

PROJET M1 ISEN

Sujet de Spécialité Big Data : DevOps/Data Engineering

ARCHITECTURE DE DONNÉES DISTRIBUÉES : COMMENT GÉRER UNE HAUTE VOLUMÉTRIE DE DONNÉES DE MANIÈRE SCALABLE ET PERFORMANTE

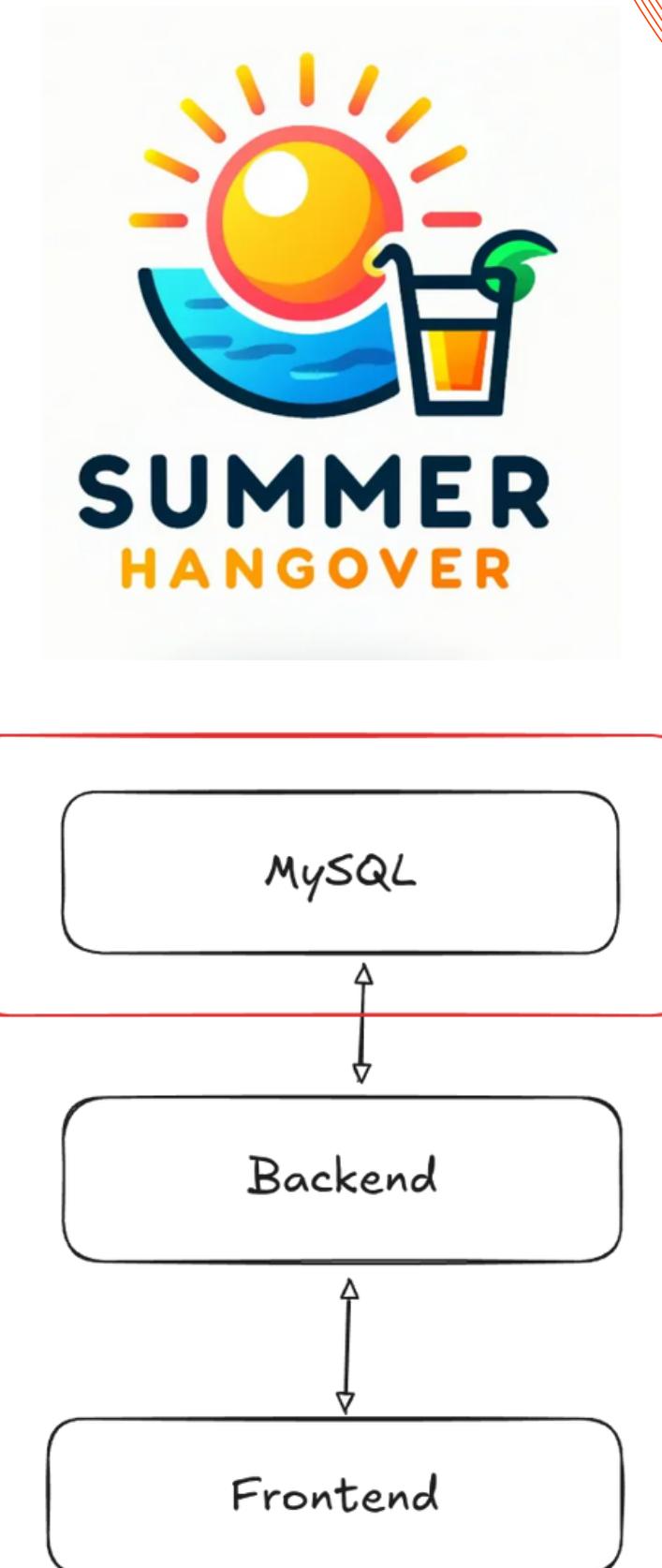
INTRODUCTION

Avant tout, d'où vient ce projet ?

- Nous avons réalisé en 3ème année un projet d'informatique ensemble
- Application web de réseau social pour créer des groupes et organiser des sorties entre amis
- Système fonctionnel, mais pensée pour le projet uniquement

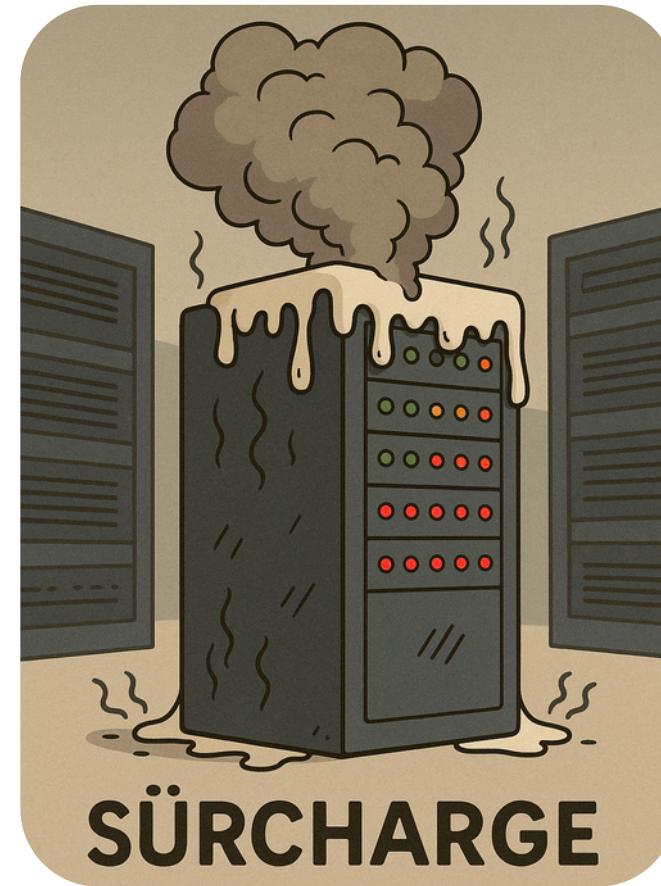
Qu'elles sont ses limites ?

- Système de stockage de données très basique
- Une unique base de données MySQL pour le stockage de toutes les informations.
- On peut juste stocker les données, mais impossible de les exploiter (Analyse, Machine Learning, Monitoring, etc...)



Conséquences :

- Inutilisable en production réel.
- Ne supporte pas l'évolution rapide des metrics.
- Oblige à utiliser du scaling vertical (très chère).
- Manque à gagner à ne pas exploiter les données.



Comment résoudre ces problèmes ?

- En passant d'une simple DB à une véritable architecture de données.
- En utilisant des techno modernes qui assurent scalabilité et résilience



OBJECTIF FINAL

Problématique : Comment passer d'une simple base de données à une architecture complète de stockage et traitement de données ?

Objectifs :

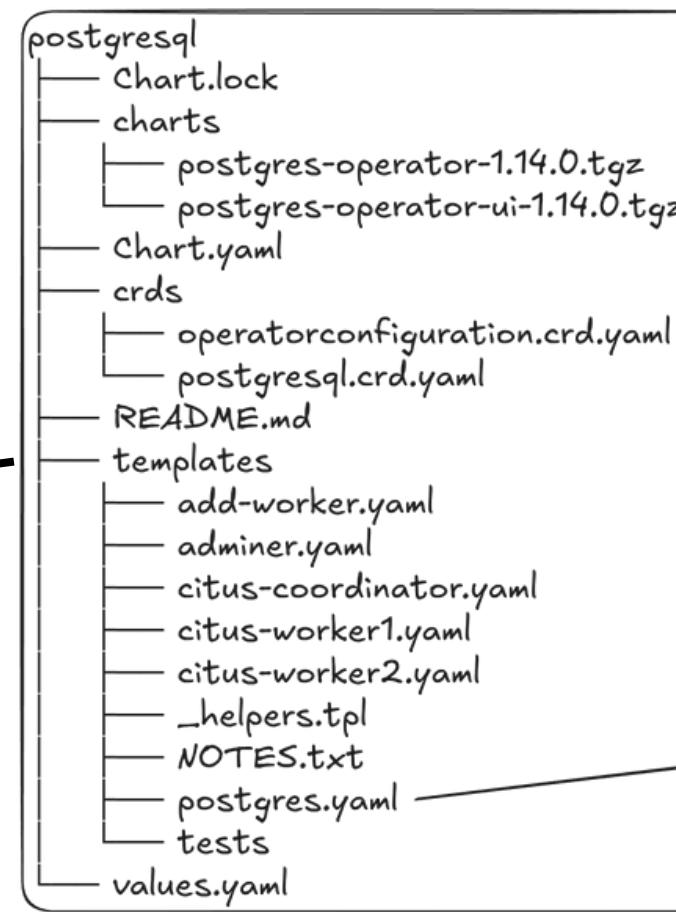
- Construire une infrastructure orchestrée, permettant d'assurer une scalabilité automatique et une forte résilience.
- Avoir un système de qualité d'entreprise, fiable, reproductible et extensible facilement.
- Non seulement stocker, mais surtout exploiter les données, pour de l'analyse, du Machine Learning, etc...



