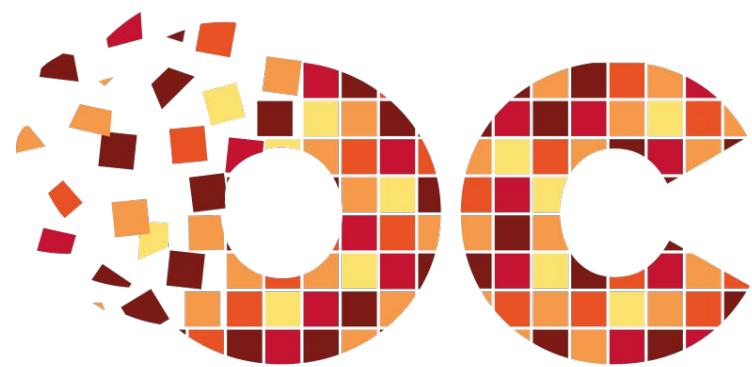


Traitez des données en temps réel

Projet 3: real-time data processing

Présenté par:

RASAMBATRA Freonel



Sommaire:

1. Présentation du sujet
 - a. ce qui est demandé
 - b. Données d'entrées
 - c. enjeux
2. Choix techniques:
 - a. Architecture fonctionnelle
 - b. Architecture technique
 - c. Topologie Storm
 - d. Flux de données sur ES
3. Métriques:
 - a. Présentation du tableau de bord
4. Présentations des différents scénarios:
 - a. Kafka / Zookeeper
 - b. Storm
 - c. ElasticSearch
5. Conclusion
6. Questions / Réponses

1. Présentation du sujet:

- Récolter toutes les informations du bitcoins :
 - Prix du bitcoins en Euro, Dollar, Livre Sterling
 - Opérations en cours:
 1. Transactions
 2. Minages
- Traiter ces informations pour visualiser en temps-réel:
 1. Le cours du bitcoin en euros.
 2. Volume de bitcoins échangés par heure:
 - en bitcoins
 - en euros.
 3. La valeur maximale des transactions réalisées par heure:
 - en bitcoins
 - en euros.
 4. Les mineurs qui se sont le plus enrichis:
 - par jour: en bitcoins et en euros.
 - par mois: en bitcoins et en euros.

1. Présentation du sujet:

```
{  
  "time":{  
    "updated":"Sep 18, 2013 17:27:00 UTC",  
    "updatedISO":"2013-09-18T17:27:00+00:00"  
  },  
  "disclaimer":"This data was produced from the Coindesk Bitcoin API. It is NOT real time data.",  
  "bpi":{  
    "USD":{  
      "code":"USD",  
      "symbol": "$",  
      "rate":"126.5235",  
      "description":"United States Dollar",  
      "rate_float":126.5235  
    },  
    "GBP":{  
      "code":"GBP",  
      "symbol": "£",  
      "rate":"79.2495",  
      "description":"British Pound Sterling",  
      "rate_float":79.2495  
    },  
    "EUR":{  
      "code":"EUR",  
      "symbol": "€",  
      "rate":"94.7398",  
      "description":"Euro",  
      "rate_float":94.7398  
    }  
  }  
}
```

- API Coindesk: donne les cours du bitcoin ==> format JSON
 - <http://api.coindesk.com/v1/bpi/currentprice.json>
- Récupération par minutes équivaut à ~ 0,03 message / s
- Intérêt:
 - a. date instant t: type date
 - b. "rate_float": type de double
 - euro
 - usd
 - gbp

```
'op': 'utx',
'x': {
  'lock_time': 0,
  'ver': 1,
  'size': 226,
  'inputs': [
    {
      'sequence': 4294967295,
      'prev_out': {
        'spent': False,
        'tx_index': 0,
        'type': 0,
        'addr': '1PX9rXFojSvpbdTVbYTAvcc6BwJKeBZbMS',
        'value': 362086,
        'n': 1,
        'script': '76a914f706a30b8829dc3f431e35561e0f8339b56d734c88ac'
      },
      'script': '483045022100925d94ef935fa231bd6dd45356522ad7bc2560e0c7ef
    },
    {
      'time': 1587758416,
      'tx_index': 0,
      'vin_sz': 1,
      'hash': '2806b055b5496552d934f40ee3e99f55978e0e86f2771e8f9055386c7b61d9a7',
      'vout_sz': 2,
      'relayed_by': '',
      'out': [
        {
          'spent': False,
          'tx_index': 0,
          'type': 0,
          'addr': '15Eo881Cs6J1LZVLYLqBcqMoPv5g9oW38n',
          'value': 16302,
          'n': 0,
          'script': '76a9142e7c69781f09144b81a5feebc4c6dd341a69adbdb88ac'
        },
        {
          'spent': False,
          'tx_index': 0,
          'type': 0,
          'addr': '1M2M2rJoiUBV6Qiv6H9HfMMGtaQMSzEd1N',
          'value': 338100,
          'n': 1,
          'script': '76a914dba3a746cf2f4effed38158bacd7bc2bb65760d688ac'
        }
      ]
    }
  ]
}
```

1. Présentation du sujet:

- API Blockchain: donne les opérations en temps en temps réel ==> format JSON
 - websocket:
<ws://ws.blockchain.info/inv>
- Opération **Transaction**: “utx”
 - Flux ~ 3,1 messages / s
 - Intérêt:
 - a. type d’opération op: type string
 - b. montant de la transaction: type de double
 - x(out) -> out(value) / 100000000.0

```
{  
  'op': 'block',  
  'x': {  
    'txIndexes': [0, 0, 0, 0, 0, 0, 0, 0, 0,  
    'nTx': 2183,  
    'estimatedBTCSent': 1156519235881,  
    'totalBTCSent': 6119077446876,  
    'reward': 0,  
    'size': 1143392,  
    'weight': 3993221,  
    'blockIndex': 0,  
    'prevBlockIndex': 0,  
    'height': 628276,  
    'hash': '000000000000000000000000000000007b32d939ac  
    'mrklRoot': '93069b12e5e69d8e3f0099f60  
    'difficulty': 15958652328578.422,  
    'version': 805298176,  
    'time': 1588253494,  
    'bits': 387031859,  
    'nonce': 573381017,  
    'foundBy': {  
      'description': '',  
      'ip': '',  
      'link': '',  
      'time': 0  
    }  
  }  
}
```

