TP1 : NIFI

RABHI Sofiene

28 novembre 2023

INSTALLATION DE NIFI

Avant l'Installation :

Contenus du Fichier ZIP Apache NiFi, une plateforme open source pour automatiser les flux de données, offre une structure de répertoires dans son fichier ZIP qui inclut:

1. bin: Contient des scripts pour gérer NiFi, comme run-nifi.bat pour Windows et nifi.sh pour Linux.
2. conf: Répertoire des fichiers de configuration, par exemple nifi.properties, pour personnaliser les paramètres de NiFi.
3. docs: Fournit la documentation de NiFi, y compris des guides d'utilisation et d'administration.
4. extensions: Contient les extensions de NiFi, telles que les processeurs et les services, pour étendre ses capacités.
5. lib: Répertoire des bibliothèques Java nécessaires au fonctionnement de NiFi et de ses extensions.
6. license: Fichier indiquant les termes de la licence Apache 2.0 sous laquelle NiFi est distribué.
7. notice: Contient des informations légales sur les composants tiers utilisés par NiFi.
8. readme: Fichier offrant un aperçu général de NiFi, incluant les prérequis et les instructions d'installation.

Après l'Installation :

Répertoires Créés Une fois Apache NiFi installé et lancé, plusieurs répertoires sont créés pour gérer et suivre les données :

1. flowfile_repository: Stocke les FlowFiles en cours de traitement, contenant des fichiers binaires représentant ces FlowFiles et leurs attributs.
2. logs: Répertoire de journalisation, enregistrant les événements, erreurs, et activités de NiFi.
3. provenance_repository: Garde une trace des événements de provenance, offrant une traçabilité des flux de données.
4. run: Contient des fichiers temporaires pour la gestion du cycle de vie de NiFi.
5. state: Stocke l'état des composants de NiFi, comme les contrôleurs et les services.
6. work: Répertoire pour les fichiers de travail utilisés par NiFi et ses extensions.

L'ajout de fichiers supplémentaires après l'installation d'Apache NiFi est crucial pour son fonctionnement optimal. Ces nouveaux répertoires, créés au démarrage initial de NiFi, jouent des rôles spécifiques dans la gestion et le suivi des flux de données. Par exemple, le répertoire **flowfile_repository** stocke les FlowFiles actifs, essentiels pour traiter les données en transit. Le répertoire **logs** est indispensable pour le débogage et le suivi des opérations, enregistrant chaque événement et anomalie. Les répertoires **provenance_repository**, **run**, **state**, et **work** fournissent respectivement la traçabilité des données, la gestion des fichiers temporaires, le stockage de l'état des composants, et un espace pour les données de travail. Ces ajouts reflètent la nature dynamique de NiFi, où la configuration initiale est étendue par des fonctionnalités en temps réel, essentielles pour une automatisation efficace et sécurisée du flux de données.

INTRODUCTION : Apache NiFi est une plateforme logicielle puissante et fiable conçue pour automatiser le flux de données entre les systèmes. Notre projet a pour objectif d'exploiter cette plateforme pour orchestrer un processus complet de récupération de données via un web service. Une fois les données reçues au format JSON, notre but est de les traiter de manière à obtenir un enregistrement par message, chacun avec un nom de fichier unique. Enfin, nous convertissons ces messages au format CSV et les sauvegardons individuellement dans un répertoire spécifique sur le disque dur.

→ Appel d'un web service, API et récupérer des données au format JSON

Dans la première étape de mon travail pratique, j'ai débuté par la recherche d'APIs gratuits, ce qui m'a conduit à utiliser le générateur d'API de Retool (https://retool.com/api-generator). J'ai pu ainsi personnaliser les champs de mon API, qui génère des données au format JSON. La structure souhaitée des données a été définie via l'interface de Retool, permettant la création d'un tableau de données structurées, comme illustré dans le lien fourni (https://retoolapi.dev/ARiEXD/data).