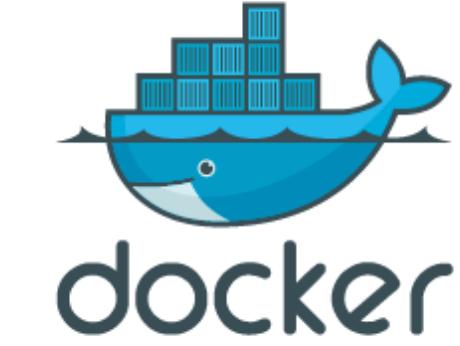
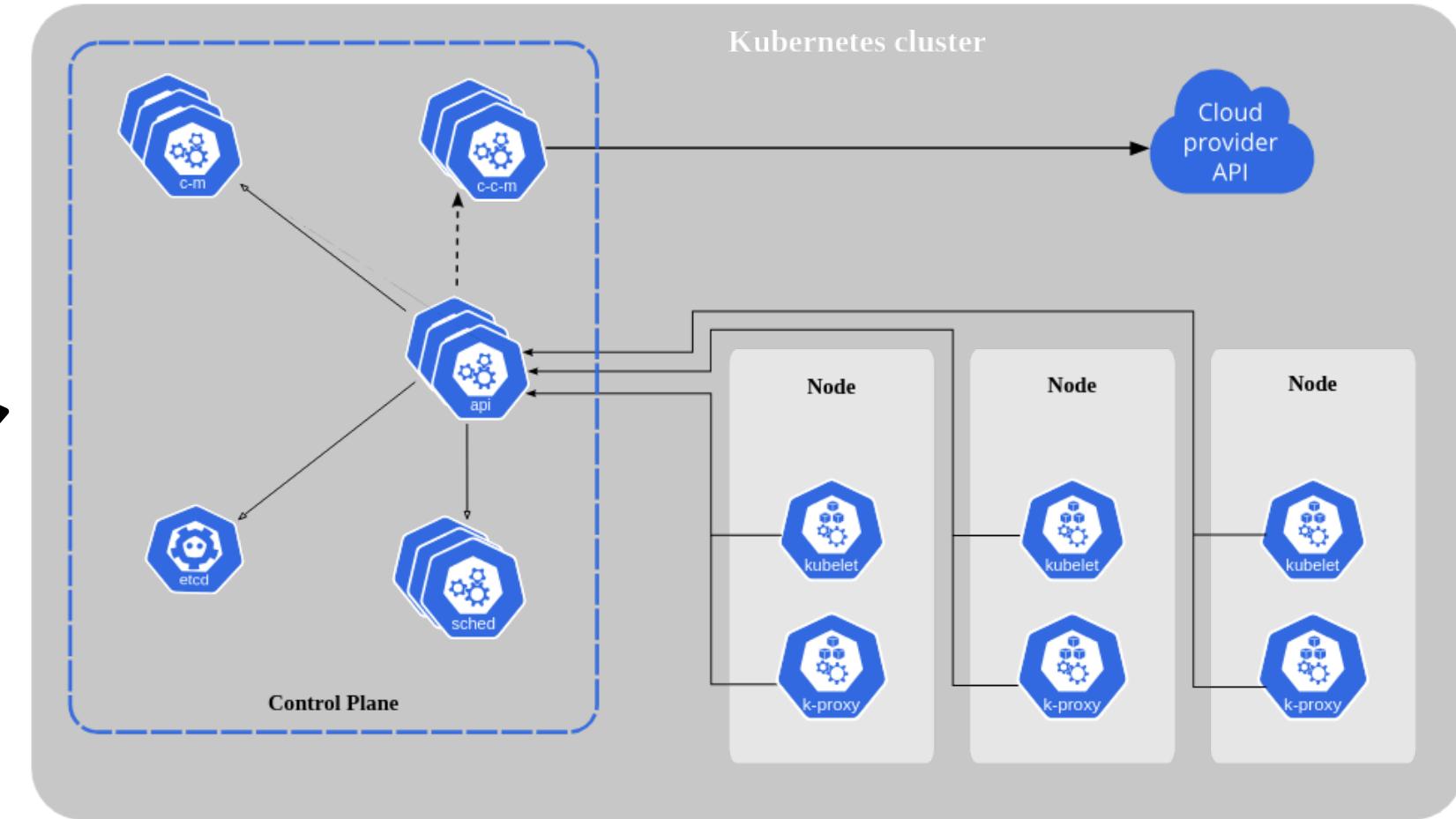
SOMMAIRE

| | |
|-----------------------------|----|
| • 1) Orchestration | 6 |
| ◦ Kubernetes / Helm | 6 |
| ◦ Architecture globale | 7 |
| • 2) Stockage des données | 8 |
| ◦ Choix de base de données | 9 |
| ◦ Mise en place | 11 |
| ◦ Model final de données | 12 |
| • 3) Pipeline de traitement | 13 |
| ◦ Concept général | 15 |
| ◦ Structure globale | 16 |
| ◦ Exemple concret | 18 |
| • 4) Conclusion | 18 |

ORCHESTRATION



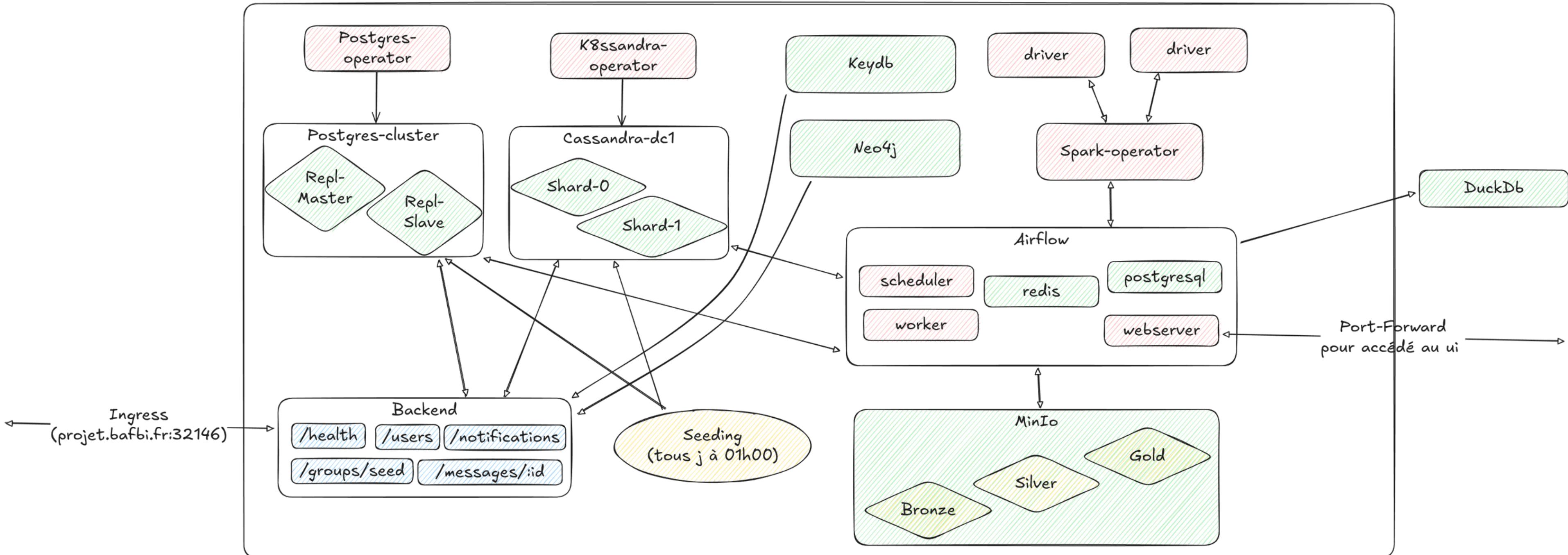
```
# postgres.yaml
apiVersion: "acid.zalan.do/v1"
kind: postgresql
metadata:
  name: postgres-minimal-cluster
  namespace: {{ .Release.Namespace }}
spec:
  teamId: "stmg"
  volume:
    size: 1Gi
  numberOfInstances: 2
  users:
    stmg: # database owner
      - superuser
      - createdb
    backend: [] # role for application foo
  databases:
    demo: stmg # dbname: owner
  postgresql:
    version: "17"
```



