

Il y a quelque temps, j'ai découvert Apache NiFi et en tant qu'ingénieur Big Data je l'ai immédiatement trouvé très attractif. En effet, cet outil permet de travailler sur la donnée avec une grande facilité.

Présentation de Nifi

Apache NiFi est un **projet open source** de la fondation Apache, supporté par Hortonworks. Il permet **d'injecter automatiquement des flux de données** entre différents systèmes sources en direction d'autres systèmes en cible.

Par exemple, NiFi peut être très utile dans un cas d'usage comme **l'alimentation d'un DataLake Hadoop** à partir de plusieurs sources de données.

Basé sur le paradigme de programmation [flow-based programming](#), NiFi fournit une interface web qui permet de **construire un flux de données en Drag et Drop**. Ainsi, il est possible de **définir**, de **contrôler en temps réel**, et d'une certaine manière, de **sécuriser** l'acheminement de données.

Apache NiFi **assure l'intégralité du flux de données**, il est tolérant aux pannes, est **scalable** et a été conçu pour gérer de gros volumes de données en temps réel.

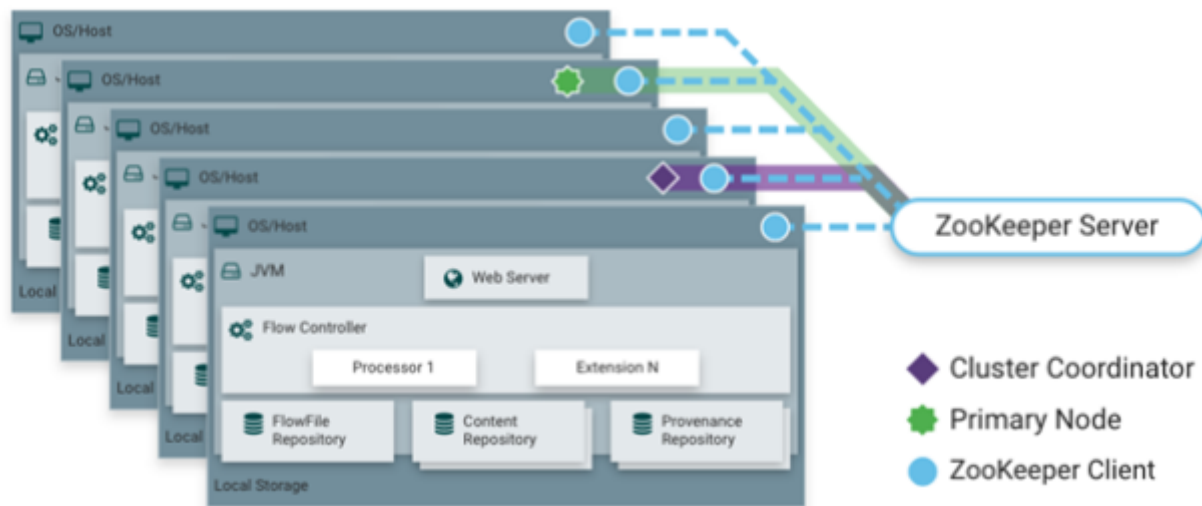
Apache NiFi est compatible avec [Kerberos](#) qui **assure l'authentification**, avec [Apache Ranger](#) qui permet la **sécurité des autorisations d'accès** et avec [Apache Knox](#) qui gère la **sécurité au niveau authentification** et celle des appels REST and HTTP.

Comment l'utiliser ?

NiFi tourne dans une JVM en mode local sur le système d'exploitation hôte (Windows, Linux ou Mac) mais **on peut aussi déployer Apache NiFi en mode cluster**, un de ses grands atouts une fois en production.

Il est conseillé d'installer le **cluster NiFi sur le Framework Hadoop**, mais vous pouvez également configurer un cluster NiFi en dehors de Hadoop.

Sachez qu'en **mode cluster**, vous aurez besoin d'[Apache Zookeeper](#) pour la gestion de la configuration afin **d'assurer la haute disponibilité des services**.



Apache NiFi en mode Cluster

Source : Hortonworks

Apache ZooKeeper élit un nœud unique en tant que coordinateur du cluster.

Tous les nœuds du cluster envoient un signal au coordinateur pour l'informer sur son état.

Le coordinateur de cluster est responsable de la **déconnexion et de la connexion des nœuds**. En quelque sorte, il joue le rôle du master sans en être réellement un.

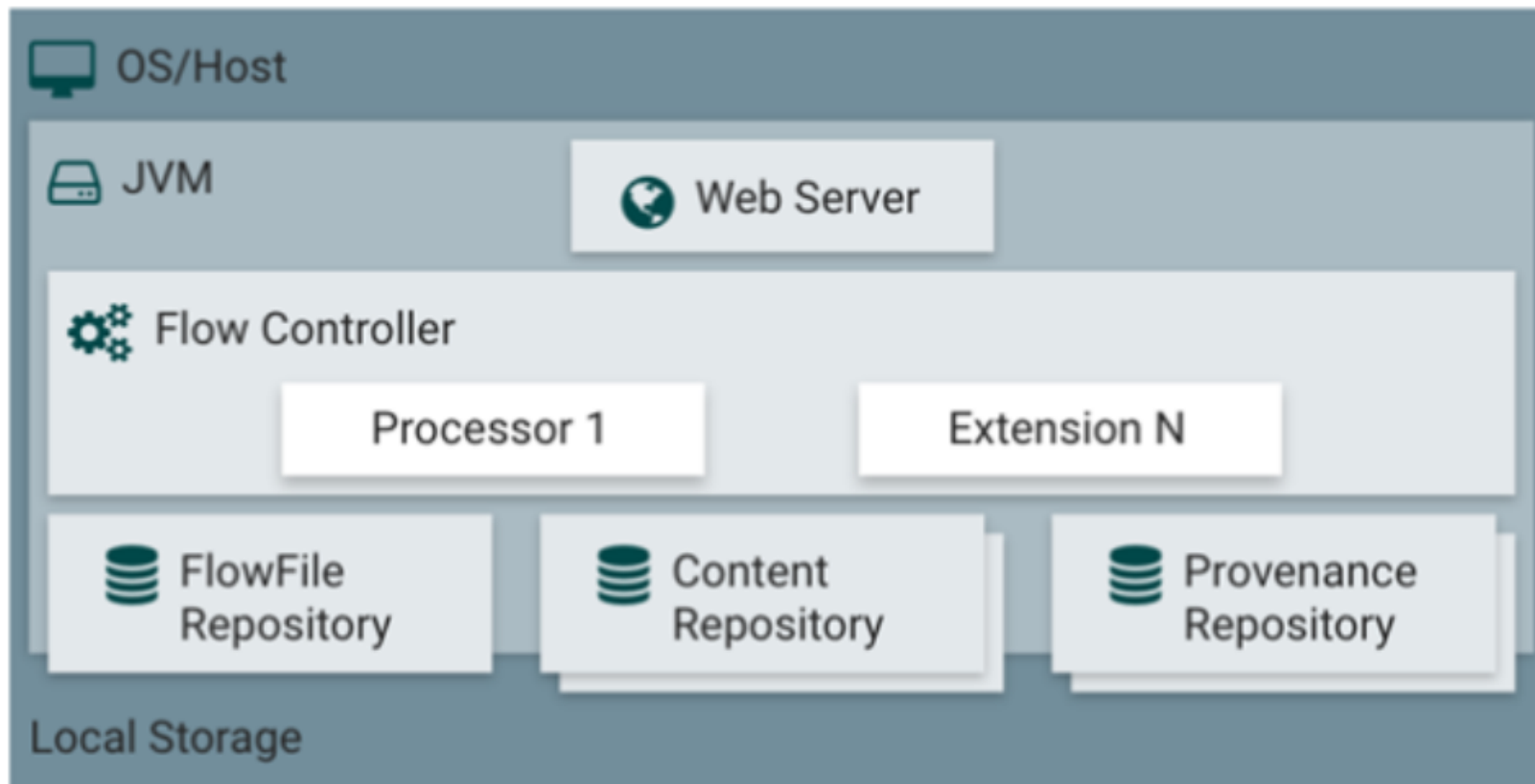
En outre, **chaque cluster possède un nœud principal**, également élu par ZooKeeper.

