TRAITEMENT DONNEES TEMPS REEL

Réaliser des traitements distribués de données en temps réel
Créer et visualiser des métriques à partir de données générées en temps réel
- Créer une plateforme distribuée de flux de messages en temps réel

Sommaire

01
- Contexte, présentation du **projet**.
  - Les enjeux

02
- Choix **techniques réalisés**.
  - Technologies.
  - Présentation technique de l'application.

03
- Présentation des **métriques obtenues**.
  - Dashboard Kibana.

04
- Les **scénarios "catastrophes"**
  - Vélocité
  - Traitement des erreurs
  - Elasticsearch

05
- **Conclusion**.

01

PRESENTATION DU PROJET

Le contexte

Notre petit neveu a une opinion assez tranchée : pour s'enrichir sans se fatiguer, il faut miser sur le Bitcoin.
Le Bitcoin est une cryptomonnaie qui fut présentée pour la première fois en novembre 2008 par une personne, ou un groupe de personnes, sous le pseudonyme de Satoshi Nakamoto.

Le Bitcoin

Pour créer et gérer les bitcoins, Bitcoin s'appuie sur un logiciel. Dans ce logiciel, les bitcoins sont créés conformément à un protocole qui rétribue les agents qui ont traité des ***transactions***. Ces agents mettent à contribution leur puissance de calcul informatique afin de vérifier, de sécuriser et d'inscrire les transactions dans un registre virtuel, appelé la ***blockchain***, en français chaîne de blocs, nom qui vient du fait que l'entité de base de Bitcoin s'appelle un bloc, et que les blocs sont ensuite reliés en une chaîne, la chaîne de blocs.

Pour chaque nouveau bloc accepté, l'activité de vérification-sécurisation-enregistrement, appelée ***minage***, est rémunérée par des bitcoins nouvellement créés (***12.5 btc*** actuellement) et par les frais des transactions traitées. En tant que monnaie ou commodité, les bitcoins peuvent être échangés contre d'autres monnaies ou commodités, biens ou services. Le ***taux d'échange*** de la cryptomonnaie est fixé principalement sur des places de marché spécialisées et fluctue selon la loi de l'offre et de la demande.

Notre besoin

Afin d'investir sereinement, nous allons réaliser quelques analyses à partir des données publiquement disponibles.

Notre mission

Nous allons croiser des informations publiques relatives au cours du bitcoin, au minage et aux transactions en bitcoins afin d'obtenir un tableau de bord qui affiche en temps réel les métriques suivantes :

- Le cours du bitcoin en euros.
- Le volume de bitcoins échangés par heure, en bitcoins et en euros.
- La valeur maximale des transactions réalisées par heure, en bitcoins et en euros.
- Les mineurs qui se sont le plus enrichis, par jour et par mois, en bitcoins et en euros.