Application

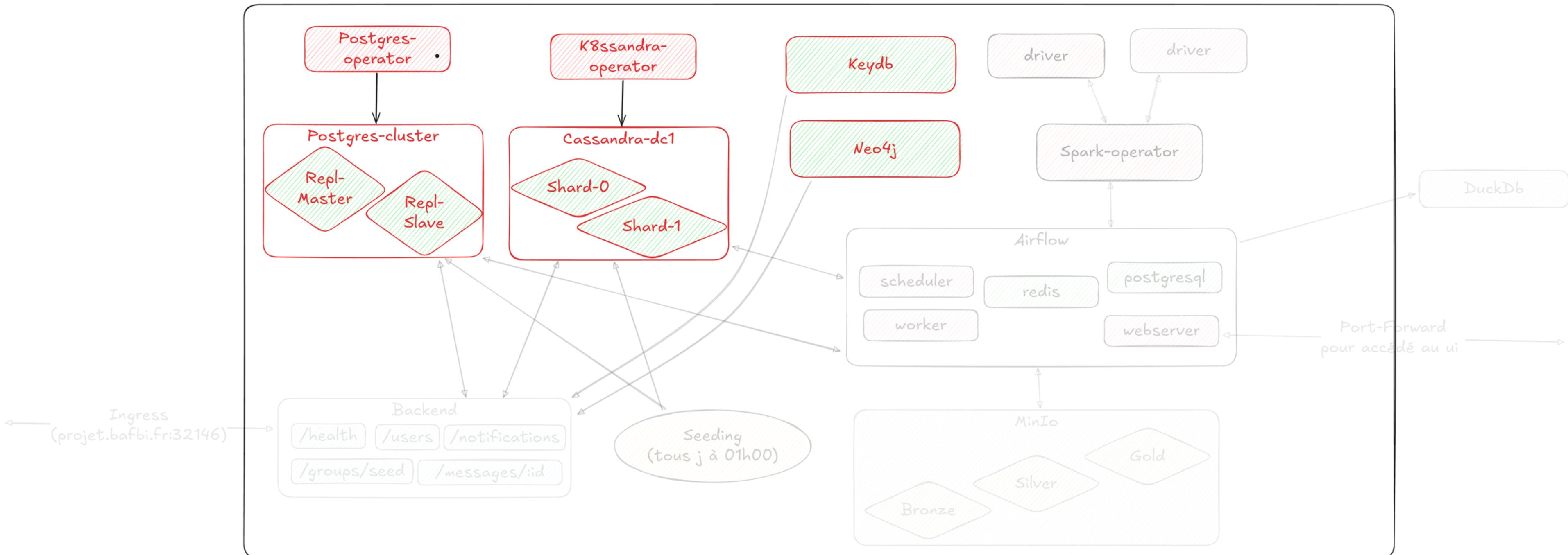
ARCHITECTURE GLOBALE

cluster kubernetes (k3s)



STOCKAGE DES DONNÉES

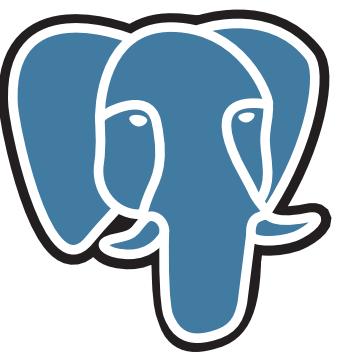
cluster kubernetes (k3s)



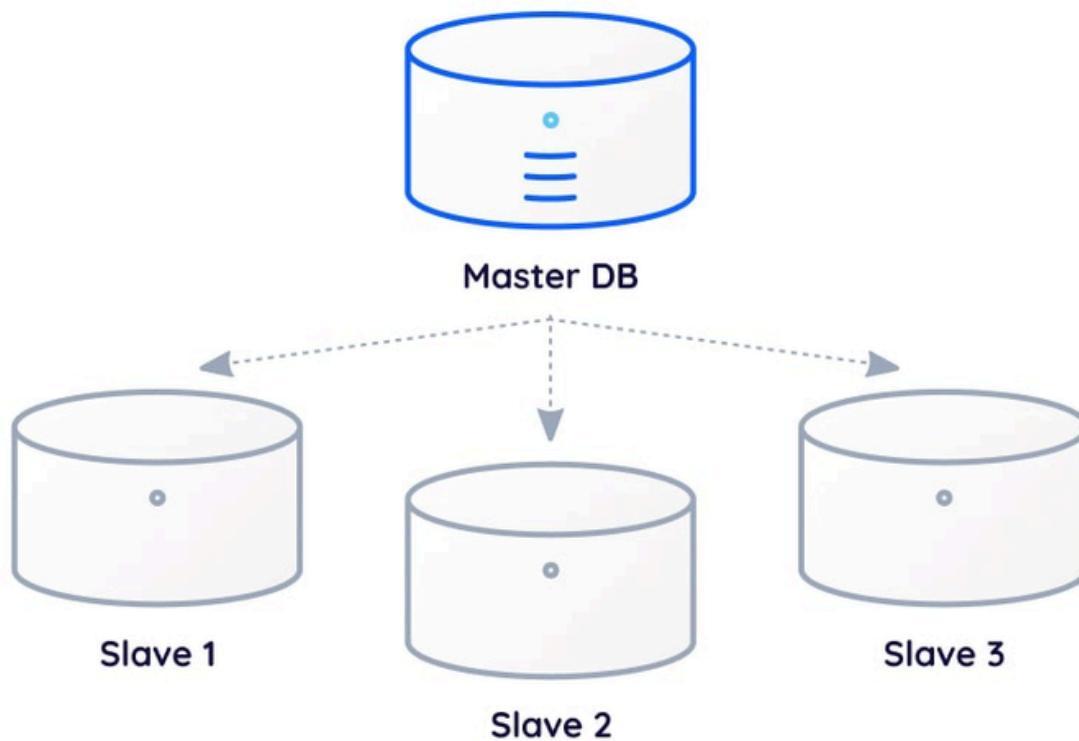
CHOIX DES BASES DE DONNÉES

1) Bases de données relationnelles : PostgreSQL

- Moderne, très puissante
- Possibilités de faire des replications de données
- Transaction ACID
- Utile pour stocker les données les plus cruciales



