Le Data Warehouse, cet accès au Data Lake AWS de 700 To

Notre présentation

Simon Cocula
Data Engineer
3 ans d'expérience Data et Cloud

Arnaud Milleker
Tech Lead Data Lake
10 ans d'expérience Data

Ogury: la problématique

Vision partielle

Les marques n’ont qu’une vision partielle de base de l’activité de leurs utilisateurs

01

Intégration complexe

Les différents outils de la chaîne marketing sont peu connectés ou de manière très complexe

02

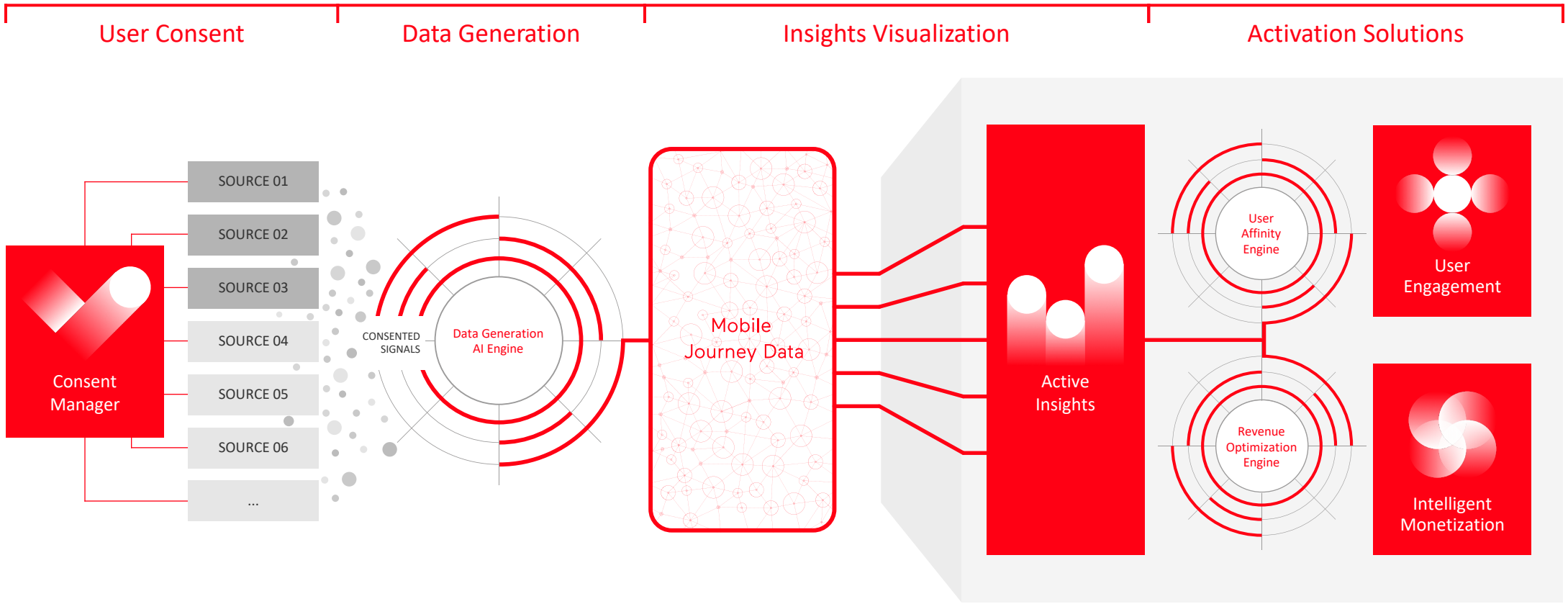
Data toxique

Les données utilisés par les marques ne sont pas traçables et pas systématiquement en règle avec la RGPD

03

Ogury Mobile Journey Marketing Cloud

The first end-to-end, fully integrated consent, data, insights and activation solution.



Le Data Warehouse, cet accès au Data Lake AWS de 700 To

Ogury en quelques chiffres

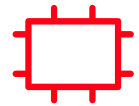
+3 500 éditeurs partenaires

+300M d'utilisateurs consentis

+340 collaborateurs

15 bureaux à travers le monde

+100M$ de chiffre d'affaire en 2019



+10 différentes technologies de bases de données

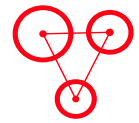
+10 différents langages de programmation

