UCNNCT

Plateforme de veille sociale multi-sources

Pipeline de traitement de données sur AWS

Eloka Michel & Gaby NJONOU • 2025-2026

Veille sociale • Big Data • AWS

Agenda

1. **Contexte & cas d'usage** – le besoin client
2. **Les 5 V du Big Data** – caractéristiques des données
3. **Architecture du pipeline** – vue détaillée
4. **Démonstration** – réalisée en live
5. **Analyse des coûts** – budget AWS
6. **Conclusion** – synthèse et perspectives

Contexte & cas d’usage

Contexte client

Le besoin

« Objectif »

« Comprendre ce qui se dit sur les réseaux sociaux et forums tech en temps réel pour anticiper les tendances du marché. »

Client fictif : Cabinet de conseil en innovation technologique

Problématique :

- Veille manuelle $\Rightarrow$ chronophage et incomplète
- Données dispersées sur plusieurs plateformes
- Pas de vision consolidée (tendances, signaux faibles, viralité)

Cas d'usage : UCNNCT

Unified Cloud-Native Content Tracking

Aspect	Description
Objectif	Collecter, traiter et analyser les contenus de **5 sources sociales**
Sources	Bluesky, Nostr, HackerNews, StackOverflow, RSS
Valeur	Détection de tendances tech, **keywords viraux**, patterns d'activité

Originalité : agrégation multi-sources avec **détection cross-platform** des sujets viraux.

Les 5 V du Big Data

Les 5 V appliqués au projet

V	UCNNCT
Volume	$\sim$ 1 million+ posts collectés
Vélocité	Ingestion temps réel (Kafka) + batch quotidien (Glue)
Variété	5 formats (JSON, API REST, WebSocket, RSS/XML)
Véracité	Déduplication Redis, nettoyage NLP, validation de schémas
Valeur	KPI actionnables, tendances, alertes business

Volume – données collectées

Source	Posts/jour	Format
Bluesky	∼ 50 000	JSON
Nostr	∼ 30 000	JSON
HackerNews	∼ 5 000	JSON
StackOverflow	∼ 10 000	JSON
RSS Feeds	∼ 2 000	XML
Total	**∼ 97 000/j**	

Stockage (ordre de grandeur)

- RAW : ∼ 500 MB / jour
- Processed : ∼ 150 MB / jour
- Curated : ∼ 5 MB / jour

⇒ **Data Lake S3 en 3 couches**

Vélocité – flux de données

Temps réel
Collectors → **Kafka** → **S3 RAW**
Consumer : batch ∼ 2000 messages

Latence cible : < 5 s

Batch (quotidien)
Glue Jobs à 6h UTC

- Transform (5 jobs en parallèle)
- Aggregation (KPIs)

Latence typique : ∼ 3 min

Variété – 5 sources, 5 formats

Source	Protocole	Spécificités
Bluesky	AT Protocol	Posts, reposts, likes
Nostr	WebSocket	Events signés, tags imbriqués
HackerNews	REST API	Stories, comments, scores
StackOverflow	REST API	Questions, réponses, votes
RSS	XML/Atom	Articles multi-sites

Défi : normaliser ces formats hétérogènes vers un **schéma unifié**.

Véracité – qualité des données

Étape	Technique	Outil
Déduplication	Hash ID + cache TTL	Redis (24h)
Nettoyage	Trim, regex, normalisation	PySpark (Glue)
Validation	Enforcement de schéma	Glue Catalog
Enrichissement	Extraction de keywords	NLP basique

Exemple (nettoyage)

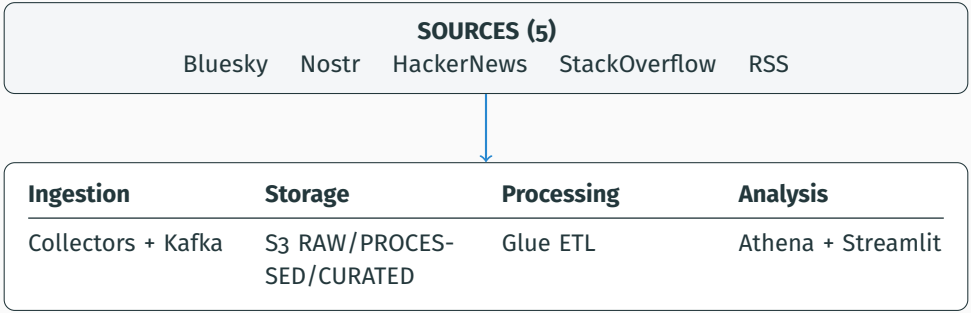
```
# Exemple nettoyage
content_clean = trim(regexp_replace(content, r'[\n\r\t]', ' '))
```