Pour plus de détails sur le **rôle de chaque nœud sur le cluster**, je vous invite à lire la documentation **NiFi System Administrator's Guide**.

Composants et fonctionnalités

Les **composants principaux de NiFi sur la JVM** sont les suivants :

- **Serveur Web** qui héberge l'interface graphique,
- Un **flow controller** qui orchestre les opérations. Il fournit des tâches à exécuter aux extensions et gère leur ordonnancement,
- **FlowFile Repository** dans lequel NiFi enregistre l'état d'un FlowFile,
- **Content Repository** où les données d'un FlowFile sont stockées,
- **Répertoire de la provenance** où toutes les données d'événement de provenance sont stockées.



Architecture NiFi

Source : Hortonworks

En outre, NiFi possède **plus de 200 connecteurs ou processeurs** qui permettent de **collecter en temps réel des données** issues de plus de 80 sources : bases de données, messages, fichiers, flux Twitter, etc.

Add Processor

Source

Displaying 242 of 242

Filter

all groups ▾	Type ▲	Version	Tags
amazon attributes	AttributeRollingWindow	1.4.0	rolling, data science, Attribute ...
avro aws consume	AttributesToJSON	1.4.0	flowfile, json, attributes
csv database	Base64EncodeContent	1.4.0	encode, base64
delete fetch get	CaptureChangeMySQL	1.4.0	cdc, jdbc, mysql, sql
hadoop ingest	CompareFuzzyHash	1.4.0	fuzzy-hashing, hashing, cyber-...
ingress insert json	CompressContent	1.4.0	lzma, decompress, compress, ...
listen logs	ConnectWebSocket	1.4.0	subscribe, consume, listen, We...
message pubsub	ConsumeAMQP	1.4.0	receive, amqp, rabbit, get, cons...
put record	ConsumeEWS	1.4.0	EWS, Exchange, Email, Consu...
restricted source	ConsumeIMAP	1.4.0	Imap, Email, Consume, Ingest, ...
text update	ConsumeJMS	1.4.0	jms, receive, get, consume, me...
	ConsumeKafka	1.4.0	PubSub, Consume, Ingest, Get

AttributeRollingWindow 1.4.0 org.apache.nifi - nifi-stateful-analysis-nar

Track a Rolling Window based on evaluating an Expression Language expression on each FlowFile and add that value to the processor's state. Each FlowFile will be emitted with the count of FlowFiles and total aggregate value of values processed in the current time window.

CANCEL

ADD

Recherche / Ajout d'un processeur

Source : *Apache NiFi*

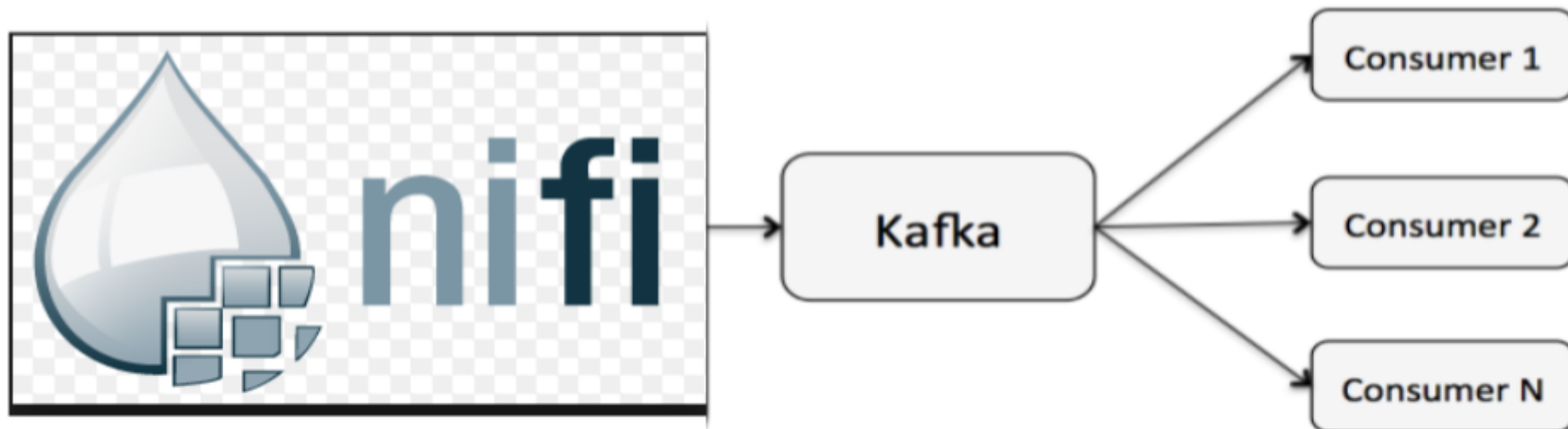
La richesse de la bibliothèque des processeurs permet de réaliser :

- La **data Transformation** : compresser ou décompresser des données, convertir un format de données vers un autre (Xml, Json, Txt to Parquet, Avro, etc.) ou bien faire de la crypto et de l'encodage (ConvertCharacterSet, EncryptContent, etc.),
- Les **accès aux bases de données** : par exemple le processeur Database Access ExecuteSQL ou bien PutHiveQL,
- La **Data Ingestion** : des processeurs permettent de récupérer des données issues de différentes sources (GetTwitter, GetMongo, GetFile, GetHttp, etc.),
- La **Data Egress/Sending Data** : des processeurs pour envoyer des données vers des systèmes cible (PutSQL, PutKafka, PutMongo, etc.).