5 prix gagnés depuis 2014

+100 000 requêtes par seconde

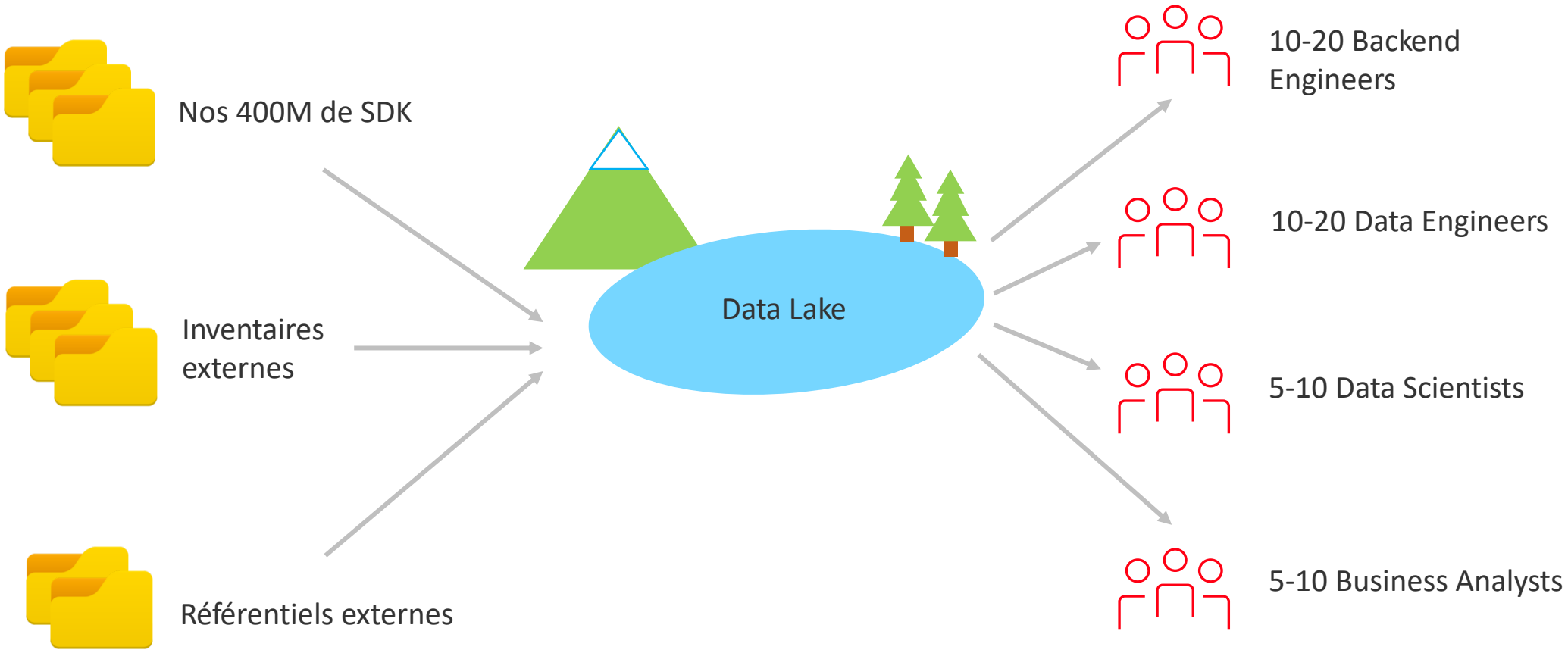
+1000 instances sur AWS

+3 TB de données ingérées par jour

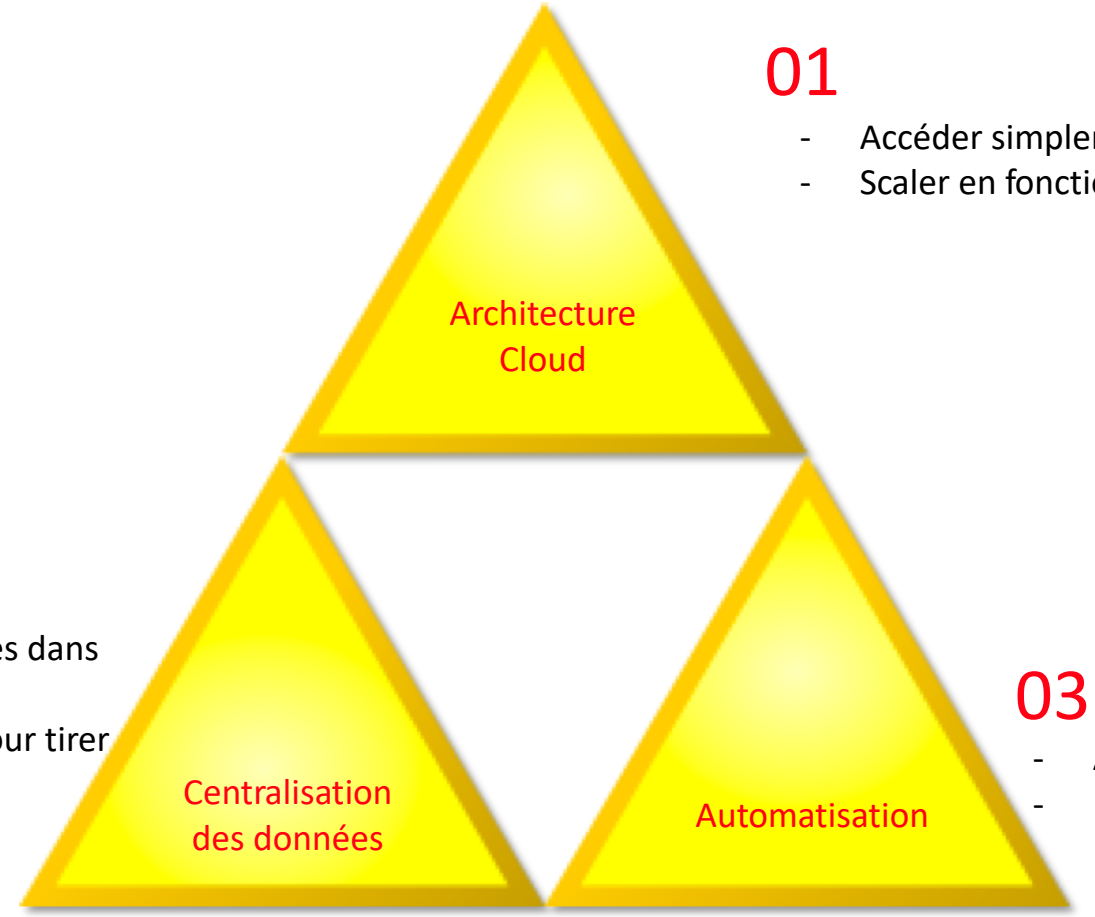


Data Lake de **+700 TB**

La squad Data Lake



Notre vision



01

- Accéder simplement à toute la data
- Scaler en fonction du volume

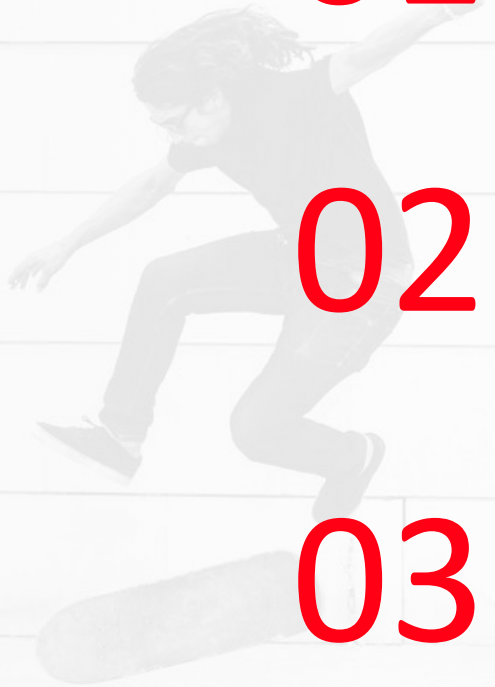
02

- Regrouper les données importantes dans un schéma clair et précis
- Croiser nos sources de données pour tirer au mieux parti de notre donnée

03

- Automatiser la création d'ETL
- Uniformiser les ETL

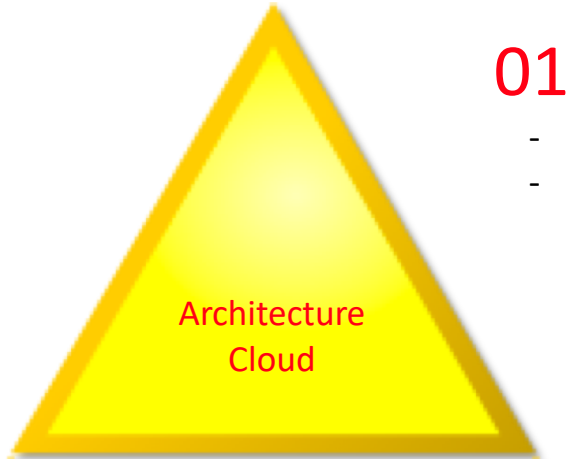
Summary



01

L’architecture Cloud

Notre vision



01

- Accéder simplement à toute la data
- Scaler en fonction du volume

Fiche technique : Redshift

Définition
Data Warehouse clusterisée

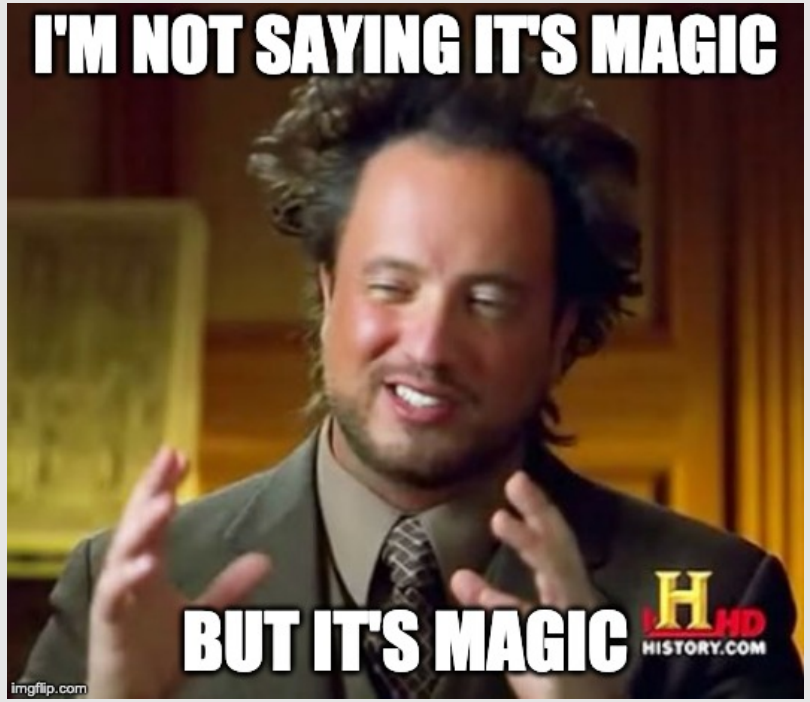
Avantages
Le compute facile pour tous, haute performance, scalable

Inconvénients
Investissement important au départ

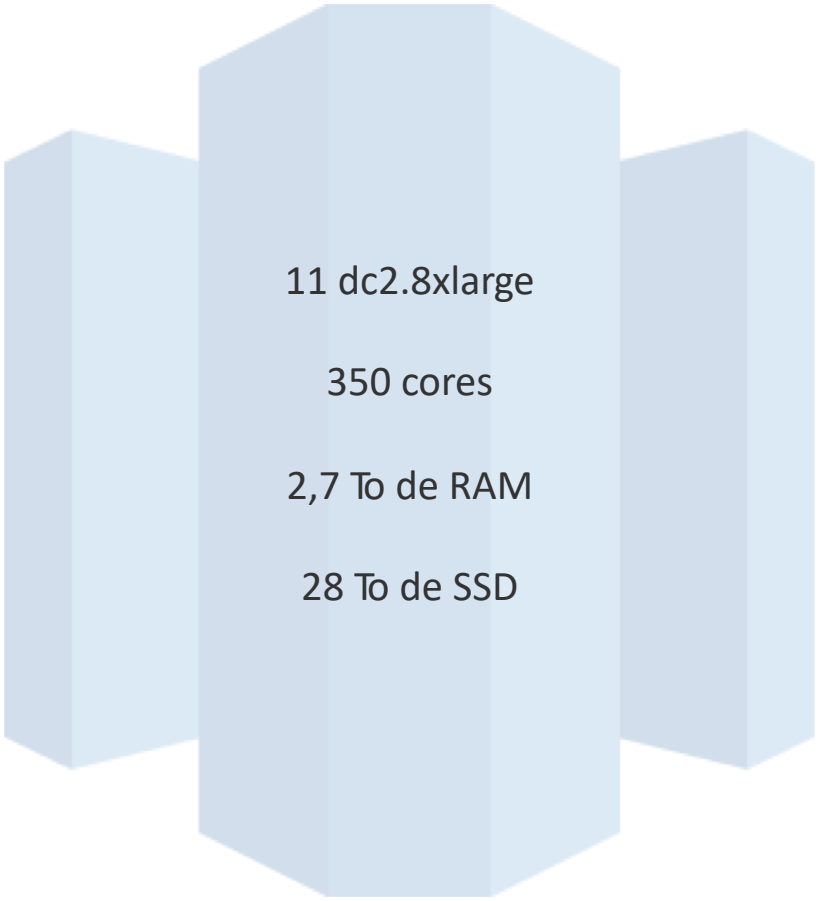
Astuces
- Gestion des distkey (et sortkey)
- Externalisation des datas sur Athena/Spectrum

