

[Ouvrir dans l'application](#)[Poursuivre](#)

582K Abonnés



Ceci est votre **dernière** histoire gratuite réservée aux membres ce mois-ci.
[Surclassement pour un accès illimité.](#)



Exemple de docker Apache NiFi et Kafka



Cory Maklin · 12 mai 2019 · 6 min de lecture ★

Dans l'écosystème Hadoop, Apache NiFi est couramment utilisé pour la phase d'ingestion. Apache NiFi offre un moyen évolutif de gérer le flux de données entre les systèmes. Lorsque vous essayez d'obtenir des informations d'un point A à un point B, de nombreux problèmes peuvent survenir. Par exemple, les réseaux peuvent tomber en panne, les logiciels se bloquent, les gens font des erreurs, les données peuvent être trop volumineuses, trop rapides ou au mauvais format. NiFi gérera ces problèmes sous le

[Ouvrir dans l'application](#)

Certaines des fonctionnalités de NiFi incluent:

Livraison garantie

Nifi garantie est la fourniture de données. Ceci est réalisé grâce à l'utilisation efficace d'un journal à écriture anticipée persistant et d'un référentiel de contenu spécialement conçus.

Mise en mémoire tampon des données / libération de pression

Nifi peut mettre en mémoire tampon les données lorsqu'une source de données donnée dépasse une partie de la chaîne de traitement ou de livraison. NiFi prend également en charge la possibilité de supprimer les données en file d'attente après un laps de temps spécifié.

File d'attente prioritaire

NiFi permet de définir un ou plusieurs schémas de hiérarchisation pour la manière dont les données sont extraites d'une file d'attente. La valeur par défaut est la plus ancienne en premier, mais il y a des moments où les données doivent être extraites les plus récentes en premier, les plus grandes en premier ou un autre schéma personnalisé.

Qualité de service

Il y a des moments où les données doivent être traitées et livrées en quelques secondes pour avoir une valeur quelconque. NiFi permet à l'administrateur de donner la priorité à la latence par rapport au débit ou à la tolérance aux pertes, etc.

Terminologie

Apache NiFi tourne autour de l'idée de processeurs. Un processeur est un nœud du graphique qui fonctionne. Cela consiste généralement à effectuer une sorte d'opération sur les données, à charger les données dans NiFi ou à envoyer les données à un système externe. Quelques exemples de processeurs sont:

- GetFile: charge le contenu d'un fichier
- UpdateAttribute: met à jour les attributs FlowFile (c'est-à-dire schema.name) qui peuvent ensuite être accédés par d'autres processeurs
- PublishKafka: envoie le contenu d'un FlowFile sous forme de message à Apache Kafka