Vous l'aviez deviné, avec Apache NiFi, il est possible de construire un **pipeline d'ingestion de données riche et complète**.

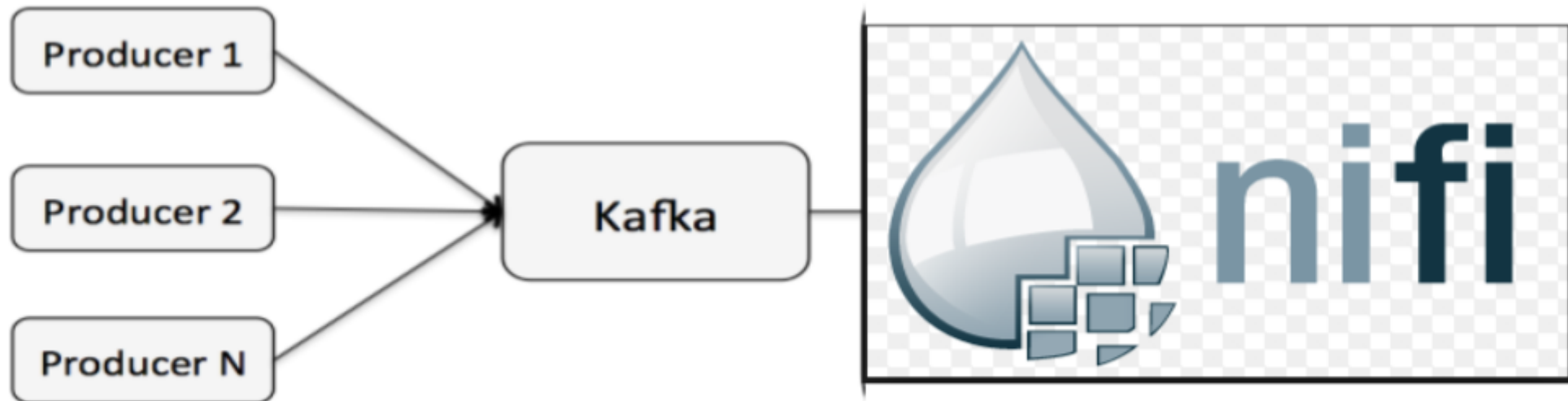
Apache NiFi s'intègre avec différents outils de streaming et sa **combinaison avec Apache Kafka est très robuste**.

NiFi peut jouer le rôle du producteur pour Kafka.



Apache NiFi en producteur de données pour Kafka

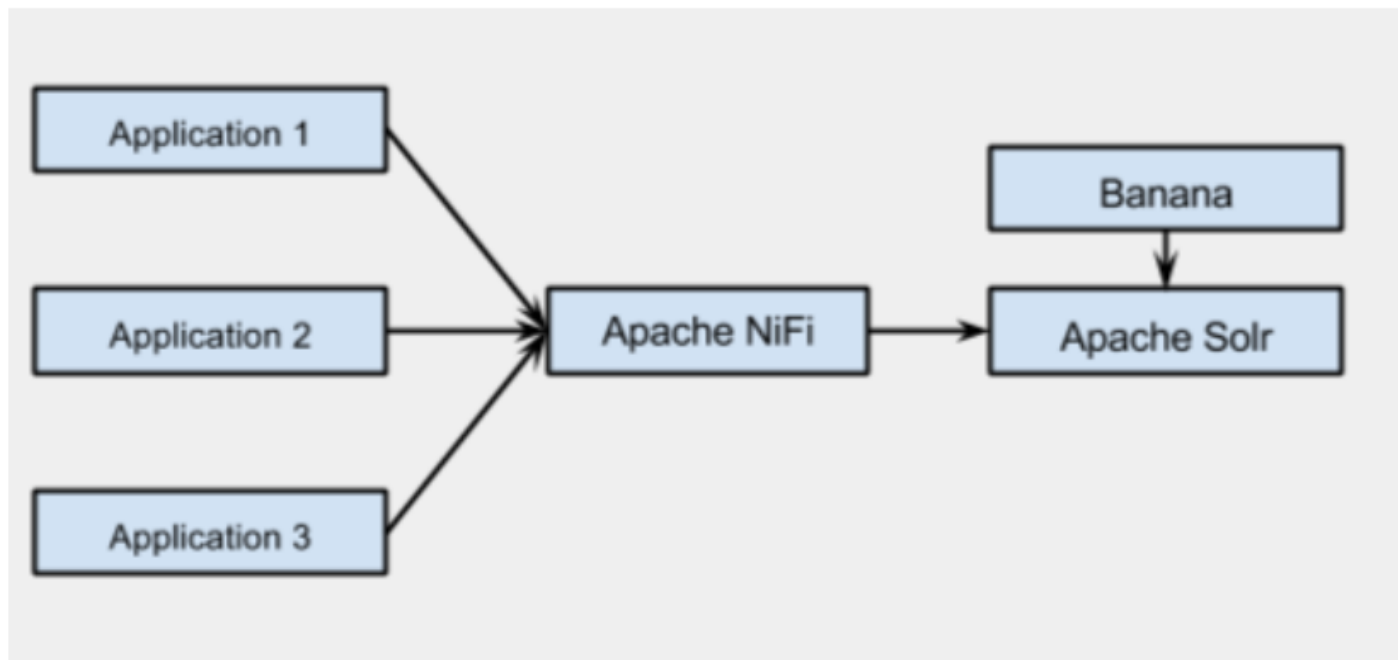
Source : *bryanbende.com*



Apache NiFi en consommateur de données

Source : *bryanbende.com*

De nombreux cas d'usage sont réalisables avec ces deux superbes outils, comme **construire un CDC** (Change Data Capture), **faire du streaming** avec Kafka ou encore **recupérer des logs de serveurs webs**, etc.