Prix
365K€ par an (173K € pour 3 ans) pour 40To de fichiers

Effet wahou
Une jointure de 2 tables (66Md et 2 Md de lignes) tourne en 31 secondes



Notre Redshift



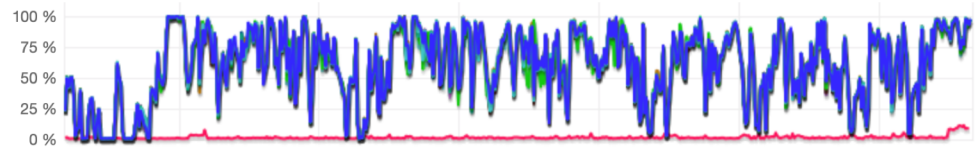
- Performance
- Coût
- Always on

Panneau d’observation

Queries

Disk

CPU



Query plan

```
XN GroupAggregate (cost=1000104732152.07..1000104732238.44 rows=200 width=113)
    -> XN Subquery Scan derived_table1 (cost=1000104732152.07..1000104732218.08 rows=1886 width=113)
      -> XN Window (cost=1000104732152.07..1000104732199.22 rows=1886 width=97)
        -> XN Sort (cost=1000104732152.07..1000104732156.79 rows=1886 width=97)
          -> XN Network (cost=44190740.08..104732049.47 rows=1886 width=97)
            -> XN Hash Right Join DS_DIST_NONE (cost=44190740.08..104732049.47 rows=1886 width=97)
              -> XN Seq Scan on users u (cost=0.00..26907240.96 rows=2690724096 width=41)
              -> XN Hash (cost=44190735.36..44190735.36 rows=1886 width=96)
                -> XN Seq Scan on user_pivot up (cost=0.00..44190735.36 rows=1886 width=96)
```

Fiche technique : Data Pipeline

Définition
L'ETL complètement intégré à AWS

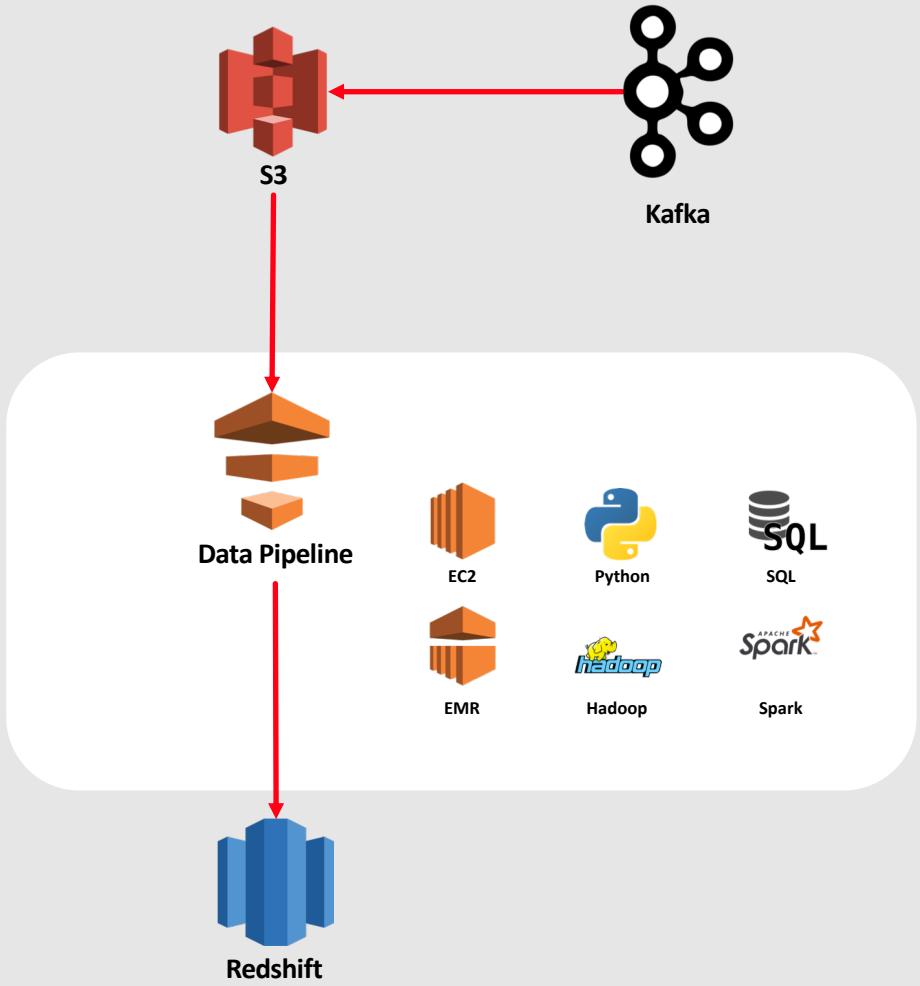
Avantages
Un coût économique faible

Inconvénients
Lourd à manipuler

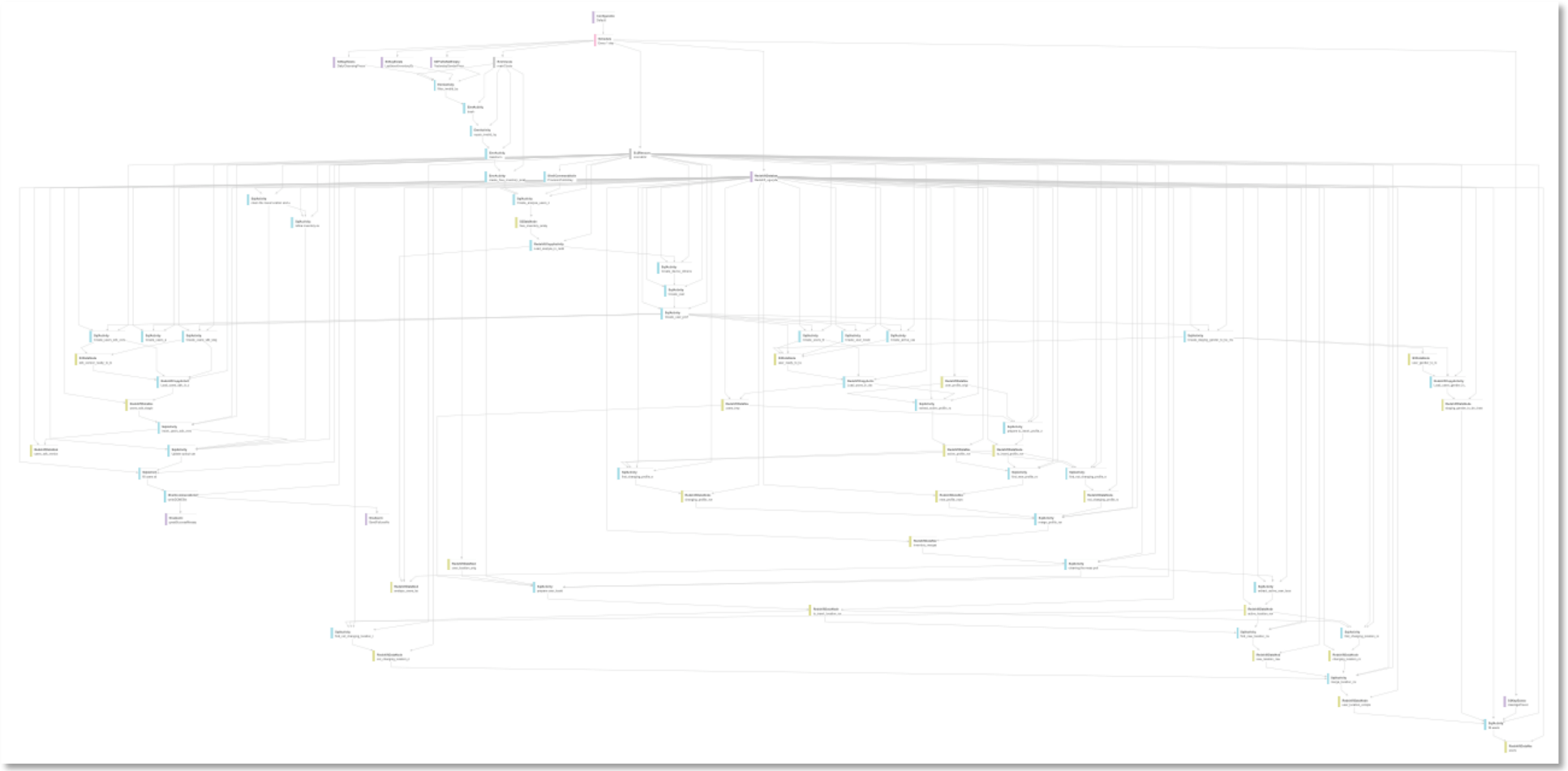
Astuces
Utiliser le JSON pour plus de liberté, un autre outil ?