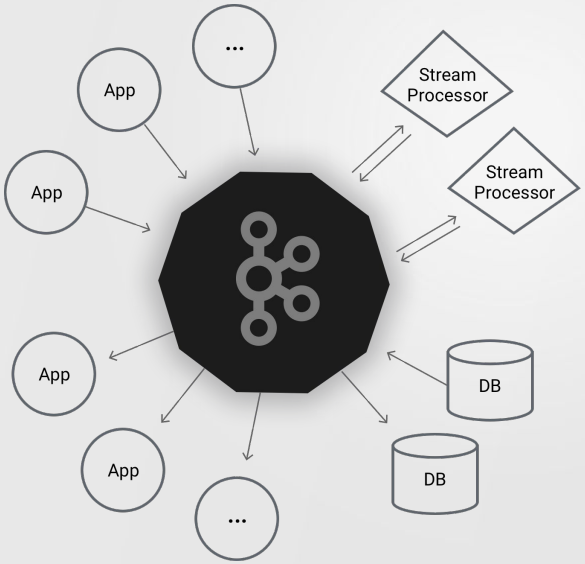
02

CHOIX TECHNIQUES

Technologies

APACHE KAFKA

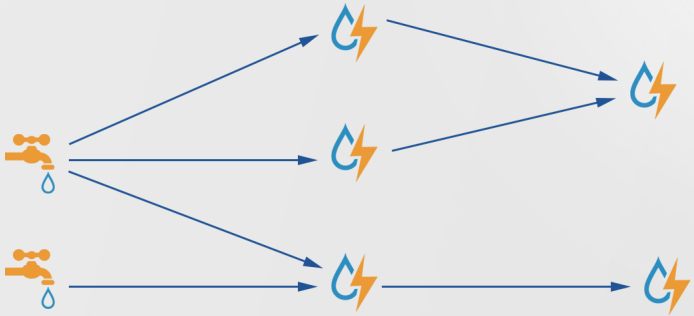
- **Apache Kafka** est un projet à code source ouvert d'agent de messages développé par l'Apache Software Foundation et écrit en Scala. Le projet vise à fournir un système unifié, en temps réel à latence faible pour la manipulation de flux de données. Kafka peut agir comme une plateforme distribuée qui centralise tous les messages qui transitent entre différentes applications.



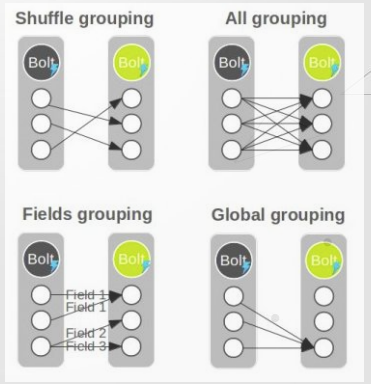
Technologies

APACHE STORM

- **Apache Storm** est un framework de calcul de traitement de flux distribué, écrit principalement dans le langage de programmation Clojure. Créé à l'origine par Nathan Marz et l'équipe de BackType le projet est rendu open source après avoir été acquis par Twitter. Il utilise des "**spouts**" et des "**bolts**" créés sur mesure pour définir les sources d'informations et les manipulations permettant un traitement par lots et distribué des données en continu. Une application Storm est conçue comme une "**topologie**" sous la forme d'un graphe acyclique dirigé (**DAG**) avec des spouts et des bolts faisant office de sommets du graphe. Les bords du graphique sont des flux nommés et dirigent les données d'un nœud à un autre. Ensemble, la topologie agit comme un pipeline de transformation de données. Les topologies Storm s'exécutent indéfiniment jusqu'à ce qu'elles soient supprimées(tandis qu'un DAG de travail MapReduce doit finir).



Quelques regroupements de flux



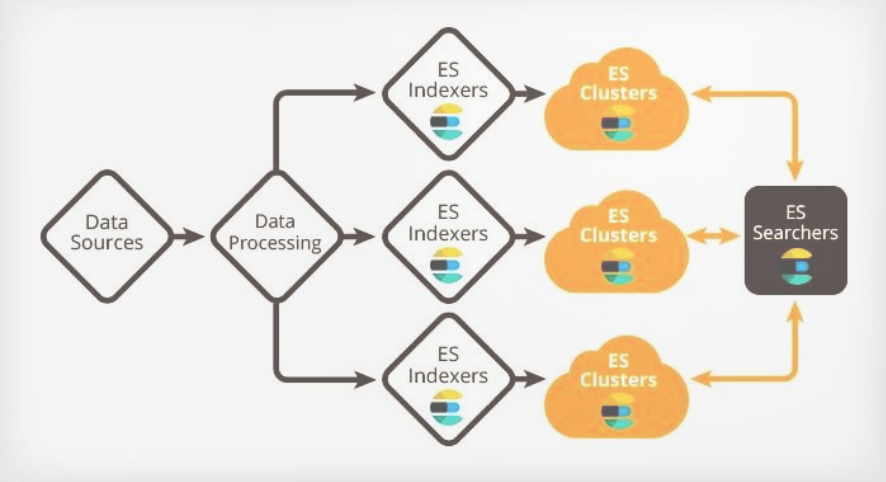
Technologies

ELASTIC SEARCH

elasticsearch

- **Elasticsearch** est un serveur utilisant Lucene pour l'indexation et la recherche des données. Il fournit un moteur de recherche distribué et multi-entité à travers une interface REST. C'est un logiciel libre écrit en Java et publié en open source sous licence Apache.

