UNIVERZITA KOMENSKÉHO V BRATISLAVE

BACKUP SYSTEM – TEST SCENARIOS

Renis Çenga

Alberto Cortés Herranz

Eric Ayestaran Guillorme

Subject: Development of Information Systems
Project Title: Backup System for Web Applications

1. SCHEDULING AND RUNNING A FULL BACKUP

Scenario A: Scheduling a Full Backup for Server1

1. **Action**: The user opens the CLI and enters:

backup-system schedule_backup --target Server1 --type full --schedule "0 3 * * 0"

Result: The system sets up a cron job for a full backup of “Server1” every Sunday at 3 AM.

2. **Action**: The user types status in the CLI.
   Result: The CLI shows that a scheduled job for Server1 will run each Sunday at 3 AM.

Scenario B: Manually Running a Full Backup

1. **Action**: The user enters:

backup-system run_backup --target Server1

Result: The system immediately starts a **full** backup for Server1 (assuming full is the default if no --type is specified).

2. **Action**: The user watches the CLI output.
   Result: It displays logs about compressing files and, once finished, shows “Backup completed successfully.”

2. SCHEDULING AND RUNNING AN INCREMENTAL BACKUP

Scenario A: Scheduling an Incremental Backup for Database1

1. **Action**: The user opens the CLI and enters:

backup-system schedule_backup --target Database1 --type incremental --schedule "0 1 * * *"

Result: The system schedules an incremental backup of Database1 every day at 1 AM.

2. **Action**: The user types list_backups.
   Result: No previous backups appear for Database1 yet, since it hasn’t run.

Scenario B: Manually Running an Incremental Backup

1. **Action**: The user enters:

backup-system run_backup --target Database1 --type incremental

Result: The system performs an incremental dump for Database1, compresses it, and logs success.

2. **Action**: The user runs list_backups again.
   Result: The new incremental backup shows up with its backup ID, status, and file location.

3. RESTORING FROM A PREVIOUS BACKUP

Scenario: Listing and Choosing a Backup to Restore

1. **Action**: The user enters list_backups in the CLI.
   Result: The system lists all known backups, showing **backupId**, **targetName**, **backupType**, and **timestamp**.
2. **Action**: The user picks a backup ID (e.g., 12345).
   Result: The system is ready to restore from this ID.
3. **Action**: The user enters:

backup-system restore_backup --id 12345

Result: The system finds the archive with backupId=12345, restores it to the original location, and logs “Restore completed successfully.”

4. APPLYING RETENTION POLICIES AND CHECKING STORAGE

Scenario: Adjusting the Full Backups to Keep

1. **Action**: The user enters:

backup-system configure_settings --setting retention_policy --value 1

Result: The system updates its retention policy to keep only **1** full backup at a time.

2. **Action**: The user triggers or waits for a new full backup.
   Result: After the backup finishes, old backups beyond the limit of 1 are removed. The CLI shows messages like “Deleted old backup: …”.

5. ENABLING AND DISABLING A SERVER

Scenario A: Disabling a Server

1. **Action**: The user types:

backup-system disable_server --target Server1

Result: The system marks Server1 as disabled so it won't be backed up.

2. **Action**: The user tries to run a backup on Server1:

backup-system run_backup --target Server1

Result: The CLI says "Server1 is disabled. Backup aborted."

Scenario B: Enabling the Server Again

1. **Action**: The user types:

backup-system enable_server --target Server1

Result: The system sets Server1 to enabled.

2. **Action**: The user runs a backup on Server1 again:

backup-system run_backup --target Server1

Result: This time, the backup proceeds successfully.

6. VALIDATING PRE/POST-BACKUP SCRIPTS

Scenario: Validate a Script

1. **Action**: The user enters:

backup-system validate_script --path /scripts/pre_backup_server1.sh

Result: The system tests the script. If it returns exit code 0, the CLI shows "Script validated successfully."

2. **Action**: The user updates Server1's pre-backup script:

backup-system configure_settings --setting pre_backup_script --target Server1 --value /scripts/pre_backup_server1.sh

Result: The system assigns this script to Server1.

7. CHECKING SYSTEM STATUS

Scenario: Viewing All Scheduled Jobs

1. **Action**: The user enters status in the CLI.
   Result: The system shows:
   - Whether the scheduler is active.
   - Next run times for each scheduled backup.
   - Last completed backup times.
2. **Action**: The user verifies the displayed information matches recent backups and cron schedules.
   Result: Everything matches correctly.

8. BACKUP WITH SYMBOLIC LINKS

Scenario: Handling Symbolic Links in a Backup

1. **Action**: The user creates a directory with symbolic links and sets it as a backup target (e.g., Server2).
2. **Action**: The user enters:

backup-system run_backup --target Server2

Result: The system correctly stores the **links** themselves in the archive rather than copying the original files multiple times. The CLI logs “Backup completed successfully.”

9. BACKUP UPLOAD TO REMOTE SFTP STORAGE

Scenario: Uploading a Backup to an SFTP Server

1. **Action**: The user configures valid SFTP credentials in the config.yaml (host, user, password, etc.).
2. **Action**: The user runs any backup (e.g., run_backup --target Server1).
3. **Action**: The system finishes compressing the backup and attempts to upload it.
   Result: The CLI logs “Backup uploaded to remote location” if everything is correct. If credentials fail, it logs an error.

10. INVALID CONFIGURATION ERROR HANDLING

Scenario: Starting the System with a Bad Config

1. **Action**: The user supplies a configuration file missing required fields (like remoteStorage.host or a cron expression).
2. **Action**: The user tries to run the system
   Result: The system detects that required fields are missing, logs an error (e.g., “Error: Missing host for remote storage”), and does **not** proceed with backups or scheduling.