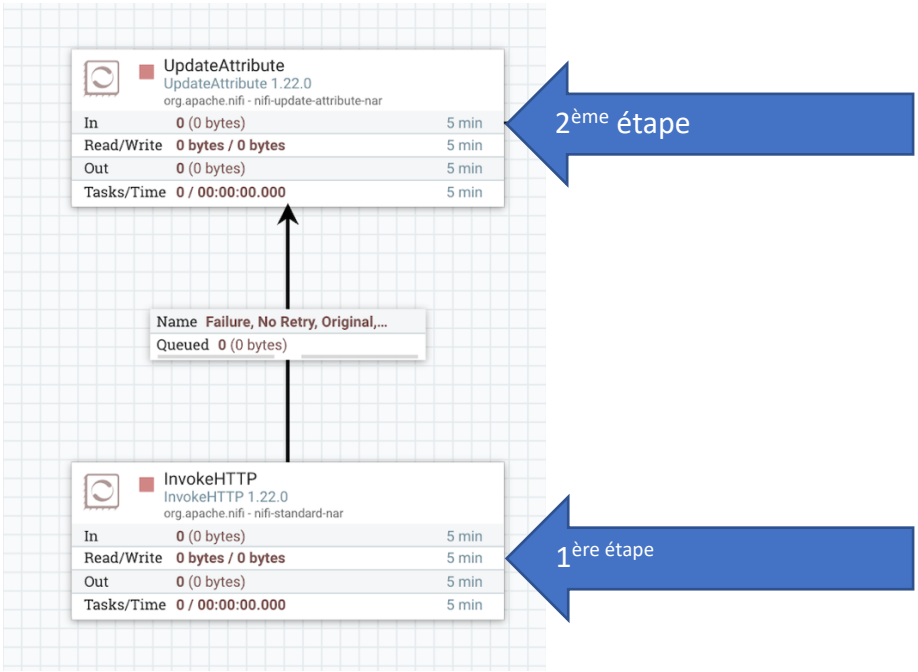

```
[
  {
    "id": 1,
    "name ": "Noelle Leverage",
    "location": "Florissant, Missouri, United States",
    "phone number ": "(555) 580-6782"
  },
  {
    "id": 2,
    "name ": "Purcell Brockman",
    "location": "Knoxville, Tennessee, United States",
    "phone number ": "(555) 157-8736"
  },
  {
    "id": 3,
    "name ": "Adrien Cathro",
    "location": "League City, Texas, United States",
    "phone number ": "(555) 847-2847"
  },
  {
    "id": 4,
    "name ": "Tory Cornels",
    "location": "Muskegon, Michigan, United States",
    "phone number ": "(555) 351-9495"
  },
  {
    "id": 5,
    "name ": "Mar Shields",
    "location": "Palo Alto, California, United States",
    "phone number ": "(555) 245-5025"
  },
  {
    "id": 6,
    "name ": "Warden Esposi",
    "location": "Daly City, California, United States",
    "phone number ": "(555) 962-3771"
  },
  {
    "id": 7,
    "name ": "Karalee Davidi",
    "location": "Yuba City, California, United States",
    "phone number ": "(555) 927-6524"
  },
  {
    "id": 8,
    "name ": "Anatollo Trousdale",
    "location": "Stamford, Connecticut, United States",
    "phone number ": "(555) 305-7346"
  },
  {
    "id": 9,
    "name ": "Keane Oliver",
    "location": "Gresham, Oregon, United States",
    "phone number ": "(555) 680-7075"
  },
```


API Preview

id	Column 1
1	Ashil Windrum
2	Briant Egdale
3	Harmony Boyle
4	Aigneis Pordal
5	Haley Alvy
6	Imogene McCardle
7	Phylis Eich
8	Zaneta Roebottom
9	Ashil Beyn
10	Ardeen Olenchenko
11	Haley Ducker
12	Itch Peter
13	Geno Gecks
14	Netti Cassell
15	Alvinia Biggin
16	Allyn Albin
17	Johannah Wathall
18	Katie Leverage
19	Caron Kentwell
20	Trixy Pyne
21	Marisa Loache
22	Sibylle Worviell

Showing 50 results

Ensuite, sur Apache NiFi, j'ai configuré un processus **InvokeHTTP** pour appeler l'API et recevoir les données JSON. Ce composant est essentiel pour intégrer les flux de données externes dans notre environnement de traitement. Pour compléter la configuration, j'ai ajouté un processeur **UpdateAttribute** pour modifier les attributs des fichiers récupérés, ce qui est crucial pour la gestion des métadonnées et la préparation des données pour les étapes suivantes. Cette configuration initiale a établi les fondations nécessaires pour le traitement et la manipulation des données reçues. Voici en dessous en image nos processus.

