The current version is #ident
"@(#)$Format:LocalFoodAI_lanfr144:disaster_recovery_plan.md:Francois
Lange:lanfr144@school.lu:2026/06/11 08:26:59:Francois
Lange:lanfr144@school.lu:2026/06/11
08:26:59:1701828b122e0c319e59134ca6511a42ecad9297::$"

Disaster Recovery & Backup Plan

This document outlines the backup and restore procedures, as well as the Disaster Recovery (DR) plan for the Local Food AI stack.

1. Backup Procedures

Given the massive 3GB+ size of the OpenFoodFacts dataset, backing up the entire MySQL data volume dynamically is resource-intensive. The strategy is split into **Code Backup** and **Data Backup**.

1.1 Source Code & App Configuration

The entire application infrastructure (Dockerfiles, Python scripts, configuration) is tracked in the Git repository. **Backup Command:** `git push origin main` *Frequency: Triggered automatically by developers after every Sprint.*

1.2 Database (MySQL) Backup

We use `mysqldump` to create a cold-standby physical backup of the user data and dietary profiles, while ignoring the massive, immutable OpenFoodFacts partition (which can be re-ingested from the source CSV).

Backup Command:

```
sudo docker exec food_project-mysql-1 mysqldump -u root -proot_pass food_db users
user_health_profiles plate_items > /backup/food_db_users_$(date +%F).sql
```

Frequency: Daily via a server-side cron job (`0 3 * * *`).

2. Restore Procedures

2.1 Database Restore (Warm Recovery)

If the database container crashes or the volumes are corrupted:

- Stop the application container to prevent write conflicts: `sudo docker-compose stop app`
- Wipe and re-initialize the MySQL container.
- Restore the user tables from the SQL dump:

```
cat /backup/food_db_users_2026-05-12.sql | sudo docker exec -i food_project-
mysql-1 mysql -u root -proot_pass food_db
```

4. Restart the background ingestion script (`./data_sync.sh`) to rebuild the massive 3GB OpenFoodFacts `products_core` tables. 5. Restart the application: `sudo docker-compose start app`