Bucket Creation:** A bucket named mongodb-data-backup-recovery was created to store backup files.
- **Resource Allocation:** Sufficient storage space was allocated based on expected backup sizes.

Step 2: MongoDB Backup Creation

- **Tool Used:** mongodump
- **Process:**
  - Data from MongoDB collections was exported using the mongodump tool.
  - A structured directory was created to hold BSON and metadata files.
- **Compression:**
  - The backup folder was compressed into a .zip file using PowerShell commands.

Step 3: Backup Upload to IBM Cloud Object Storage

- **Tool Used:** IBM Cloud CLI
- **Process:**
  - The .zip file was uploaded securely to the bucket mongodb-data-backup-recovery.
  - Verification was performed to ensure the backup file was successfully uploaded.

Step 4: MongoDB Recovery

- **Tool Used:** mongorestore
- **Process:**
  - The backup file was downloaded from IBM Cloud Object Storage using the IBM Cloud CLI.
  - Files were extracted to a temporary location.
  - Data was restored using the mongorestore tool.
- **Verification:**
  - Collections were validated manually using **MongoDB Compass** and **mongosh**.

Step 5: Automation and Scripting

- **Scripts Developed:**
  - **Backup Script:** Handles automated backups, compression, and uploads.
  - **Recovery Script:** Handles automated downloads, extraction, and restoration.
- **Scheduling:** Scripts were configured to run automatically using **Windows Task Scheduler**.

3. Ensure Proper Integration of APIs or SDKs

i. **Authentication and API Integration:** API keys were generated via IBM Cloud to authenticate all CLI operations. Authentication was tested for both upload and download commands.

ii. **CLI Automation:** IBM Cloud CLI commands (ibmcloud cos upload, ibmcloud cos download) were integrated into PowerShell scripts.

iii. **Environment Compatibility:** Tested the backup and recovery scripts on local Windows environments. Ensured compatibility with MongoDB Atlas.

4. Test the Solution/Product to Ensure and Fix Bugs (if any)

i. **Functional Testing**
   - Performed end-to-end tests for backup and recovery scripts.
   - Verified .zip files were uploaded and downloaded correctly.
   - Ensured database collections matched pre-backup states.

ii. **Error Simulation**
   - **Scenario 1:** Simulated missing backup files.
   - **Scenario 2:** Tested invalid API keys.
   - **Scenario 3:** Created manual data corruption scenarios.

iii. **Debugging and Issue Resolution**
   - Addressed issues such as invalid file paths, script execution failures, and permission errors.
   - Adjusted error-handling blocks in scripts.

5. Measure Performance Using Cloud Monitoring Tools or Analytics Dashboards

i. **Monitoring Metrics**
   - **Backup Time:** Monitored the time taken for backup creation and uploads.
   - **Recovery Time:** Tracked time for download and data restoration.
   - **Storage Usage:** Monitored storage consumption in IBM Cloud Object Storage.

ii. **Alerts and Notifications**
   - Configured alerts for the following Backup job failure, Excessive storage consumption and Authentication failures.

iii. **Optimization Strategies**
   - Improved script efficiency for compression and extraction. Scheduled non-peak hours for backup processes to minimize resource contention.