Valeur – KPIs business

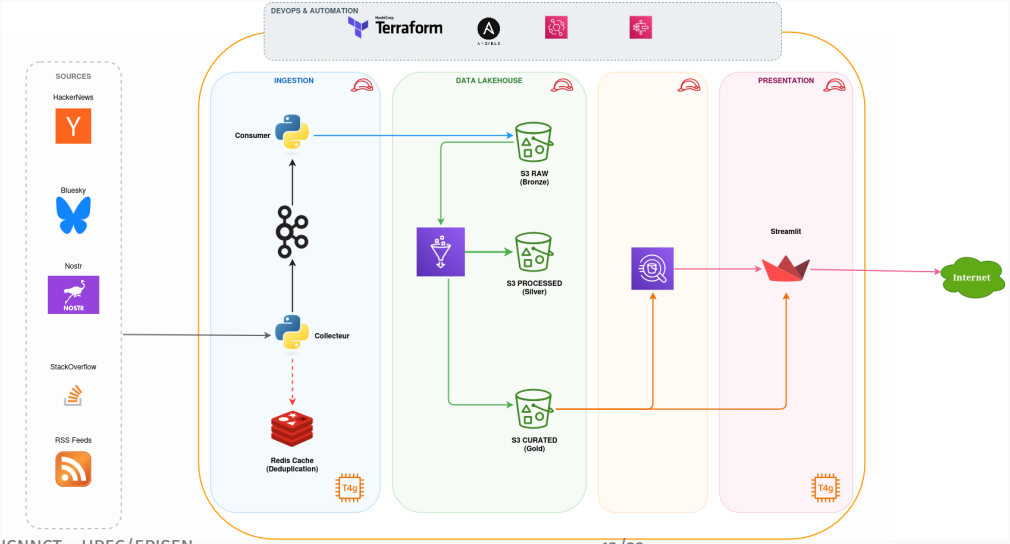
KPI	Question business
Volume par source	Quelle plateforme est la plus active ?
Trending keywords	Quels sujets émergent ?
Cross-source keywords	Quels sujets deviennent viraux (multi-plateformes) ?
Taux de croissance	Le buzz augmente ou diminue ?
Patterns horaires	Quand publier pour maximiser la visibilité ?

Architecture du pipeline

Architecture – vue globale



Architecture – détail



Phase 1 – ingestion

Collecte

- 5 collectors Python
- 5 topics Kafka
- Consumer $\rightarrow$ écriture S3 RAW (batch $\sim$ 2000 msg)

Cibles

- Latence ingestion : < 5 s
- Déduplication : TTL 24h
- Robustesse : retry + backoff

Infra
AWS : 2 instances EC2 (ARM64) : 1× t4g.small (collecte/consumer) + 1× t4g.micro (Streamlit)
Interne : Kafka + Redis sur VM dédiée

Phase 2 – stockage (Data Lake S3)

3 couches (Bronze / Silver / Gold)

	RAW (Bronze)	PROCESSED (Silver)	CURATED (Gold)
Format	JSON brut	Parquet nettoyé	Parquet agrégé (KPIs)
Volume	$\sim$ 500 MB/j	$\sim$ 150 MB/j	$\sim$ 5 MB/j
Organisation	partitions source/date	partitions source	tables prêtes à consommer

Service AWS : Amazon S3

Phase 3 – processing (AWS Glue)

Transform (5 jobs parallèles)

- Nettoyage du contenu
- Normalisation du schéma
- Extraction de keywords
- Déduplication

Service AWS : AWS Glue (Spark serverless)

Aggregation (1 job)

- Volume par période
- Trending keywords
- Cross-source analysis
- Stats de contenu

Phase 4 – analyse & visualisation

Athena
SQL directement sur S3 (Parquet/partitions)

```sql
SELECT source, COUNT(*) AS total
FROM volume_by_source
GROUP BY source;
```

10+ tables disponibles

Services AWS : Amazon Athena, EC2 (Streamlit)

Dashboard Streamlit

- Vue d'ensemble (métriques)
- Tendances & croissance
- Filtres source/date
- Requêtes SQL intégrées