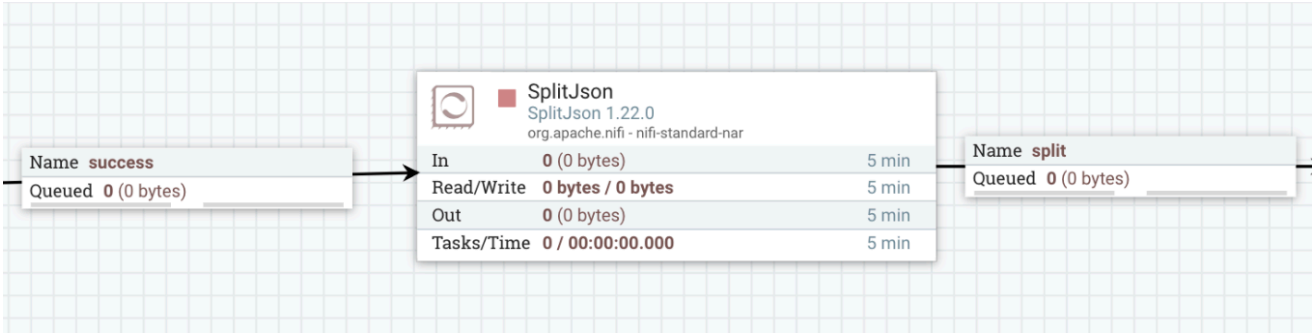


→ Spliter les données reçues pour avoir un enregistrement par message

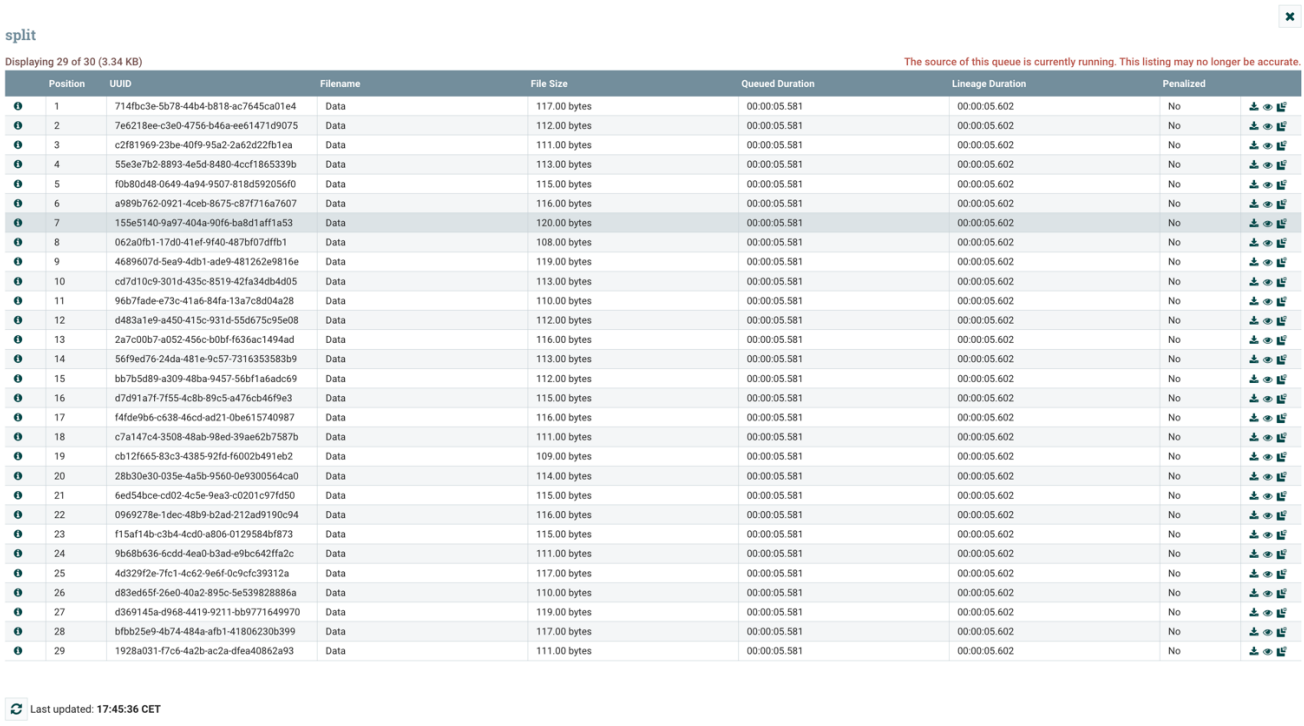
Après avoir récupéré les données JSON du web service, la deuxième étape consistait à séparer ces données en enregistrements distincts. J'ai utilisé le processeur SplitJson dans Apache NiFi, qui est spécialement conçu pour décomposer un flux de données JSON en plusieurs parties plus petites. Ce composant analyse les données et les divise en fonction d'un chemin spécifié ou de la structure du JSON. En l'occurrence, chaque objet JSON représentant un message a été séparé, permettant ainsi un traitement individuel ultérieur.

L'objectif de cette opération était de faciliter la gestion des messages, en veillant à ce que chaque message soit traité comme une entité unique lors des étapes de transformation et de stockage.

Comme cela nous avons pu garantir que chaque message serait correctement identifié et routé pour le traitement subséquent. Cette étape est essentielle pour maintenir l'intégrité des données et pour préparer le terrain pour leur conversion et leur sauvegarde individuelles. Voici une image du processus SplitJson sur notre machine.



La capture d'écran suivante montre le résultat du processus SplitJson dans Apache NiFi, où l'on peut voir une liste d'enregistrements séparés prêts pour la prochaine étape de traitement. Chaque enregistrement est maintenant isolé avec une taille de fichier spécifique, ce qui indique que les données JSON ont été correctement fractionnées. Grâce à cette étape, nous disposons désormais d'une base solide pour la manipulation individuelle des messages, qui sont prêts à être renommés et convertis en format CSV.


split

Displaying 29 of 30 (3.34 KB) — The source of this queue is currently running. This listing may no longer be accurate.