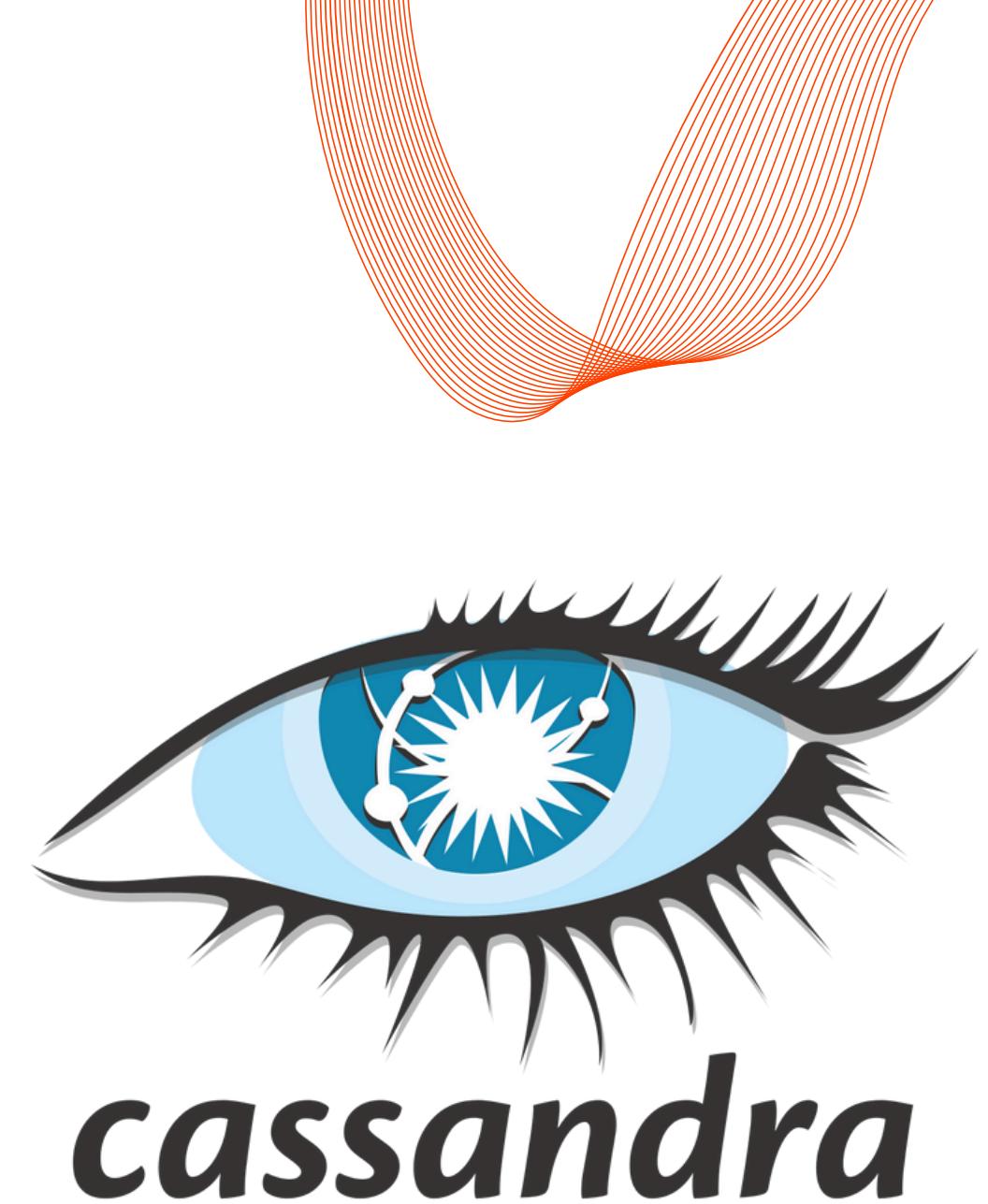
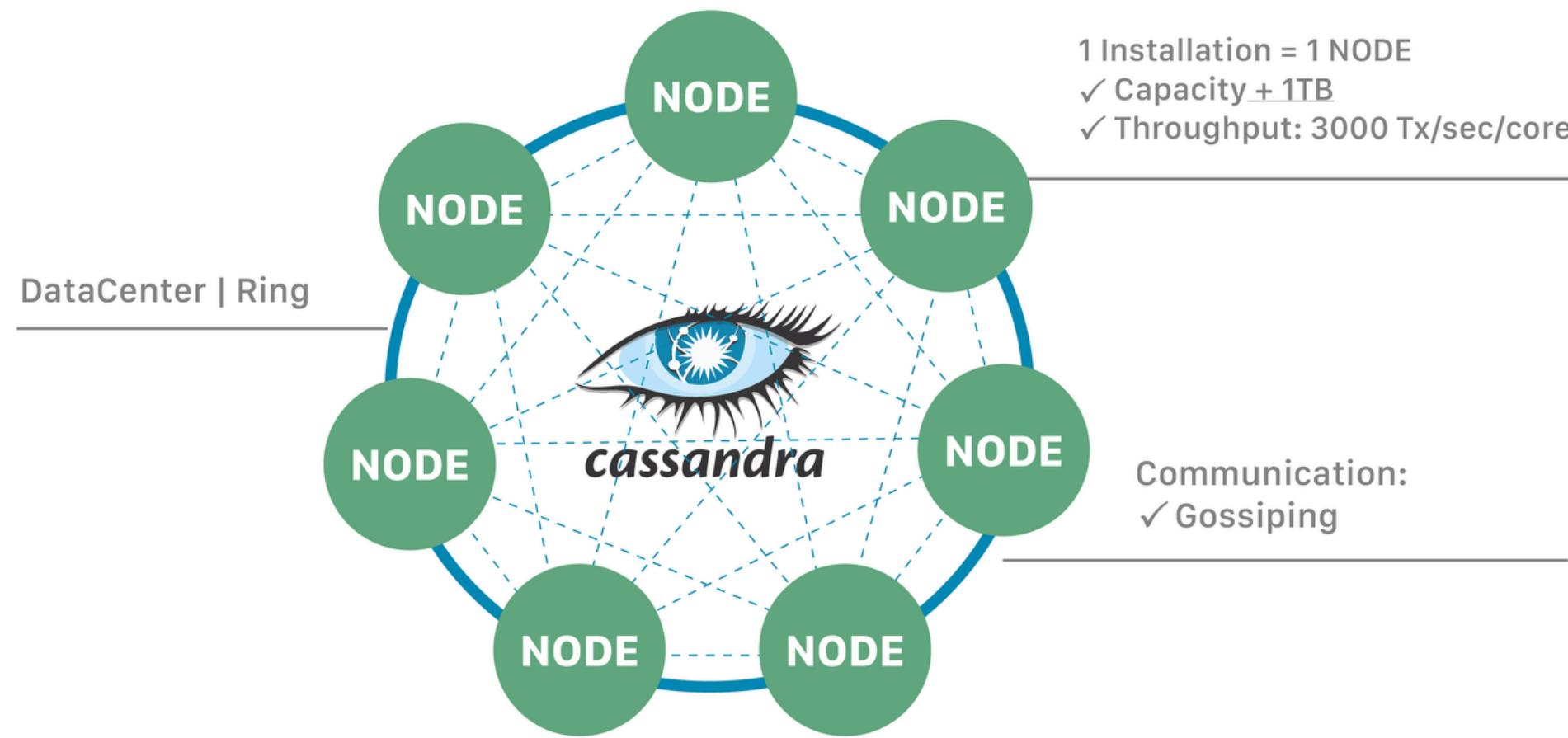
PostgreSQL



CHOIX DES BASES DE DONNÉES

1) NoSQL Orienté colonne : Cassandra

- Conçu pour des usages intensifs d'entreprise
- Ecriture ultra-rapide de nouvelles données
- Fonctionne en Peer-To-Peer
- Permet de faire du Sharding naturellement