Ouvrir dans l'application

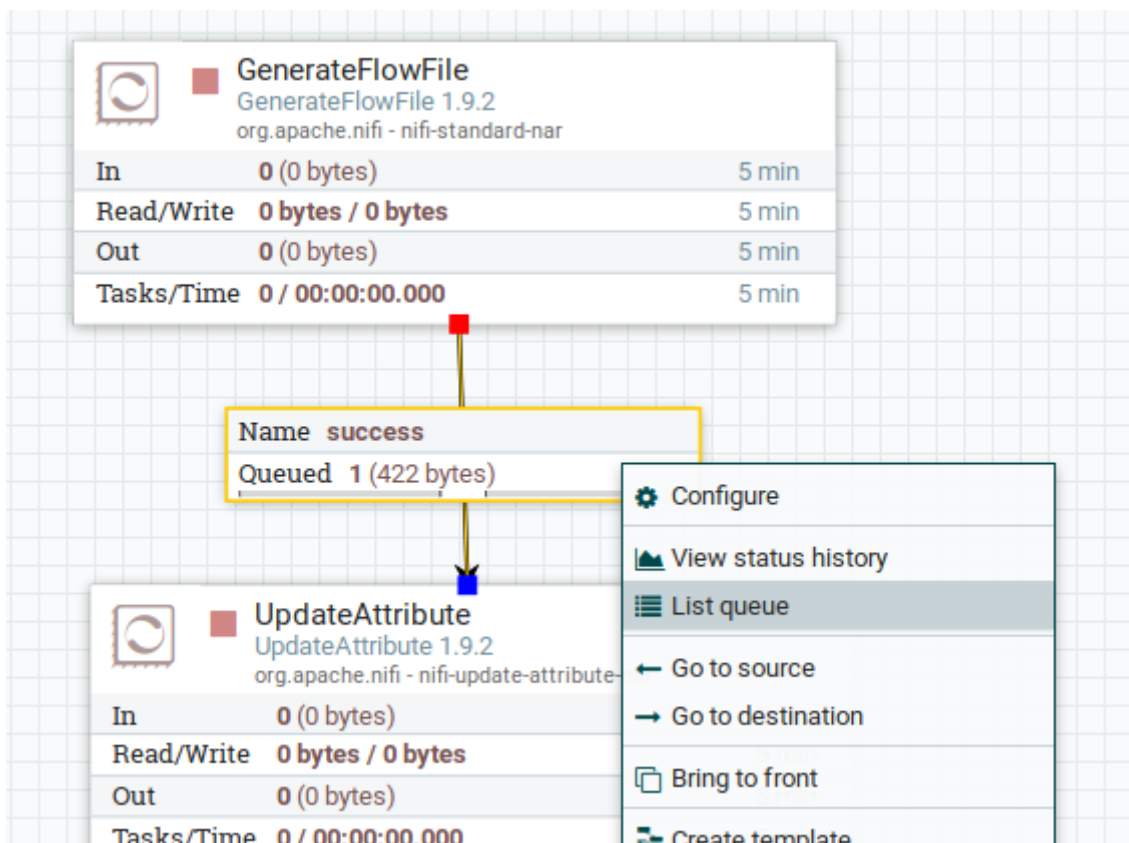


groupe de processus dans lequel ils ont été créés.

Voici quelques exemples de services de contrôleur:

- AvroSchemaRegistry: stocke les schémas Avro dans le registre qui peuvent ensuite être récupérés par d'autres services de contrôleur
- AvroRecordSetWriter: Écriture et code des données au format Avro

Un autre concept clé de NiFi est le FlowFile. Un FlowFile est constitué de données à une position donnée dans le graphique et de quelques métadonnées supplémentaires. Nous pouvons afficher FlowFiles en cliquant sur l'option de menu déroulant «*Liste des files d'attente*» d'une connexion.



Par exemple, le Flowfile suivant a un identifiant et un nom de fichier uniques.

FlowFile


DETAILS

ATTRIBUTES

FlowFile Details

Content Claim

[Ouvrir dans l'application](#)

79c97f20-85f1-496f-8b07-54867503953e	1
File Size	Identifier
422 bytes	1557575506693-1
Queue Position	Offset
No value set	3376
Queued Duration	Size
00:00:11.547	422 bytes
Lineage Duration	<div> DOWNLOAD</div> <div> VIEW</div>
00:00:11.547	
Penalized	
No	
<div>OK</div>	

