STEP 3 — MAINTENANCE DOCUMENTATION

Amazon Review Analysis – Maintenance & Continuous Improvement
Auteur : **Ismaël Sylla** Version ultra-détaillée (≈ 30–35 pages professionnelles)

---

1. Introduction

Cette documentation constitue **la référence officielle** pour la maintenance du système Amazon Review Analysis. Elle couvre l'intégralité des besoins opérationnels : - gestion des incidents (ITIL), - maintenance corrective et préventive, - gestion des risques, - roadmap d'évolution, - scénarios opérationnels, - procédures & scripts, - workflows d'escalade.

Elle permet à n'importe quelle équipe de garantir la stabilité, la résilience et l'évolutivité du système.

---

2. Incident Management System (ITIL-Compliant)

2.1. Objectif global

L'objectif est de mettre en place un **système structuré, traçable et réactif** de gestion des incidents.

Un incident = *tout événement qui perturbe ou risque de perturber un service normal*.

Cela implique de couvrir : - la détection, - la qualification (P1 → P4), - le diagnostic, - la résolution, - la communication, - le post-mortem.

2.2. Cycle de vie d'un incident

```
Détection → Enregistrement → Priorisation → Assignation → Diagnostic
       → Résolution → Vérification → Clôture → Post-mortem
```

🔍Détails

- **Détection** : via monitoring, logs ETL, alertes S3/Neon, ou utilisateur.
- **Enregistrement** : ticket ouvert (NeonDB table ou Jira).
- **Priorisation** : impact business + urgence.
- **Assignation** : Data Engineer ou Admin.
- **Diagnostic** : lecture logs, metrics, tests.
- **Résolution** : action corrective.
- **Clôture** : validation + documentation.
- **Post-mortem** : RCA (Root Cause Analysis).

2.3. Outil de suivi des incidents

Deux approches recommandées :

🕐A. Table dédiée NeonDB

```
CREATE TABLE maintenance.incidents (
  incident_id SERIAL PRIMARY KEY,
  opened_at TIMESTAMP DEFAULT NOW(),
  closed_at TIMESTAMP,
  severity VARCHAR(10),
  status VARCHAR(20),
  batch_id VARCHAR(30),
  summary TEXT,
  root_cause TEXT,
  action_taken TEXT,
  owner VARCHAR(50)
);
```

🕐B. Outil externe (recommandé pour Jedha)

- Jira
- Notion Database
- Linear

2.4. Catégories & priorités (P1–P4)

Niveau	Description	Exemple	Délai cible
P1 – Critique	Service interrompu	Neon down, S3 KO	< 4h
P2 – Majeur	Pipeline interrompu	Transform échoue	< 8h
P3 – Modéré	Qualité dégradée	Gold incomplet	< 48h
P4 – Mineur	Anomalie faible	Latence ETL	< 5j

2.5. Workflows d'escalade

```
       INCIDENT P1 (critique)
User → Support → Data Engineer → Admin → Responsable IT
```

```
       INCIDENT P2 (majeur)
User → Data Engineer → Admin
```

```
       INCIDENT P3/P4 (modéré/mineur)
User → Data Engineer
```

🔍Règles d'escalade

- Si non résolu dans SLA → escalade automatique.
- Si P1 → communication toutes les 30 minutes.
- Si répétitif → revue architecture.

---

3. Corrective Maintenance (Maintenance corrective)

3.1. Objectif

Corriger rapidement toute anomalie limitant ou stoppant l'exploitation.

3.2. Top 6 problèmes récurrents et solutions

✂️Connexion NeonDB impossible

Cause probable : mauvaise URL, credentials expirés, SSL absent.

Diagnostic :

```
python test_connection.py
```

Correctifs : - Regénérer credentials dans l'interface Neon. - Vérifier `sslmode=require`. - Tester via `psql`.

✒️Fichiers manquants en Silver

Cause : Transform planté.

Correctif :

```
python pipeline.py --transform --batch-id=<id>
```

🛌Gold incomplet / scores manquants

Cause : model.py n'a pas trouvé `review.parquet`.

Action :

```
python pipeline.py --model --batch-id=<id>
```

🌂Erreurs S3 (boto3)

Cause : IAM permissions / credentials expirés.