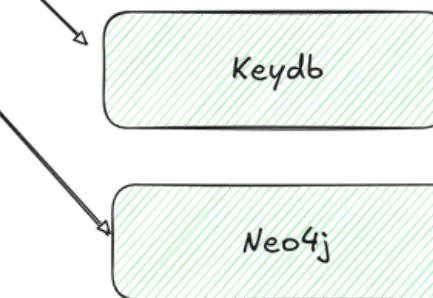
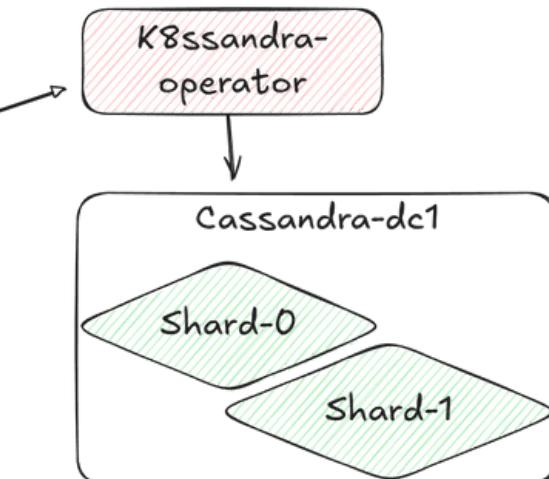
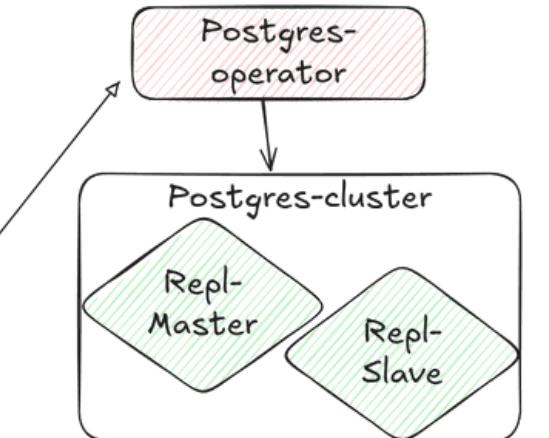
MISE EN PLACE

```
# k8c1.yaml
apiVersion: k8ssandra.io/v1alpha1
kind: K8ssandraCluster
metadata:
  name: demo
spec:
  cassandra:
    serverVersion: "4.1.3"
    datacenters:
      - metadata:
          name: dc1
        size: 1
    softPodAntiAffinity: true
  resources:
    requests:
      cpu: 1200m
      memory: 1.5Gi
    limits:
      cpu: 2400m
      memory: 2.5Gi
  storageConfig:
    cassandraDataVolumeClaimSpec:
      storageClassName: local-path
      accessModes:
        - ReadWriteOnce
    resources:
      requests:
        storage: 5Gi
```

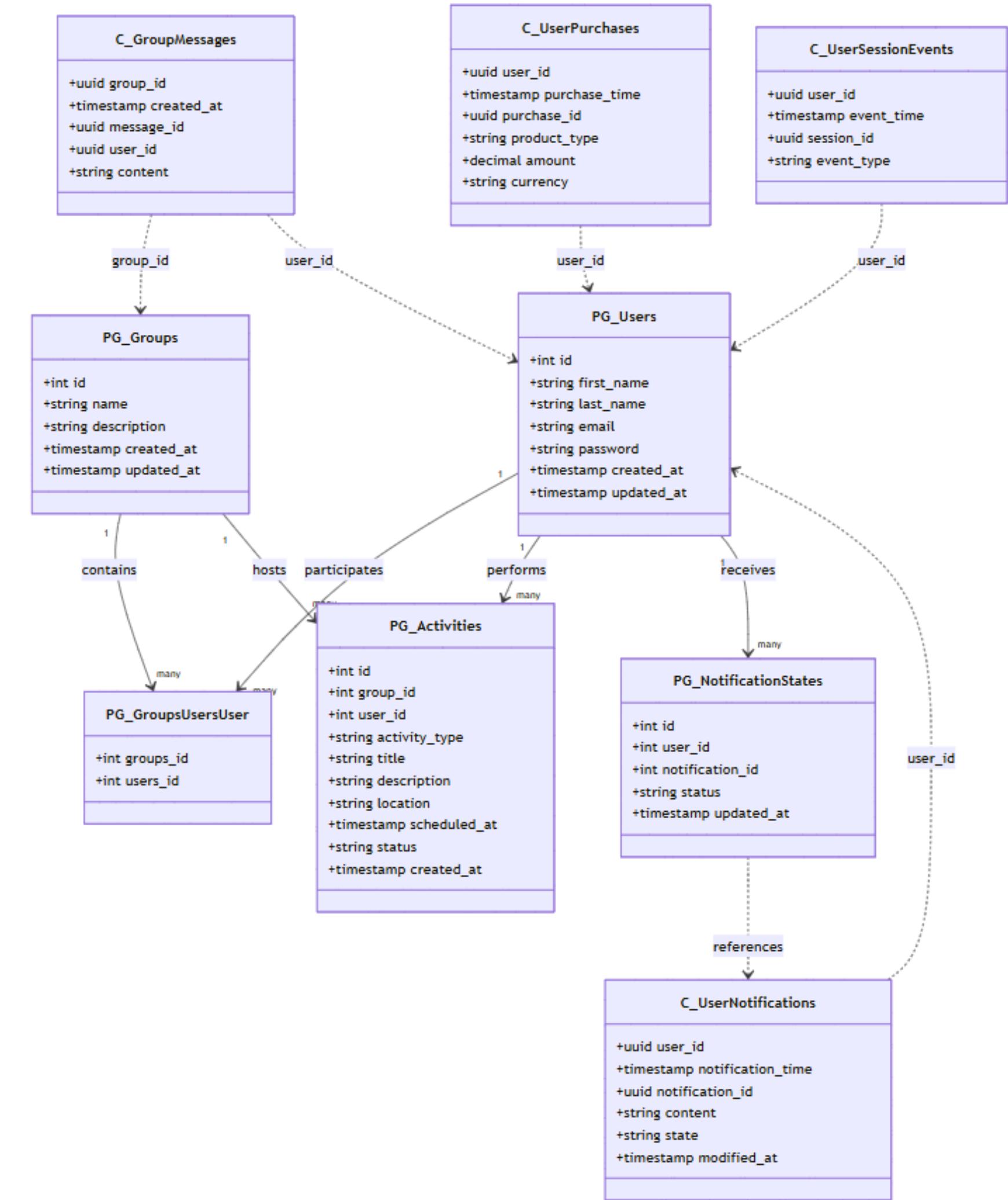
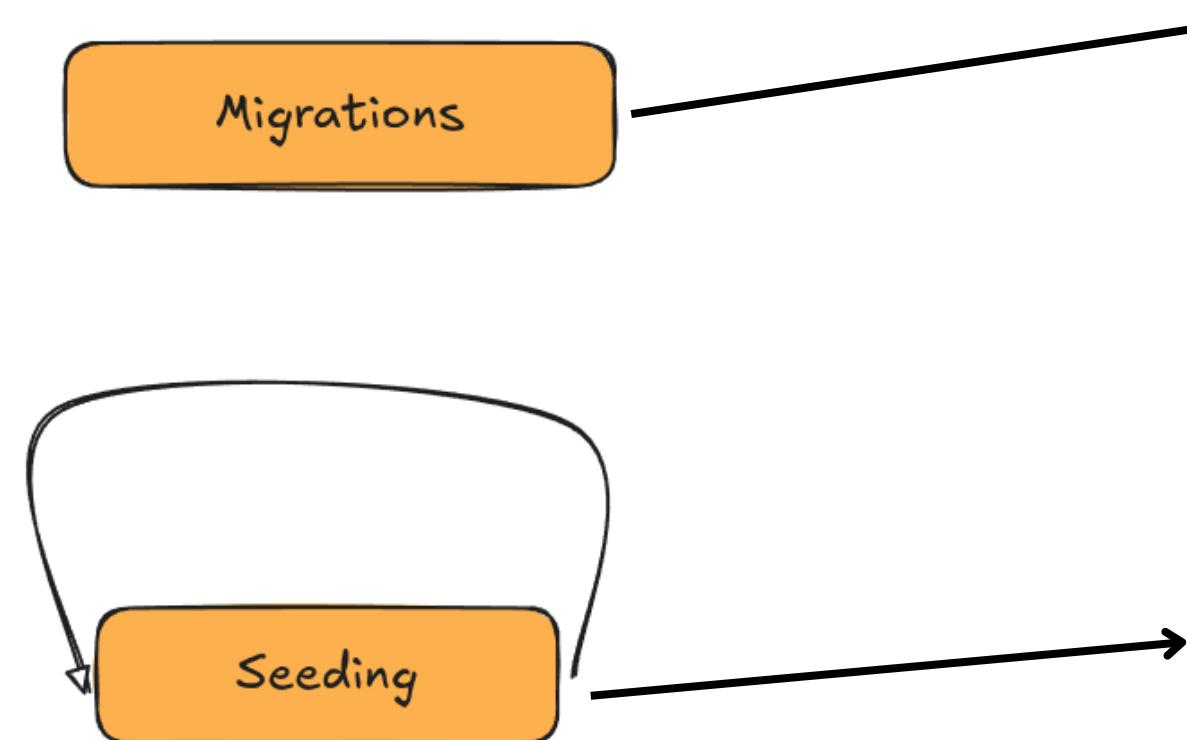
```
# postgres.yaml
apiVersion: "acid.zalan.do/v1"
kind: postgresql
metadata:
  name: postres-minimal-cluster
  namespace: {{ .Release.Namespace }}
spec:
  teamId: "stmg"
  volume:
    size: 1Gi
  numberofInstances: 2
  users:
    stmg: # database owner
    - superuser
    - createdb
    backend: [] # role for application foo
  databases:
    demo: stmg # dbname: owner
  postgresql:
    version: "17"
```