Prix
Négligeable

Effet wahou
… Hmmm, on passe



Un exemple de Data Pipeline



- Bonne intégration des outils AWS

- Limitations :
  - D'ETL
  - D'objets
- Paramétrages
- Pas modulaire

Fiche technique : Glue

Définition

Solution serverless pour créer des catalogues (structure de données des fichiers plats) ou scripter du Spark avec Python ou Scala (/!\ librairies spécifiques AWS)

Astuces

Bien initier une table rapidement

Prix

Négligeable

Effet wahou

Création d'un catalogue de données en 2 minutes

Fiche technique : Athena

Définition

Solution serverless de SQL sur fichier plat (via Glue)

Avantages

\+ Très pratique pour exploration ou ETL (si données pas trop grosses)

Inconvénients

\- Attention au prix qui peut s'envoler

Astuces

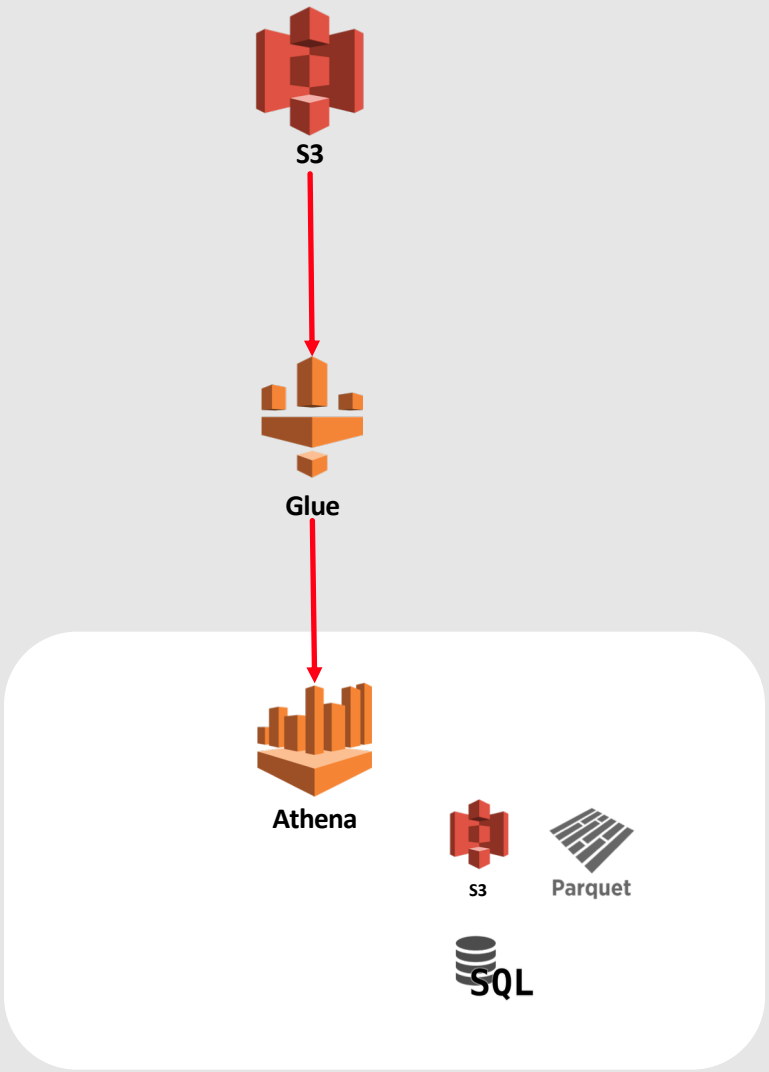
- Transformer les sources en Parquet ou ORC
- Lambda pour créer les partitions
- Utiliser les CTAS

Prix

Scan de 2 milliards de lignes pour 35 centimes (5$/To)

Effet wahou

Exploiter des fichiers de plusieurs Go pour quelques centimes en 1 minute tout compris



Fiche technique : Redshift Spectrum

Définition
Solution de SQL sur fichier plat depuis Redshift

Avantages
\+ Récupération de données de grands volumes et/ou semi-structurées, possibilité de joindre avec les Data de Redshift, facilité d'utilisation

Inconvénients
\- Attention au prix qui peut s'envoler, distributions à refaire

Astuces
- Redistribuer la table (distkey)
- Réduire la table avant utilisation ou stockage dans Redshift

Prix
Scan de 2 milliards de lignes pour 35 centimes (5$/To)

Effet wahou
Exploiter des fichiers de plusieurs Go pour quelques centimes en 1 minute tout compris … Et l'avoir dans son Data Warehouse !

BETWEEN THE DATA LAKE AND THE USER

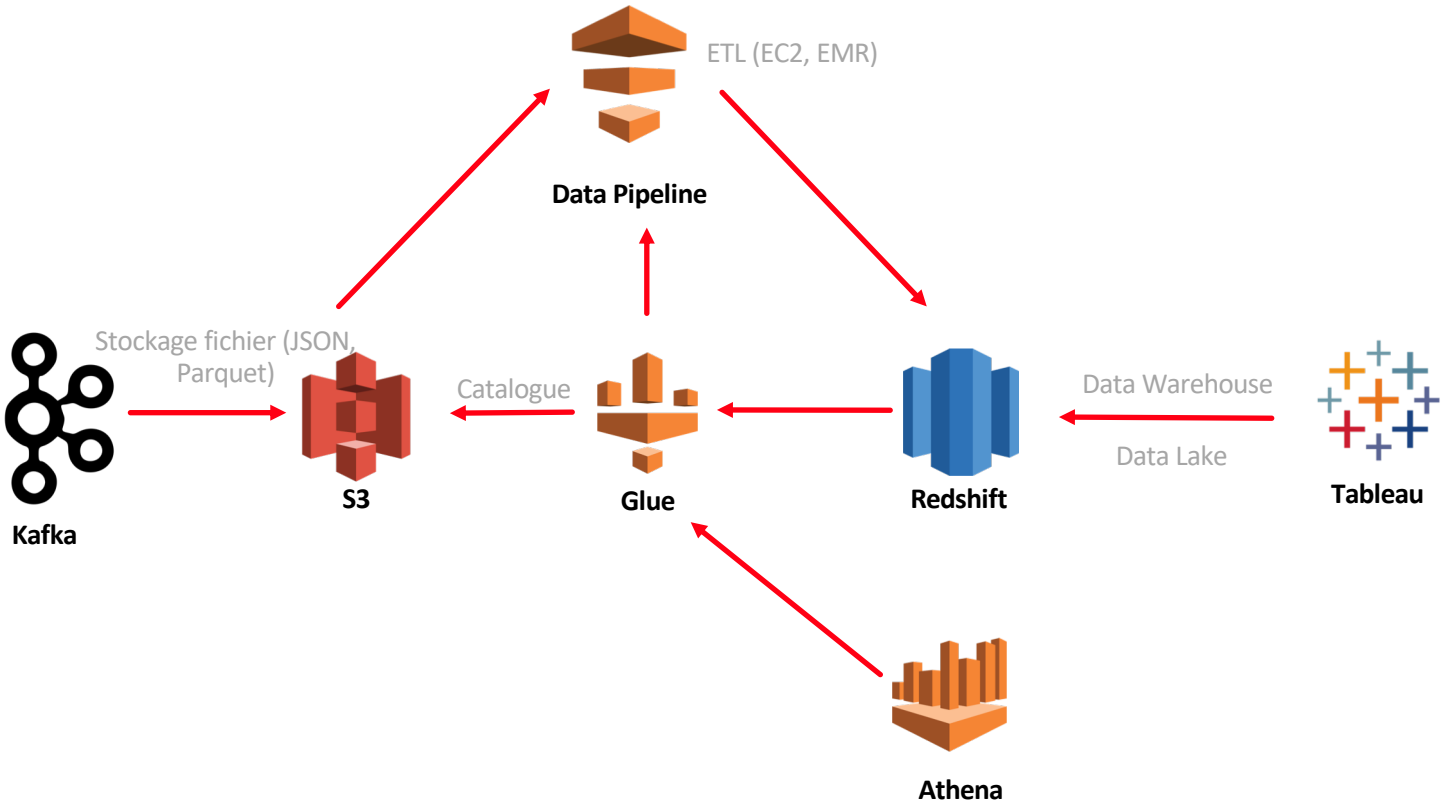
Notre architecture de Data Lake

Contexte

- 35 développeurs Data
- 5-10 projets critiques de DScience déployés
- Business
- Monitoring
- Data Lake de 700 To