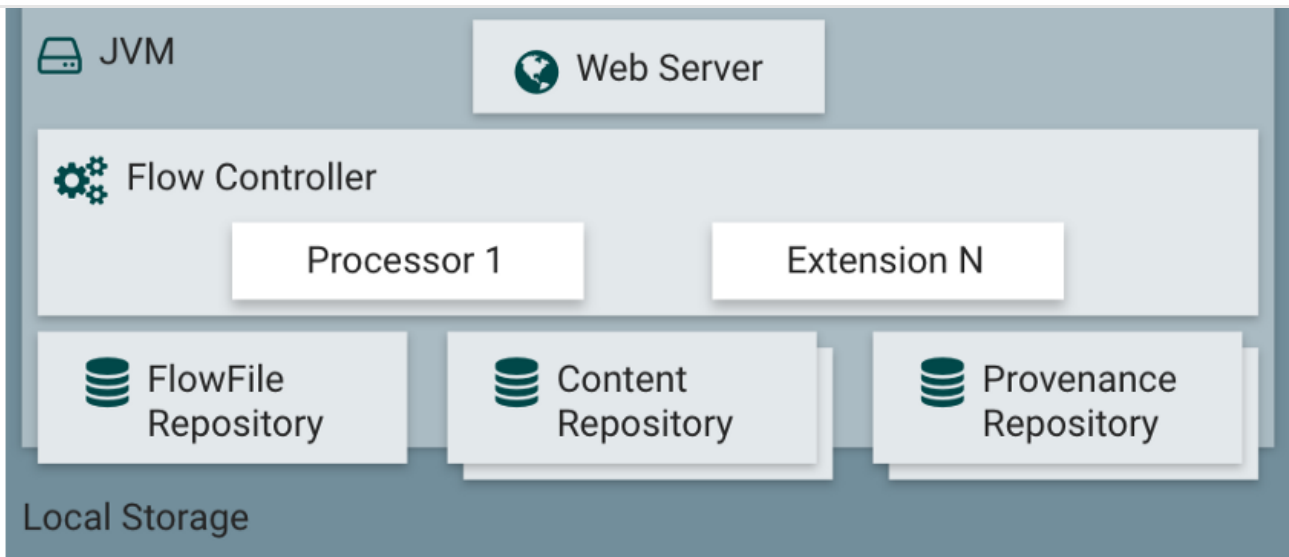
En cliquant sur la *vue* , nous pouvons voir les données réelles se déplacer d'un processeur à un autre.

View as: original		Filename: 79c97f20-85f1-496f-8b07-54867503953e
		Content Type: text/plain
1	YearsExperience,Salary	
2	1.1,39343.00	
3	1.3,46205.00	
4	1.5,37731.00	
5	2.0,43525.00	
6	2.2,39891.00	
7	2.9,56642.00	
8	3.0,60150.00	
9	3.2,54445.00	
10	3.2,64445.00	
11	3.7,57189.00	
12	3.9,63218.00	
13	4.0,55794.00	
14	4.0,56957.00	
15	4.1,57081.00	
16	4.5,61111.00	
17	4.9,67938.00	
18	5.1,66029.00	
19	5.3,83088.00	
20	5.9,81363.00	
21	6.0,93940.00	
22	6.8,91738.00	
23	7.1,98273.00	
24	7.9,101302.00	
25	8.2,113812.00	
26	8.7,109431.00	
27	9.0,105582.00	
28	9.5,116969.00	
29	9.6,112635.00	
30	10.3,122391.00	
31	10.5,121872.00	

Architecture

NiFi s'exécute dans une JVM sur un système d'exploitation hôte.

Ouvrir dans l'application



Serveur Web

Contrairement à la plupart des logiciels, l'administration d'Apache NiFi se fait via une interface utilisateur.

Contrôleur de débit

Gère toute la logique liée aux processeurs.

Référentiel FlowFile

Le référentiel FlowFile est l'endroit où NiFi stocke les métadonnées d'un FlowFile actuellement actif dans le flux.

Référentiel de contenu

Le référentiel de contenu est l'endroit où vit le contenu réel d'un FlowFile donné. Plusieurs emplacements de stockage du système de fichiers peuvent être spécifiés afin de réduire les conflits.

Référentiel de provenance

Le référentiel de provenance est l'endroit où toutes les données d'événement de provenance sont stockées. En substance, les données d'événement de provenance vous indiquent ce qui s'est produit et quand.