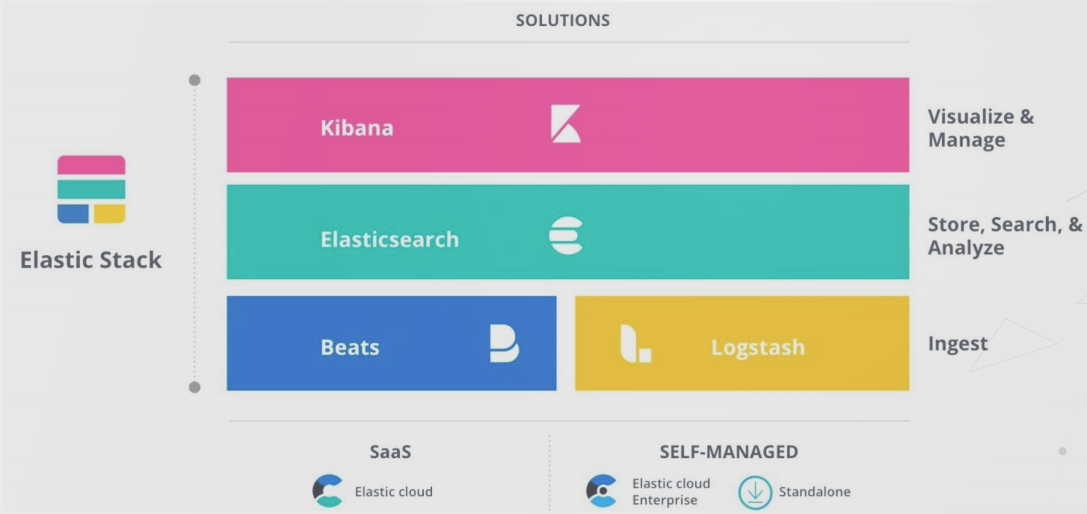
  Elasticsearch assure la disponibilité des données, mais pas leur cohérence. Il ne doit donc pas être utilisé comme base de données principale (exception sur ce projet).