	Position	UUID	Filename	File Size	Queued Duration	Lineage Duration	Penalized
	1	714fbc3e-5b78-44b4-b818-ac7645ca01e4	Data	117.00 bytes	00:00:05.581	00:00:05.602	No
	2	7e6218ee-c3e0-4756-b46a-ee61471d9075	Data	112.00 bytes	00:00:05.581	00:00:05.602	No
	3	c2f81969-23be-40f9-95a2-2a62d22fb1ea	Data	111.00 bytes	00:00:05.581	00:00:05.602	No
	4	55e3e7b2-8893-4e5d-8480-4ccf1865339b	Data	113.00 bytes	00:00:05.581	00:00:05.602	No
	5	f0b80d48-0649-4a94-9507-818d592056f0	Data	115.00 bytes	00:00:05.581	00:00:05.602	No
	6	a989b762-0921-4ceb-8675-c87f716a7607	Data	116.00 bytes	00:00:05.581	00:00:05.602	No
	7	155e5140-9a97-404a-90f6-ba8d1aff1a53	Data	120.00 bytes	00:00:05.581	00:00:05.602	No
	8	062a0fb1-17d0-41ef-9f40-487bf07dffb1	Data	108.00 bytes	00:00:05.581	00:00:05.602	No
	9	4689607d-5ea9-4db1-ade9-481262e9816e	Data	119.00 bytes	00:00:05.581	00:00:05.602	No
	10	cd7d10c9-301d-435c-8519-42fa34db4d05	Data	113.00 bytes	00:00:05.581	00:00:05.602	No
	11	96b7fade-e73c-41a6-84fa-13a7c8d04a28	Data	110.00 bytes	00:00:05.581	00:00:05.602	No
	12	d483a1e9-a450-415c-931d-55d675c95e08	Data	112.00 bytes	00:00:05.581	00:00:05.602	No
	13	2a7c00b7-a052-456c-b0bf-f636ac1494ad	Data	116.00 bytes	00:00:05.581	00:00:05.602	No
	14	56f9ed76-24da-481e-9c57-7316353583b9	Data	113.00 bytes	00:00:05.581	00:00:05.602	No
	15	bb7b5d89-a309-48ba-9457-56bf1a6adc69	Data	112.00 bytes	00:00:05.581	00:00:05.602	No
	16	d7d91a7f-7f55-4c8b-89c5-a476cb46f9e3	Data	115.00 bytes	00:00:05.581	00:00:05.602	No
	17	f4fde9b6-c638-46cd-ad21-0be615740987	Data	116.00 bytes	00:00:05.581	00:00:05.602	No
	18	c7a147c4-3508-48ab-98ed-39ae62b7587b	Data	111.00 bytes	00:00:05.581	00:00:05.602	No
	19	cb12f665-83c3-4385-92fd-f6002b491eb2	Data	109.00 bytes	00:00:05.581	00:00:05.602	No
	20	28b30e30-035e-4a5b-9560-0e9300564ca0	Data	114.00 bytes	00:00:05.581	00:00:05.602	No
	21	6ed54bce-cd02-4c5e-9ea3-c0201c97fd50	Data	115.00 bytes	00:00:05.581	00:00:05.602	No
	22	0969278e-1dec-48b9-b2ad-212ad9190c94	Data	116.00 bytes	00:00:05.581	00:00:05.602	No
	23	f15af14b-c3b4-4cd0-a806-0129584bf873	Data	115.00 bytes	00:00:05.581	00:00:05.602	No
	24	9b68b636-6cdd-4ea0-b3ad-e9bc642ffa2c	Data	111.00 bytes	00:00:05.581	00:00:05.602	No
	25	4d329f2e-7fc1-4c62-9e6f-0c9cfc39312a	Data	117.00 bytes	00:00:05.581	00:00:05.602	No
	26	d83ed65f-26e0-40a2-895c-5e539828886a	Data	110.00 bytes	00:00:05.581	00:00:05.602	No
	27	d369145a-d968-4419-9211-bb9771649970	Data	119.00 bytes	00:00:05.581	00:00:05.602	No
	28	bfbb25e9-4b74-484a-afb1-41806230b399	Data	117.00 bytes	00:00:05.581	00:00:05.602	No
	29	1928a031-f7c6-4a2b-ac2a-dfea40862a93	Data	111.00 bytes	00:00:05.581	00:00:05.602	No

Last updated: 17:45:36 CET

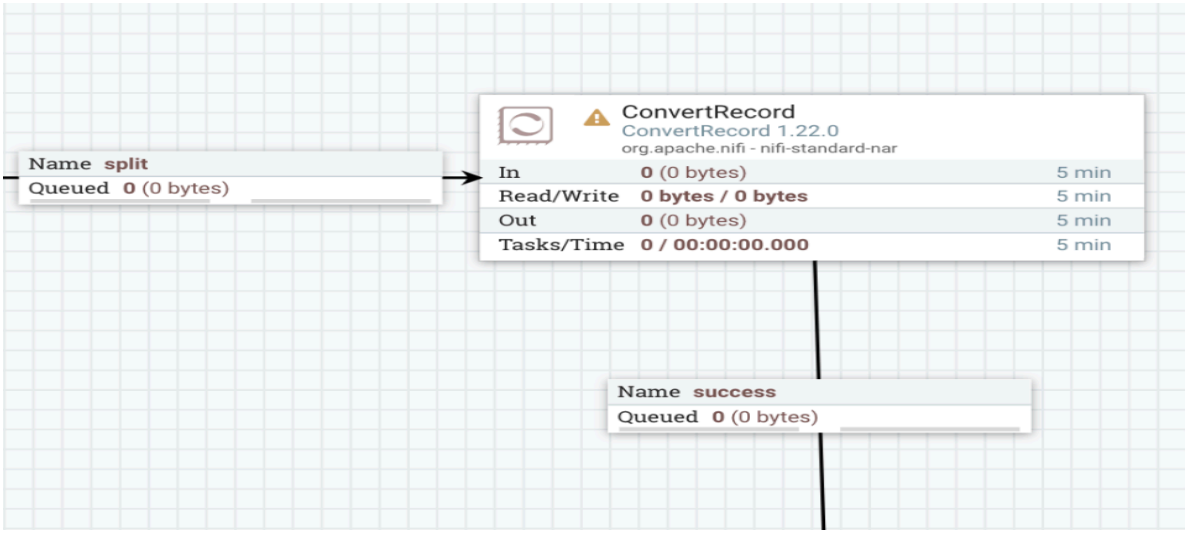
Voici le premier fichier affiche un enregistrement individuel avec des champs pour l'identifiant, le nom etc... Qui sont prêt pour la conversion en CSV.