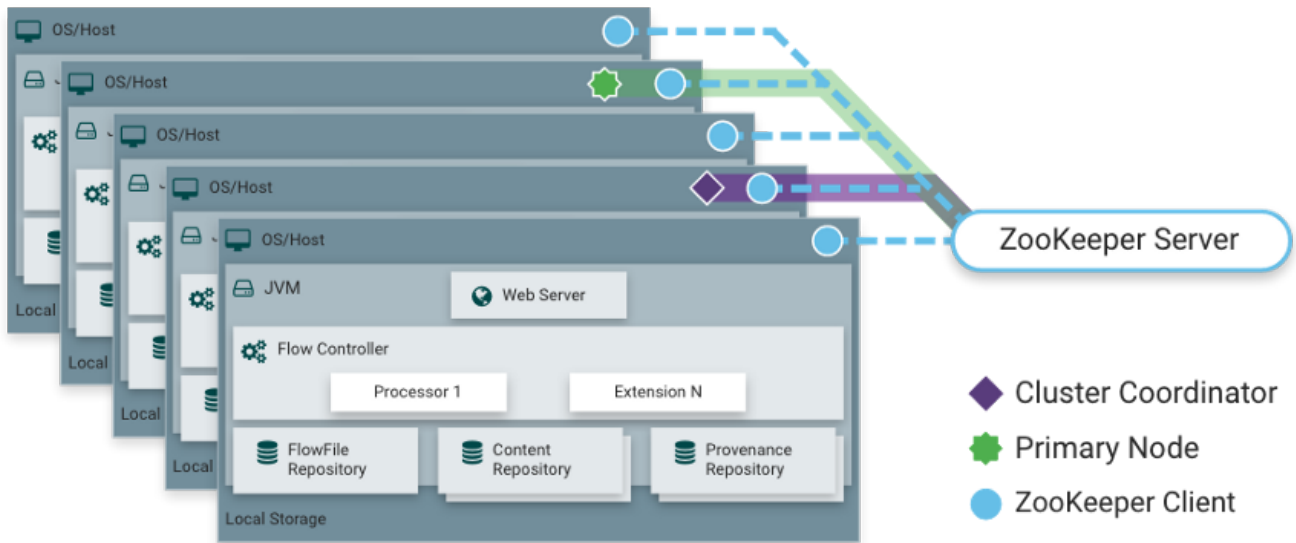
NiFi Data Provenance

Displaying 2 of 2

Oldest event available: 05/11/2019 11:51:46 UTC

Showing the events that match the specified query. [Clear search](#)

Date/Time	Type	FlowFile UUID	Size	Component Name	Component Type
05/11/2019 17:59:04.911 UTC	DOWNLOAD	79c97f20-85f1-496f-8b07-54867503953e	422 bytes	No value set	NiFi Flow
05/11/2019 17:58:09.283 UTC	CREATE	79c97f20-85f1-496f-8b07-54867503953e	422 bytes	GenerateFlowFile	GenerateFlowFile



Chaque nœud du cluster NiFi effectue les mêmes tâches sur les données, mais chacun fonctionne sur un ensemble de données différent. Apache ZooKeeper est utilisé pour élire le coordinateur de cluster et gérer automatiquement le basculement.

L'administrateur peut interagir avec le cluster NiFi via l'interface utilisateur de n'importe quel nœud et toute modification est répliquée sur tous les nœuds du cluster.

Code

Dans cet exemple, nous prendrons un fichier CSV et le publierons sur Kafka. Nous utiliserons docker pour configurer notre environnement. Copiez le contenu suivant dans `docker-compose.yml` et exécutez `docker-compose up -d`.

```

1 ---
2 version : ' 2 '
3 services :
4   gardien de zoo :
5     image : confluentinc / cp-zookeeper: dernier
6     environnement :
7       ZOOKEEPER_CLIENT_PORT : 2181
8       ZOOKEEPER_TICK_TIME : 2000
9
10  kafka :
11    # "` -._, - '"` -._, -' " `-._, - '"` -._, -' " `-._, - '"` -._, -'"
12    # Une note importante sur l'accès à Kafka à partir de clients sur d'autres machines:
13    # -----
14    #
15    # La configuration utilisée ici expose le port 9092 pour les connexions externes au court

```


[Ouvrir dans l'application](#)All nodes 

Nous utiliserons un ensemble de données simple contenant les années d'expérience et le salaire de 30 personnes.