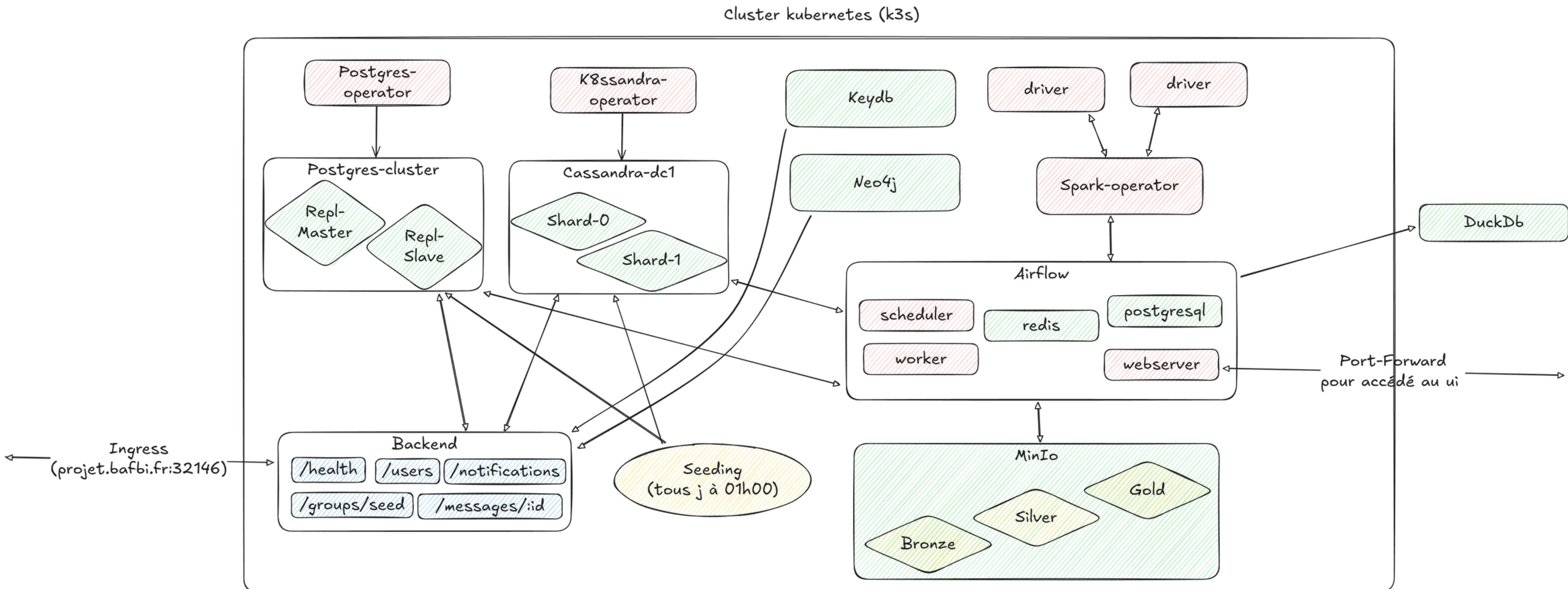
```
# helm install projet-m1 ./projetm1-chart/ -n projet-m1 --create-namespace
```



