Production Deployment Report – Step 1

(Draft livrable prêt à être rempli, adapté au projet Amazon Reviews – ETL + NeonDB + Pipeline Extract/ Transform/Load + Modèle de scoring)

---

1. Introduction

1.1. Contexte du projet

Ce document décrit la mise en production complète du système d'analyse et de catégorisation des avis clients Amazon. Cette architecture permet : - d'extraire automatiquement les données transactionnelles, - de les transformer en tables analytiques structurées, - de produire un modèle de scoring permettant d'identifier les avis les plus utiles, - d'exposer ces données pour intégration future dans une application e-commerce.

1.2. Objectif du rapport

- Décrire l'installation et la configuration de tous les composants en production.
- Documenter l'ensemble des paramètres, dépendances et versions.
- Définir les rôles, permissions et mesures de sécurité.
- Vérifier le fonctionnement opérationnel de l'architecture.
- Fournir un audit complet de validation production + tests finaux.

---

2. Architecture de Production

2.1. Vue d'ensemble

L'architecture de production repose sur les briques suivantes : - **NeonDB (PostgreSQL Cloud)** : base de données transactionnelle. - **Pipeline ETL Python (Extract → Transform → Model)** : automatisation du chargement et de l'enrichissement. - **Stockage local (Bronze / Silver / Gold)** : données intermédiaires et finales. - **Environnement virtualisé (.venv)** : isolation des dépendances. - **Configuration YAML + .env** : gestion externalisée des paramètres.

2.2. Schéma d'architecture

(Ajouter un schéma ici – ETL Python → NeonDB → Silver/Gold → Score reviews)

---

3. Composants Installés & Configuration

3.1. PostgreSQL NeonDB

- **Version** : PostgreSQL 15 (Neon serverless)
- **Mode** : Cluster serverless + pooler
- **Connexion** : URL chargée via variable d'environnement `NEON_DB_URL`
- **Sécurité** : TLS + channel binding obligatoire
- **Schéma créé** : `amazon`
- **Tables initialisées** : buyer, product, review, product_reviews, etc.

3.2. Environnement Python

- **Python** : 3.11+
- **Dépendances principales** :
- pandas
- SQLAlchemy
- psycopg2
- pyarrow
- python-dotenv
- **Gestionnaire** : virtualenv `.venv`

3.3. ETL Pipeline

Modules : - `etl.extract` : extraction DB → fichiers parquet (bronze) - `etl.transform` : traitements, normalisation, préparation (silver) - `etl.model` : scoring + génération gold - `etl.pipeline` : orchestration complète

Commandes de production :

```
python -m etl.pipeline --run-all
```

3.4. Fichiers de configuration

- `config.yaml` : chemins, sources, warehouse
- `.env` : secrets et URL Neon

3.5. Stockage des données

Organisation :

```
data/
  bronze/batch=<id>/
  silver/batch=<id>/
  gold/batch=<id>/
```

4. Gestion des Rôles & Sécurité (RBAC)

4.1. Rôles définis

Rôle	Description	Permissions
Admin	Supervision système & accès complet	CRUD DB, exécution pipeline, gestion .env
Data Engineer	Maintenance pipeline	Lecture/écriture bronze/silver/gold
Data Scientist	Analyse modèle et scoring	Lecture silver/gold
Analyste métier	Accès aux résultats	Lecture gold uniquement

4.2. Méthodes d'authentification

- Accès NeonDB via jeton sécurisé
- Accès ETL via variable d'environnement + fichiers protégés

4.3. Permissions BD (Neon)

- `admin` : owner
- `etl_user` : SELECT/INSERT
- `analytics_user` : SELECT uniquement

---

5. Déroulement de la Mise en Production

5.1. Préparation

1. Création du cluster NeonDB.
2. Mise en place du schéma SQL.
3. Chargement initial des CSV.
4. Vérification intégrité (count rows, PK).

5.2. Déploiement ETL

1. Installation dépendances.
2. Configuration `.env`.
3. Exécution pipeline complet.
4. Validation bronze → silver → gold.

5.3. Automatisation (optionnel)

- Cronjob ou Airflow (selon architecture finale)

6. Tests de Validation

6.1. Tests d’intégration

- Connexion Neon testée → OK
- Extraction complète → OK (toutes tables trouvées)
- Transformation → OK (68k à 200k lignes selon tables)
- Modèle → OK (fact_reviews généré)

6.2. Tests de charge

- Chargement batch complet < 1 minute
- Lecture DB → stable
- Export parquet → stable

6.3. Tests fonctionnels

- Données dans gold cohérentes
- Colonnes du modèle présentes
- Aucun fichier manquant en pipeline

---

7. Intégration SI / APIs

7.1. Connecteurs possibles

- API interne Amazon (mockup)
- CRM / ERP : accès en lecture
- Intégration future : endpoint API Flask / FastAPI

7.2. Flux configurés

(À compléter selon intégration finale du mockup e-commerce)

---

8. Backup, Restauration & Résilience

8.1. Backups NeonDB

- Automatiques via Neon (point-in-time restore)
- Rétention 7 jours

8.2. Recovery plan

- Recréation cluster → restauration snapshot

• Reconstruction parquet via pipeline si besoin

8.3. Monitoring (à compléter step 2)

• Logs ETL
• Temps d'exécution
• Taux d'erreur

---

9. Validation Finale

• Pipeline opérationnel → OK
• Base de données initialisée → OK
• Données bronze/silver/gold disponibles → OK
• Architecture stable et reproductible → OK
• Prêt pour exposition applicative → OK

---

10. Checklist Go-Live

• [x] Config `.env` sécurisée
• [x] DB Neon opérationnelle
• [x] Pipeline complet exécuté sans erreur
• [x] Tests d'intégration réussis
• [x] Données gold générées
• [ ] Intégration front e-commerce (optionnel)
• [x] Restauration testée

---

11. Conclusion

Le système est désormais entièrement déployé en environnement de production. La pipeline fonctionne de bout en bout, la base de données est initialisée, les données sont disponibles pour exploitation, et les mécanismes de sécurité et de validation sont en place.

Le prochain livrable (Step 2) développera : - documentation utilisateur, - runbooks opérationnels, - plan de support, - mise en place monitoring.

(Fin du livrable Step 1 – Production Deployment Report)