Project 4: Automated Backup & Rotation Script with Google Drive

Intern: Shyam Chamapkbhai Chotaliya **Company:** Fortune Cloud Technologies **Date:** [14-07-2025]

Project Objective

The objective of this project was to develop a fully automated backup management system using a shell script on a Linux environment. The system is designed to:

1. Create compressed `.zip` archives of a specified project directory.
2. Securely upload these archives to a designated folder in Google Drive using `rclone`.
3. Implement a rotational backup strategy to automatically delete old backups, conserving storage space.
4. Send a success or failure notification to a webhook endpoint using `curl` after each backup attempt.
5. Run automatically on a daily schedule using `cron`.

This solution provides a robust, low-cost, and reliable method for protecting important project data.

Tools Used

- **Host System:** AWS EC2 Instance (Ubuntu 22.04 LTS)
- **`rclone`:** A command-line tool used to sync files and directories with cloud storage providers. It was used here to authorize access and upload backups to Google Drive.
- **`zip`:** A standard Linux utility used to compress the project directory into a single `.zip` archive.
- **`curl`:** A tool to transfer data from or to a server, used here to send a `POST` request with a JSON payload to a webhook URL for notifications.
- **`cron`:** A standard Linux job scheduler used to automate the execution of the backup script on a daily basis.

Setup and Implementation Instructions

1. `rclone` Configuration with Google Drive

`rclone` was configured to connect to a Google account by running `rclone config` on the EC2 instance. Since the server is headless (no GUI), the "auto config" option was disabled. An authorization link was run on a local machine with a browser, and the resulting verification code was pasted back into the EC2 terminal to grant `rclone` access. The remote was named `gdrive`.

2. Shell Script (`backup.sh`) Logic

The script performs the following actions in sequence:

1. **Timestamping:** Creates a unique filename for the backup using the current date and time (e.g., `backup_2025-07-14_103000.zip`).
2. **Archiving:** Uses the `zip` command to compress the target project directory into the timestamped `.zip` file.
3. **Uploading:** Uses `rclone copy` to upload the newly created archive to a specific folder (`Automated_Backups`) in Google Drive.
4. **Notification:** After a successful upload, it uses `curl` to send a JSON payload to a pre-configured webhook URL, indicating the status, filename, and size of the backup.
5. **Cleanup & Rotation:** It deletes the local `.zip` file to save space and then applies a retention policy by deleting old backups from Google Drive based on daily, weekly, and monthly rules.

3. Retention Rotation Implementation

The rotation logic is implemented using `rclone lsf`, `grep`, `sort`, `tail`, and `rclone deletefile`.

- **Daily:** Keeps the last 7 daily backups.
- **Weekly:** Keeps the last 4 backups made on a Sunday.
- **Monthly:** Keeps the last 3 backups made on the 1st of the month. This tiered approach ensures a balance between granular recovery points and long-term storage efficiency.

4. Scheduling with `cron`

The script was scheduled to run automatically by editing the crontab with `crontab -e` and adding the following entry: `30 2 * * * /bin/bash /home/ubuntu/scripts/backup.sh >> /home/ubuntu/scripts/backup.log 2>&1` This command executes the script every day at 2:30 AM and logs all output to `backup.log` for troubleshooting.

How to install and configure rclone

`rclone` is the core utility used to communicate with Google Drive.

1. **Installation:** Install `rclone` on an Ubuntu system using the following command:

```
sudo apt update && sudo apt install rclone -y
```

2. **Configuration (`rclone config`):**
    - Run `rclone config` in the terminal to start the interactive setup.
    - Create a **New remote** (`n`) and give it a name (e.g., `gdrive`).
    - Select the storage type `drive` from the list.
    - Leave `client_id` and `client_secret` blank to use rclone's default.
    - Choose scope `1` for full read/write access.

- When asked to **Use auto config?**, select **no (`n`)** because we are on a headless server (like EC2).
- `rclone` will provide a command (`rclone authorize ...`). Run this on your personal machine (with a web browser) to authenticate with your Google account.
- After granting permissions, a verification code will be displayed. Copy this code and paste it back into the EC2 terminal.
- Complete the remaining steps to save the remote.

3. **Verification:** Confirm access by listing the contents of your Google Drive:

```
rclone lsd gdrive:
```

Importance of Each Component

- **Backup Automation:** Automating the process with a script and `cron` eliminates human error and ensures backups are performed consistently and reliably without manual effort.
- **Retention Policy:** A rotation strategy is crucial for managing storage costs and preventing unlimited data growth. It ensures that storage space on Google Drive is used efficiently.
- **Webhook Notification:** The notification system provides vital visibility. It confirms that backups are succeeding and can be configured to send alerts on failure, allowing for immediate action.

Testing

The system was tested by:

1. Running the script manually to confirm the successful creation and upload of the backup.
2. Checking the target Google Drive folder to verify the backup file's presence.
3. Monitoring the `webhook.site` endpoint to ensure the JSON payload was received correctly.
4. Creating "fake" old backup files on Google Drive using `rclone touch` and re-running the script to verify that the rotation logic correctly deleted them.

List of Included Files

- **README.md:** This report.
- **backup.sh:** The complete, commented shell script.
- **sample-webhook.json:** An example of the JSON payload sent by the script.
- **cron-entry.txt:** The crontab entry used for scheduling.
- **/screenshots/:** A folder containing screenshots of the process.

> ***Note:*** *This submission covers only Project 4 out of the 6 assigned internship projects. The remaining projects are in progress and will be submitted separately.*