Préparer les données pour les indexer dans Apache Solr

Source : *bryanbende.com*

On peut également **stocker les données sortantes d'Apache Nifi sur le Cloud**, notamment sur le cloud d'Amazon ([S3](#) ou [Readshift](#)).

Je ne vais pas détailler plus les processeurs fournis par Apache NiFi, vous les trouverez à [cette adresse](#). Sachez qu'Apache NiFi vous permet de vous **connecter à beaucoup d'outils** producteurs de données, **d'effectuer des transformations** sur les données récoltées et de les **stocker** et de les **envoyer** vers différents **outils de stockage ou analytiques**.

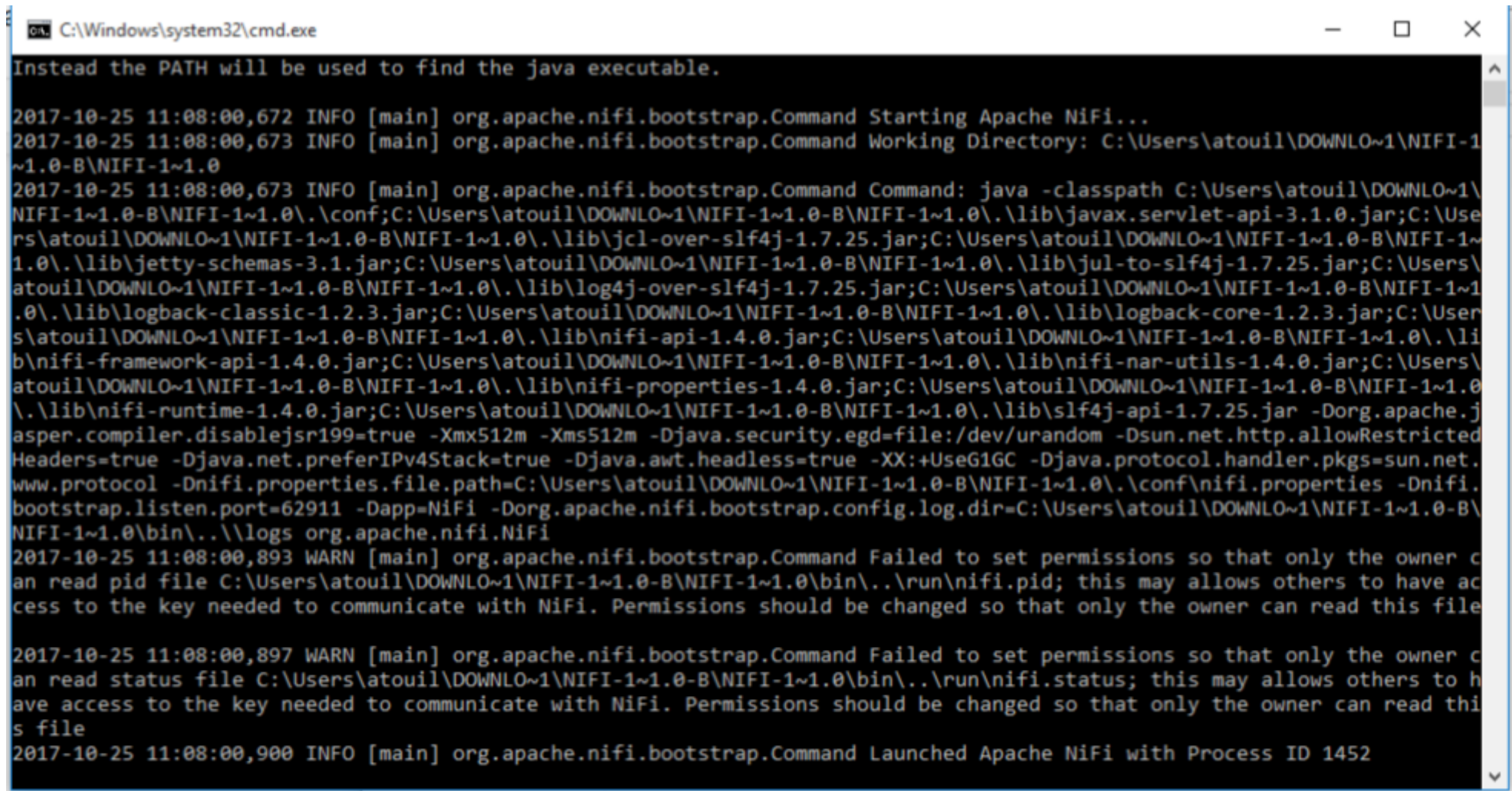
Exemple d'utilisation

Installation

Pour pouvoir utiliser et tester Apache NiFi, il suffit de **télécharger le fichier source ou binaire** sur le site Apache ou hortonworks (<https://nifi.apache.org/download.html>).

Lancez ensuite le fichier run-nifi.bat qui se trouve dans le répertoire nifi-1.4.0-bin\nifi-1.4.0\bin si vous êtes sur Windows, ou bien bin\nifi.sh run pour les utilisateurs linux/mac.

