Flink

Réalisé Par :

-Chaymae Ben el filali - Ghali Oumaima

-Khalil Chaimaa - Darhouane Assia

-Lahmama Fatima-Zahraa -Siham Hafsi

Encadré Par :

Pr. Soussi Nassima

2023-2024

Plan

C'est une technologie qui a révolutionné la manière dont nous traitons et analysons les données en continu. Dans un monde où l'information circule à la vitesse de la lumière, il est essentiel d'avoir des outils capables de suivre le rythme effréné des données. C'est là qu'entre en jeu Apache Flink. Cette plateforme de traitement de données en temps réel est devenue un acteur clé dans l'ère du Big Data, offrant des capacités de traitement, d'analyse et de calcul distribué sans précédent.

I POURQUOI APACHE FLINK?

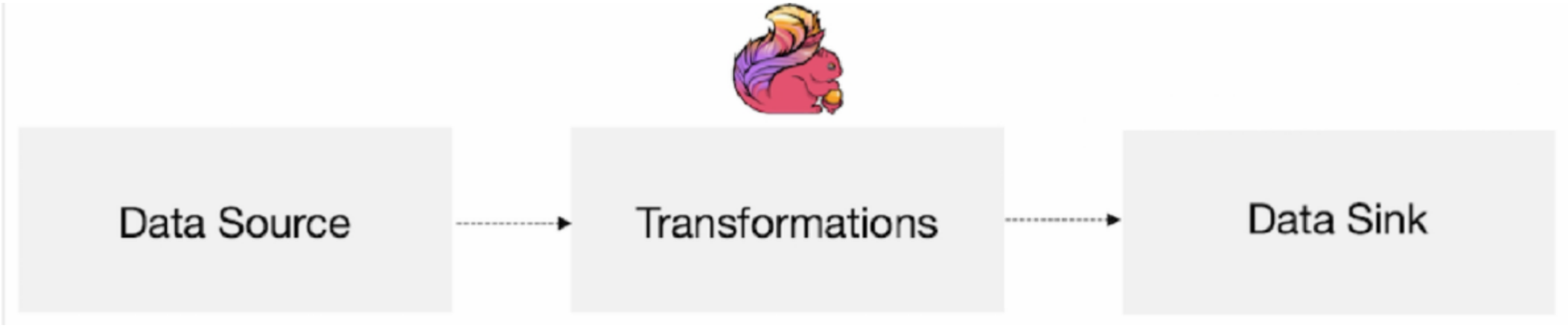
Haute performance : Flink est reconnu pour ses performances élevées et sa capacité à gérer de grandes quantités de données. Il exploite le parallélisme et la distribution pour traiter efficacement les charges de travail Big Data.

Flexibilité : Flink prend en charge à la fois le traitement en batch et en streaming, ce qui signifie qu'il peut être utilisé pour une variété de cas d'utilisation, de l'analyse de données en temps réel à la transformation de données par lots.

Écosystème : Il s'intègre bien avec d'autres technologies Big Data, ce qui en fait un choix judicieux pour des entreprises qui utilisent déjà des composants tels que Hadoop, Kafka, et d'autres systèmes.

Tout commence par le développeur qui écrit un programme Flink en utilisant l'API Flink. Ce programme peut être écrit en Java, Scala ou Python et est conçu pour traiter des flux de données ou des ensembles de données en batch.

Les programmes Flink se composent de trois parties : Source, Transformation et Sink.
La source est responsable de la lecture des données à partir de sources de données telles que HDFS, Kafka et du texte.
Transformation est responsable des opérations de transformation des données.
Sink est responsable des sorties de données finales (telles que HDFS, Kafka et texte).



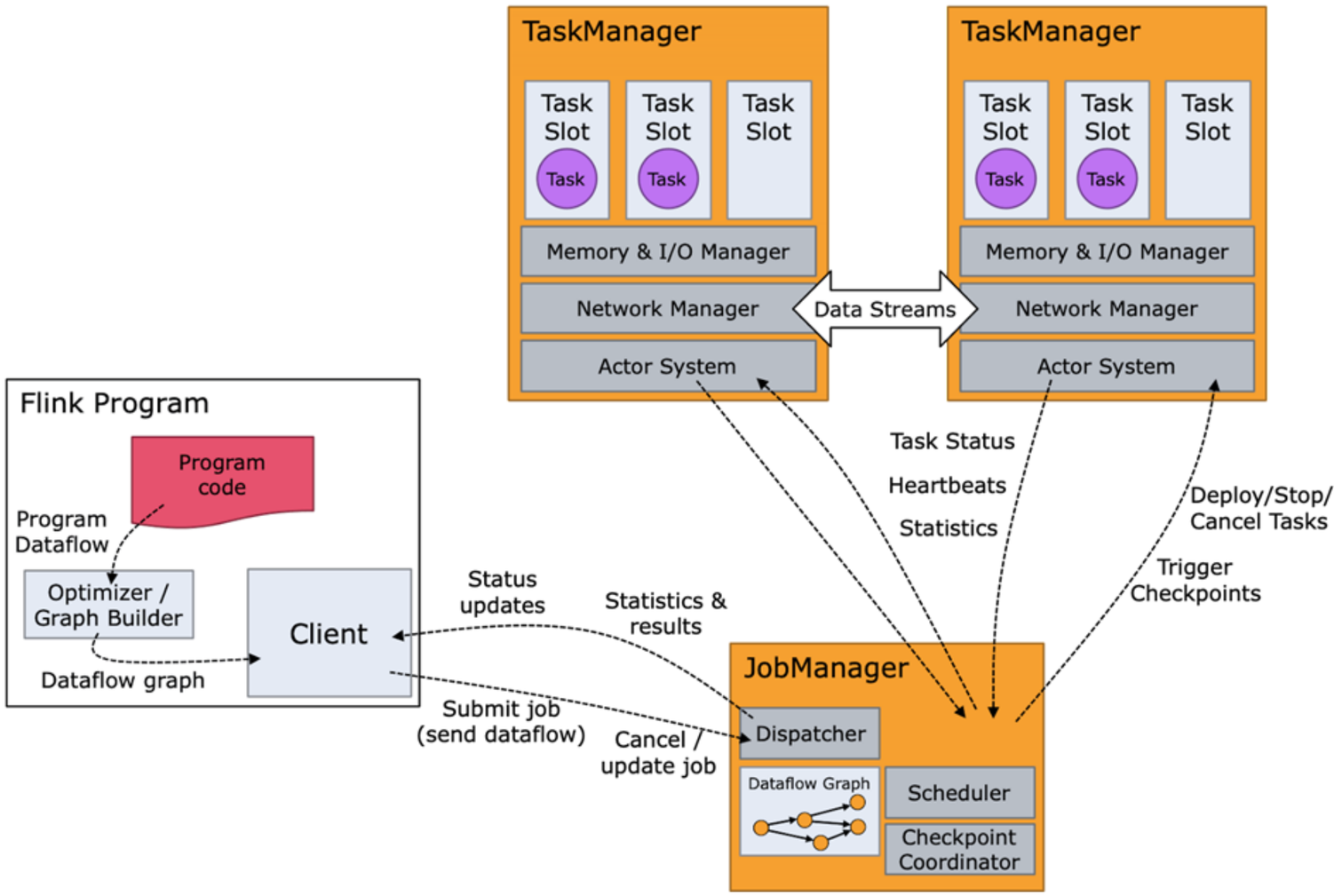
Transformation	Description
Map	Applique une transformation définie par l'utilisateur à chaque élément du flux et produit exactement un élément.
FlatMap	Prend un élément du flux et produit zéro, un ou plusieurs éléments.
Filter	Filtre les éléments d'un flux en fonction d'une ou plusieurs conditions.
KeyBy	Attribue une clé aux éléments d'un flux, ce qui revient à la partitionner en fragments partageant la même clé. Le flux sortant est de type *KeyedDataStream*.
Reduce	Applique une réduction sur un *KeyedDataStream* en combinant le nouvel élément du flux au dernier résultat de la réduction.
Fold	Applique une réduction sur un *KeyedDataStream* en combinant le nouvel élément du flux à un accumulateur dont la valeur initiale est fournie.
Aggregations	Fonctions prédéfinies `sum`, `max`, `min`, `maxBy` et `minBy`, pour agréger les valeurs numériques des éléments d'un *KeyedDataStream*.
Union	Union de deux flux ou plus, incluant tous les champs de tous les flux (ne supprime pas les doublons).
Connect	Connecte deux flux indépendamment de leur type, permettant d'avoir un état partagé.
Split	Partitionne le flux selon des critères.
Select	Sélectionne un ou plusieurs éléments d'un flux partitionné.
Extract Timestamps	Extrait l'horodatage d'un *stream*.