Forces

- Architecture simple et dynamique
- Tous accès simple
- Exploration facile et rapide



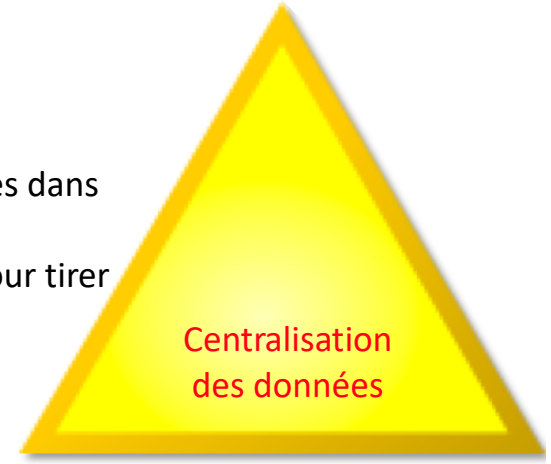
02

Centralisation des données

Notre vision

02

- Regrouper les données importantes dans un schéma clair et précis
- Croiser nos sources de données pour tirer au mieux parti de notre donnée



Rappel: Qu'est-ce qu'un schéma en étoile?

Votre base de données est un Data Warehouse qui rassemble un ou plusieurs domaines métier dans des schémas en étoile (ou Datamart)

Le schéma en étoile est constitué de :

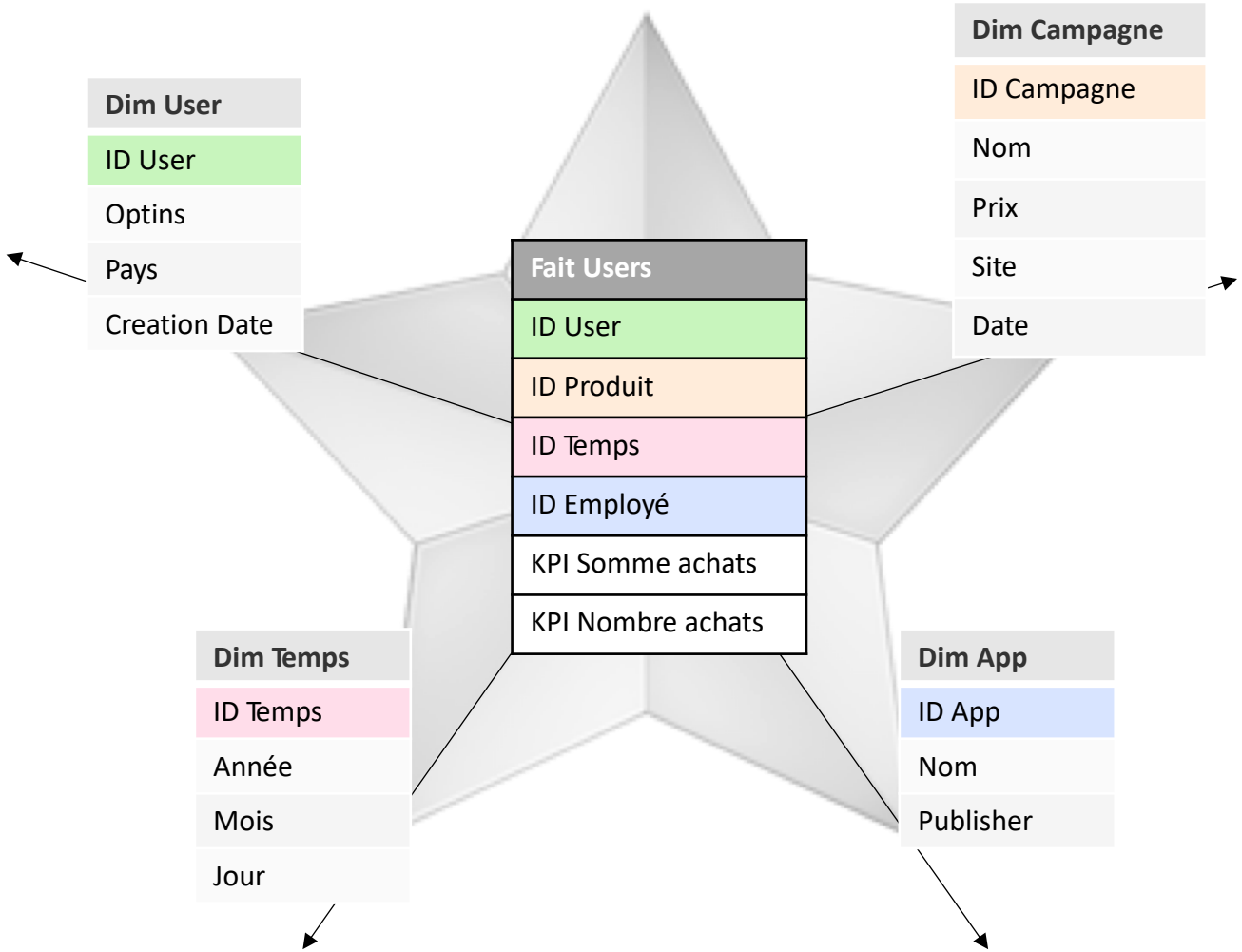
- Table de fait
- Table de dimensions

La table de fait :

- Regroupe la donnée d'un sujet précis (ex : achats)
- Agrège les KPIs de l'entreprise (ex : unités, chiffre d'affaire)

Les tables de dimensions :

- Structurent des référentiels d'entreprise (Date, produits, …)
- Filtrent la donnée

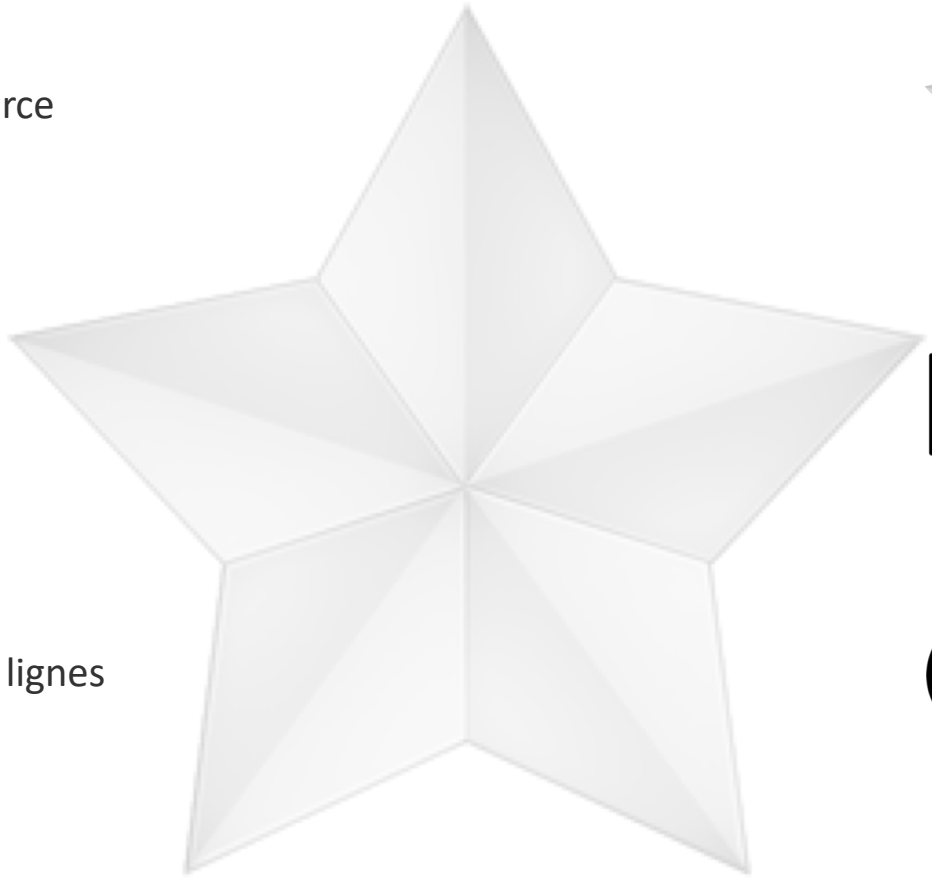


Notre « étoile » et ses faits

25 Dossiers source

0,5 To scannés
par jour

240 millions de lignes
par jour



7 dimensions
116 métriques

28 milliards de lignes
3,55% du Data Warehouse

1 minute en moyenne
par requête

Dans un système distribué

Les tables de fait sont:

- Distribuées par nœud avec une clé :
  - Également répartie
  - Présente dans les jointures

Distkey sur Redshift

- Trié par filtre fréquent : la date, le pays, …

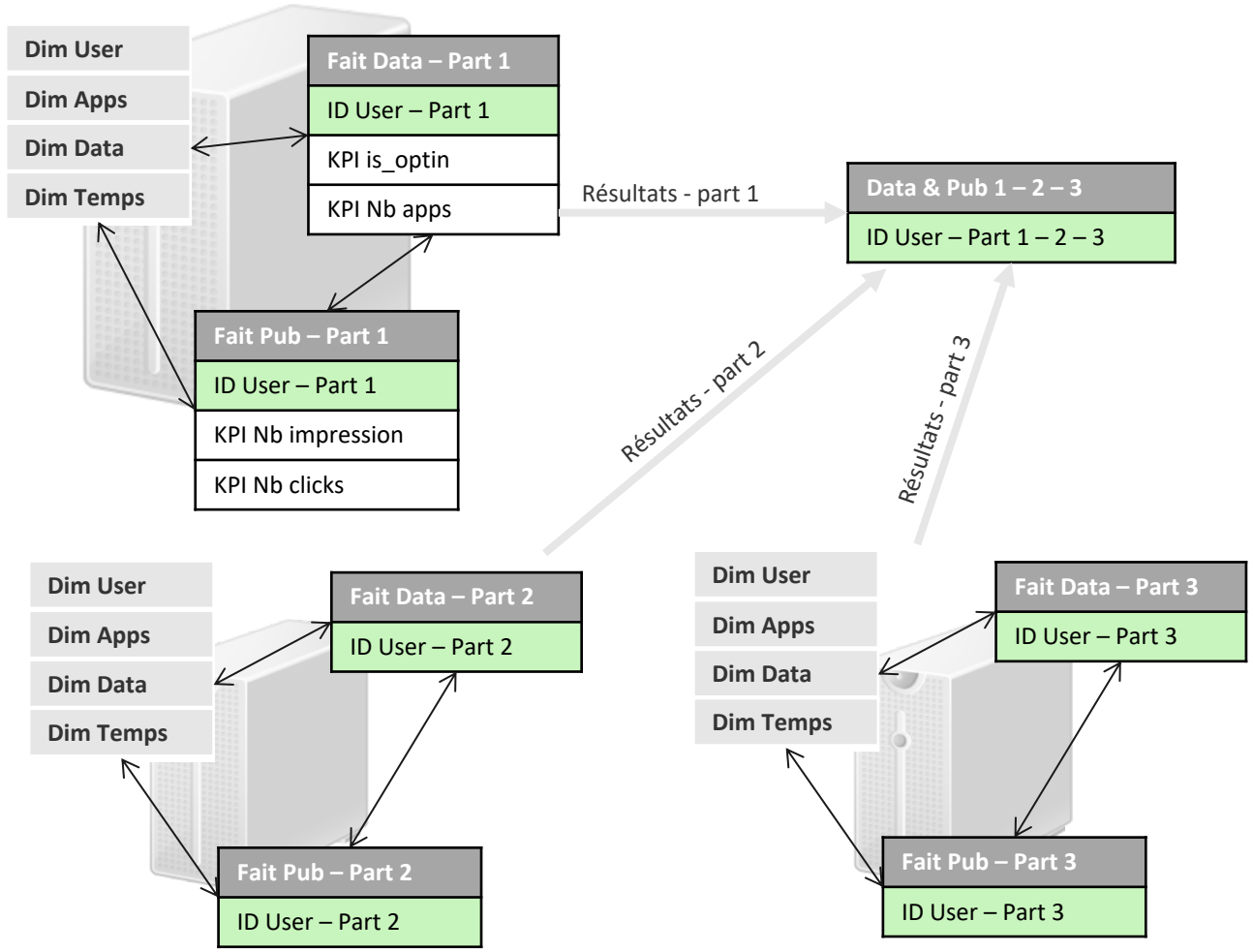
Sortkey sur Redshift

Chaque table de dimension est:

- Recopiée entièrement (broadcast) sur chaque nœud pour permettre des jointures rapides avec la table de fait
- Triée par la clef de jointure avec la table de fait

Effet wahou

Une jointure de 2 tables (66Md et 2 Md de lignes) tourne en 31 secondes



Avantages et Inconvénients d’un schéma en étoile

+	−
• Agrégations très performantes • Scalabilité linéaire • Data centralisées • Requêtes simplifiées	• Evolutions plus complexes • Choix structurant à la conception

Cycle de développement d'un ETL

Analyse :
- ❖ Recherche des sources
- ❖ Compréhension de la donnée

Développement :
- ❖ Cleaning
- ❖ Transformations
- ❖ Structure (Compression, format)
- ❖ en Python/Scala

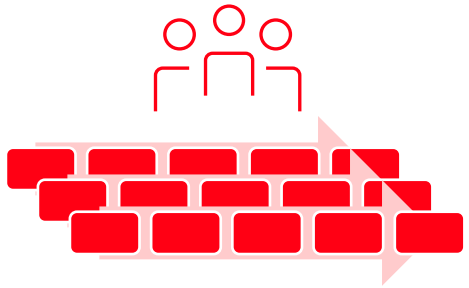
Tests

Déploiement et intégration continue

Communication sur le déploiement

+ Documentation

Conclusion:
- ❖ Travail répétitif
- ❖ Développements très long



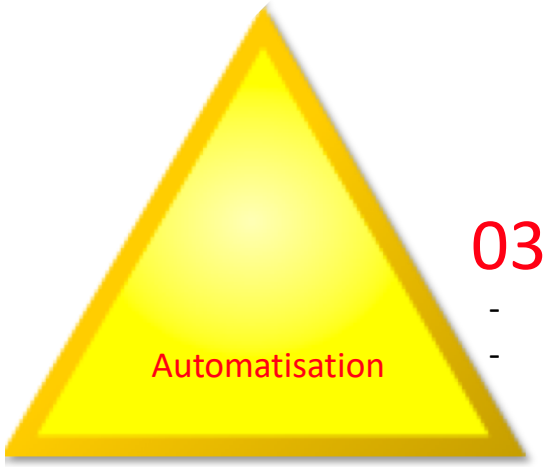
Dans le cas de plusieurs équipes :
- Travail souvent recommencé
- Tests différents
- Données interprétées différemment

⇒ Incohérences et perte de confiance du business

03

Automatisation de création d’ETL

Notre vision

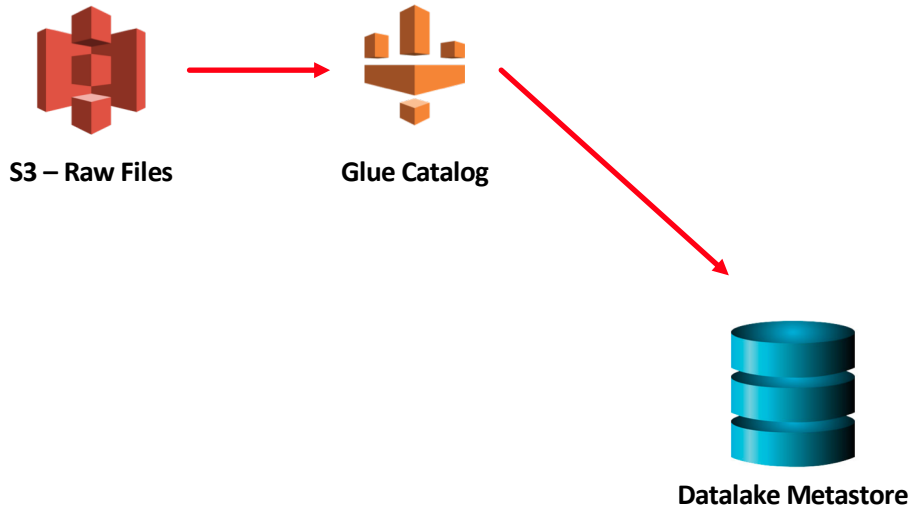


03

- Automatiser la création d’ETL
- Uniformiser les ETL

I. Metastore: Catalogue

- Référencer l'emplacement des données
- Documenter les données
- Simplifier l'accès aux données (Hive, Spark, …) par les ETL



II. Metastore: Cleaning

- Les opérations de cleaning faites sur chaque champs sont écrites dans des scripts SQL templétisés