Technologies

KIBANA

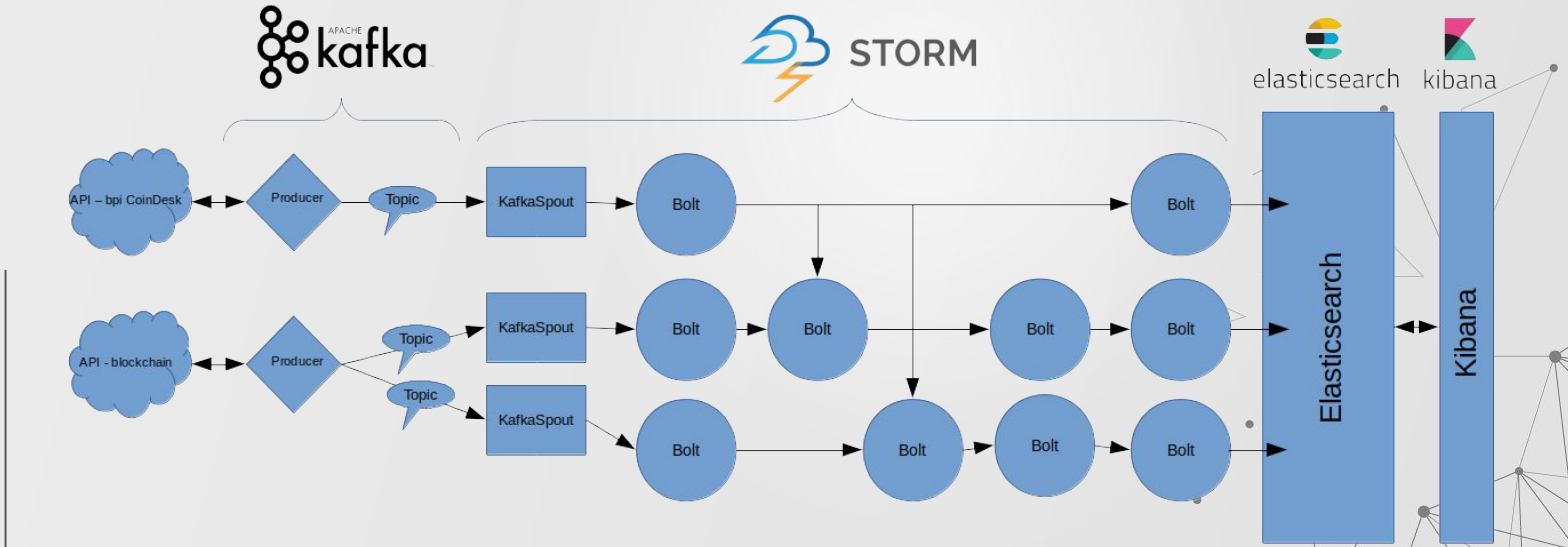
- **Kibana** est un greffon de visualisation de données pour Elasticsearch publié sous la licence libre Apache. Il fournit des fonctions de visualisation sur du contenu indexé dans une grappe Elasticsearch. Il permet également de manager Elasticsearch, notamment avec son dev tools qui permet d'administrer la bdd avec une interface.



Présentation technique de l'application

- **Schéma d'architecture des différents composants**

Architecture - topologie(Storm) Lien vers diagramme détaillé



Présentation technique de l'application

- **Démarrage serveurs**

  Utilisation de supervisord qui permettra de manière centralisée la gestion des processus et leur redémarrage en cas de panne/arrêt.

```
francois@alaska:~/ElasticSearch/elasticsearch-7.4.0/config$ sudo supervisorctl start storm: es-kibana:
storm:zookeeper: started
storm:kafka-9093: started
storm:kafka-9092: started
storm:kafka-9094: started
storm:storm-nimbus: started
storm:kafka-manager: started
storm:storm-ui: started
storm:storm-workers: started
es-kibana:elasticsearch1: started
es-kibana:elasticsearch3: started
es-kibana:elasticsearch2: started
es-kibana:kibana: started
```

Les 3 noeuds Elasticsearch se connectent entre eux pour former un cluster. Regroupés sous un même nom de cluster oc-da-p3.

- **Création des topics Kafka**

  La réplication se fait sur les 3 serveurs kafka démarrés, ports: 9092, 9093, 9094.

```
francois@alaska:~/Kafka/kafka_2.12-2.3.0$ ./bin/kafka-topics.sh --create --zookeeper localhost:2181 --replication-factor 3 --partitions 9 --topic p3-btc-bpi
Created topic p3-btc-bpi.
francois@alaska:~/Kafka/kafka_2.12-2.3.0$ ./bin/kafka-topics.sh --create --zookeeper localhost:2181 --replication-factor 3 --partitions 9 --topic p3-btc-blocks
Created topic p3-btc-blocks.
francois@alaska:~/Kafka/kafka_2.12-2.3.0$ ./bin/kafka-topics.sh --create --zookeeper localhost:2181 --replication-factor 3 --partitions 9 --topic p3-btc-transactions
Created topic p3-btc-transactions.
```

Présentation technique de l'application

- **Création des Index Elasticsearch**

  Chaque index aura **3 shards** (1 shard par defaut Depuis ES6), et **2 réplicas**.