1. Présentation du sujet:

- Opérations: “block”
- Intérêt:
 - a. type d’opération op: type string
 - b. “toalBTCSent”: type de double
 - c. hash: type string
 - d. identité du mineur “foundBy”: type str
 - Si description vide, on le récupère sur un autre API:
<https://chain.api.btc.com/v3/block/hash>
 - Si toujours vide => Miner = “anonymous”

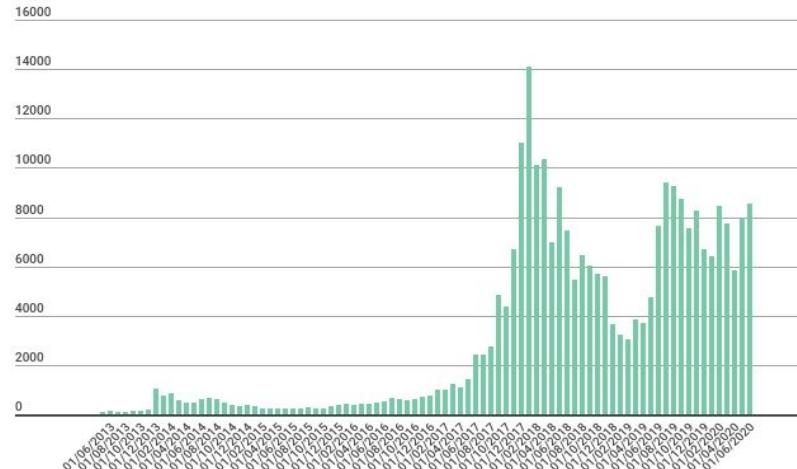
1. c) Les enjeux:

- Dit Tableau de bord dit observation du résultat en temps-réel:
 - a. traiter les données à la même vitesse qu'elles arrivent → latence faible
 - b. s'adapter aux volumes (possibilité très élevés) → passage à l'échelle de manière horizontale
- Contrainte est de mettre en place une architecture robuste:
 - a. Tolérant aux pannes: pas de single point of failure
 - b. déployables sur de grosses volumes de données
 - c. ne pas perdre de données
 - d. Gérer les erreurs : anomalie ou

1. c) Les enjeux:

- Surveiller le moment opportun pour investir dans le Bitcoins:
 - Croisement des info prix + opérations
- Observer les tendances des opérations
- Découvrir les plus grand utilisateur du Bitcoin
- Analyse en temps réel peut s'appliquer à surveiller dans d'autres activité, comme par exemple:
 - Supervision
 - Bourse
 - ...
- Adaptation au niveau de l'API et le JSON