View as: original

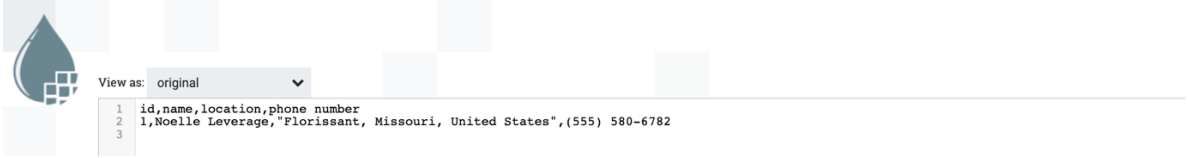

```
1 {"id":2,"name ":"Purcell Brockman","location":"Knoxville, Tennessee, United States","phone number ":"(555) 157-8736"}
```

→ Convertir chaque message vers le format CSV

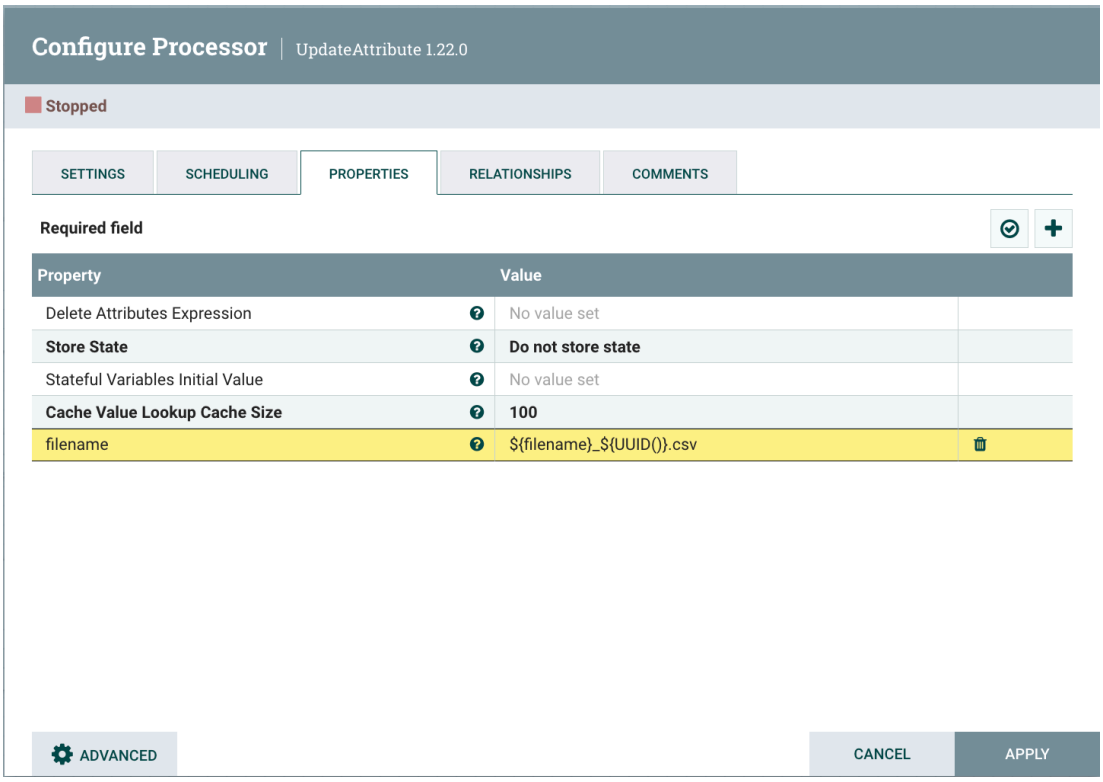
Le processeur ConvertRecord dans Apache NiFi joue un rôle clé à cette étape du projet. Il est configuré pour recevoir les flux de données JSON séparés précédemment par le SplitJson. L'objectif ici est de convertir ces données structurées au format JSON en un format CSV standardisé, qui est largement utilisé pour le stockage de données tabulaires.