Sink	Description
writeAsText	Transforme les éléments en texte en utilisant la méthode *toString()*
writeAsCsv	Ecrit les éléments dans un fichier csv. Les délimiteurs de champs et de ligne sont paramétrables.
print / printToErr	Imprime les éléments sur la console.
writeUsingOutputFormat / FileOutputFormat	Permet d'écrire avec un format défini par l'utilisateur.
addSink	Permet d'ajouter un connecteur défini par l'utilisateur en dehors des fonctions *sink* existantes.

-Flink a son optimiseur qui optimise l'application du point de vue de l'efficacité de l'exécution.

-L'application Flink sera convertie en « dataflow graph» : c'est un graphe d'opérateurs défini par le développeur, où chaque opérateur représente une étape de traitement (map, filter, reduce, etc.).et sera soumise au cluster Flink pour exécution.

-Le cluster est composé de deux types de composants principaux : le Job Manager et le Task Manager.



Job manager : (maître)

planifie les tâches, les distribue aux TaskManagers, suit l'avancement de l'exécution, alloue les ressources et compile les résultats.

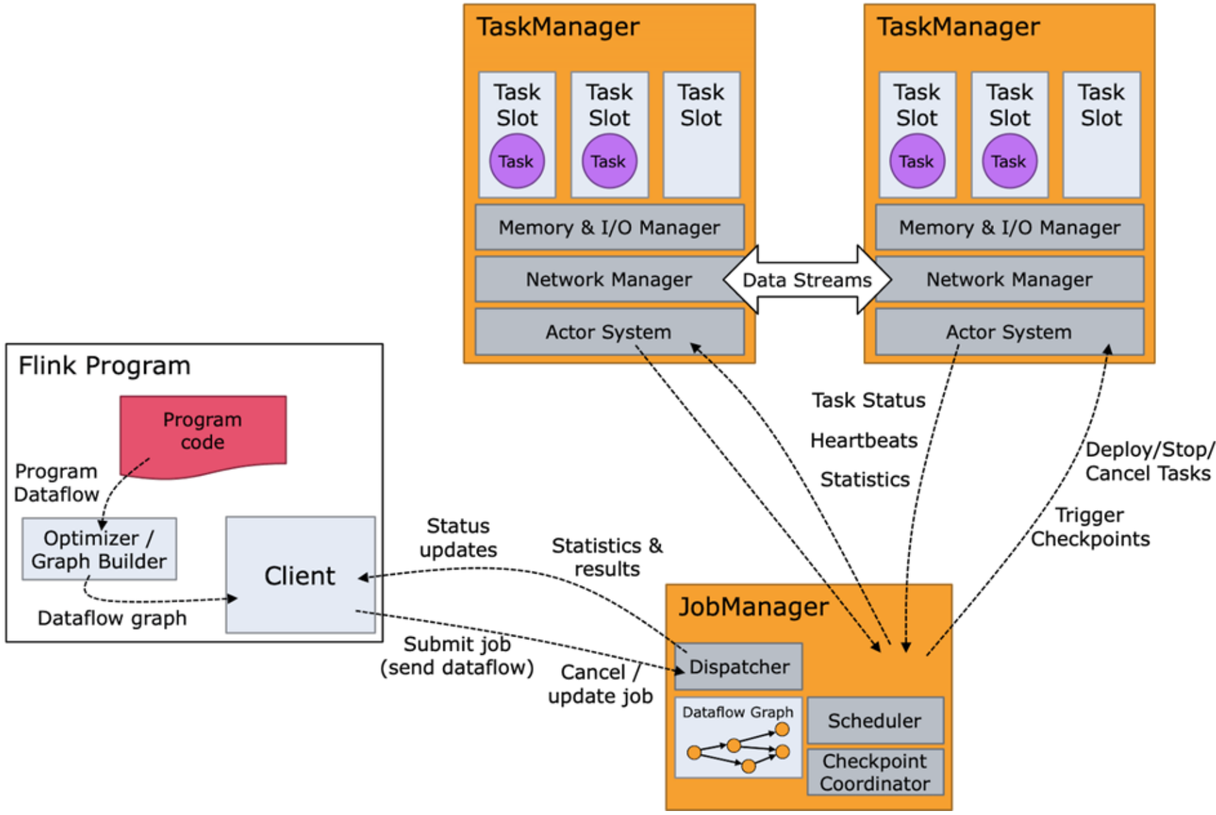
il se compose de quatre composantes différentes :

Dispatcher : Fournir une interface pour soumettre un graphique de tâches et démarrer un nouveau job master pour chaque tâche soumise.