  GET _cluster/health?pretty

```
{
  "cluster_name" : "oc-da-p3",
  "status" : "green",
  "timed_out" : false,
  "number_of_nodes" : 3,
  "number_of_data_nodes" : 3,
  "active_primary_shards" : 9,
  "active_shards" : 21,
  "relocating_shards" : 0,
  "initializing_shards" : 0,
  "unassigned_shards" : 0,
  "delayed_unassigned_shards" : 0,
  "number_of_pending_tasks" : 0,
  "number_of_in_flight_fetch" : 0,
  "task_max_waiting_in_queue_millis" : 0,
  "active_shards_percent_as_number" : 100.0
}
```

```
PUT btc-value
{
  "settings": {
    "number_of_shards": 3,
    "number_of_replicas": 2
  }
}

PUT btc-volume
{
  "settings": {
    "number_of_shards": 3,
    "number_of_replicas": 2
  }
}

PUT btc-miner
{
  "settings": {
    "number_of_shards": 3,
    "number_of_replicas": 2
  }
}

PUT btc-raw-bpi
{
  "settings": {
    "number_of_shards": 3,
    "number_of_replicas": 2
  }
}

PUT btc-raw-blocks
{
  "settings": {
    "number_of_shards": 3,
    "number_of_replicas": 2
  }
}

PUT btc-raw-transactions
{
  "settings": {
    "number_of_shards": 3,
    "number_of_replicas": 2
  }
}
```

```
{
  "acknowledged" : true,
  "shards_acknowledged" : true,
  "index" : "btc-value"
}
```

Présentation technique de l'application

- **Lancement des producers**

  Utilisation de supervisord

```
francois@alaska:~$ sudo supervisorctl start producer:
[sudo] Mot de passe de francois :
producer:bpi: started
producer:blockchain: started
```