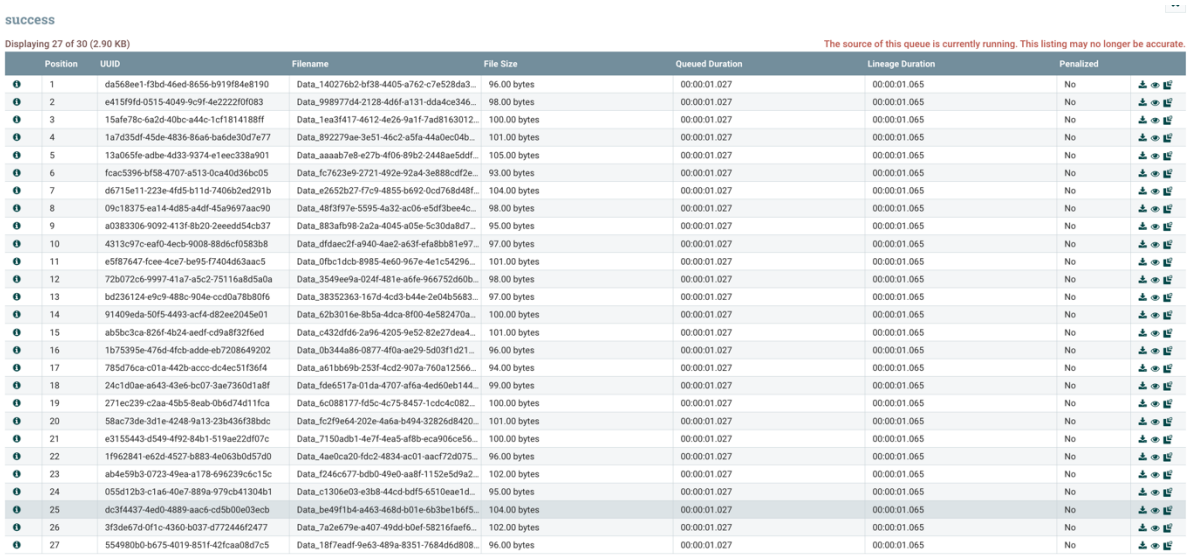
Voici le résultat que nous obtenons après la conversion :



→ Renommer le nom du fichier avec un nom unique pour chaque message



Le processeur UpdateAttribute de Apache NiFi a été configuré pour renommer chaque fichier CSV avec un nom unique. Cela a été réalisé en ajoutant un identifiant universel unique (UUID) au nom de fichier existant, assurant ainsi que chaque message ait un nom de fichier distinct. Cette pratique est cruciale pour éviter les conflits de noms et garantir l'unicité de chaque fichier lors de la sauvegarde sur le disque dur. Voici en image ce que cela nous donne.



success

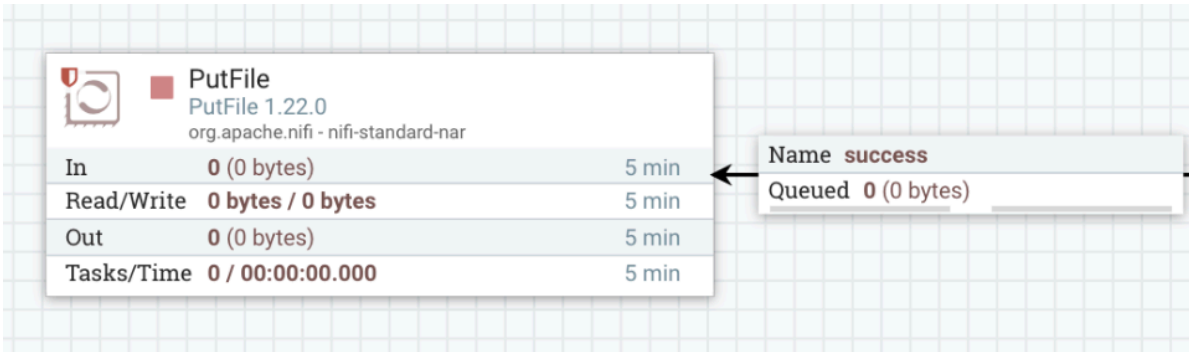
Displaying 27 of 30 (2.90 KB)

The source of this queue is currently running. This listing may no longer be accurate.