Job master : responsable de la gestion de l'exécution d'un seul graphe de tâche.

Resource manager / Task scheduler : permet de gérer task managers et d'affecter les tâches aux task managers pour exécution.

Checkpoint coordinator : permet d'activer la tolérance aux pannes.



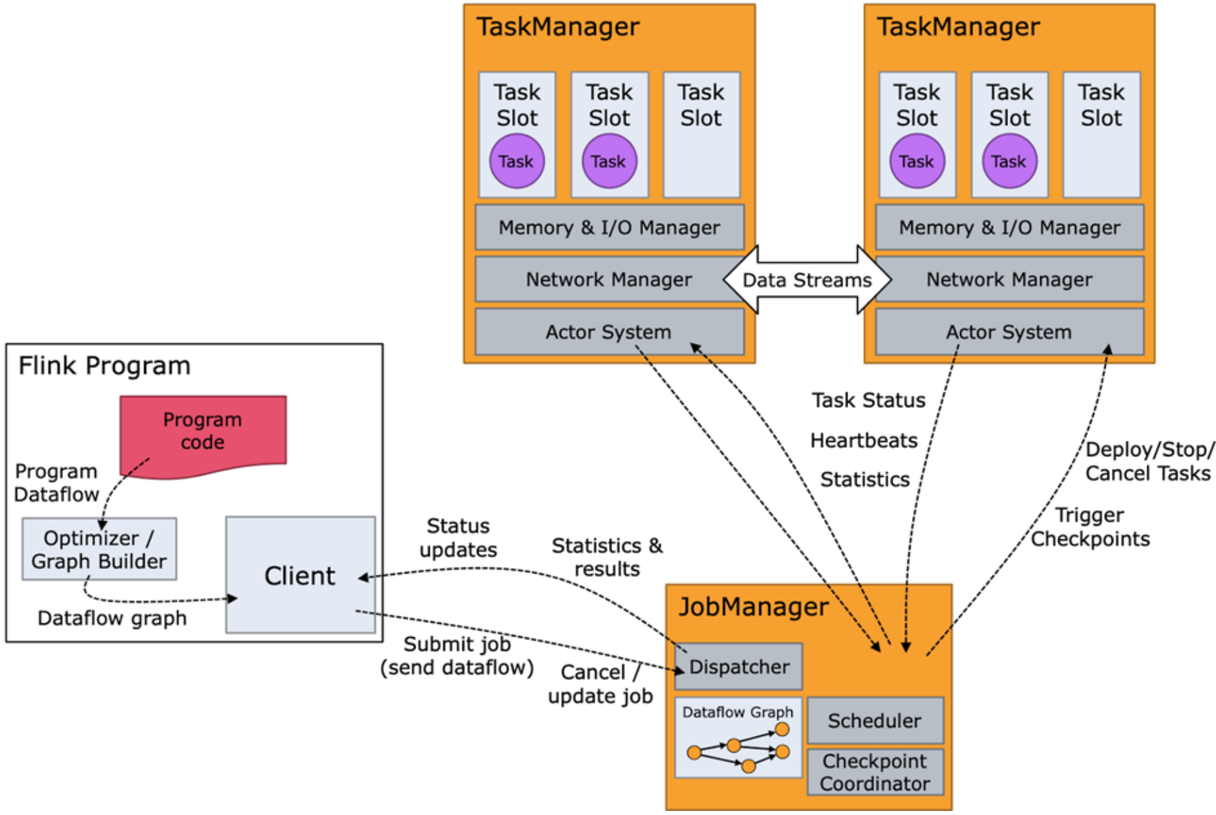
task manager : (esclave)

-Chaque Task Manager est responsable de l'exécution réelle des tâches. Un cluster Flink peut avoir plusieurs Task Managers répartis sur différentes machines.

-Un TaskManager peut avoir un ou plusieurs Task slots.Le Task Slot représente un espace mémoire isolé dédié à l'exécution d'un ou plusieurs fils de traitements (Threads)