Années d'expérience, salaire

```
1.1.39343.00
1.3.46205.00
1.5.37731.00
2.0.43525.00
2.2.39891.00
2.9.56642.00
3.0.60150.00
3.2.54445.00
3.2.64445.00
3.7.57189.00
3.9.63218.00
4.0.55794.00
4.0.56957.00
4.1.57081.00
4.5.61111.00
4.9, 67938,00
5,1,66029,00
5,3,83088,00
5,9,81363,00
6,0,93940,00
6,8,91738,00
7,1,98273,00
7,9,101302,00
8,2,113812,00
8,7,109431,00
9,0,105582,00
9,5,116969,00
9,6,112635,00
10,3,122391,00
10,5,121872,00
```

Again, configure the **GenerateFlowFile** by pasting the data into the custom text property.

Configure Processor

SETTINGS

SCHEDULING

PROPERTIES

COMMENTS

Ouvrir dans l'application



File Size	0B
Batch Size	1
Data Format	Text
Unique FlowFiles	
Custom Text	
Character Set	

1 YearsExperience,Salary
2 1.1,39343.00
3 1.3,46205.00
4 1.5,37731.00
5 2.0,43525.00
6 2.2,39891.00

☐ Set empty string

CANCEL
OK

CANCEL
APPLY

Note: any property in bold is mandatory, the rest are optional.

Drag a processor onto the main panel and select **UpdateAttribute**. For the **UpdateAttribute** processor, under properties, click on the plus sign in the top right corner. Then, create a **schema.name** property with a value of **test.schema**.

Configure Processor

SETTINGS

SCHEDULING

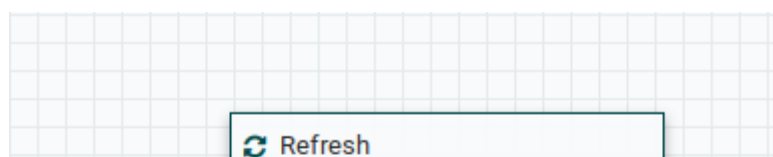
PROPERTIES

COMMENTS

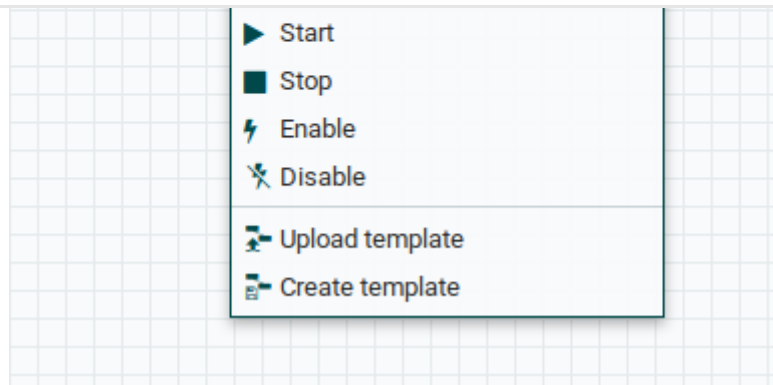
Required field
+

Property	Value
Delete Attributes Expression	No value set
Store State	Do not store state
Stateful Variables Initial Value	No value set
Cache Value Lookup Cache Size	100
schema.name	test-schema 🗑

Now, we'll create our controller services. Right click on the main panel and select **Configure**.



Ouvrir dans l'application



From here we can create all the controller services for our processor group.