Automatisation – déploiement (Terraform)

Infrastructure as Code

```
terraform init
terraform apply
```

Ressources provisionnées

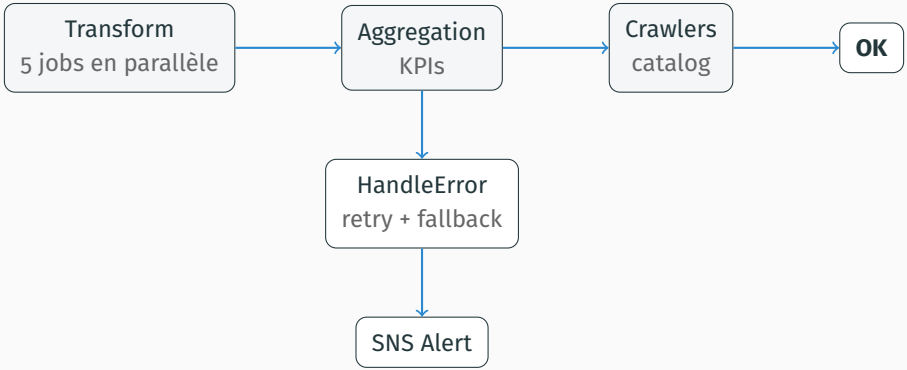
- VPC, Security Groups
- Buckets S3 (raw, processed, curated, scripts)
- Instance(s) EC2
- Glue Jobs (6) + Crawlers (2)
- Step Functions + EventBridge
- Athena Workgroup + Tables (10+)

12 fichiers Terraform ⇒ déploiement complet reproductible.

Automatisation – exécution (Step Functions)

Planification
EventBridge : `cron(0 6 * * ? *)`

Tous les jours à 6h UTC



Démonstration

Démonstration – réalisée en live

Ce que nous allons montrer

1. Kafka : messages en temps réel (topics actifs)
2. S3 RAW : arrivée des fichiers et partitions
3. Step Functions : orchestration du batch Glue
4. Athena : requête KPI (cross-source keywords)
5. Streamlit : dashboard et filtres

Analyse des coûts

Analyse des coûts – budget AWS

Service	Hypothèse	Coût/mois
EC2 (ARM64)	t4g.medium (24/7) + t4g.micro (14h/j)	(24,53 + 3,57) = $28.10
EBS (gp3)	58 Go total (50 Go + 8 Go)	0,08 × 58 = $4.64
S3 Standard	100 Go stockés	0,023 × 100 = $2.30
Glue ETL	6 jobs/j × 3 min (2 DPU)	18 DPU-h × 0,44 = $7.92
Glue Crawlers	2 runs/j (10 min min/run)	10 DPU-h × 0,44 = $4.40
SES (Mail)	10k mails (quota EC2 déduit)	7k × 0,0001 = $0.70
Athena	∼ 10 Go scannés / mois (via Streamlit)	0,01 To × 5,00 = $0.05
Step Functions	∼ 6000 transitions/mois	(6k - 4k grat.) × tarif = $0.05
CloudWatch Logs	Ingestion (5 Go grat.) + Stockage	(Incl. Free Tier) ≈ $1.50
SNS	Notifications (100k HTTP/Email grat.)	(Incl. Free Tier) ≈ $0.50
TOTAL		$50.16/mois

**Note : La t4g.medium est active 24/7. La t4g.micro (Streamlit) est active de 06h à 20h.*

Optimisations possibles

Optimisation	Impact attendu
Reserved Instances / Savings Plans	-20% à -40% sur EC2 selon engagement
Arrêt planifié (nuit/week-end)	forte baisse si usage non 24/7
Spot Instances	jusqu'à -60% si tolérant aux interruptions
Partitionnement + Parquet	réduit le scan Athena $\Rightarrow$ coût quasi nul
Glue tuning (DPUs, bookmarks)	ajuste la conso au besoin réel

Objectif : réduire le coût sans sacrifier la fiabilité.