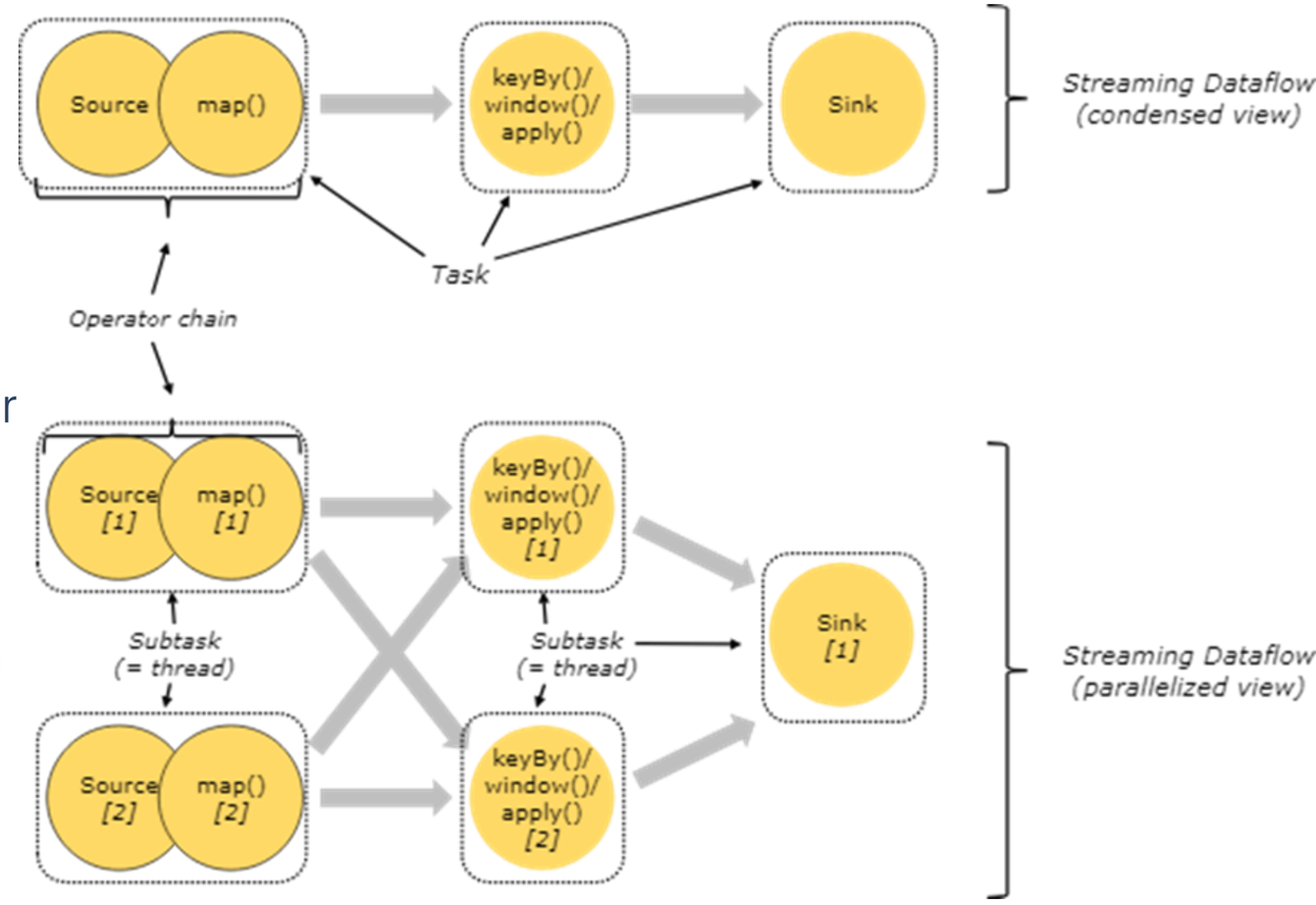
-Chaque TaskManager envoie un message (heartbeats) à intervalle régulier et en reçoit du JobManager pour signaler qu'il n'est pas en panne

-Si des données doivent être échangées entre les tâches (par exemple, lors de regroupements ou de jointures), les TaskManagers gèrent efficacement cet échange en utilisant des canaux de communication.



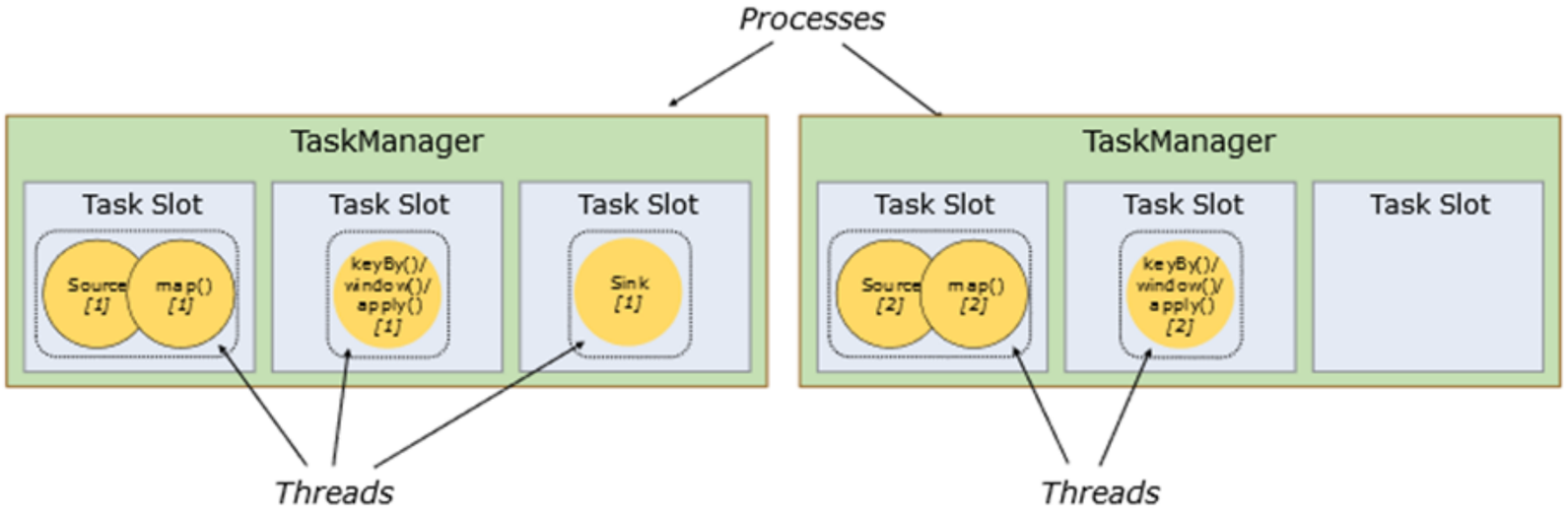
-Les Task Managers commencent à exécuter les tâches qui leur sont attribuées. Cela peut inclure la lecture de données, l'application des opérateurs, la gestion de l'état, etc. Les tâches sont exécutées en parallèle sur plusieurs Task Managers pour accélérer le traitement.

-Operator Chains (Chaînes d'opérateurs) : Les Operator Chains sont des pipelines d'opérateurs qui sont exécutés ensemble dans une seule tâche pour minimiser les coûts de communication.



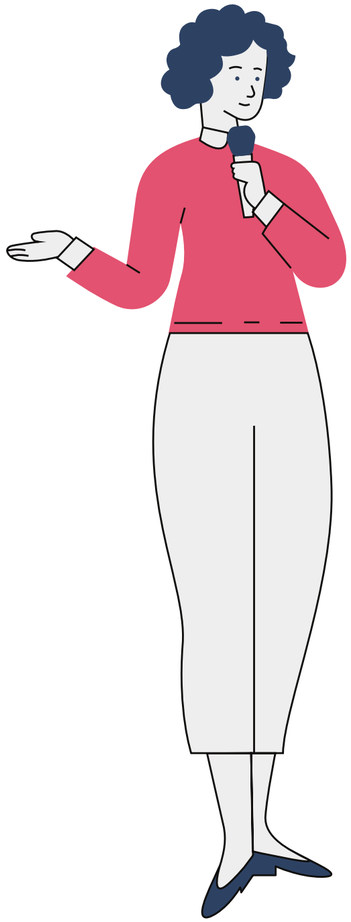
-Une sous-tâche est une subdivision d'une tâche. Elle permet de paralléliser davantage le traitement à l'intérieur d'une tâche.