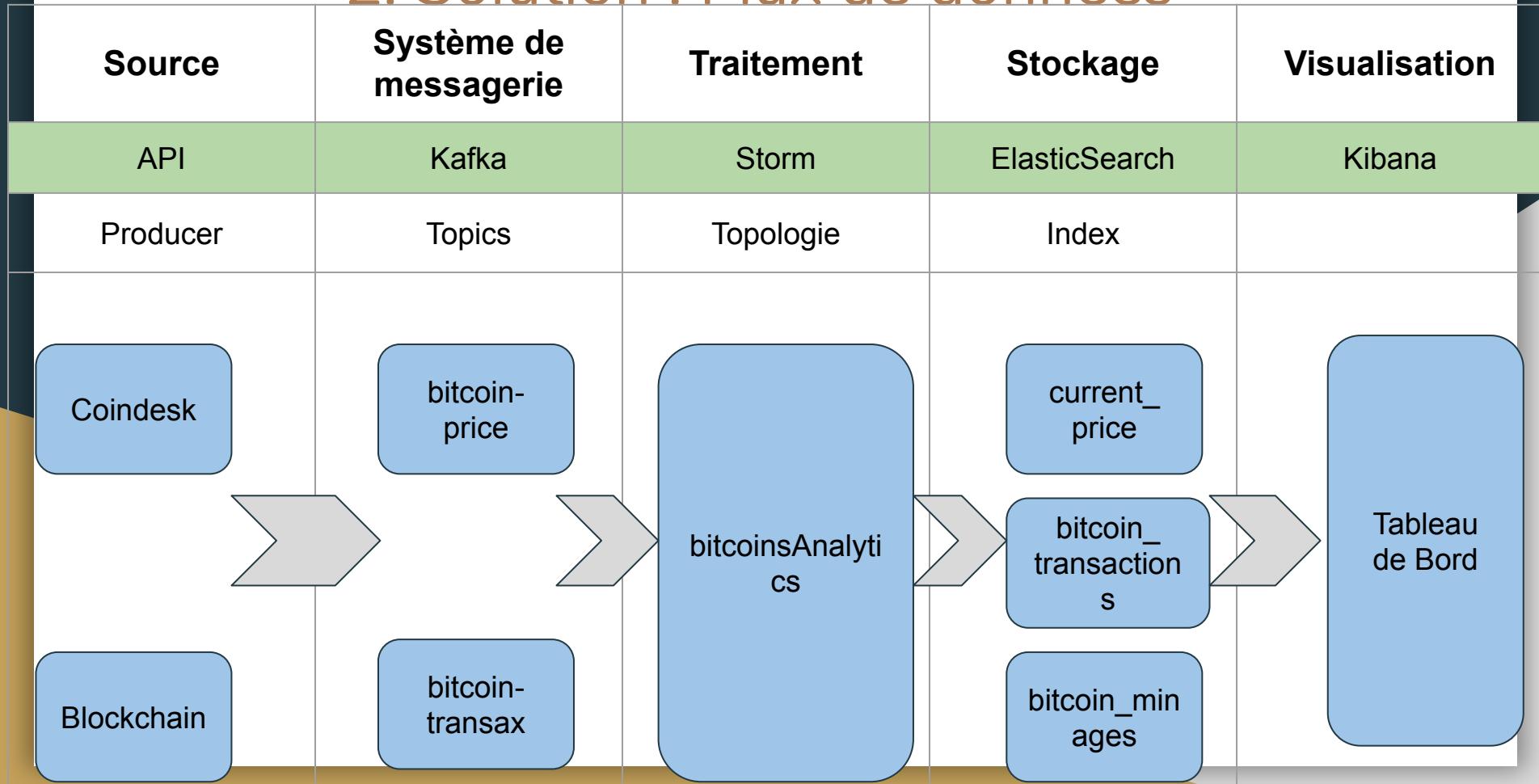
Cours mensuel du bitcoin en euros



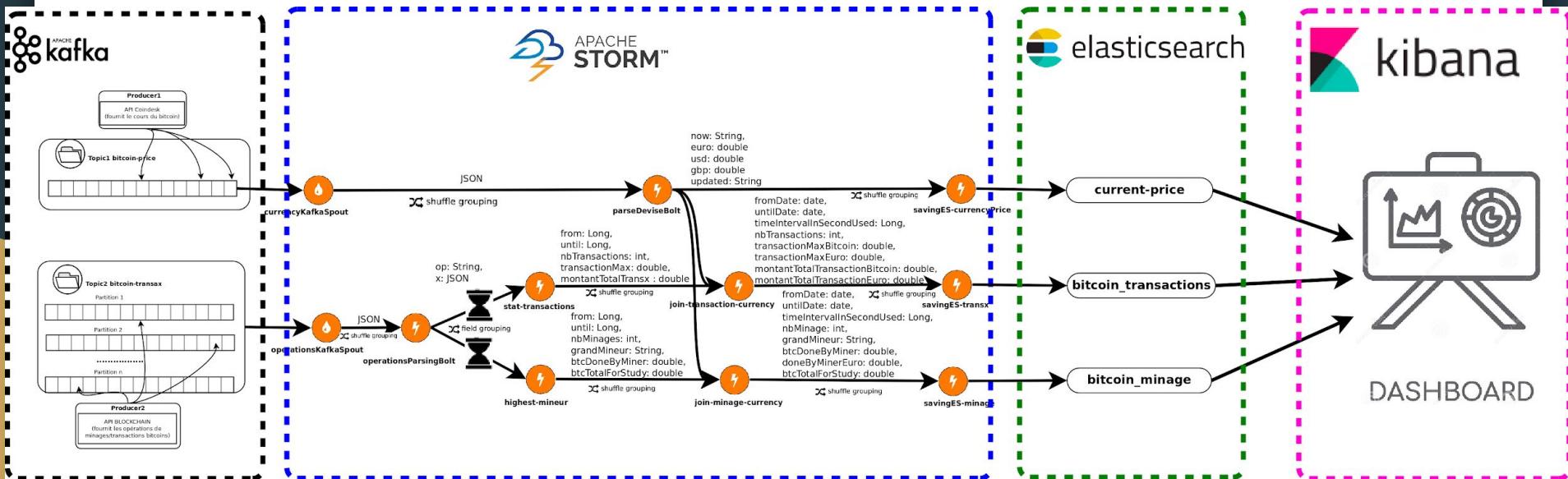
Lexique:

- ❖ Kafka:
 - Producer: produit les messages qui passeront à travers le cluster
 - Consumer: consomme les messages dans les fils d'attentes
 - Partitions: canaux de fils de messages
- ❖ Storm:
 - Spouts: génèrent les tuples (messages)
 - Bolts: traitent les tuples
 - Topologie: graphe pour représenter les connexions entre les spouts et les bolts.
- ❖ Zookeeper: gestionnaire de cluster présent dans Kafka et Storm. Il synchronise les différents éléments du cluster.