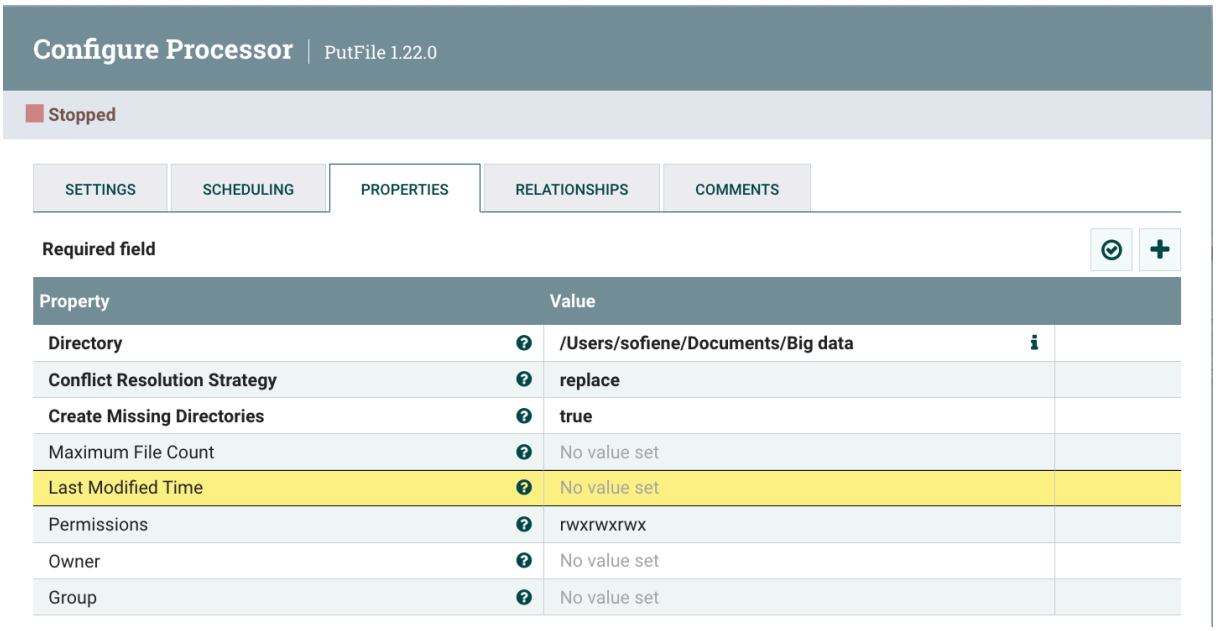
Position	UUID	Filename	File Size	Queued Duration	Lineage Duration	Penalized
1	da568ee1-f3bd-46ed-8656-b919f84e8190	Data_140276b2-bf38-4405-a762-c7e528da3...	96.00 bytes	00:00:01.027	00:00:01.065	No
2	e415f9fd-0515-4049-9c9f-4e2222f0f083	Data_998977d4-2128-4d6f-a131-dda4ce346...	98.00 bytes	00:00:01.027	00:00:01.065	No
3	15afe78c-6a2d-40bc-a44c-1cf1814188ff	Data_1ea3f417-4612-4e26-9a1f-7ad8163012...	100.00 bytes	00:00:01.027	00:00:01.065	No
4	1a7d35df-45de-4836-86a6-ba6de30d7e77	Data_892279ae-3e51-46c2-a5fa-44a0ec04b...	101.00 bytes	00:00:01.027	00:00:01.065	No
5	13a065fe-adbe-4d33-9374-e1eec338a901	Data_aaaab7e8-e27b-4f06-89b2-2448ae5ddf...	105.00 bytes	00:00:01.027	00:00:01.065	No
6	fcac5396-bf58-4707-a513-0ca40d36bc05	Data_fc7623e9-2721-492e-92a4-3e888cdf2e...	93.00 bytes	00:00:01.027	00:00:01.065	No
7	d6715e11-223e-4fd5-b11d-7406b2ed291b	Data_e2652b27-f7c9-4855-b692-0cd768d48f...	104.00 bytes	00:00:01.027	00:00:01.065	No
8	09c18375-ea14-4d85-a4df-45a9697aac90	Data_48f3f97e-5595-4a32-ac06-e5df3bee4c...	98.00 bytes	00:00:01.027	00:00:01.065	No
9	a0383306-9092-413f-8b20-2eeedd54cb37	Data_883afb98-2a2a-4045-a05e-5c30da8d7...	95.00 bytes	00:00:01.027	00:00:01.065	No
10	4313c97c-eaf0-4ecb-9008-88d6cf0583b8	Data_dfdaec2f-a940-4ae2-a63f-efa8bb81e97...	97.00 bytes	00:00:01.027	00:00:01.065	No
11	e5f87647-fcee-4ce7-be95-f7404d63aac5	Data_0fbc1dcb-8985-4e60-967e-4e1c54296...	101.00 bytes	00:00:01.027	00:00:01.065	No
12	72b072c6-9997-41a7-a5c2-75116a8d5a0a	Data_3549ee9a-024f-481e-a6fe-966752d60b...	98.00 bytes	00:00:01.027	00:00:01.065	No
13	bd236124-e9c9-488c-904e-ccd0a78b80f6	Data_38352363-167d-4cd3-b44e-2e04b5683...	97.00 bytes	00:00:01.027	00:00:01.065	No
14	91409eda-50f5-4493-acf4-d82ee2045e01	Data_62b3016e-8b5a-4dca-8f00-4e582470a...	100.00 bytes	00:00:01.027	00:00:01.065	No
15	ab5bc3ca-826f-4b24-aedf-cd9a8f32f6ed	Data_c432dfd6-2a96-4205-9e52-82e27dea4...	101.00 bytes	00:00:01.027	00:00:01.065	No
16	1b75395e-476d-4fcb-adde-eb7208649202	Data_0b344a86-0877-4f0a-ae29-5d03f1d21...	96.00 bytes	00:00:01.027	00:00:01.065	No
17	785d76ca-c01a-442b-accc-dc4ec51f36f4	Data_a61bb69b-253f-4cd2-907a-760a12566...	94.00 bytes	00:00:01.027	00:00:01.065	No
18	24c1d0ae-a643-43e6-bc07-3ae7360d1a8f	Data_fde6517a-01da-4707-af6a-4ed60eb144...	99.00 bytes	00:00:01.027	00:00:01.065	No
19	271ec239-c2aa-45b5-8eab-0b6d74d11fca	Data_6c088177-fd5c-4c75-8457-1cdc4c082...	100.00 bytes	00:00:01.027	00:00:01.065	No
20	58ac73de-3d1e-4248-9a13-23b436f38bdc	Data_fc2f9e64-202e-4a6a-b494-32826d8420...	101.00 bytes	00:00:01.027	00:00:01.065	No
21	e3155443-d549-4f92-84b1-519ae22df07c	Data_7150adb1-4e7f-4ea5-af8b-eca906ce56...	100.00 bytes	00:00:01.027	00:00:01.065	No
22	1f962841-e62d-4527-b883-4e063b0d57d0	Data_4ae0ca20-fdc2-4834-ac01-aacf72d075...	96.00 bytes	00:00:01.027	00:00:01.065	No
23	ab4e59b3-0723-49ea-a178-696239c6c15c	Data_f246c677-bdb0-49e0-aa8f-1152e5d9a2...	102.00 bytes	00:00:01.027	00:00:01.065	No
24	055d12b3-c1a6-40e7-889a-979cb41304b1	Data_c1306e03-e3b8-44cd-bdf5-6510eae1d...	95.00 bytes	00:00:01.027	00:00:01.065	No
25	dc3f4437-4ed0-4889-aac6-cd5b00e03ecb	Data_be49f1b4-a463-468d-b01e-6b3be1b6f5...	104.00 bytes	00:00:01.027	00:00:01.065	No
26	3f3de67d-0f1c-4360-b037-d772446f2477	Data_7a2e679e-a407-49dd-b0ef-58216faef6...	102.00 bytes	00:00:01.027	00:00:01.065	No
27	554980b0-b675-4019-851f-42fcaa08d7c5	Data_18f7eadf-9e63-489a-8351-7684d6d808...	96.00 bytes	00:00:01.027	00:00:01.065	No