Cette fenêtre s'affiche :



```
C:\Windows\system32\cmd.exe

Instead the PATH will be used to find the java executable.

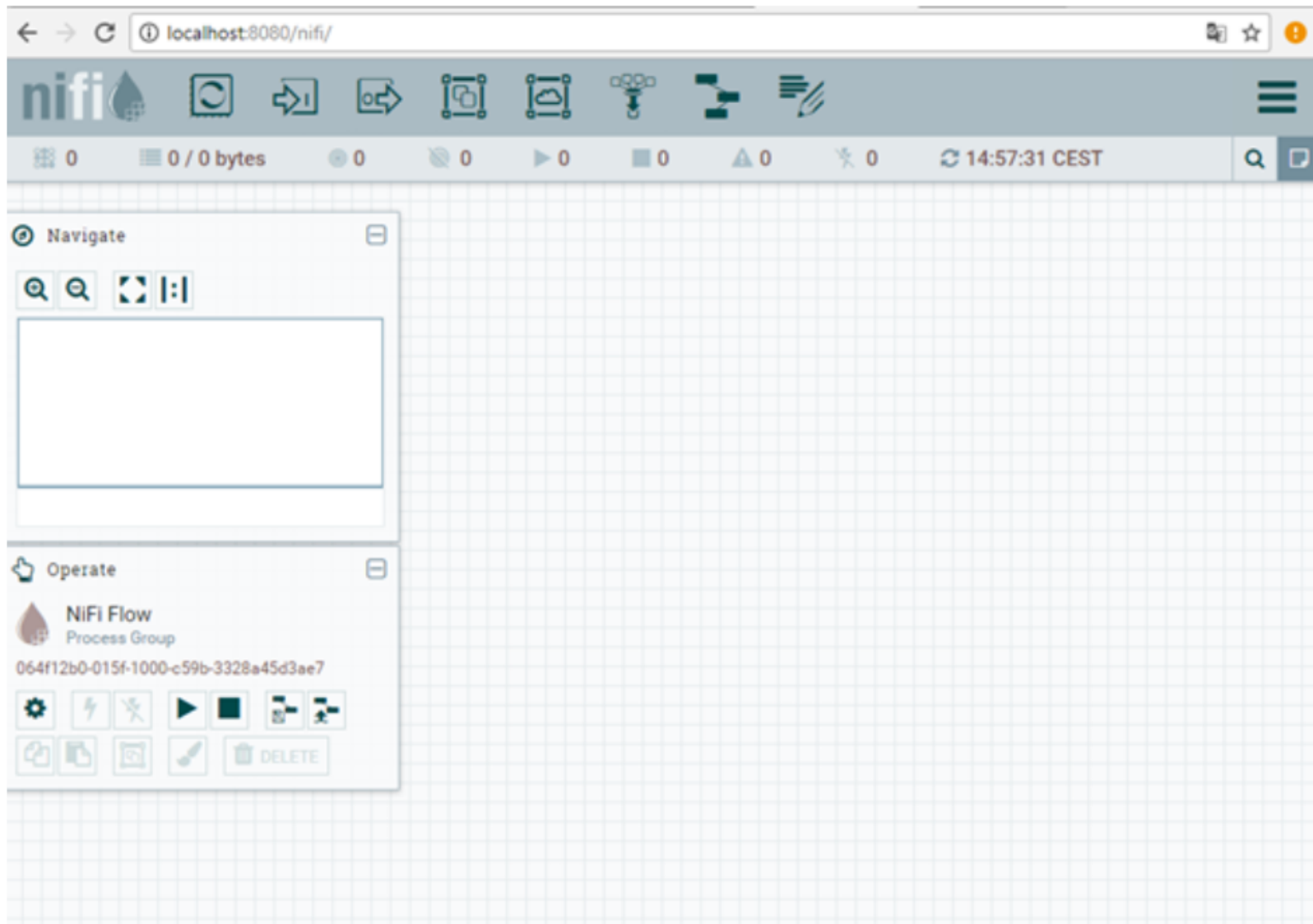
2017-10-25 11:08:00,672 INFO [main] org.apache.nifi.bootstrap.Command Starting Apache NiFi...
2017-10-25 11:08:00,673 INFO [main] org.apache.nifi.bootstrap.Command Working Directory: C:\Users\atouil\DOWNLO~1\NIFI-1~1.0-B\NIFI-1~1.0
2017-10-25 11:08:00,673 INFO [main] org.apache.nifi.bootstrap.Command Command: java -classpath C:\Users\atouil\DOWNLO~1\NIFI-1~1.0-B\NIFI-1~1.0\.\conf;C:\Users\atouil\DOWNLO~1\NIFI-1~1.0-B\NIFI-1~1.0\.\lib\javax.servlet-api-3.1.0.jar;C:\Users\atouil\DOWNLO~1\NIFI-1~1.0-B\NIFI-1~1.0\.\lib\jcl-over-slf4j-1.7.25.jar;C:\Users\atouil\DOWNLO~1\NIFI-1~1.0-B\NIFI-1~1.0\.\lib\jetty-schemas-3.1.jar;C:\Users\atouil\DOWNLO~1\NIFI-1~1.0-B\NIFI-1~1.0\.\lib\jul-to-slf4j-1.7.25.jar;C:\Users\atouil\DOWNLO~1\NIFI-1~1.0-B\NIFI-1~1.0\.\lib\log4j-over-slf4j-1.7.25.jar;C:\Users\atouil\DOWNLO~1\NIFI-1~1.0-B\NIFI-1~1.0\.\lib\logback-classic-1.2.3.jar;C:\Users\atouil\DOWNLO~1\NIFI-1~1.0-B\NIFI-1~1.0\.\lib\logback-core-1.2.3.jar;C:\Users\atouil\DOWNLO~1\NIFI-1~1.0-B\NIFI-1~1.0\.\lib\nifi-api-1.4.0.jar;C:\Users\atouil\DOWNLO~1\NIFI-1~1.0-B\NIFI-1~1.0\.\lib\nifi-framework-api-1.4.0.jar;C:\Users\atouil\DOWNLO~1\NIFI-1~1.0-B\NIFI-1~1.0\.\lib\nifi-nar-utils-1.4.0.jar;C:\Users\atouil\DOWNLO~1\NIFI-1~1.0-B\NIFI-1~1.0\.\lib\nifi-properties-1.4.0.jar;C:\Users\atouil\DOWNLO~1\NIFI-1~1.0-B\NIFI-1~1.0\.\lib\nifi-runtime-1.4.0.jar;C:\Users\atouil\DOWNLO~1\NIFI-1~1.0-B\NIFI-1~1.0\.\lib\slf4j-api-1.7.25.jar -Dorg.apache.jasper.compiler.disablejsr199=true -Xmx512m -Xms512m -Djava.security.egd=file:/dev/urandom -Dsun.net.http.allowRestrictedHeaders=true -Djava.net.preferIPv4Stack=true -Djava.awt.headless=true -XX:+UseG1GC -Djava.protocol.handler.pkgs=sun.net.www.protocol -Dnifi.properties.file.path=C:\Users\atouil\DOWNLO~1\NIFI-1~1.0-B\NIFI-1~1.0\.\conf\nifi.properties -Dnifi.bootstrap.listen.port=62911 -Dapp=NiFi -Dorg.apache.nifi.bootstrap.config.log.dir=C:\Users\atouil\DOWNLO~1\NIFI-1~1.0-B\NIFI-1~1.0\bin\..\logs org.apache.nifi.NiFi
2017-10-25 11:08:00,893 WARN [main] org.apache.nifi.bootstrap.Command Failed to set permissions so that only the owner can read pid file C:\Users\atouil\DOWNLO~1\NIFI-1~1.0-B\NIFI-1~1.0\bin\..\run\nifi.pid; this may allows others to have access to the key needed to communicate with NiFi. Permissions should be changed so that only the owner can read this file
2017-10-25 11:08:00,897 WARN [main] org.apache.nifi.bootstrap.Command Failed to set permissions so that only the owner can read status file C:\Users\atouil\DOWNLO~1\NIFI-1~1.0-B\NIFI-1~1.0\bin\..\run\nifi.status; this may allows others to have access to the key needed to communicate with NiFi. Permissions should be changed so that only the owner can read this file
2017-10-25 11:08:00,900 INFO [main] org.apache.nifi.bootstrap.Command Launched Apache NiFi with Process ID 1452
```

Démarrage de Apache NiFi

Source : CMD Windows

Puis, sur un votre navigateur Web, saisissez l'adresse : **localhost:8080**.