Par exemple, si une tâche de map est configurée pour fonctionner avec un parallélisme de 2, elle sera divisée en 2 sous-tâches distinctes, chacune s'occupant d'une partie des données d'entrée.



=>Une fois que toutes les tâches sont terminées, le Job Manager collecte les résultats et les renvoie au client.

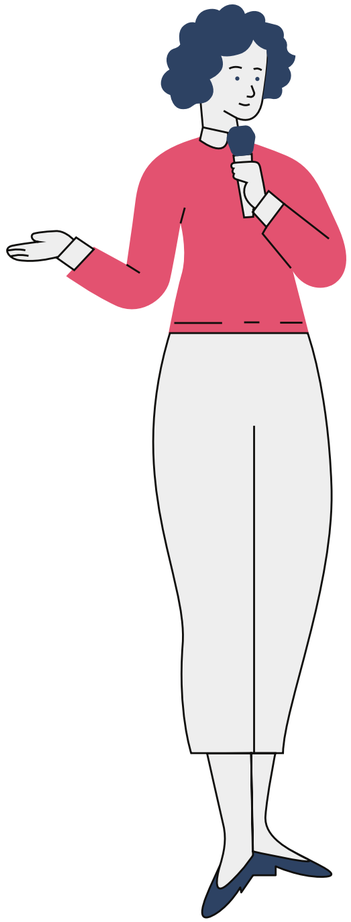
la tolérance aux pannes et la haute disponibilité.



Snapshotting : Flink prend des "snapshots" périodiques de l'état de l'application en cours d'exécution. Un snapshot est une capture de l'état de l'ensemble de l'application, y compris les données intermédiaires, les métadonnées de configuration, et les informations sur la progression. Ces snapshots sont stockés dans un système de stockage distribué comme HDFS ou Amazon S3.

Reprise après panne : En cas de panne d'un composant (Task Manager, Job Manager, etc.), Flink peut récupérer à partir du dernier snapshot connu. Les composants défaillants sont redémarrés, et l'état précédemment capturé est restauré. Cela permet de reprendre le traitement là où il s'était arrêté, minimisant ainsi la perte de données et de temps de traitement.

la tolérance aux pannes et la haute disponibilité.



Durabilité des résultats : Les résultats intermédiaires et finaux du traitement sont stockés de manière durable, ce qui signifie qu'ils sont écrits dans un système de stockage distribué. Cela garantit que les résultats ne sont pas perdus en cas de panne.

Rollback et Retransmission : En cas de panne d'une tâche, Flink est capable de revenir en arrière et de relancer des tâches à partir du dernier snapshot pour s'assurer que le traitement est cohérent et fiable.

Haute disponibilité : Flink offre également des options pour une haute disponibilité, ce qui signifie que vous pouvez configurer le système pour qu'il continue de fonctionner même en cas de panne du Job Manager principal en ayant un Job Manager de secours prêt à prendre le relais.

III. CAS D'UTILISATIONS DE FLINK

CAS D'UTILISATIONS

BATCH PROCESSING

Analyse Historique:
Traitement d'énormes ensembles de données historiques pour obtenir des insights, par exemple, l'analyse des ventes annuelles d'une entreprise.

MICRO-BATCH PROCESSING

Tableau de Bord en Temps Réel:
Actualisation fréquente d'un tableau de bord en temps réel avec des données agrégées provenant de micro-batches, idéal pour la surveillance continue de l'activité d'un site web.

STREAM PROCESSING

Détection d'Anomalies en Temps Réel:
Identification immédiate de comportements anormaux dans un flux de données, comme la détection en temps réel d'attaques informatiques.

III Cas d'utilisations de Flink

DANS LE DOMAINE DE L'INFORMATIQUE ET DE LA TECHNOLOGIE, APACHE FLINK EST ÉGALEMENT LARGEMENT UTILISÉ POUR PLUSIEURS CAS D'UTILISATION EN TEMPS RÉEL. VOICI QUELQUES EXEMPLES SUPPLÉMENTAIRES DE CAS D'UTILISATION DE FLINK DANS LE DOMAINE DE L'IT ET DE LA TECHNOLOGIE :

1. IOT (INTERNET DES OBJETS) :
   - FLINK EST ADAPTÉ POUR LE TRAITEMENT EN TEMPS RÉEL DES DONNÉES GÉNÉRÉES PAR LES DISPOSITIFS IOT. IL PEUT INGÉRER DES DONNÉES DE CAPTEURS EN CONTINU, LES ANALYSER POUR DÉTECTER DES MODÈLES OU DES ÉVÉNEMENTS SPÉCIFIQUES, ET DÉCLENCHER DES ACTIONS EN TEMPS RÉEL, COMME DES ALERTES EN CAS DE DÉFAILLANCE D'UN CAPTEUR.

2. GESTION DES LOGS :
   - FLINK PEUT ÊTRE UTILISÉ POUR L'AGRÉGATION, L'ANALYSE ET LA CORRÉLATION EN TEMPS RÉEL DES JOURNAUX D'APPLICATION, DES JOURNAUX DE SERVEUR ET DES JOURNAUX DE SÉCURITÉ, CE QUI FACILITE LE DÉBOGAGE ET LA SURVEILLANCE DES SYSTÈMES.

3. INTELLIGENCE ARTIFICIELLE ET APPRENTISSAGE AUTOMATIQUE :
   - FLINK PEUT ÊTRE UTILISÉ DANS LE TRAITEMENT DE FLUX DE DONNÉES POUR DES APPLICATIONS D'APPRENTISSAGE AUTOMATIQUE EN TEMPS RÉEL, TELLES QUE LA DÉTECTION D'ANOMALIES, LA PRÉDICTION DE LA DEMANDE ET LA PERSONNALISATION.