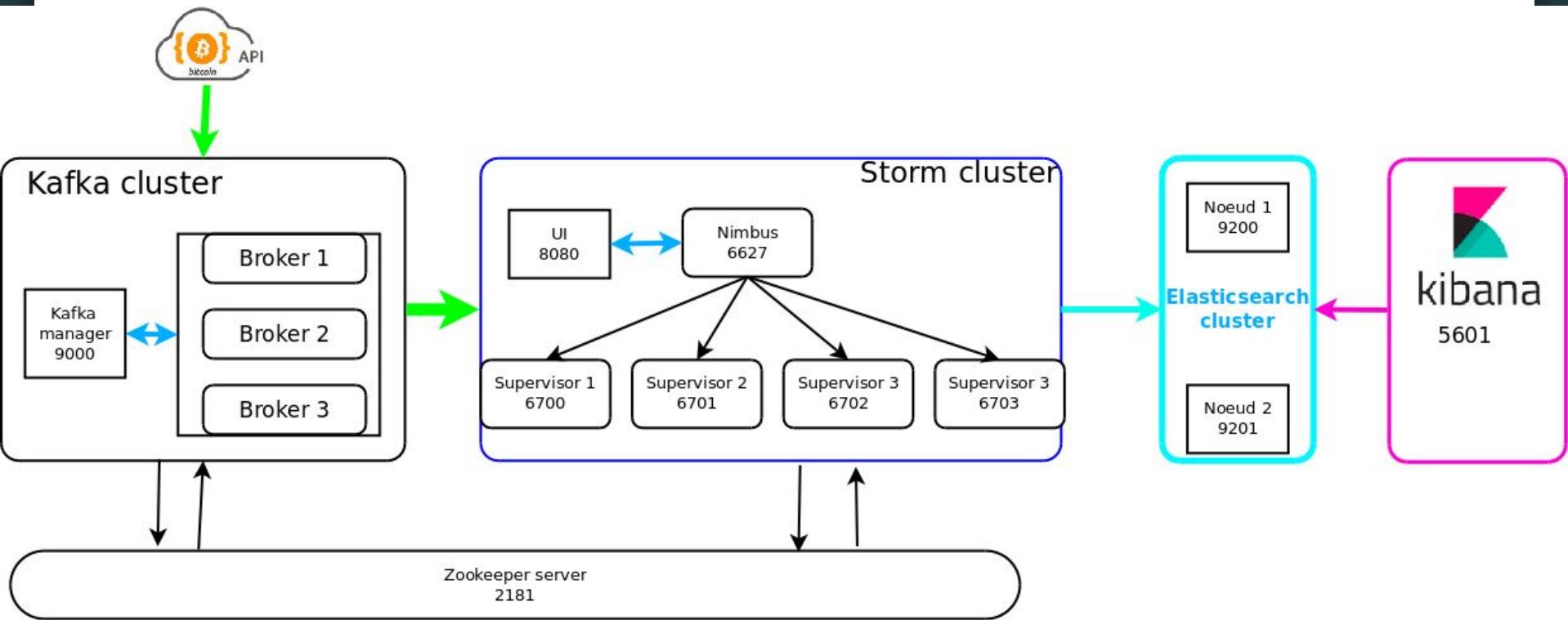
2. Solution : Flux de données



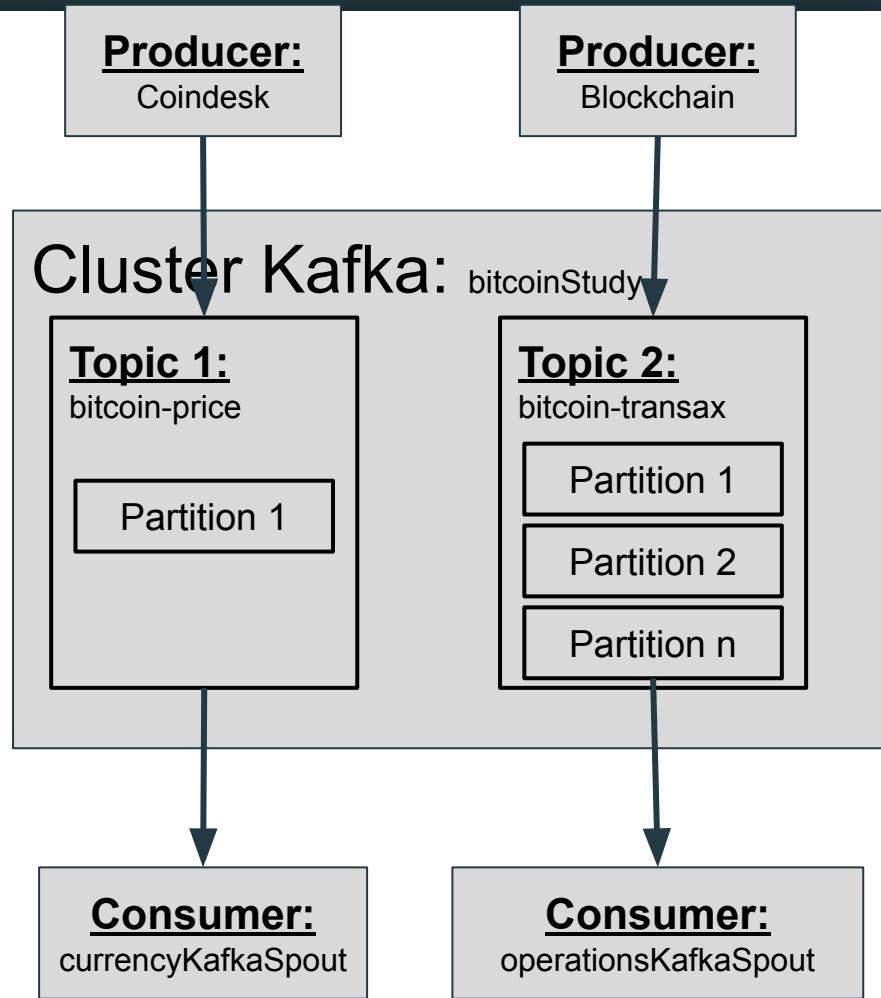
2.a) Architecture fonctionnelle



2.b) Architecture technique:

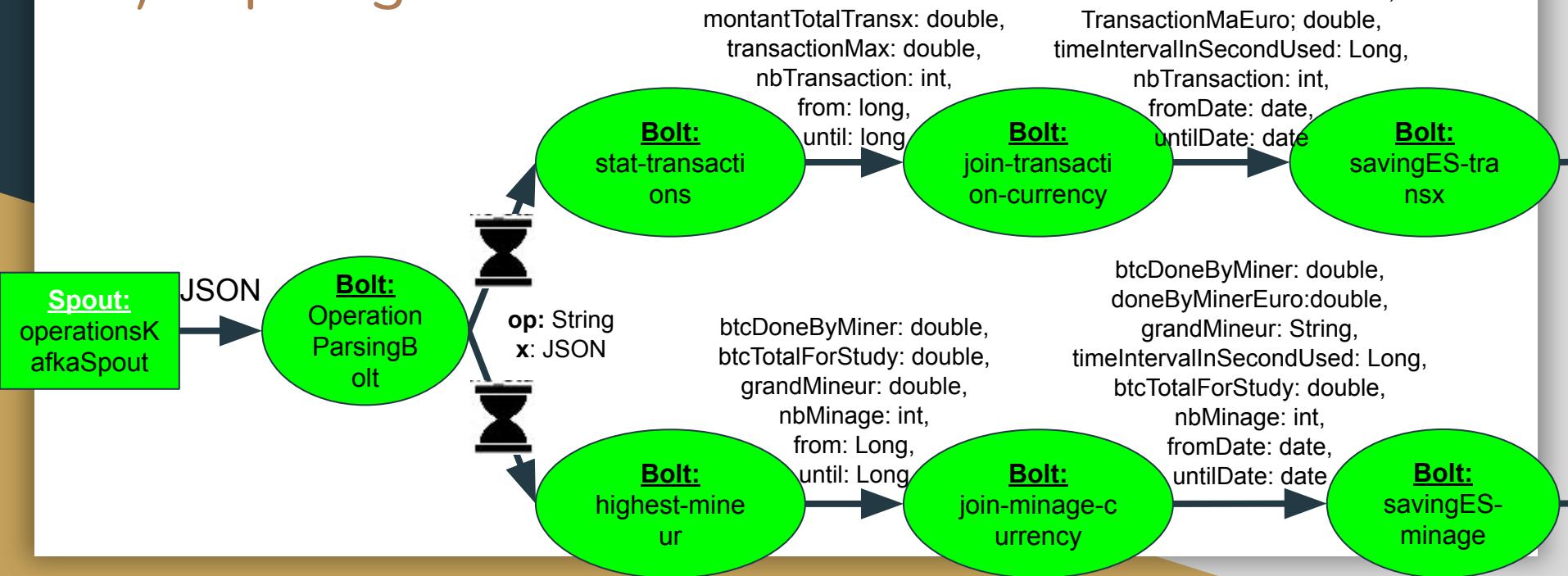


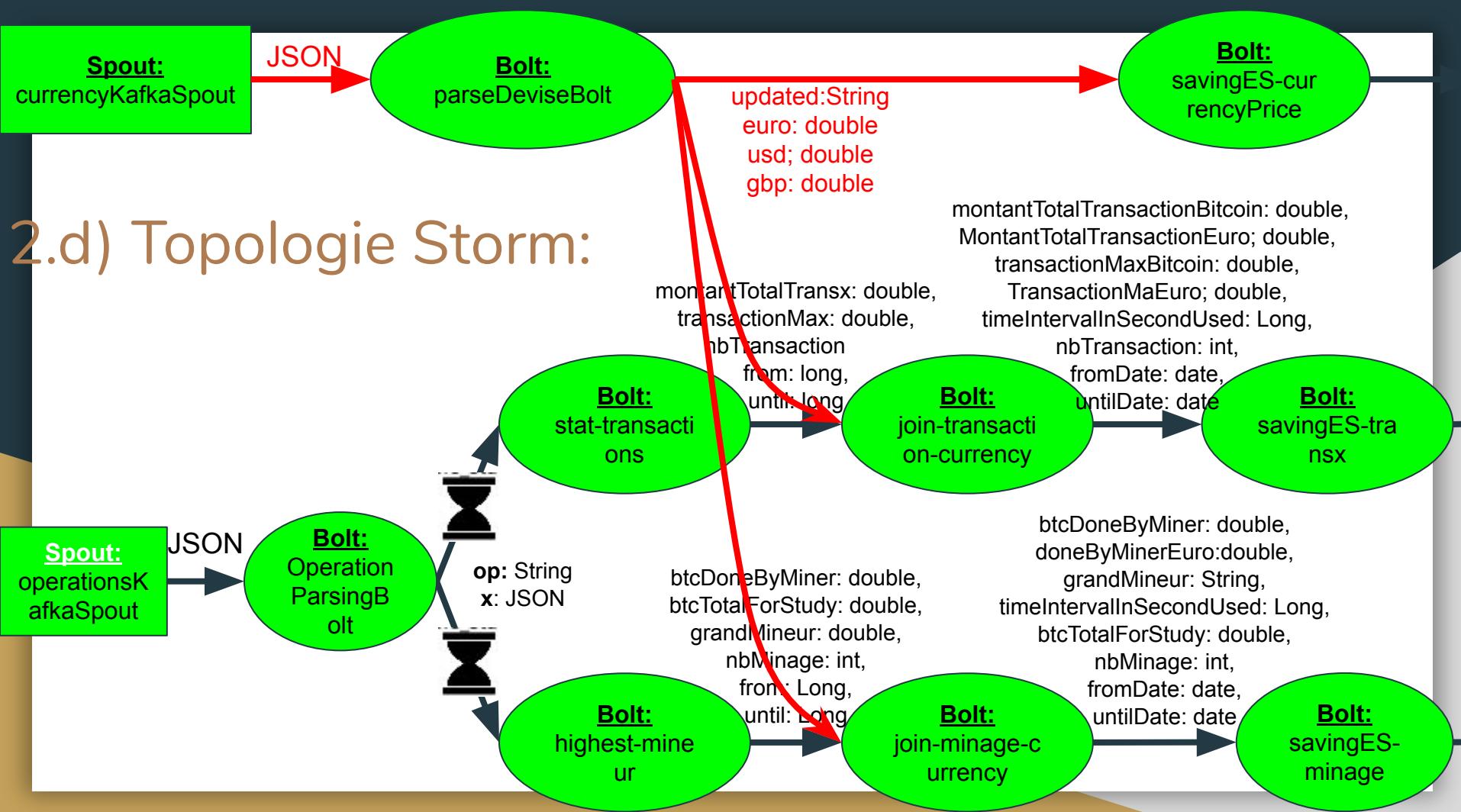
2.c) Kafka:





2.d) Topologie Storm:

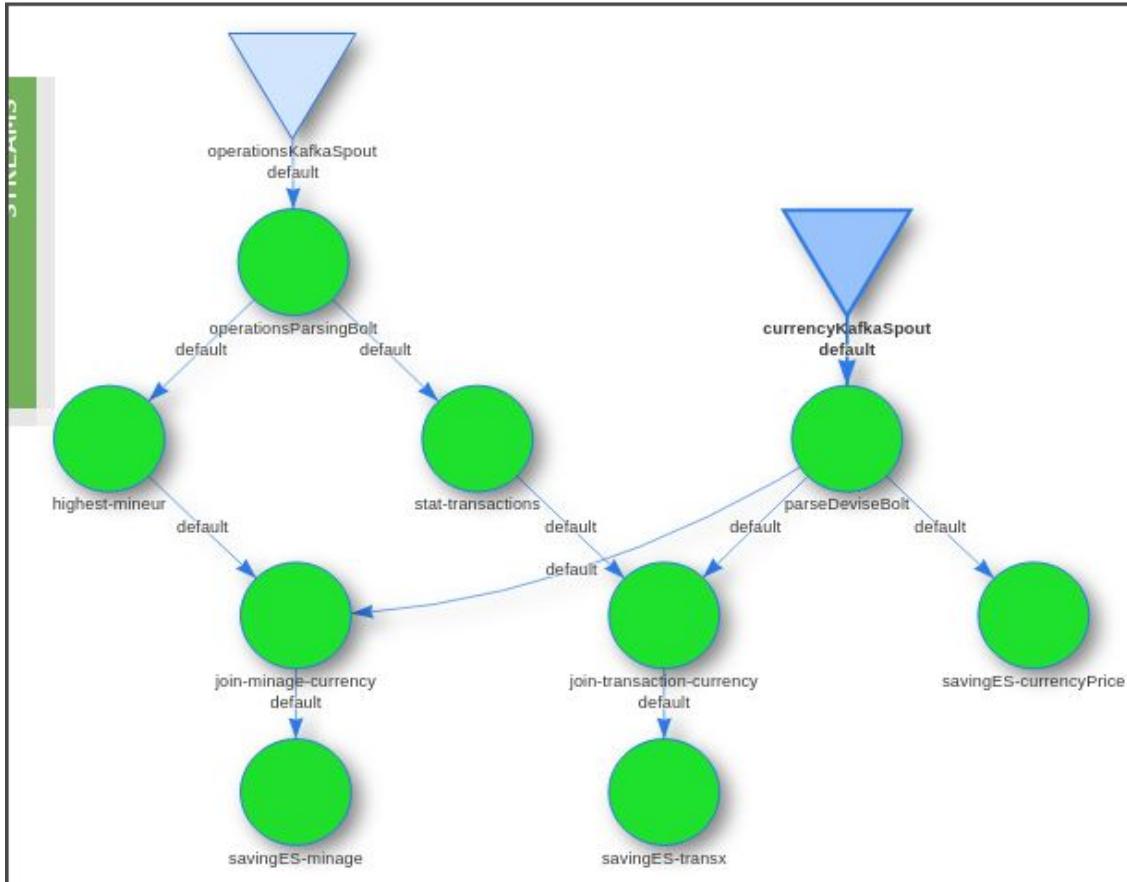




Topology Visualization

[Hide Visualization](#)[Open Visualization](#)

2.d) Topologie Storm:



2.e) Structures de données: Elasticsearch



Time	_source
Jun 6, 2020 @ 14:17:00.196	now: Jun 6, 2020 @ 14:17:00.196 euro: 8,616.078 usd: 9,728.843 gbp: 7,681.369 updated: Jun 6, 2020 12:16:00 UTC _id: JveQixIB970SF6tcE_Un _type: devises _index: current_price _score: -

Index bitcoin_transactions

Time	_source
Jun 6, 2020 @ 14:23:28.000	fromDate: Jun 6, 2020 @ 14:18:15.000 untilDate: Jun 6, 2020 @ 14:23:28.000 timeIntervalInSecondsUsed: 313 nbTransactions: 1,200 transactionMaxBitcoin: 159.823 transactionMaxEuro: 1,375,526.578 montantTotalTransactionBitcoin: 1,472.783 montantTotalTransactionEuro: 12,675,575.445 _id: LfeWiXIB970SF6tcBvUH _type: stats _index: bitcoin_transactions _score: -

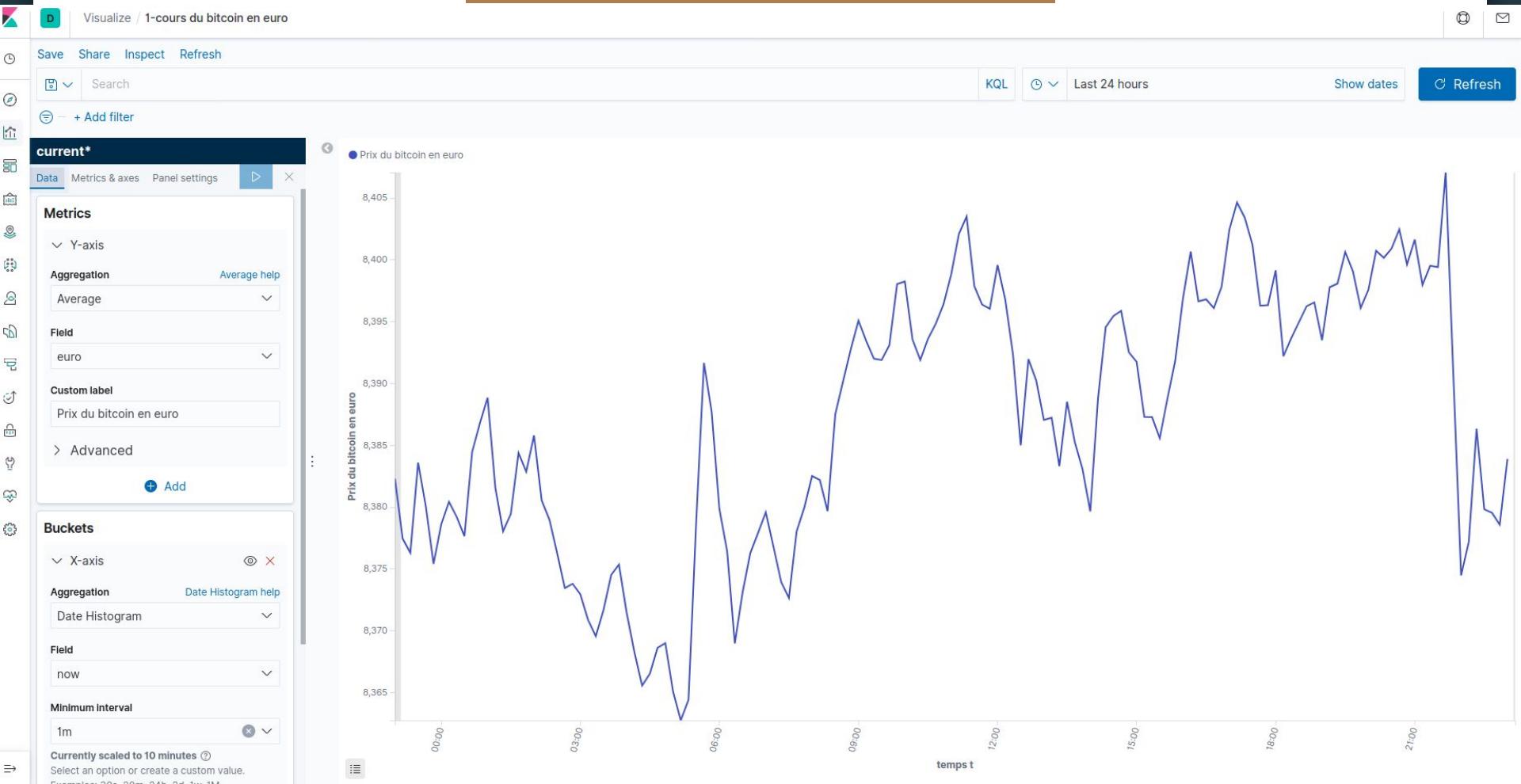
Index bitcoin_minages

Time	_source
Jun 6, 2020 @ 13:50:16.000	fromDate: Jun 6, 2020 @ 13:38:49.000 untilDate: Jun 6, 2020 @ 13:50:16.000 timeIntervalInSecondsUsed: 687 nbMineur: 3 grandMineur: Poolin btcDoneByMiner: 3,132.992 doneByMinerEuro: 26,869,386.876 btcTotalForStudy: 5,702.506 _id: Evd_iXIB970SF6tcCvUA _type: stats _index: bitcoin_minages _score: -