- **Lancement de l'application**

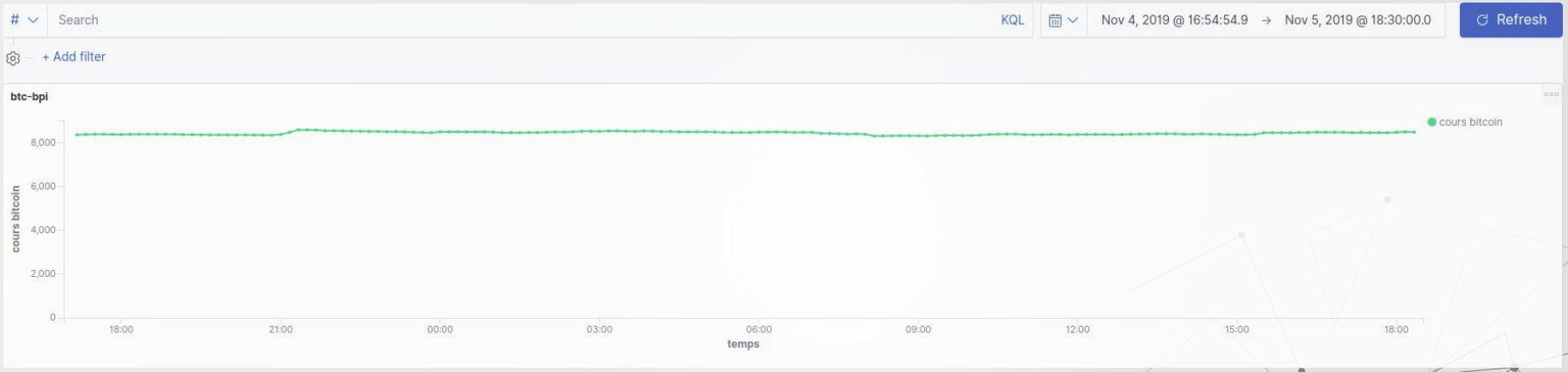
  App.java

03

METRIQUES OBTENUES

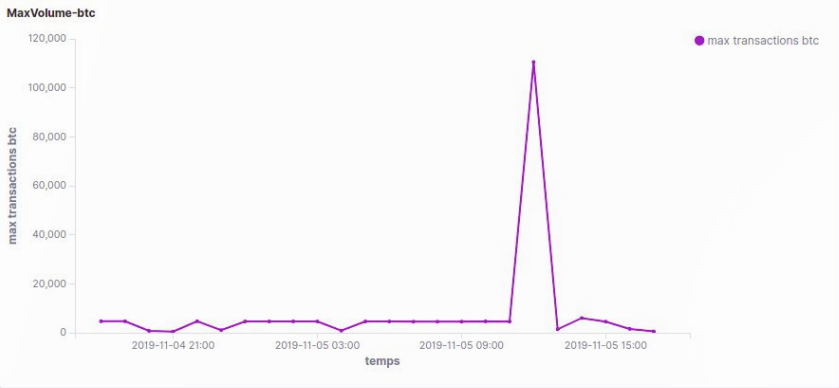
Métriques

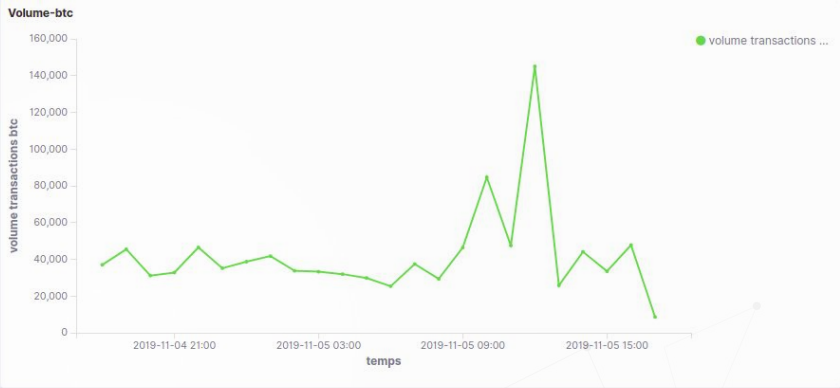
Bitcoin price index

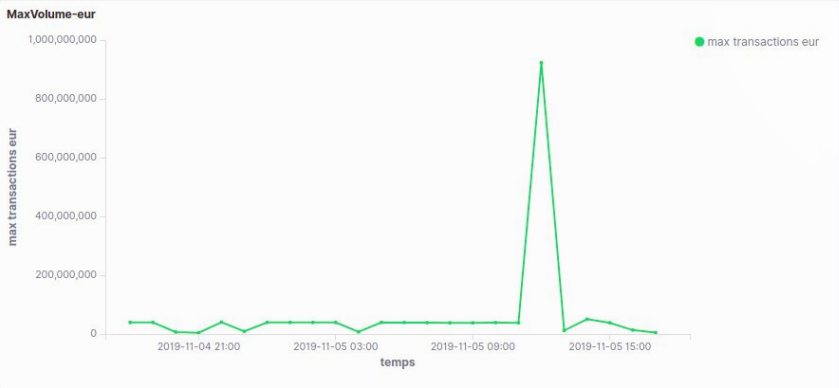


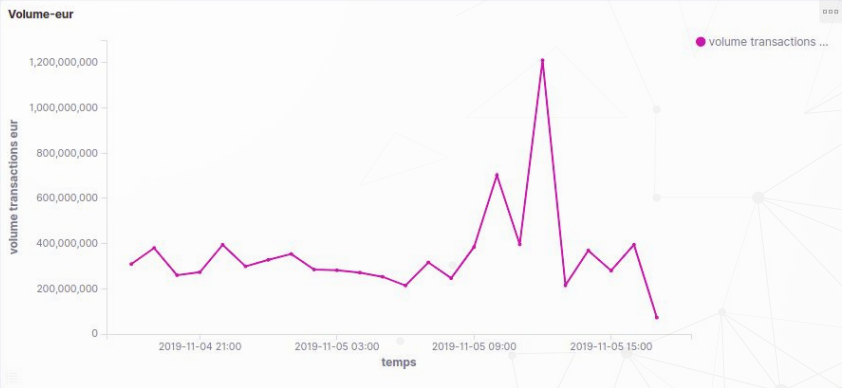
Métriques

Volume