SPARK STREAMING VS FLINK

VS

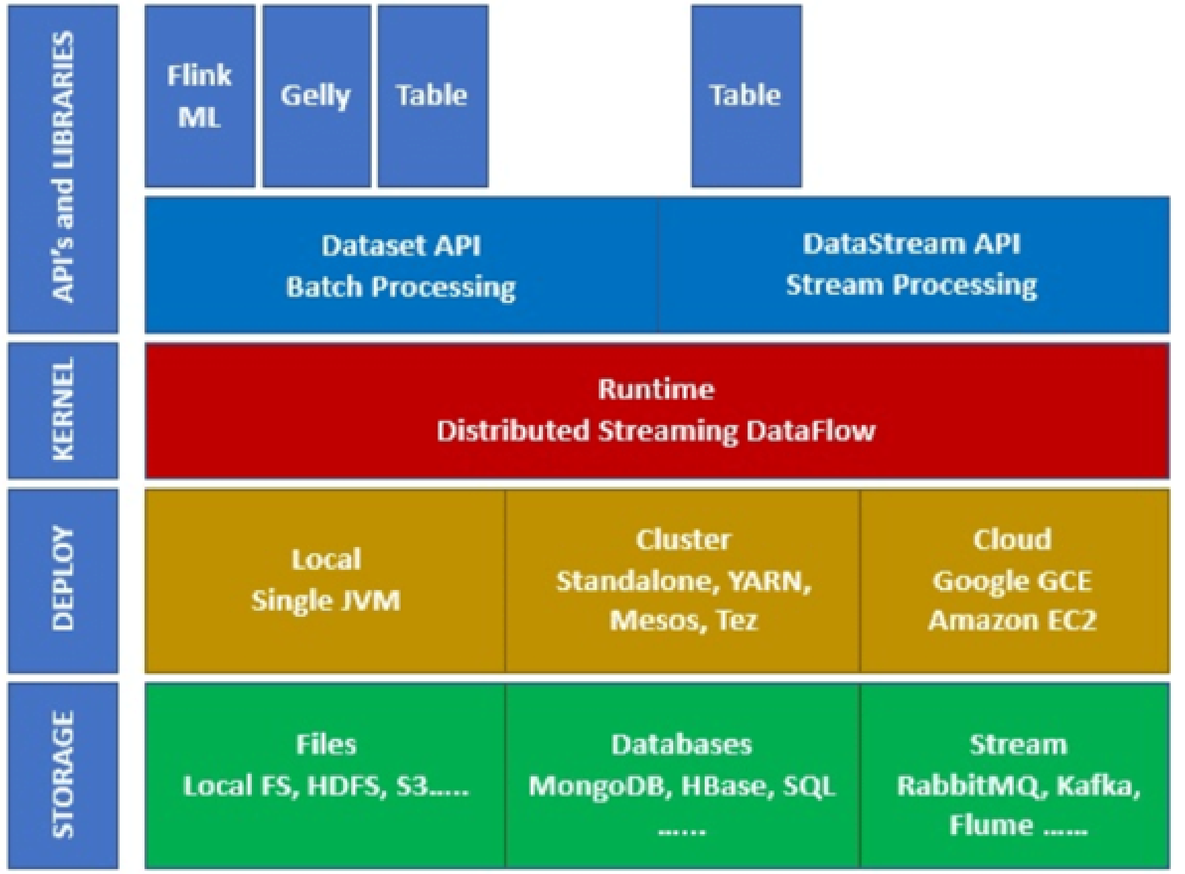
Spark Streaming

	FLINK	SPARK STREAMING
LATENCE	Une latence très faible, ce qui en fait un choix solide pour les applications nécessitant une réactivité en temps réel	Une latence légèrement plus élevée par rapport à Flink
MODÈLE DE TRAITEMENT	Traite les données de manière continue, que ce soit en mode micro-batching ou en traitement de flux pur	Basé sur le modèle de micro-batching, traitant les données par petits lots à intervalles réguliers
COMPLEXITÉ DES OPÉRATIONS	Offre une API unifiée pour le traitement de flux et de batch, simplifiant le développement	Peut nécessiter une transition entre les API de batch et de streaming, ce qui peut introduire une complexité supplémentaire.
ÉCOSYSTÈME	En croissance, avec un écosystème solide, mais parfois moins mature que celui de Spark	Appartient à l'écosystème Spark bien établi, avec une large adoption et de nombreuses bibliothèques.
FACILITÉ D'UTILISATION	Une API unifiée rend le développement plus fluide, mais peut nécessiter une courbe d'apprentissage.	Familiarité avec l'écosystème Spark, ce qui peut être un avantage pour ceux déjà habitués à Spark

- A. Flink et les outils connexes : Table API, Flink SQL, FlinkML, etc.
- B. Intégration avec d'autres technologies (Kafka, Hadoop).

L'écosystème de Apache Flink est composé de plusieurs couches et niveaux d'abstraction, comme illustré dans la figure suivante :

- **Storage (stockage)** : Flink dispose de plusieurs possibilités pour lire / écrire les données, telles que HDFS (Hadoop), S3, en local, Kafka, et d'autres
- **Deploy (déploiement) :** Flink peut se déployer en mode local, sur des clusters ou dans le cloud
- **Kernel (noyau) :** Il s'agit ici de la couche d'exécution, qui donne la tolérance aux pannes, les différents calculs distribués, etc.
- **API's & Libraries (API et bibliothèques) :** Il s'agit de la couche de plus haut niveau de l'écosystème Flink. On retrouve l'API Datastream, en charge du traitement des flux, l'API Dataset, en charge du traitement par lot (batch processing), et d'autres bibliothèques comme Flink ML (Machine Learning), Gelly (graph processing) et Table (pour le SQL).