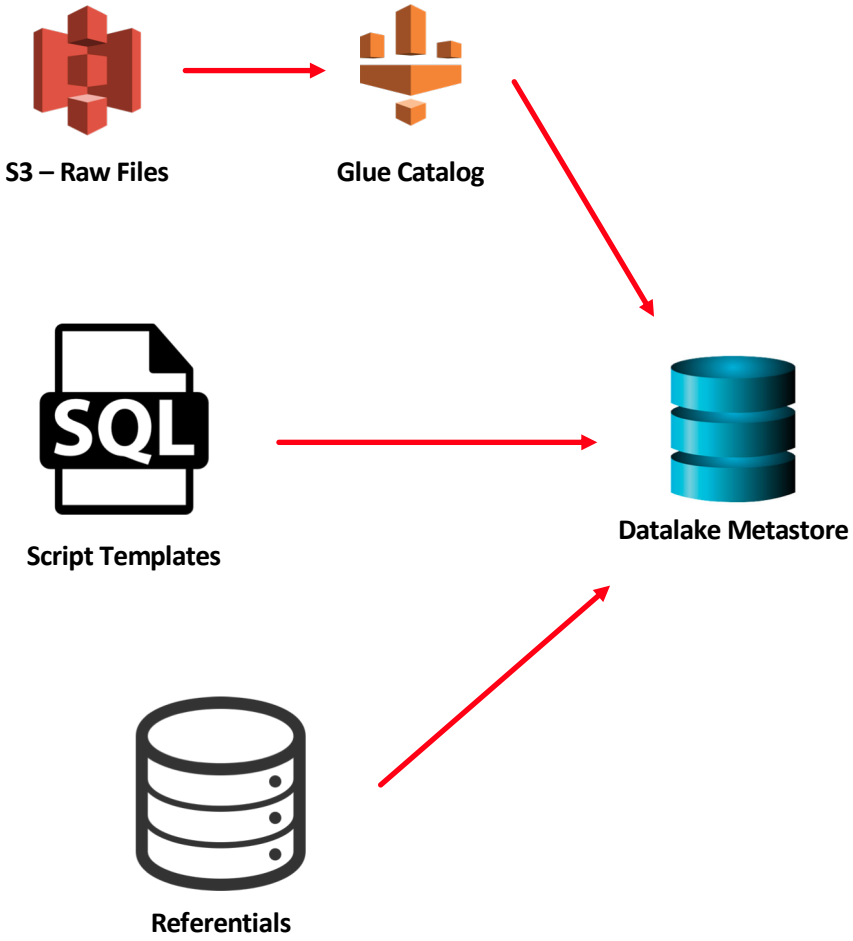
```
CASE
  WHEN "{{ field[0] }}" RLIKE '^(IAB[0-9_]{1,})(,IAB[0-9_]{1,})*$'
    THEN CAST( SPLIT( \"{{ field[0] }}\", ',') AS {{ type }} )
  ELSE CAST( SPLIT( '{{ null_value }}', ',') AS {{ type }} )
END
```

- Des référentiels permettent de valider une partie de nos données

Countries' Codes
US
FR
GB

Android Versions
4.0.1
7.0.3
9.0.0

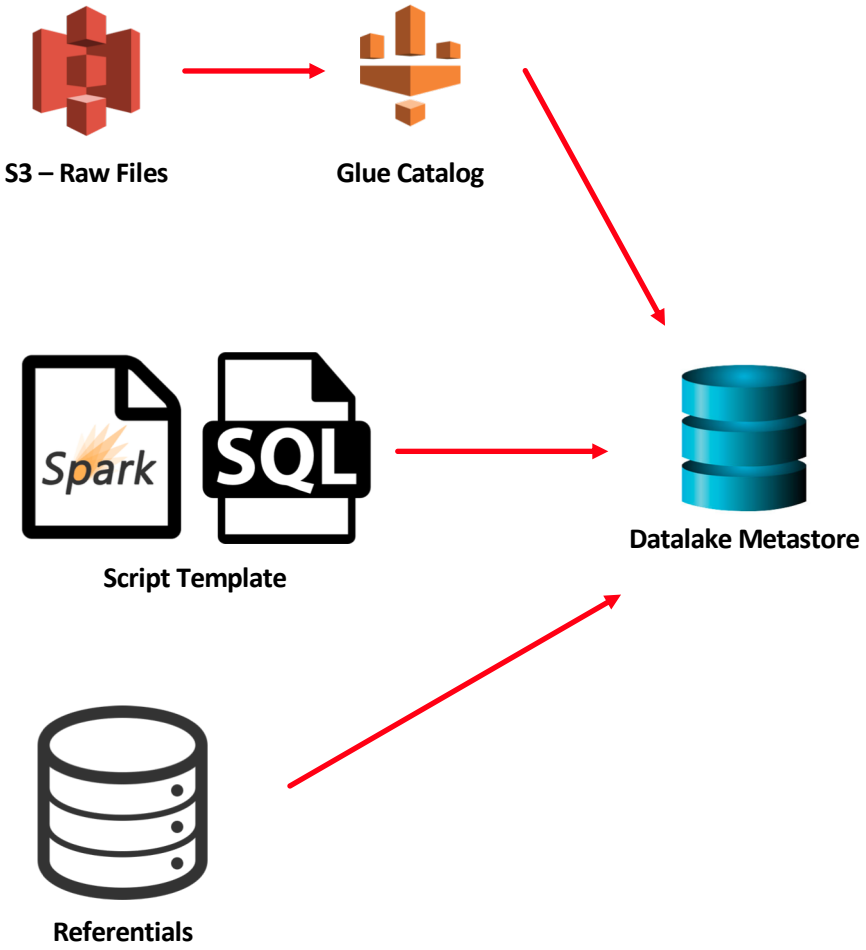
Users
837806a7-6c37-4630-9f6c-9aa7ad0129ed
a0eebc99-9c0b-4ef8-bb6d-6bb9bd380a11



II. Metastore: Transformation

- Les transformations complexes sont écrites dans des fichiers PySpark templétisés

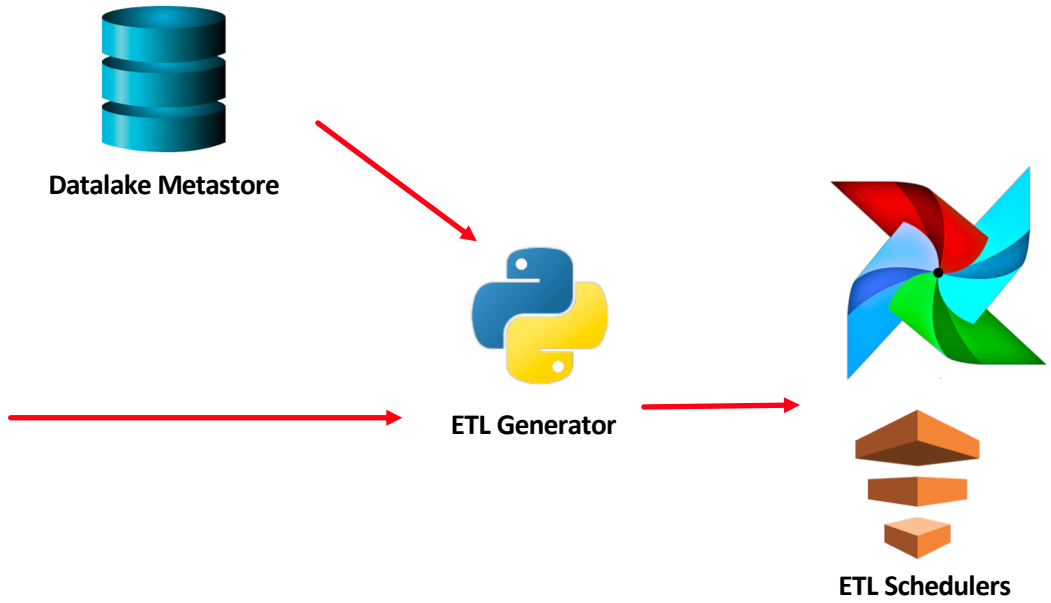
```
def windowingFuntion1(dataframe):

df = dataframe.groupBy( '{{ fields[0] }}', '{{ fields[1] }}' ).\
  agg( countDistinct('source').alias('count'), max( '{{ fields[2] }}' ).\
  alias('{{ fields[2] }}') )

dfFiltered = ad.\
  where("'{{ fields[1] }}' != {{ null_values[1] }}")

windowdf = Window.partitionBy( dfFiltered['{{ fields[0] }}'] ).\
  orderBy( df['count'].desc(), df['{{ fields[2] }}'].desc() ).\
  rowsBetween( Window.unboundedPreceding, Window.currentRow )

output = df.select( '{{ fields[0] }}', first( '{{ fields[1] }}' ).\
  over(windowdf).alias( '{{ fields[1] }}' ) ).\
  na.fill('{{ null_values[1] }}').distinct()

return output
```