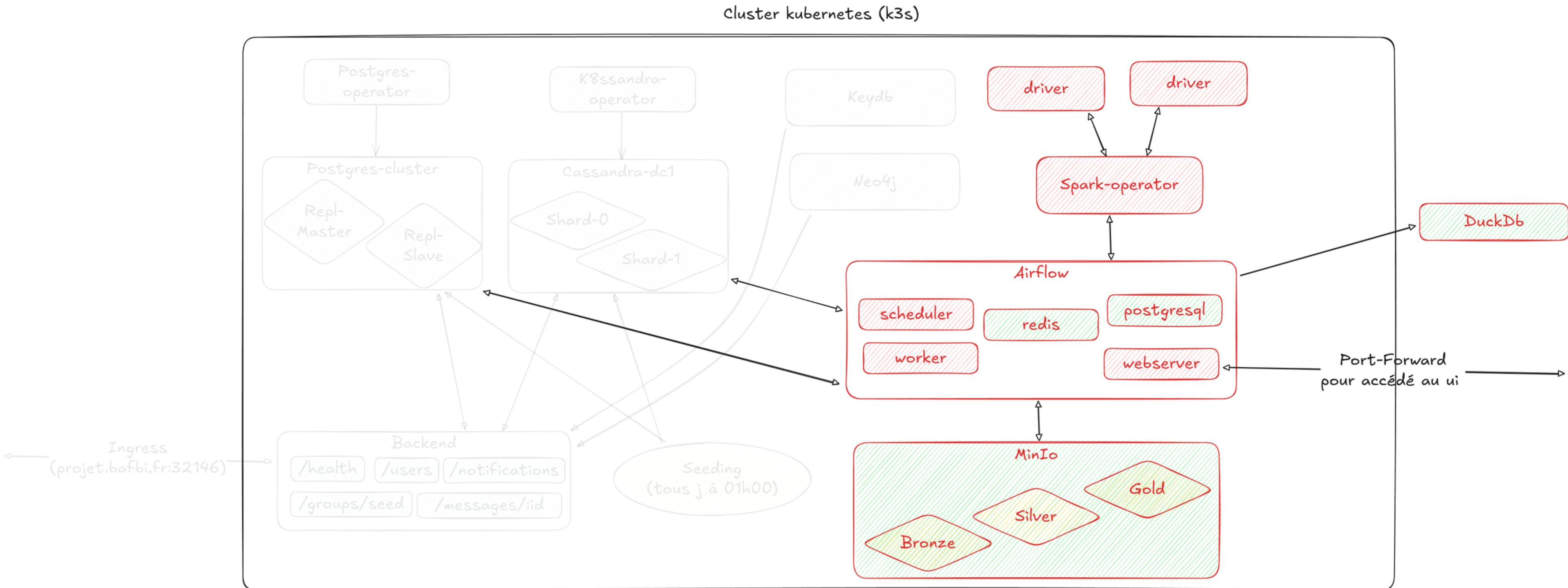
MODEL FINAL DE DONNÉES



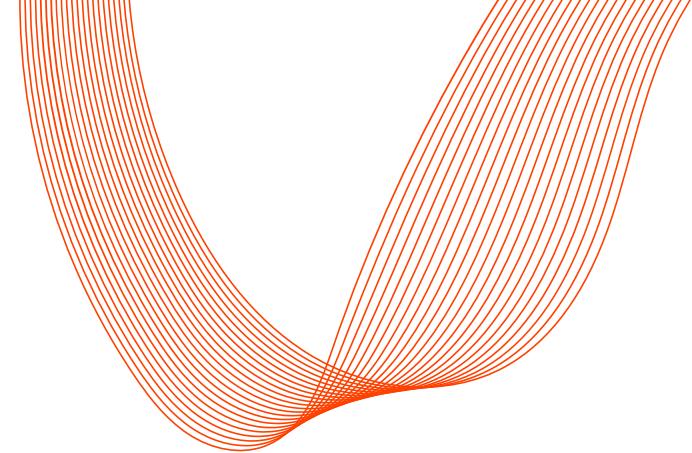
PIPELINE DE TRAITEMENT



PIPELINE DE TRAITEMENT

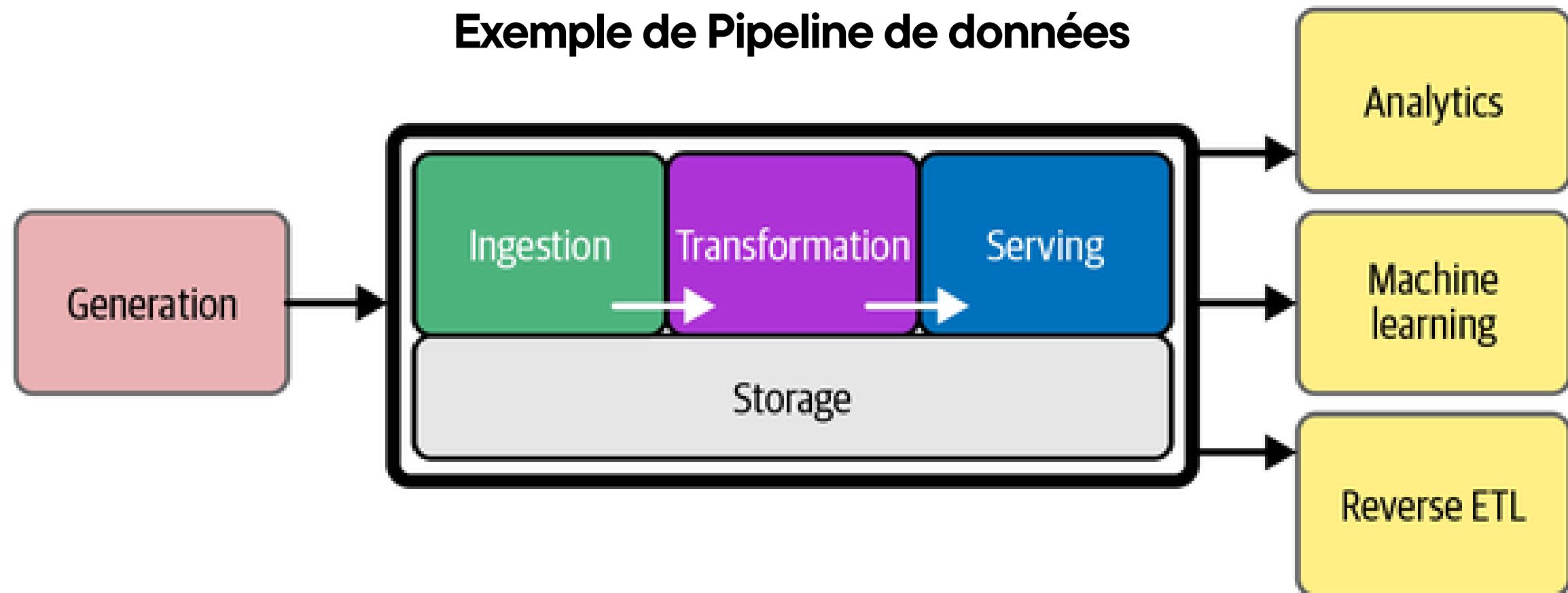


CONCEPT GENERAL



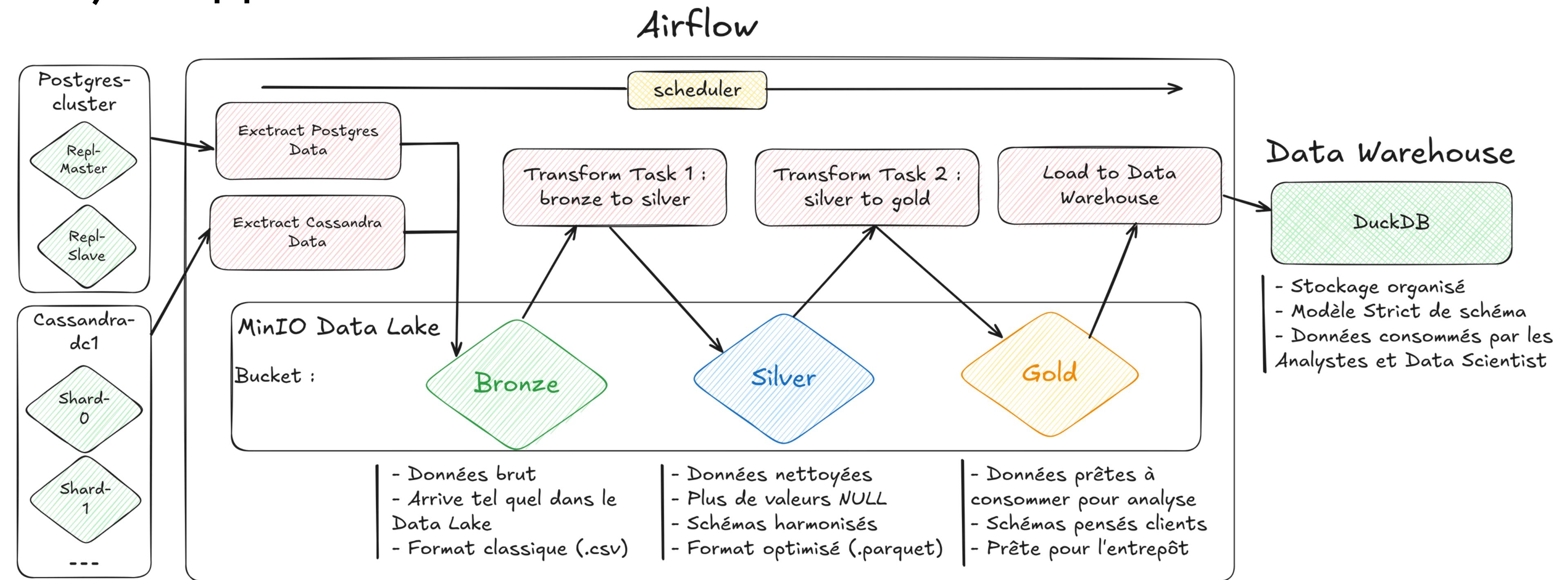
1) Pourquoi avoir besoin d'une pipeline ?