→Sauvegarder chaque message dans un fichier distinct dans le même répertoire disque dur

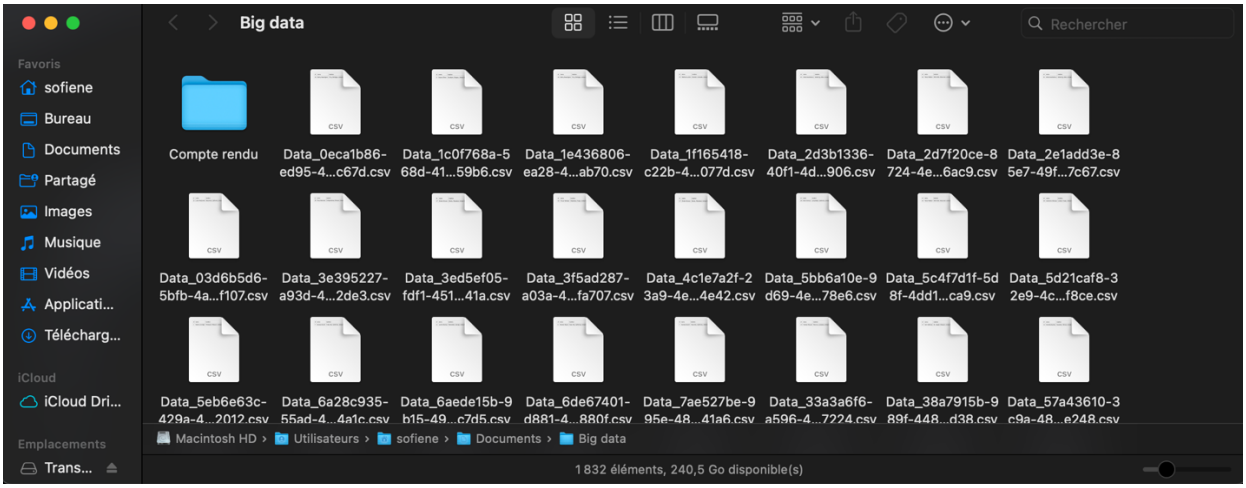
L'étape finale de ce processus consiste à sauvegarder chaque message, désormais au format CSV et avec un nom de fichier unique, dans un répertoire spécifique sur le disque dur. La première image montre la configuration du processeur PutFile dans Apache NiFi.



Ce processeur est utilisé pour écrire les fichiers dans le système de fichiers local. Il a été configuré pour placer les fichiers dans le répertoire /Users/sofiene/Documents/Big data, et pour remplacer tout fichier existant ayant le même nom, ce qui ne devrait pas se produire grâce à l'ajout d'un UUID unique à chaque nom de fichier comme on peut le voir ci-dessous.



L’image suivante montre le résultat de cette opération sur le système de fichiers local. On y voit un dossier appelé Big data contenant une grande quantité de fichiers CSV, chacun avec un nom distinct, conformément à la configuration NiFi. Cela démontre que le processus de sauvegarde s'est déroulé avec succès et que les fichiers sont prêts à être utilisés ou analysés ultérieurement.



CONCLUSION : Le projet démontre l'utilisation d'Apache NiFi pour gérer efficacement des données JSON récupérées via un web service. Il utilise des processeurs comme InvokeHTTP pour l'intégration, SplitJson pour la division des données, et ConvertRecord pour la conversion en CSV. Chaque fichier CSV, nommé de manière unique, est ensuite stocké sur le disque dur grâce au processeur PutFile. Ce processus illustre la capacité de NiFi à orchestrer le traitement des données de leur récupération à leur stockage, soulignant son potentiel dans le big data et l'analyse de données.

Schéma global Nifi