Data base backup and recovery strategy

Derakhshan Radbareh, Yaroslov Oleinychenko

Table Of Contents

1. Overview

This document outlines the strategy for keeping the PostgreSQL database used by a Netflix-like API safe, secure, and recoverable in case of data loss or unexpected failures. This system is crucial for managing user data, subscription information, streaming history, and operational metrics that ensure the service runs smoothly. Given the dynamic nature of the platform, a robust, efficient, and frequent backup plan is essential.

2. Why is a Backup Strategy Necessary?

In the context of a Netflix-like API, the database is the backbone of the service. It stores critical information, including:

- User accounts and profiles
- Subscription statuses and billing details
- Streaming history (e.g., "Continue Watching" lists)
- Recommendations tailored to user preferences
- Authentication and authorization records

Losing this data could:

- Disrupt the user experience
- Lead to financial inaccuracies
- Harm the reputation of the platform

A reliable backup strategy ensures that:

- Data can be recovered quickly with minimal disruption
- Downtime and data loss are limited to the smallest possible window
- Regulatory and legal requirements for data protection are met

3. Why Daily Backups?

The database experiences constant changes through:

- User logins and content interactions
- Subscription updates
- Profile modifications
- Recommendation data generation

A backup window longer than one day would be unacceptable for such a dynamic system. Losing "Continue Watching" progress or subscription updates can frustrate users and damage credibility. Therefore, scheduling backups at 2:00 AM during off-peak hours minimizes system load and ensures minimal impact on user experience.

4. Backup Strategy

The backup strategy consists of the following key practices:

Schedule:

- Daily full backups at 2:00 AM
- Retention of backups for 30 days

Storage:

- Local storage on a secure server
- Cloud backup via Google Drive using rclone
- Optional periodic external drive backups

Backup Types:

- Logical backups using pg_dumpall
- Optional physical backups with pg_basebackup

Point-in-Time Recovery (PITR):

- WAL archiving enabled to restore up to a specific second between full backups

Encryption:

- AES-256 encryption at rest
- HTTPS/SSL/TLS encryption during transit

Automation:

- Use of task schedulers and scripts
- Error logging and notification setup

Space Management:

- Automatic deletion of backups older than 30 days

Backup Script Features:

- Environment variable use to secure credentials
- Auto-creation of backup directories
- Date-stamped filenames
- Error and event logging

Example Backup Script

Below is an example of a script that performs a daily full logical backup using `pg_dump`:

```
#!/bin/bash
pg_dump -U <username> -d <database_name> -F c -b -v -f "/path/to/backups/full_backup_$(date
+%F).backup"
```

5. Recovery Strategy

The Recovery Strategy focuses on rapid restoration with minimal data loss:

- **Recovery Time Objective (RTO):** Full database recovery within two hours
- **Recovery Point Objective (RPO):** Maximum data loss limited to 24 hours
- **Recovery Process:**
  - Use pg_restore to restore the latest full backup
  - Apply WAL logs for incremental recovery

Example Recovery Command

To restore a backup using pg_restore, use the following command:

```
pg_restore -U <username> -d <database_name> -v "/path/to/backups/full_backup_<backup_date>.backup"
```

- **Testing:**
  - Monthly recovery drills
  - Integrity checks on recovered data
- **Failover Readiness:**
  - Streaming replication to maintain a hot standby server
  - Consideration of automatic failover mechanisms (e.g., Patroni)

6. Security

Security measures are crucial and include:

- **Encryption:**
  - Encrypt backups at rest (AES-256)
  - Encrypt backups in motion (HTTPS/SSL)
- **Access Control:**
  - Restrict access to authorized personnel only
  - Use strong authentication methods for backup system access
- **Audits:**
  - Conduct regular security audits
    - Review storage locations and access logs periodically

7. Monitoring and Alerts

Monitoring ensures system health and backup reliability:

- Use of monitoring tools like pgAdmin, Zabbix, and Prometheus
- Tracking backup job success and failure rates
- Monitoring replication lag and WAL archive space
- Setting up alerts for backup failures or replication issues

8. Documentation

Proper documentation practices are maintained:

Backup Procedures:

- Maintain updated schedules and retention policies
- Document storage locations and backup types

Scripts and Automation:

- Version-control and audit all backup and recovery scripts

Testing and Recovery:

- Keep logs of recovery tests and results

Contacts:

- Maintain clear emergency contact lists for escalation

Review Process:

- Conduct quarterly reviews to update strategies

9. Conclusion

A comprehensive backup and recovery strategy is critical for protecting the database of a Netflix-like API. By implementing daily full backups, enabling point-in-time recovery, securing data through encryption, maintaining replication and monitoring systems, and ensuring thorough documentation, the platform can achieve high levels of data protection, minimize the risk of loss, and maintain consistent business operations. This structured approach ensures the resilience and reliability necessary for sustaining user trust and business continuity.