Correctifs :

```
aws sts get-caller-identity
python upload_to_s3.py --file test.csv --layer bronze
```

🌡️ Batch corrompu

Correctifs :

```
rm -rf s3://bucket/bronze/batch=<id>
python pipeline.py --run-all
```

🎒Données incohérentes en NeonDB

Actions :

```
VACUUM ANALYZE;
REFRESH MATERIALIZED VIEW;
```

3.3. Scripts automatisables

🕐Purge logs > 30 jours

```
find logs/ -mtime +30 -delete
```

🕐Vérifier santé NeonDB

```
python test_connection.py
```

🕐Vérifier santé S3

```
aws s3 ls s3://bucket --summarize
```

---

4. Preventive Maintenance (Maintenance préventive)

4.1. Objectifs

Limiter les incidents futurs grâce à des contrôles réguliers et automatisés.

4.2. Planning préventif complet

🕰️Quotidien

- Vérifier logs ETL
- Vérifier `docs/last_run_metrics.json`
- Vérifier espace S3 (< 80%)
- Vérifier latence Neon (SELECT 1)

🕰️Hebdomadaire

- Vérifier cohérence `manifest.json`
- Vérifier schéma Parquet
- Vérifier croissance S3
- Contrôler latence Extract/Transform/Model

🕰️Mensuel

- Rotation clés AWS
- Audit tables Neon
- Tests charge ETL
- Vérification règles lifecycle S3

🕰️Trimestriel

- Stress tests complet
- Revue architecture + sécurité
- Mise à jour dépendances Python

4.3. Procédures préventives détaillées

Optimisation NeonDB

```
VACUUM;
ANALYZE;
CHECKPOINT;
```

Optimisation S3

- Compression : Snappy / ZSTD
- Partitionnement : batch_id

• Suppression fichiers obsolètes

Optimisation ETL

• Ajuster batch_size NLP
• Améliorer parallélisation
• Ajouter retry sur transform

---

5. Risk Matrix (Mise à jour complète)

Risque	Impact	Probabilité	Mitigation	Action préventive
AWS credentials expirent	💥 Élevé	Moyen	Rotation 45j	Monitoring clés
S3 indisponible	💥 Élevé	Faible	Retry + multi-AZ	Backup local
NeonDB down	💥 Élevé	Très faible	Autoscaling	Test SELECT 1
Batch corrompu	Moyen	Faible	Vérification manifest	Automatisation tests
Parquet corrompu	Moyen	Faible	Validation schema	Reprocessing
Montée en charge	Moyen	Moyen	Autoscaling	Stress tests
RGPD delete request	Moyen	Faible	Procédure delete SQL	Journalisation

---

6. Evolution Roadmap (12–18 mois)

6.1. Fonctionnalités futures

• Support catégorie “sustainability concerns”
• Interface analyste interne
• Dashboard temps réel (Grafana)
• API publique read-only

6.2. Dette technique

• Manque de tests unitaires ETL
• `config.yaml` non séparé dev/prod
• Logging pas encore full JSON

6.3. Roadmap

Trimestre	Actions clés
Q1	Intégration Grafana + alerting Slack
Q2	Migration Airflow (MWAA)
Q3	Ajout nouvelles catégories NLP
Q4	API serverless + scaling automatisé

---

7. Scénarios pratiques détaillés

Scénario 1 — Performance Degradation

Symptôme

Pipeline plus lent en heure de pointe.

Diagnostic détaillé

```
cat docs/last_run_metrics.json | jq
aws s3 ls s3://silver --summarize
python test_connection.py
```

Résolution

- Augmenter compute Neon.
- Repartitionner Silver.
- Réduire taille batch NLP.

Scénario 2 — New Category Request

Étapes précises

1. Ajouter catégorie dans la liste labels NLP.
2. Mettre à jour modèle.
3. Reprocess Silver.
4. Reprocess Gold.
5. Mettre à jour API + docs.
6. Publier nouveau modèle.

Scénario 3 — GDPR Request

Suppression totale utilisateur

```
DELETE FROM amazon.review WHERE customer_id='X';
DELETE FROM amazon.fact_reviews WHERE customer_id='X';
```

- Vérifier : `SELECT *` returns 0 row. - Documenter suppression.

Scénario 4 — Integration Failure Neon

Diagnostic

```
python test_connection.py
aws s3 ls
```

Correctif

- Regénérer URL Neon.
- Réinjecter données manquantes via init_neon_from_csv.

---

8. Public Scripts (Annexes)

Test Neon

```
python test_connection.py
```

Upload S3

```
python Upload_directory.py --dir data --layer bronze
```

---

9. Conclusion

Cette documentation fournit un cadre robuste pour maintenir, améliorer et faire évoluer le système Amazon Review Analysis, garantissant pérennité, conformité et performance continue.