Vous obtenez alors cette fenêtre :



Fenêtre principale de Apache NiFi

Source : Apache NiFi

Apache NiFi est désormais **installé et lancé** sur votre poste de travail.

Pour vous permettre de prendre la main avec NiFi, je vais prendre l'exemple d'un **data flow qui récupère des logs générés par un serveur web HTTP**.

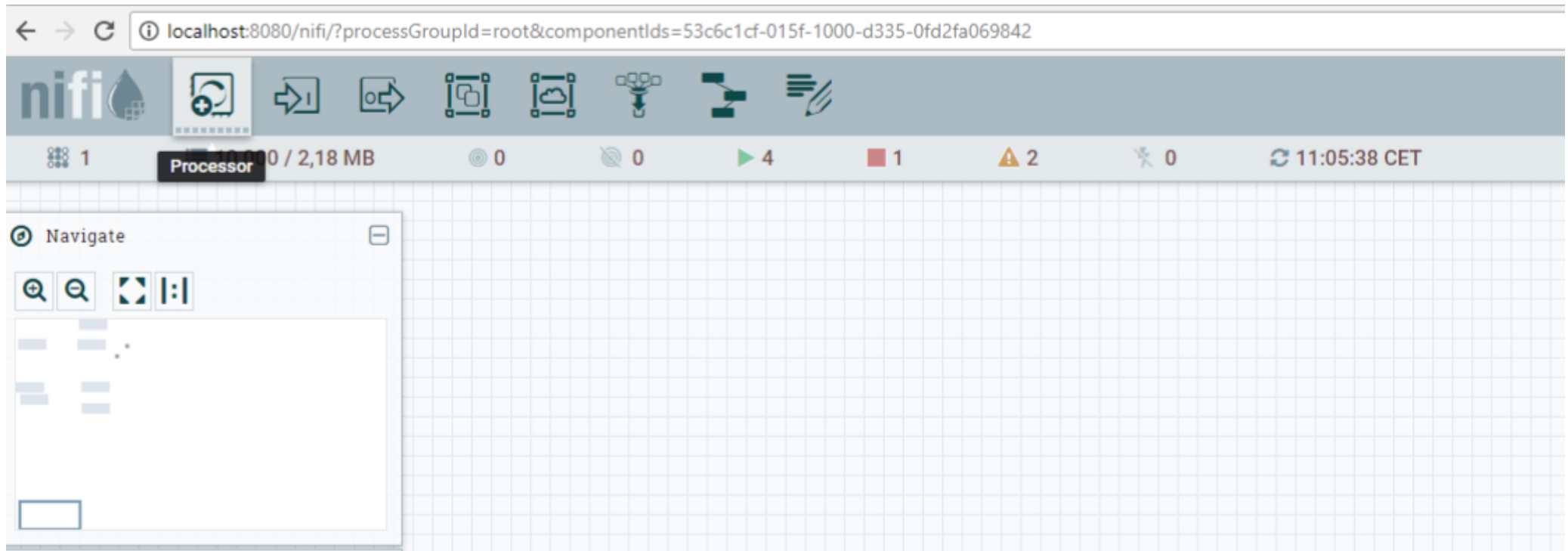
Nous allons utiliser le **processeur GetHTTP** et nous allons le **brancher sur l'API REST** du serveur Apache NiFi <http://localhost:8080/nifi-api/test-apache-nifi>.

Mise en place

On ajoute le processeur GetHTTP, pour **récupérer des données générées par un serveur HTTP** et les **écrire dans un FlowFile**.

On configure le processeur avec les paramètres qui nous intéressent, en l'**occurrence l'adresse http**.

Pour rajouter un processeur, cliquez sur le bouton **"Processor"**.



Ajout d'un processeur

Source : Apache NiFi

Puis, sélectionnez le processeur qui vous intéresse. Dans notre exemple, on rajoute le **processeur GetHTTP**.

Add Processor

Source

all groups ▼

amazon attributes
avro aws consume
csv database
delete fetch get
hadoop ingest
ingress insert json
listen logs
message pubsub
put record
restricted source
text update

Displaying 50 of 242

get

Type ▲	Version	Tags
GetFile	1.4.0	ingress, input, restricted, get, fi...
GetHBase	1.4.0	get, hbase, ingest
GetHDFS	1.4.0	restricted, get, fetch, HDFS, ha...
GetHDFSEvents	1.4.0	inotify, hadoop, events, notifica...
GetHDFSSequenceFile	1.4.0	get, fetch, sequence file, HDFS,...
GetHTML_Element	1.4.0	css, dom, get, html, element
GetHTTP	1.4.0	input, get, fetch, http, poll, http...
GetIgniteCache	1.4.0	cache, read, get, Ignite, key
GetJMSQueue	1.4.0	jms, pull, get, consume, source...
GetJMSTopic	1.4.0	jms, durable, pull, get, non-dura...
GetKafka	1.4.0	PubSub, Ingest, Get, Kafka, Ing...
GetMonno	1.4.0	read net monnoh

ConsumeAMQP 1.4.0 org.apache.nifi - nifi-amqp-nar

Consumes AMQP Message transforming its content to a FlowFile and transitioning it to 'success' relationship

CANCEL ADD

Fenêtre de recherche des processeurs

Source : Apache NiFi

Processeur log attribute

Le 2e processeur est LogAttribute qui affiche les logs générés par le serveur HTTP.

Configure Processor

SETTINGS SCHEDULING **PROPERTIES** COMMENTS

Required field +

Property	
URL	1://localhost:8080/nifi-api/test-apache-nifi
Filename	
SSL Context Service	
Username	
Password	
Connection Timeout	
Data Timeout	
User Agent	No value set
Accept Content-Type	No value set
Follow Redirects	false
Redirect Cookie Policy	default
Proxy Host	No value set
Proxy Port	No value set

☐ Set empty string

CANCEL OK

CANCEL APPLY

Configuration du processeur 1

Source : Apache NiFi

Il faut cocher la case “**Success**”. Le paramètre “**Penalty Duration**” sert à **contrôler le temps accordé à ce processeur** quand il pénalise le flux. S’il dépasse 30 secondes, relancez-le. “**Yield Duration**” permet pour sa part de **programmer à nouveau le traitement après la durée mentionnée**.