- Système de stockage actuel est dit opérationnelle (OLTP).
- On doit pouvoir exploiter les données (analyse, Machine Learning, sauvegarde de backup, etc...).
- Mais impossible de faire des requêtes lourde sur les DB en production.
- D'où le transfert vers un système dit analytique (OLAP)



STRUCTURE GLOBALE

2) Notre pipeline ELT :

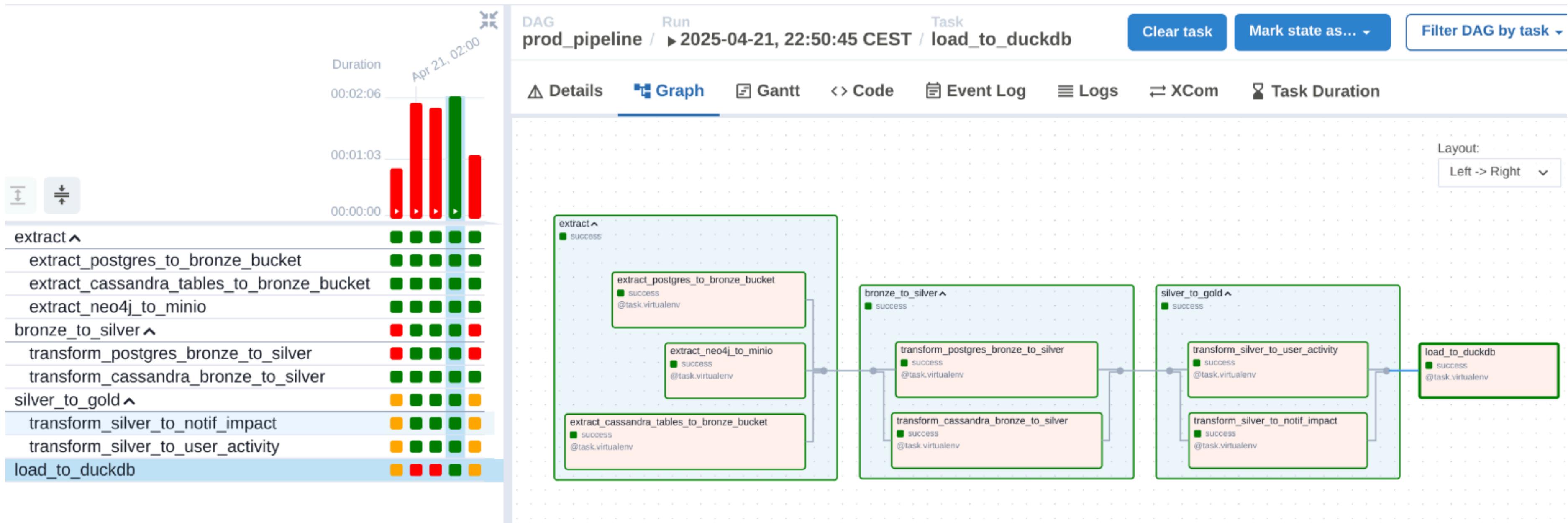


STRUCTURE GLOBALE



Apache
Airflow

3) Concrètement :



EXEMPLE CONCRET

3) Exemple d'usage pour de l'analyse :

| SELECT * FROM "notification_states" ORDER BY "updated_at" DESC LIMIT 50 | | | | | |
|---|--------------|---------|-----------------|---------|----------------------------|
| | Modification | user_id | notification_id | status | updated_at |
| | modifier | 794 | 43056 | CLICKED | 2025-04-23 00:58:15.133+00 |
| | modifier | 242 | 41303 | DELETED | 2025-04-23 00:55:58.481+00 |
| | modifier | 341 | 41686 | NULL | 2025-04-23 00:55:36.649+00 |
| | modifier | 311 | 38166 | UNSEEN | 2025-04-23 00:55:18.574+00 |
| | modifier | 680 | 9108 | SEEN | 2025-04-23 00:52:33.31+00 |
| | modifier | 839 | 28371 | NULL | 2025-04-23 00:49:37.782+00 |
| | modifier | 134 | 27557 | SEEN | 2025-04-23 00:48:57.884+00 |
| | modifier | 958 | 29381 | CLICKED | 2025-04-23 00:47:33.285+00 |
| | modifier | 256 | 45292 | SEEN | 2025-04-23 00:46:35.795+00 |
| | modifier | 112 | 31117 | CLICKED | 2025-04-23 00:46:07.152+00 |
| | modifier | 182 | 48371 | DELETED | 2025-04-23 00:45:40.066+00 |
| | modifier | 184 | 217 | NULL | 2025-04-23 00:45:25.815+00 |
| | modifier | 388 | 13138 | DELETED | 2025-04-23 00:42:13.786+00 |
| | modifier | 772 | 41699 | DELETED | 2025-04-23 00:40:33.14+00 |
| | modifier | 18 | 10723 | NULL | 2025-04-23 00:38:25.923+00 |
| | modifier | 167 | 30922 | CLICKED | 2025-04-23 00:37:15.837+00 |
| | modifier | 491 | 13355 | UNSEEN | 2025-04-23 00:36:38.24+00 |
| | modifier | 256 | 12939 | DELETED | 2025-04-23 00:35:28.961+00 |
| | modifier | 499 | 29994 | NULL | 2025-04-23 00:34:01.184+00 |
| | modifier | 949 | 49004 | UNSEEN | 2025-04-23 00:33:54.725+00 |



```
conn.sql("""
    SELECT
        notif_date,
        SUM(CASE WHEN is_success THEN 1 ELSE 0 END) AS seen,
        COUNT(*) AS total,
        ROUND(100.0 * SUM(CASE WHEN is_success THEN 1 ELSE 0
    FROM gold_notif_impact_per_day
    GROUP BY notif_date
    ORDER BY success_rate DESC LIMIT 10
""").df()
```

[24] ✓ 0.0s

| | notif_date | seen | total | success_rate |
|---|------------|--------|-------|--------------|
| 0 | 2025-03-20 | 843.0 | 2441 | 34.54 |
| 1 | 2025-04-22 | 631.0 | 1831 | 34.46 |
| 2 | 2025-03-27 | 2174.0 | 6364 | 34.16 |
| 3 | 2025-04-04 | 2121.0 | 6217 | 34.12 |
| 4 | 2025-03-29 | 2137.0 | 6274 | 34.06 |
| 5 | 2025-03-23 | 2100.0 | 6169 | 34.04 |

5. Classement des clients par montant total dépensé (du plus élevé au plus faible)

```
conn.sql("""
    SELECT
        user_id,
        SUM(total_money_spend) AS total_spent,
        COUNT(date) AS active_days,
        ROUND(AVG(total_money_spend), 2) AS avg_spent_per_day
    FROM gold_user_activity_per_day
    GROUP BY user_id
    ORDER BY total_spent DESC
    LIMIT 20
""").df()
```

✓ 0.0s

| | user_id | total_spent | active_days | avg_spent_per_day |
|---|--------------------------------------|-------------|-------------|-------------------|
| 0 | 02d499ef-07db-50bf-b430-3589e4804f17 | 4832.46 | 135 | 35.80 |
| 1 | e0ea72fb-ae2a-5955-8472-716e37b3f8e4 | 4084.98 | 117 | 34.91 |
| 2 | 65e32085-d618-5dd4-8383-403d3844b62e | 4038.29 | 118 | 34.22 |
| 3 | 861a7a83-242f-5b53-8930-860d45057f74 | 3956.33 | 115 | 34.40 |
| 4 | 49ccfc98-062a-51eb-ae40-fa2ad5963911 | 3904.48 | 122 | 32.00 |
| 5 | bbf019ae-487f-5d00-960e-764f6784aa0b | 3884.13 | 121 | 32.10 |
| 6 | 254c904a-96e5-5e55-8263-50af62adc388 | 3873.42 | 123 | 31.49 |

CONCLUSION

Difficultés rencontrées:

- Apprentissage de Kubernetes
- Mise en lien des services
- Limité par nos ressources en tant qu'étudiant
- Manque de documentation ou d'exemple pour la plupart des technologies

Axes d'améliorations:

- Etendre la pipeline à d'autres usages
- Renforcement de la sécurité
- Mise en place de Monitoring
- Intégrer du Cloud

PROJET M1 ISEN

Sujet de Spécialité Big Data : DevOps/Data Engineering

MERCI DE NOUS AVOIR ÉCOUTÉ