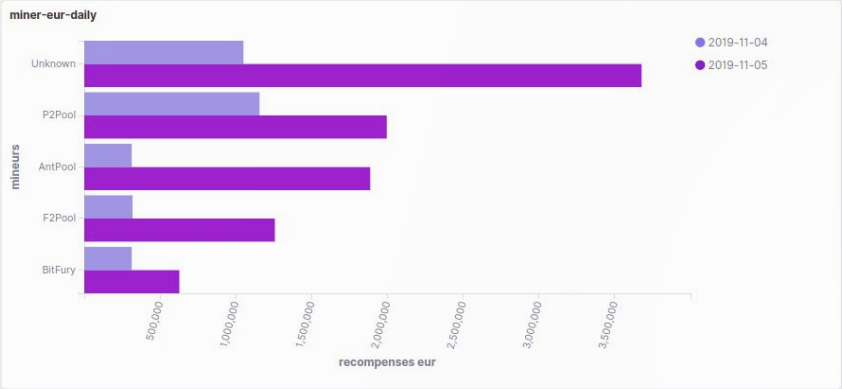


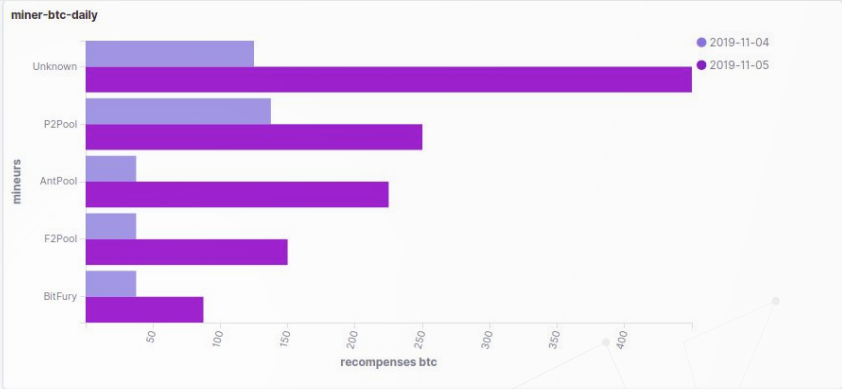


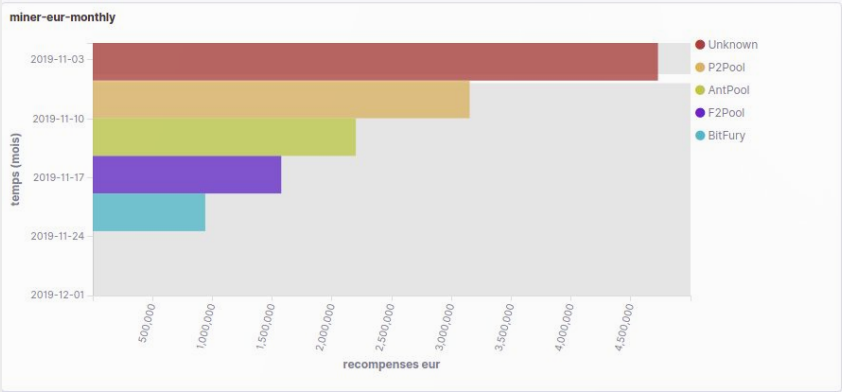


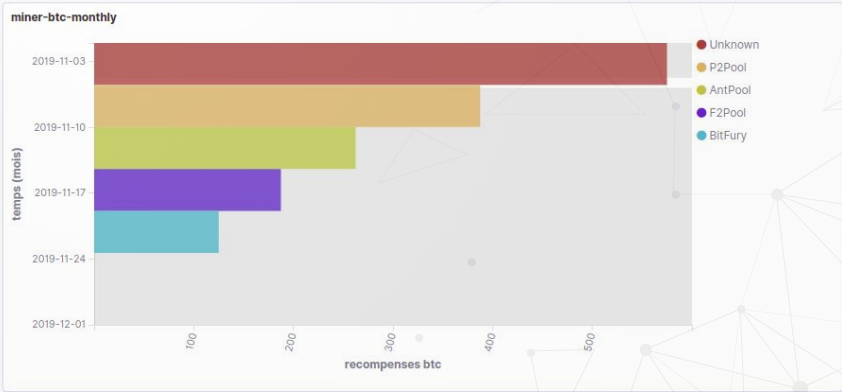
Métriques

Mineurs









04

SCENARIOS “CATASTROPHES”

Scenario 1

Que se passe-t-il lorsque la vélocité des données est multipliée par 10, 100 ou 1000 ?