Configure Processor

SETTINGS | SCHEDULING | PROPERTIES | COMMENTS

Name: LogAttribute ☒ Enabled ☐ success **Automatically Terminate Relationships** ⓘ
All FlowFiles are routed to this relationship

Id: 53ca7536-015f-1000-c0d1-a52d937c583d

Type: LogAttribute 1.4.0

Bundle: org.apache.nifi - nifi-standard-nar

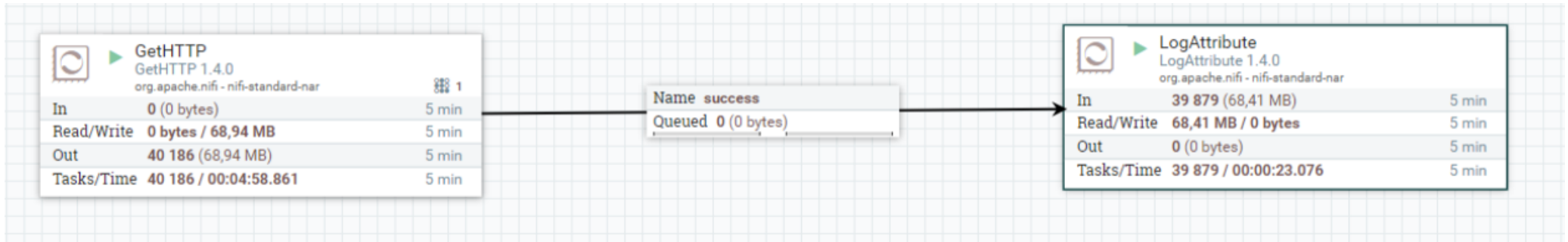
Penalty Duration ⓘ: 30 sec Yield Duration ⓘ: 1 sec

Bulletin Level ⓘ: WARN ▼

CANCEL APPLY

Configuration du processeur 2
Source : Apache NiFi

Voici ce à quoi ressemble alors le **flux de données**, assez simple :