Intégration avec d'autres technologies :

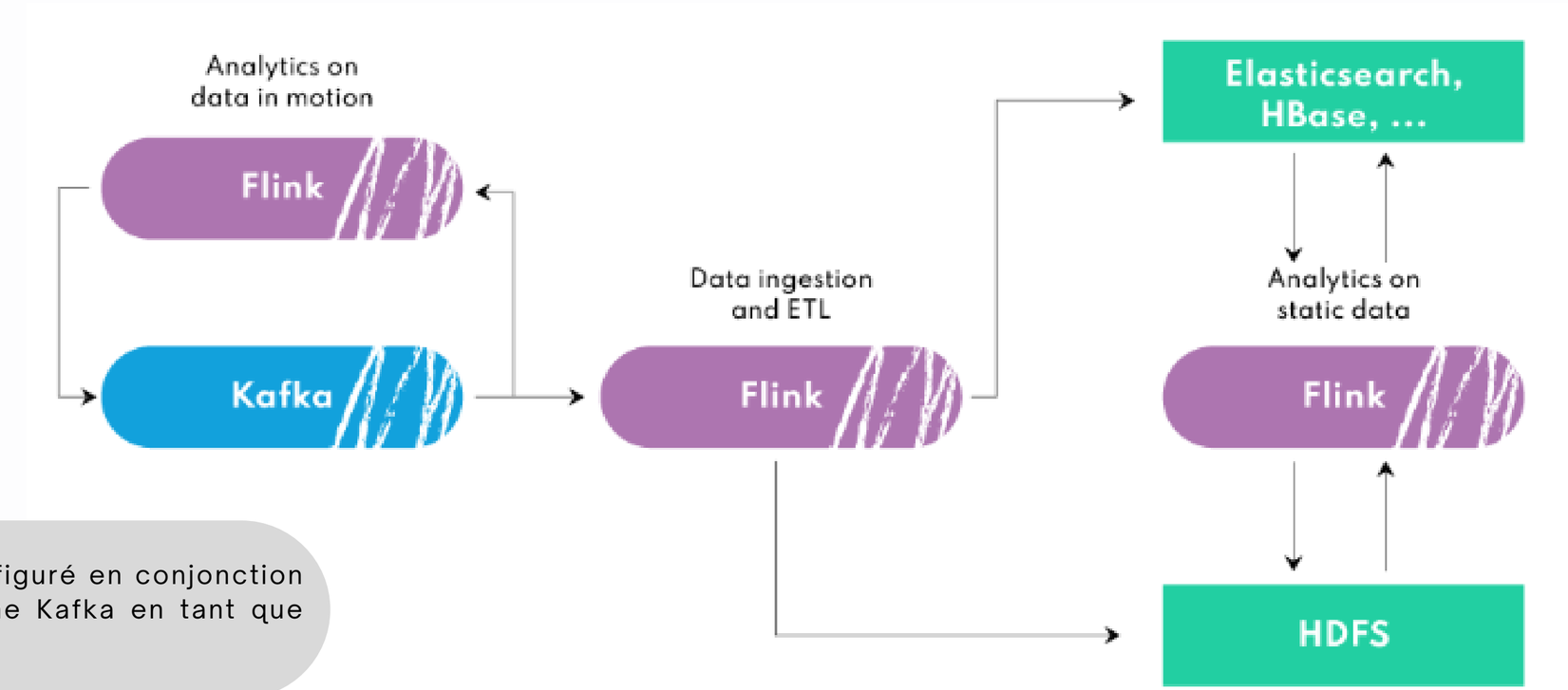
Flink est conçu pour être flexible et compatible avec un large éventail de technologies, ce qui en fait un choix populaire pour le traitement de flux en temps réel et l'analyse de données distribuées.

1. Kafka : Flink s'intègre facilement avec Apache Kafka, qui est une plateforme de streaming distribuée, permettant de lire des données en temps réel à partir de Kafka pour les traiter et les analyser avec Flink.

2. Hadoop : Flink peut s'intégrer avec Hadoop, en tirant parti de son système de stockage distribué (HDFS) et en interagissant avec les données stockées dans le cadre du traitement distribué.

.

Dans l'environnement du big data, Flink est un composant qui se concentre uniquement sur le calcul et ne propose pas de stockage. En tant que partie de la pile d'infrastructure du big data, Flink se combine avec d'autres technologies pour fournir une solution end-to-end aux organisations cherchant à analyser rapidement et efficacement de grands ensembles de données.



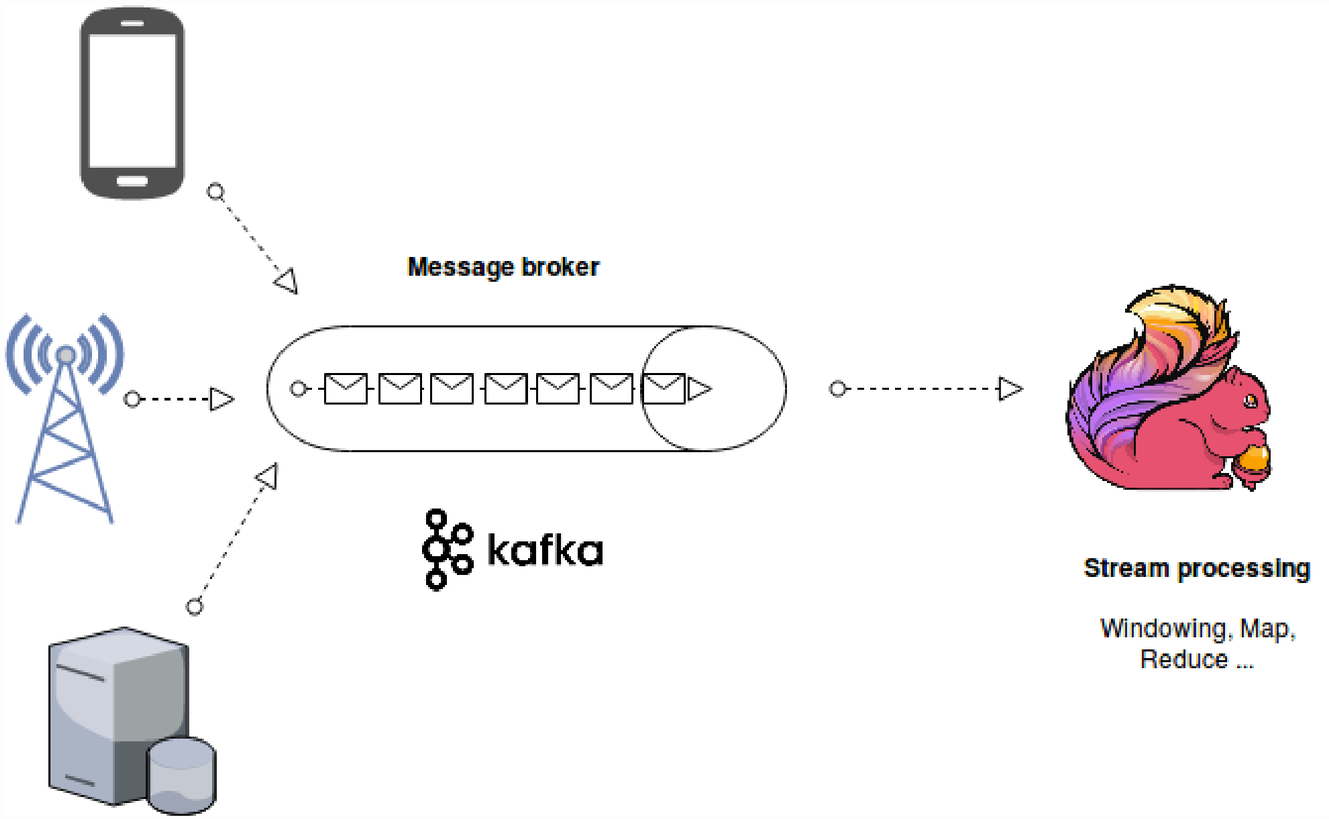
Il est configuré en conjonction avec Apache Kafka en tant que Event log.