Conclusion

Conclusion – livrables

Exigence	Statut
Pipeline 4 phases	Ingestion $\rightarrow$ Storage $\rightarrow$ Processing $\rightarrow$ Analysis
Automatisation déploiement	Terraform (12 fichiers)
Automatisation exécution	Step Functions + EventBridge
5 sources de données	Bluesky, Nostr, HN, SO, RSS
Dashboard interactif	Streamlit + Athena
Analyse des coûts	$\sim$ \$50/mois

Perspectives

- **Sentiment Analysis** : comprendre le ton des posts (ex. AWS Comprehend)
- **Alertes temps réel** : notification si un keyword dépasse un seuil
- **ML Predictions** : prédire les tendances (ex. SageMaker)
- **Plus de sources** : X/Twitter, Reddit, LinkedIn
- **Multi-tenant** : dashboard par client

Message final
UCNNCT fournit une veille multi-sources **cloud-native**, automatisée et extensible, avec un coût maîtrisé.

Merci !

Questions ?

Repository : https ://github.com/njonou01/pipepline-lab
Stack : Terraform, Python, PySpark – AWS (S3, Glue, Athena, Step Functions, EC2) – Kafka, Redis, Streamlit

Annexes

Annexe A – structure du repo

```
uccnct/
+-- terraform/ # Infrastructure as Code
| +-- main.tf
| +-- variables.tf
| +-- storage.tf
| +-- compute.tf
| +-- glue.tf
| +-- ...
+-- src/
| +-- collectors/ # Scripts de collecte
| +-- consumers/ # Kafka to S3
| +-- dashboard/ # Streamlit app
+-- scripts/
| +-- glue/ # ETL PySpark
+-- docs/
| +-- presentation.tex
| +-- architecture.png
+-- README.md
```

Annexe B – services AWS utilisés

Service	Rôle
EC2	Compute (collectors, consumer, dashboard)
S3	Data Lake (RAW/PROCESSED/CURATED)
Glue	ETL Spark serverless
Glue Catalog	Metastore (schémas)
Athena	SQL analytique
Step Functions	Orchestration
EventBridge	Scheduler
CloudWatch	Monitoring & logs
SNS	Notifications
IAM	Sécurité & rôles

Annexe C – schéma de données (ex. volume_by_source)

Colonne	Type	Description
date	DATE	Date de collecte
source	STRING	Nom de la source
total_posts	BIGINT	Nombre total de posts
unique_posts	BIGINT	Posts uniques
aggregated_at	TIMESTAMP	Date d'agrégation

Annexe C – schéma de données (ex. keywords)

Colonne	Type	Description
date	DATE	Date de l'analyse
keyword	STRING	Mot-clé technique (ex. "python")
mentions	BIGINT	Nombre d'occurrences
sources_count	BIGINT	Nombre de sources différentes
aggregated_at	TIMESTAMP	Date d'agrégation

Annexe C – schéma de données (ex. volume_by_source)

Colonne	Type	Description
date	DATE	Date de collecte
source	STRING	Nom de la source
total_posts	BIGINT	Nombre total de posts
unique_posts	BIGINT	Posts uniques
aggregated_at	TIMESTAMP	Date d'agrégation

Annexe C – schéma de données (ex. hourly_activity)

Colonne	Type	Description
source	STRING	Source des données
hour	INTEGER	Heure de la journée (0-23)
day_of_week	INTEGER	Jour de la semaine (1=Dim, 7=Sam)
post_count	BIGINT	Volume moyen
aggregated_at	TIMESTAMP	Date d'agrégation

Annexe C – schéma de données (ex. growth_rate)

Colonne	Type	Description
date	DATE	Date
source	STRING	Source
total_posts	BIGINT	Volume du jour
prev_day_posts	BIGINT	Volume de la veille
growth_pct	DOUBLE	Croissance en %
aggregated_at	TIMESTAMP	Date d’agrégation

Annexe C – schéma de données (ex. global_summary)

Colonne	Type	Description
total_posts	BIGINT	Volume global historique
active_sources	BIGINT	Nombre de sources actives
first_post	TIMESTAMP	Premier post ingéré
last_post	TIMESTAMP	Dernier post ingéré
total_remapped	BIGINT	Total enrichis (NLP)
aggregated_at	TIMESTAMP	Date d'agrégation

Annexe D – requêtes Athena utiles

```sql
-- Volume par semaine
SELECT year_week, source, total_posts
FROM volume_by_week
ORDER BY year_week DESC;

-- Croissance jour/jour
SELECT date, source, growth_pct
FROM growth_rate
WHERE growth_pct IS NOT NULL
ORDER BY date DESC;

-- Keywords viraux
SELECT keyword, total_mentions, sources_count
FROM cross_source_keywords
```