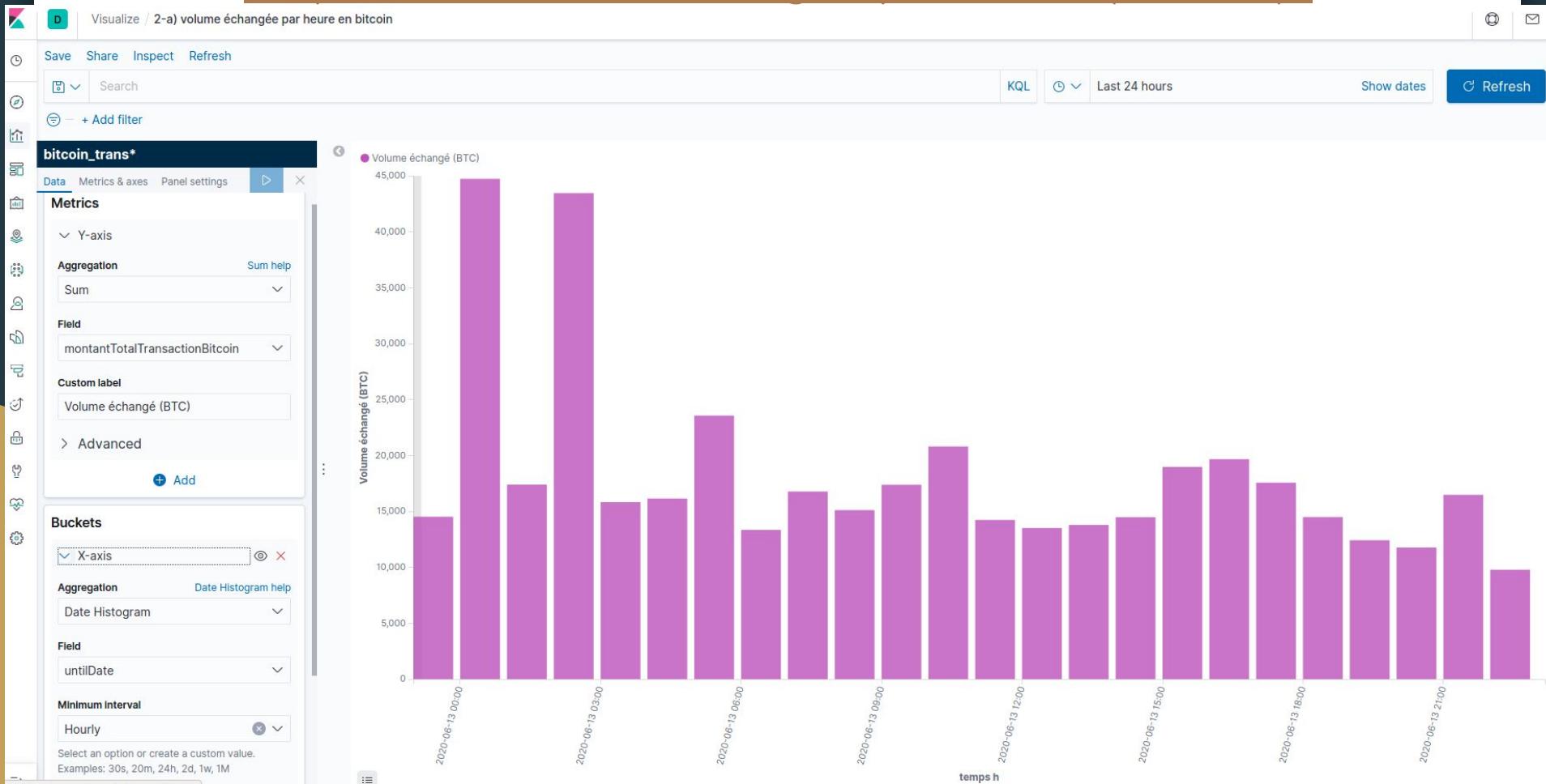
3. Métriques demandées:

1	Cours BTC (€)			
2	volume BTC échangés / h	(BTC)	a	
		(€)	b	
3	valeur maximale des transactions réalisées / heure	(BTC)	a	
		(€)	b	
4	mineur s qui se sont le plus enrichis	/ jour	(BTC)	a
			(€)	b
		/ mois	(BTC)	c
			(€)	d