Faire grandir le cluster Kafka de manière horizontale, en rajoutant des noeuds.

Augmenter le nombre de consumers et donc de partitions en respectant la règle:

- dans un groupe, le ***nombre de consumers <= nombre de partitions***.

Nous pouvons également modifier le parallelism des spouts et bolts (modifier le nombre d'executors) et le nombre de workers en cours de production: "rebalancing" par l'application webUI ou avec le CLI.

Scenario 2

Comment allez-vous gérer les erreurs dans le traitement des événements ?

Les tuples émis par les spouts (KAFKASPOUT) possèdent un identifiant qui leur est propre.

Après le traitement de chaque tuple, chaque bolt indique le succès ou l'échec du traitement du tuple à l'aide des méthodes **ack()** et **fail()**.

Un tuple donné peut être mis en échec de deux manières différentes :

- En appelant la *méthode fail()* d'un OutputCollector.
- Si le traitement d'un tuple *excède un délai maximal* défini par *TOPOLOGY_MESSAGE_TIMEOUT_SECS*.

Storm va gérer les échecs de tuple par de nouvelles tentatives d'envoie de ces tuples.

Scenario 3

Que se passe-t-il si le cluster Elasticsearch devient soudainement injoignable ?

Un cluster est un ensemble d'un ou plusieurs nœuds (serveurs) qui stockent toutes les données et permet d'indexer et de rechercher des données dans l'ensemble des nœuds. Les clusters Elasticsearch sont dotés de partitions principales et de copies pour fournir un basculement en cas de panne d'un nœud. Lorsqu'une partition principale tombe en panne, la copie prend sa place.

Pour prendre en compte une panne du cluster, il faut prévoir une réplication du cluster sur un cluster secondaire pour permettre une reprise d'activité. La réplication inter-clusters (CCR, "cross-cluster replication") est disponible pour la production dans Elasticsearch 6.7.0.

05
CONCLUSION

Un projet de traitement de données temps réel

- Réaliser des traitements distribués de données en temps réel
- Créer et visualiser des métriques à partir de données générées en temps réel
- Créer une plateforme distribuée de flux de messages en temps réel

Ressources

Web

- https://kafka.apache.org/
- https://storm.apache.org/
- https://www.elastic.co/fr/
- https://stackoverflow.com/questions/17257448/what-is-the-task-in-storm-parallelism
- https://storm.apache.org/releases/2.0.0/Understanding-the-parallelism-of-a-Storm-topology.html
- https://fr.slideshare.net/edvorkin/learning-stream-processing-with-apache-storm
- https://dzone.com/articles/storm-kafka-integration-with-configurations-and-co
- https://www.elastic.co/fr/blog/follow-the-leader-an-introduction-to-cross-cluster-replication-in-elasticsearch
- https://blog.zenika.com/2014/01/31/storm-ajouter-du-temps-reel-a-votre-bigdata/
- https://community.cloudera.com/t5/Community-Articles/How-to-write-topology-with-the-new-kafka-spout-client-in/ta-p/244661
- https://www.slideshare.net/eiichirouchiumi/storm-anatomy
- https://www.slideshare.net/Phelion/a-la-rencontre-de-kafka-le-log-distribu-par-florian-garcia?qid=e770a78c-b867-4671-8e4d-0a3bf0ce1a28

Et bien d'autres...

MERCI

Avez-vous des **questions**?

f2buttet@gmail.com
06.84.19.58.69