Click the plus sign in the top right corner and select **AvroSchemaRegistry**. Under the properties tab, create a new property called **test-schema** and paste the following schema into the value field.

```
{
  "type" : "record",
  "namespace" : "Test",
  "name" : "Employee",
  "fields" : [
    { "name" : "YearsExperience" , "type" : "float" },
    { "name" : "Salary" , "type" : "float" }
  ]
}
```

Controller Service Details

SETTINGS

PROPERTIES

COMMENTS

Required field

Property	Value
Validate Field Names	true
test-schema	{ "type" : "record", "namespace" : "Test", "name" : "Employee", ...

We're going to need **CSVReader** as well. Under the properties tab, configure it to use the schema name property, the **AvroSchemaRegistry** and treat the first line as the column headers.

Ouvrir dans l'application



Required field +

Property	Value	
Schema Access Strategy	Use 'Schema Name' Property	
Schema Registry	AvroSchemaRegistry	→
Schema Name	\${schema.name}	
Schema Version	No value set	
Schema Branch	No value set	
Schema Text	\${avro.schema}	
CSV Parser	Apache Commons CSV	
Date Format	No value set	
Time Format	No value set	
Timestamp Format	No value set	
CSV Format	Custom Format	
Value Separator	,	
Treat First Line as Header	true	
Ignore CSV Header Column Names	false	

CANCEL APPLY

Finally, create a **AvroRecordSetWriter** and configure it to use the **AvroSchemaRegistry**.

Configure Controller Service

SETTINGS PROPERTIES COMMENTS

Required field +

Property	Value	
Schema Write Strategy	Embed Avro Schema	
Schema Cache	No value set	
Schema Access Strategy	Use 'Schema Name' Property	
Schema Registry	AvroSchemaRegistry	→
Schema Name	\${schema.name}	
Schema Version	No value set	
Schema Branch	No value set	
Schema Text	\${avro.schema}	
Compression Format	NONE	
Cache Size	1000	
Encoder Pool Size	32	

CANCEL APPLY

Before continuing, make sure that you enable all the controller services by clicking on the lightning bolt on the far right.

[Ouvrir dans l'application](#)

Now that we finished setting up the controller services, we'll create a Kafka topic by running the following command.

```
docker-compose exec kafka \
kafka-topics --create --topic test --partitions 1 --replication-
factor 1 --if-not-exists --zookeeper zookeeper:2181
```

Verify it worked correctly.

```
docker-compose exec kafka \
kafka-topics --describe --topic test --zookeeper zookeeper: 2181
```

Créez **PublishKafkaRecord** et configurez-le comme suit.

Required field		Value	
Property			
Kafka Brokers	?	kafka-single-node_kafka_1:29092	
Topic Name	?	test	
Record Reader	?	CSVReader	→
Record Writer	?	AvroRecordSetWriter	→
Security Protocol	?	PLAINTEXT	

Assurez-vous de vérifier les **relations de résiliation automatique** car il s'agit du dernier processeur de notre graphique.