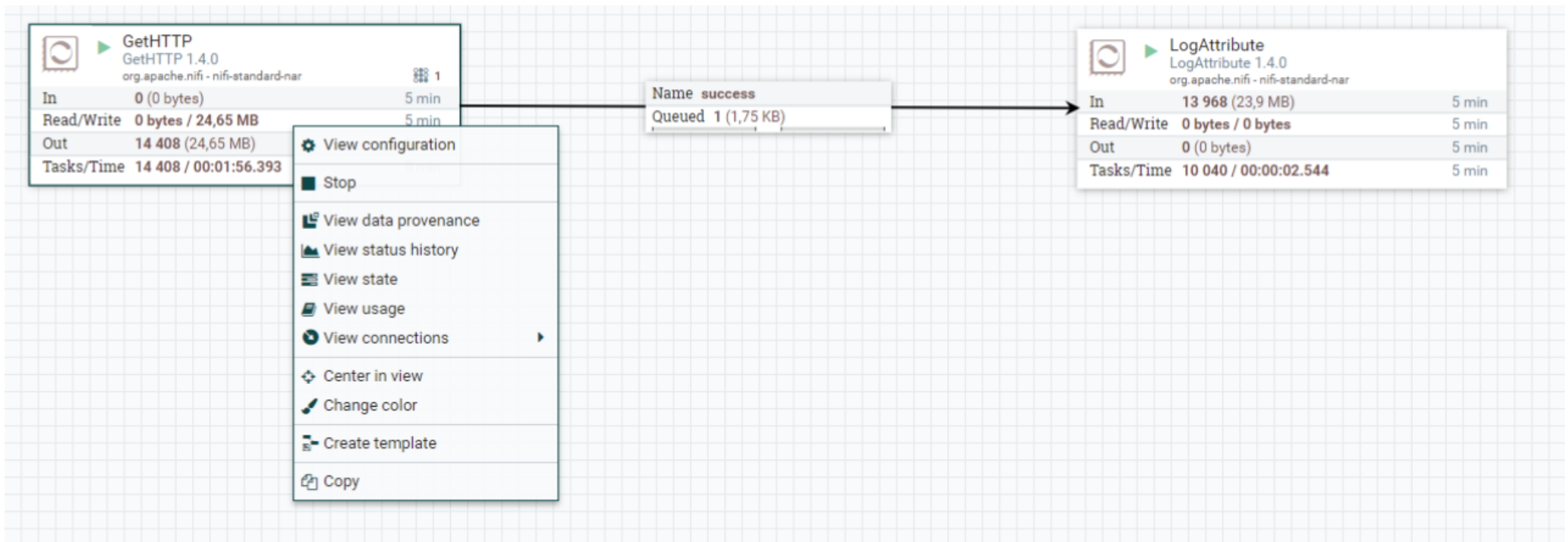


Un flux de données sur Apache NiFi

Source : Apache NiFi

Exécution

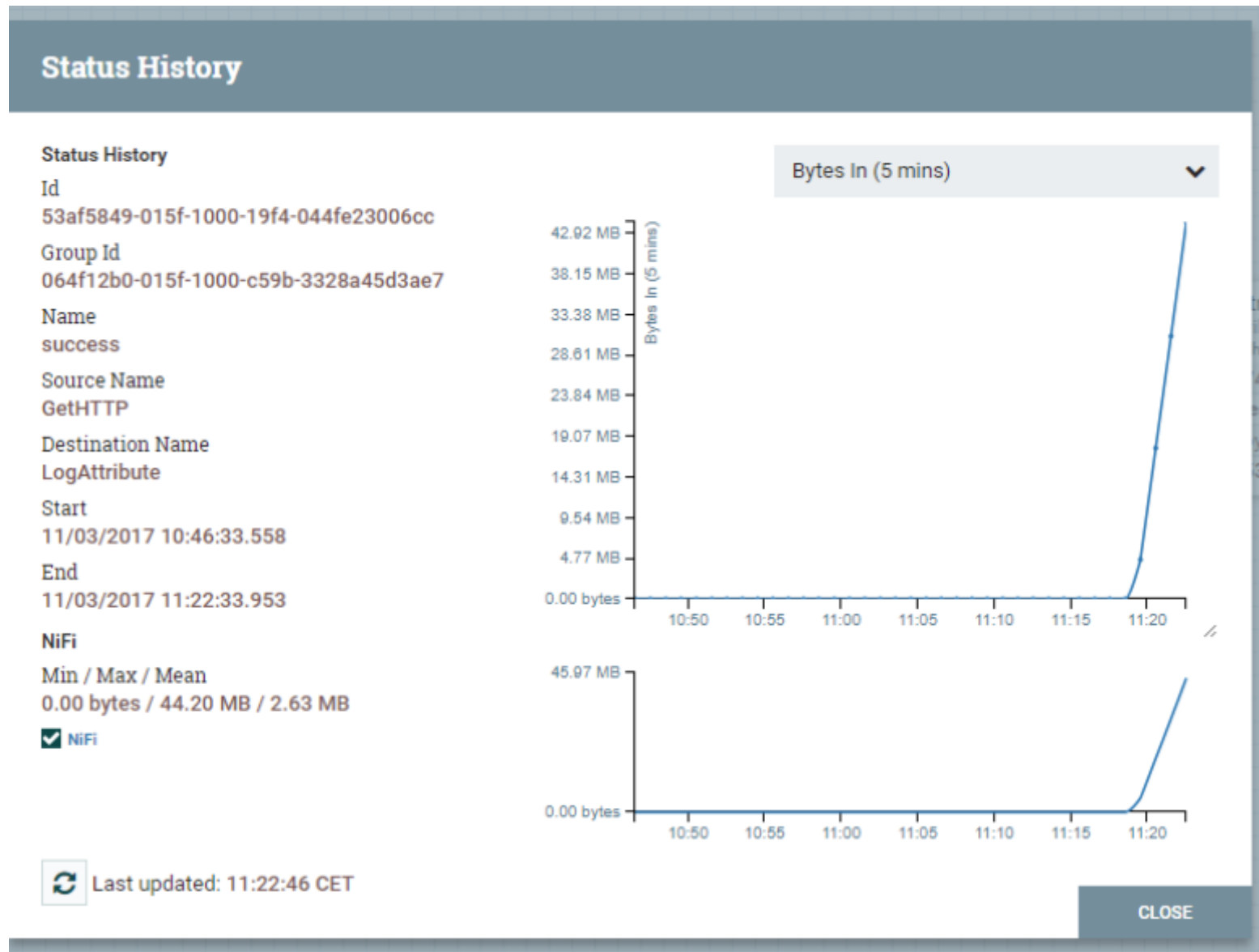
Lancez alors le flux pour voir le résultat en effectuant un clic droit sur un processeur. Choisissez **View data provenance** ou **View state** pour obtenir les informations sur l'état du processeur.



Détails du flux traité par le processeur

Source : Apache NiFi

Une autre fenêtre affiche alors les **statistiques du flux de données**.



Détails du statut du processeur

Source : Apache NiFi

Vous obtenez les données récupérées du serveur Web afin **d'effectuer une analyse**.

NiFi Data Provenance

Displaying 1,000 of 1,000

Oldest event available: 11/03/2017 11:19:09 CET

Showing the most recent 1,000 of 1 000+ events, please refine t

Date/Time	Type	FlowFile Uuid	Size	Component Name	Component Type
11/03/2017 11:34:16.096 CET	DROP				LogAttribute
11/03/2017 11:34:16.091 CET	RECEIVE				GetHTTP
11/03/2017 11:34:16.086 CET	DROP				LogAttribute
11/03/2017 11:34:16.086 CET	DROP				LogAttribute
11/03/2017 11:34:16.084 CET	RECEIVE				GetHTTP
11/03/2017 11:34:16.077 CET	RECEIVE				GetHTTP
11/03/2017 11:34:16.075 CET	DROP				LogAttribute
11/03/2017 11:34:16.070 CET	RECEIVE				GetHTTP
11/03/2017 11:34:16.064 CET	DROP				LogAttribute
11/03/2017 11:34:16.064 CET	DROP				LogAttribute
11/03/2017 11:34:16.063 CET	RECEIVE				GetHTTP
11/03/2017 11:34:16.057 CET	RECEIVE				GetHTTP
11/03/2017 11:34:16.052 CET	DROP				LogAttribute
11/03/2017 11:34:16.052 CET	DROP				LogAttribute
11/03/2017 11:34:16.050 CET	RECEIVE				GetHTTP
11/03/2017 11:34:16.043 CET	RECEIVE				GetHTTP
11/03/2017 11:34:16.041 CET	DROP				LogAttribute
11/03/2017 11:34:16.036 CET	RECEIVE				GetHTTP
11/03/2017 11:34:16.029 CET	DROP				LogAttribute
11/03/2017 11:34:16.029 CET	DROP				LogAttribute
11/03/2017 11:34:16.029 CET	RECEIVE				GetHTTP
11/03/2017 11:34:16.022 CET	RECEIVE				GetHTTP
11/03/2017 11:34:16.019 CET	DROP				LogAttribute
11/03/2017 11:34:16.019 CET	DROP	95c3fd62-8d43-49ac-bc2a-e76e36cecd1	1,75 KB	LogAttribute	LogAttribute
11/03/2017 11:34:16.016 CET	RECEIVE	c9673309-bc3b-4b4b-b698-3a983c2e4a51	1,75 KB	GetHTTP	GetHTTP
11/03/2017 11:34:16.009 CET	RECEIVE	95c3fd62-8d43-49ac-bc2a-e76e36cecd1	1,75 KB	GetHTTP	GetHTTP

Information sur un flux récupéré par Apache NiFi

Source : Apache NiFi

Une autre fenêtre s'ouvre et **détaille les informations** récupérées depuis le serveur web.



Un Json récupéré depuis le serveur Web par Apache NiFi
Source : Apache NiFi

Ma conclusion sur Apache NiFi

On peut facilement imaginer l'utilité d'un **Data flow** qui récupère des logs d'un serveur (Web, BDD ou autres) afin d'analyser le comportement ou de surveiller l'état des machines.

Pour construire un Data flow plus complexe, il sera intéressant d'utiliser le **processeur InvokeHttp** qui peut être configuré dynamiquement.

Apache NiFi facilite la **construction d'un pipeline de données**. Avec les différents processeurs de sa bibliothèque, on peut facilement **récupérer les données** d'un broker Kafka, de serveurs Web, de Twitter, etc., **puis les injecter dans différents systèmes** en sortie comme par exemple Elastic, S3 ou HDFS.

De plus, l'interface graphique interactive de NiFi permet de créer rapidement des flux de données.

Cet outil relativement peu connu dans le **monde du Big data** présente selon moi une très **grande efficacité et facilite certaines tâches dans la construction d'un pipeline de données complexes**.