Et aussi avec des systèmes tels que HDFS ou d'autres bases de données en tant que couche de stockage pour offrir des tâches ETL périodiques ou des pipelines de données en continu.

APACHE FLINK & APACHE KAFKA

Apache Flink peut se connecter directement à Apache Kafka pour consommer des flux de données en temps réel. Cela permet de traiter les événements en streaming et de réagir rapidement aux changements.

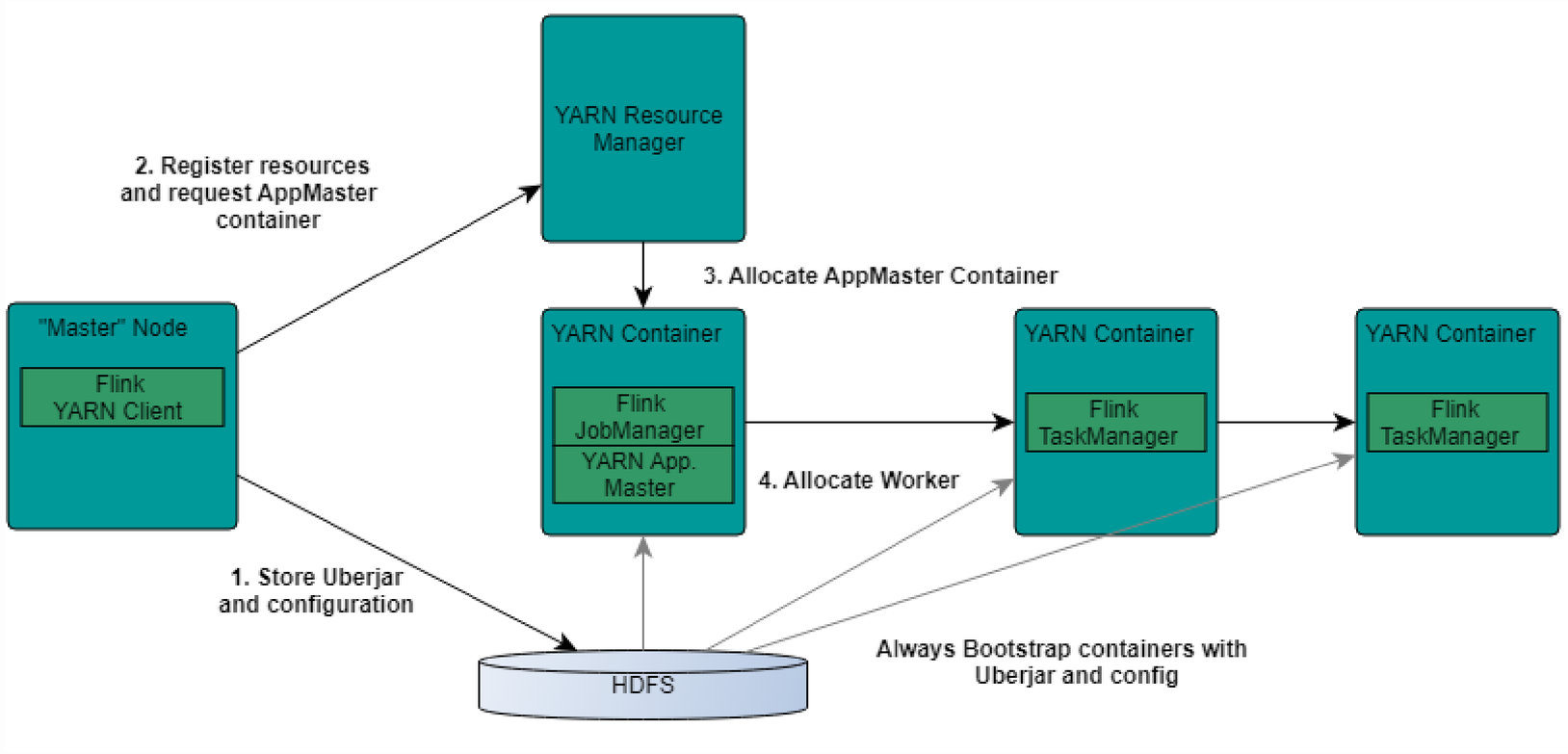
- Kafka permet de publier des messages provenant de sources hétérogènes.
- Apache Flink traitera ces données en appliquant des fonctions de fenêtrage ou de réduction.



APACHE FLINK & APACHE HADOOP

L'intégration d'Apache Flink avec Apache Hadoop (HDFS et YARN) crée un pipeline robuste de traitement de données. Les données résident dans HDFS.

Lorsqu'un job Flink est soumis, YARN alloue des conteneurs pour le JobManager et les TaskManagers de Flink. Le JobManager coordonne le job, tandis que les TaskManagers effectuent un traitement parallèle. Les données sont lues depuis HDFS, traitées, et les résultats sont stockés au besoin. Flink garantit la tolérance aux pannes en utilisant des points de contrôle dans HDFS. Une fois le job terminé, les ressources YARN sont libérées. Cette architecture optimise les ressources, autorise le traitement à grande échelle, et assure la fiabilité grâce à HDFS et YARN.



En bref, Apache Flink excelle par sa polyvalence, offrant un traitement de flux en temps réel pur et par lots. Avec une faible latence, des garanties de traitement robustes, et une gestion d'état intégrée, Flink répond aux besoins des applications modernes.

La communauté active et le support continu en font un choix fiable pour propulser vos applications de traitement de données vers l'avenir.

Merci pour votre attention