Configure Processor

SETTINGS

SCHEDULING

PROPERTIES

COMMENTS

Name

PublishKafkaRecord_0_10

Id

abc19d12-016a-1000-21db-c8c69342487f

Type

PublishKafkaRecord_0_10 1.9.2

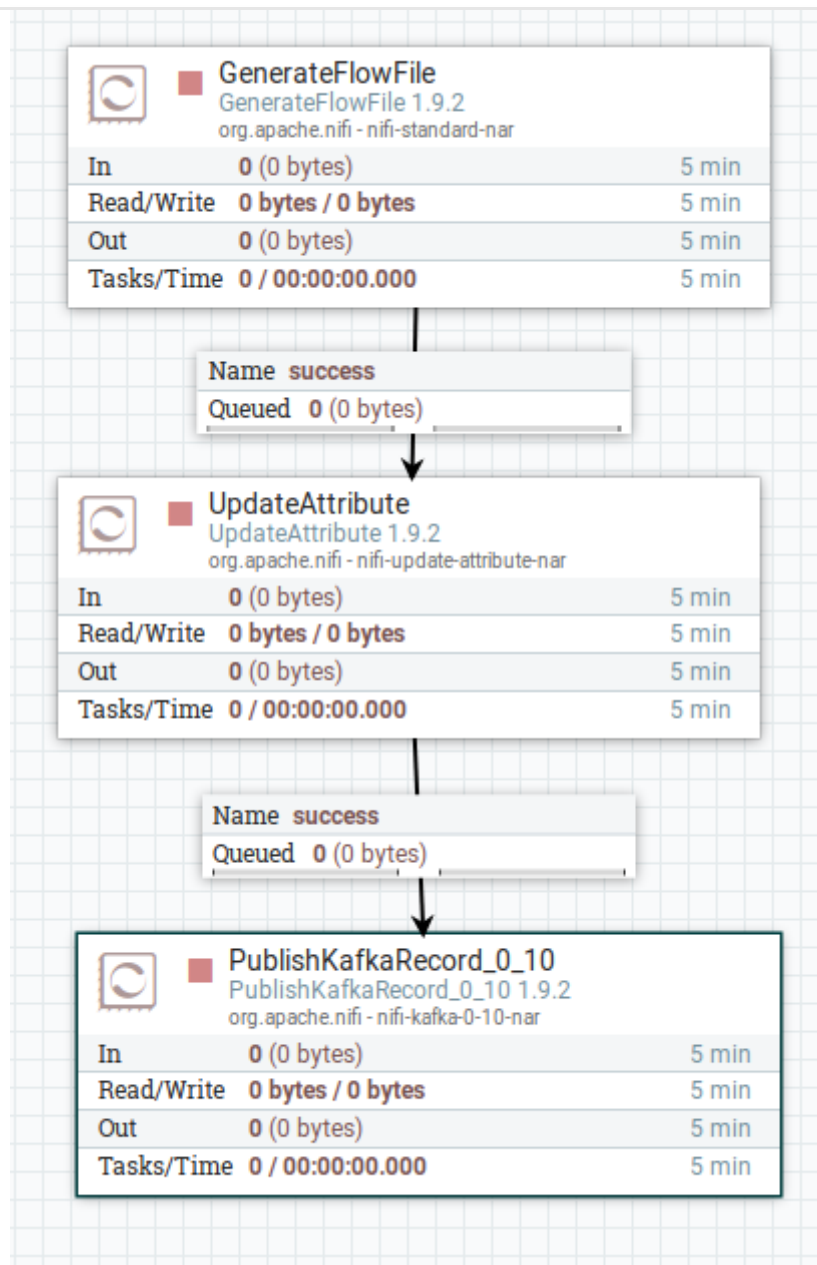
☒ Enabled

Automatically Terminate Relationships ?

☒ failure
Any FlowFile that cannot be sent to Kafka will be routed to this Relationship

☒ success
FlowFiles for which all content was sent to Kafka.

Ouvrir dans l'application



Enfin, pour démarrer le flux, faites un clic droit sur chaque processeur et sélectionnez **démarrer** . Si tout fonctionne comme prévu, chacune des lignes d'origine doit être réécrite.

```

docker-compose exec kafka \
  kafka-console-consumer --bootstrap-server localhost: 29092 --topic
test --from-begin --max-messages 30
  
```

Inscrivez-vous à The Variable

[Ouvrir dans l'application](#)

des recherches de pointe aux fonctionnalités originales a ne pas manquer. [Regarde.](#)

[Recevez cette newsletter](#)

Les e-mails seront envoyés à formationgeekjava@gmail.com.
[Pas toi?](#)

[Vers la science des données](#)[Apprentissage automatique](#)[Science des données](#)[Intelligence artificielle](#)[Programmation](#)

À AiderLégal
propos

Téléchargez l'application Medium