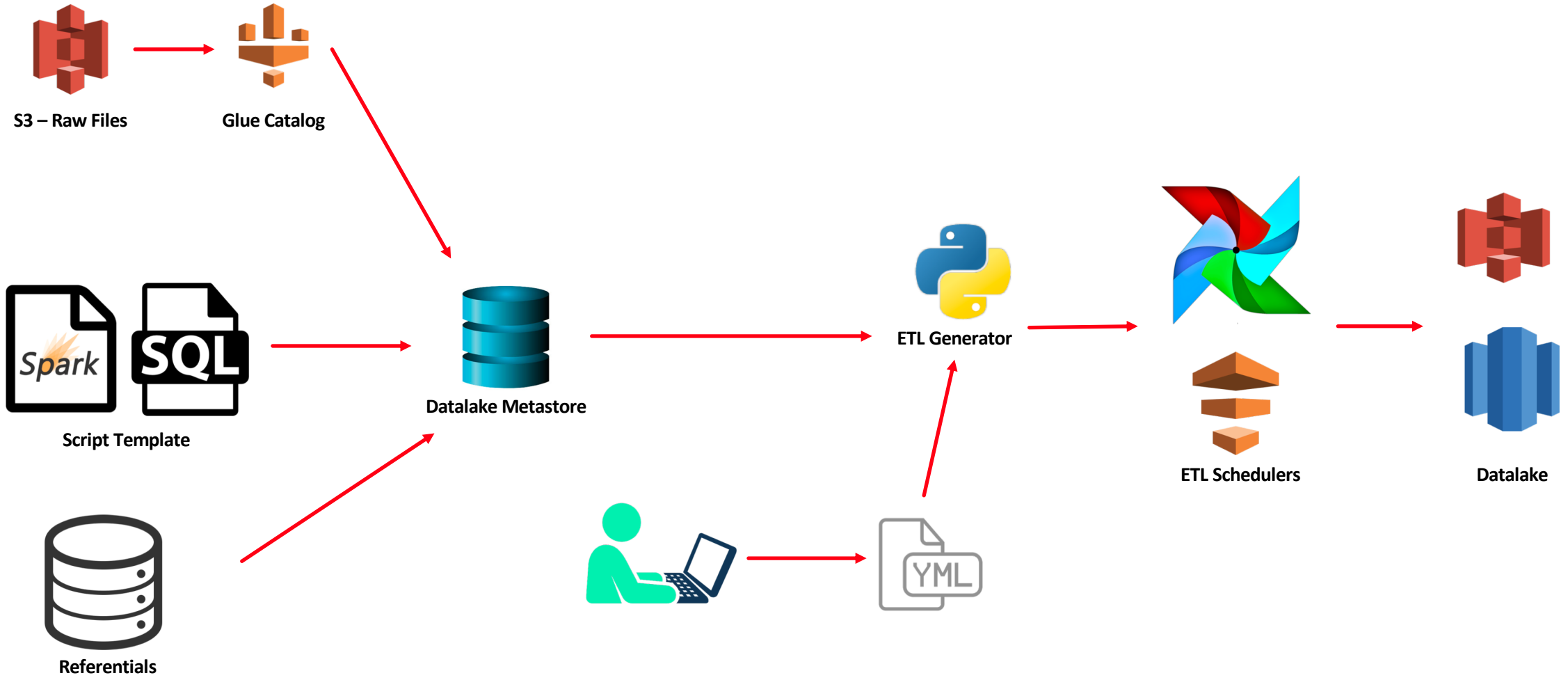
Générateur automatique d'ETL

- Grace à un fichier de configuration un utilisateur peut générer un ETL
- Avec ce système, nous simplifions grandement le processus d'accès aux données « froides »
- La création des ETL est uniformisée et rapide

```
TEMPLATE: standard

SOURCES:
- source_1
- source_2
- source_3

KEYS:
- field: key_1
sources: all
- field: key_2
sources:
- source_1
- source_2

PROPERTIES:
- name: property_1
sources: all

MEASURES:
- field: measure_1
source: source_1
aggregation: sum

- field: measure_2
source: source_3
aggregation: avg
```



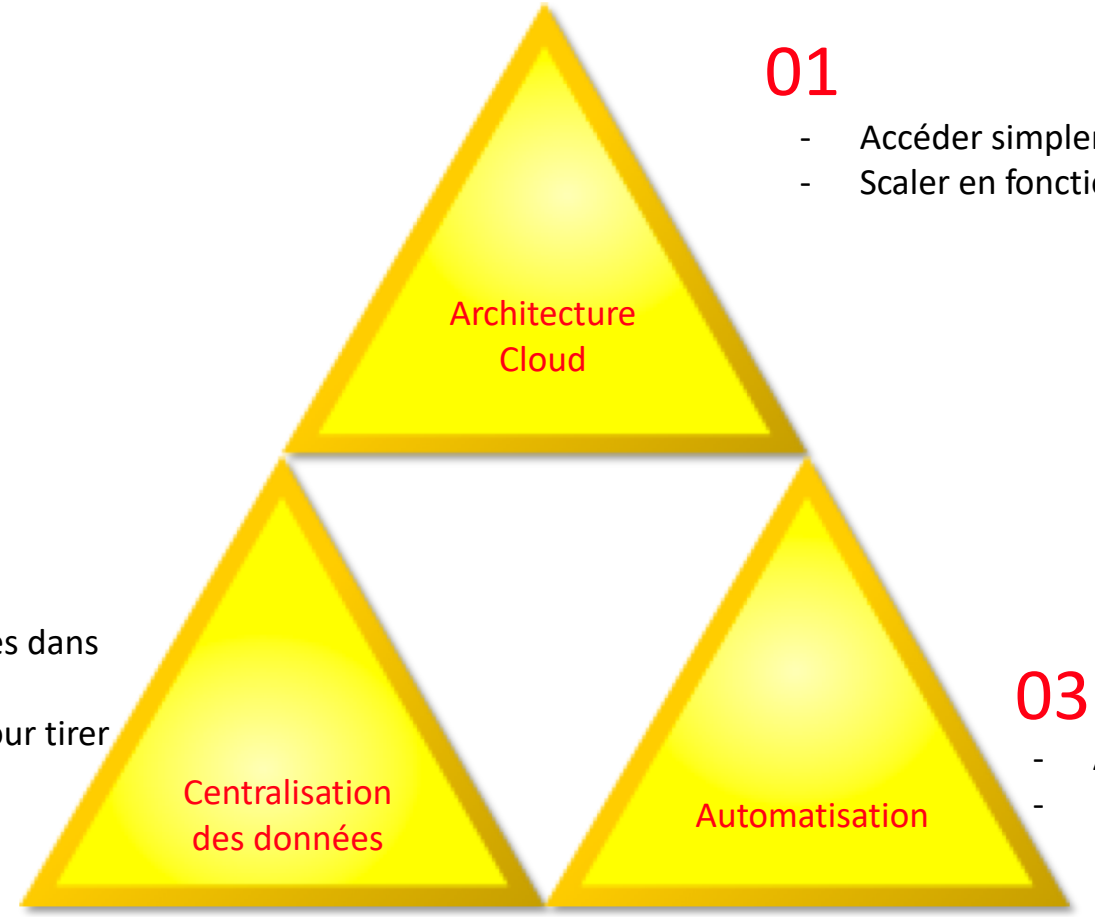
Comment l'implémenter?



04

Conclusion

Conclusion



01

- Accéder simplement à toute la data
- Scaler en fonction du volume

02

- Regrouper les données importantes dans un schéma clair et précis
- Croiser nos sources de données pour tirer au mieux parti de notre donnée

03

- Automatiser la création d'ETL
- Uniformiser les ETL

Questions ?

NOUS RECRUTONS…

80 dévelopeurs !

Rejoignez Ogury pour voir la naissance d’une licorne !

- Récupération des données en temps-reel
- Ciblage en temps réel
- Data Lake
- Data Science
- Business Analysts
- Infrastructure

MEETUP DATA & CLOUD PARIS

00

Annexes

Comment choisir ses outils ?

Data pipeline : Job scheduler

EC2 : Instance pour scripts

EMR : Instance Hadoop/Spark

+ économique

- statique

Glue : Catalogue de données

Athena : Requêteur SQL sur fichier

+ rapide à développer/utiliser

- coûts à surveiller

Redshift : Data Warehouse clusterisé

Spectrum : Service Athena dans Redshift

+ performance

- investissement