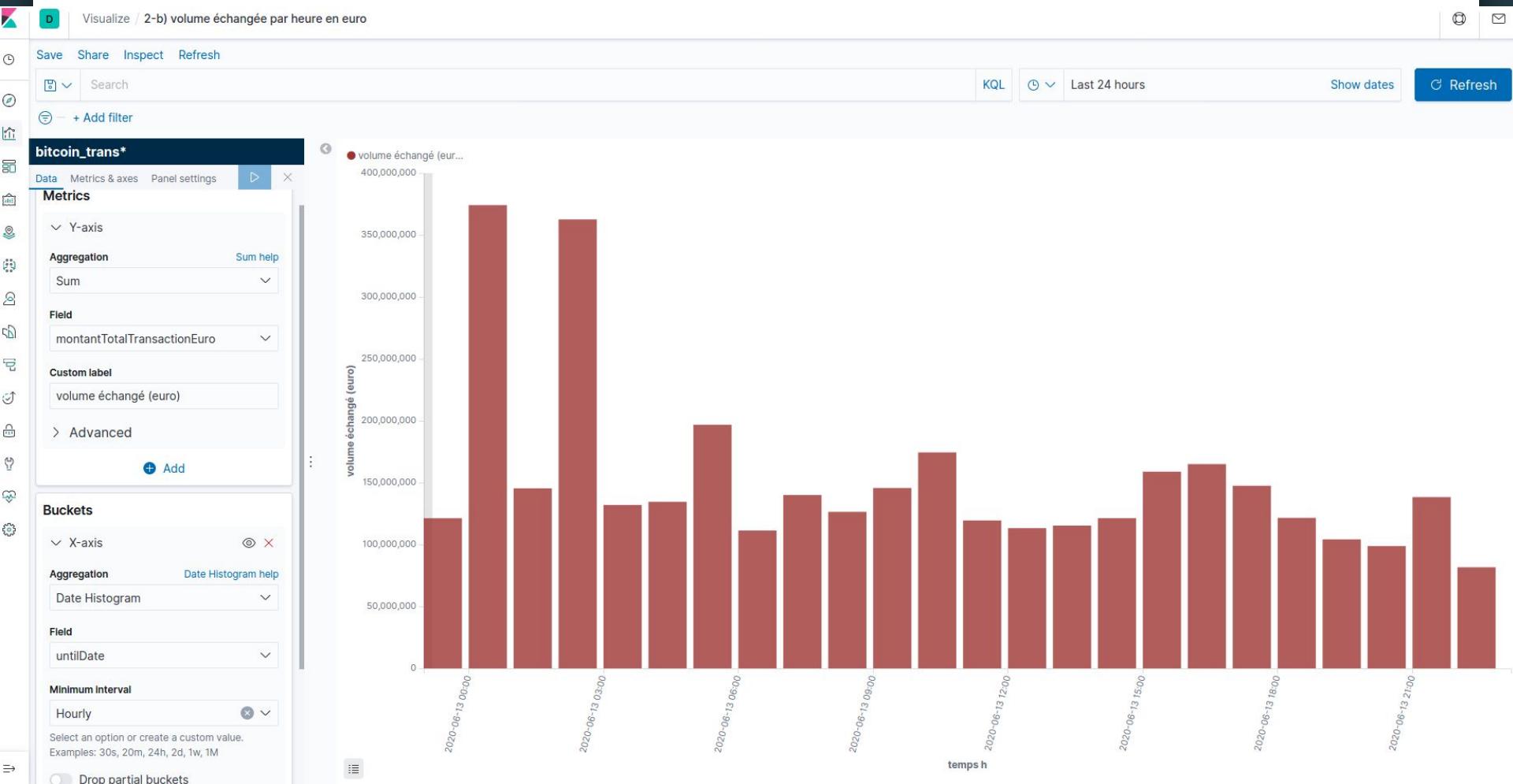
1-Cours du bitcoin en euro :



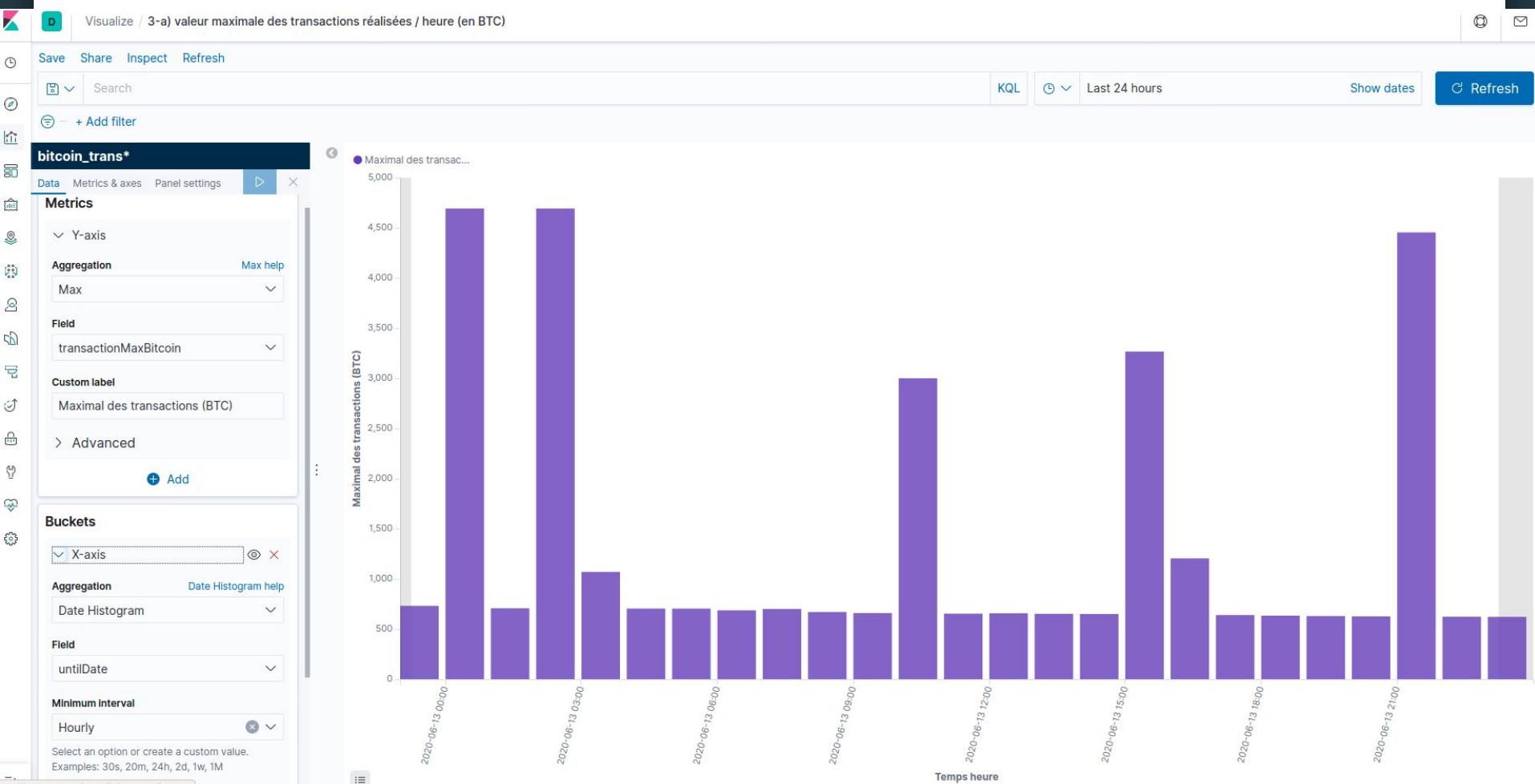
2-a) Volume bitcoin échangés par heure (en BTC) :



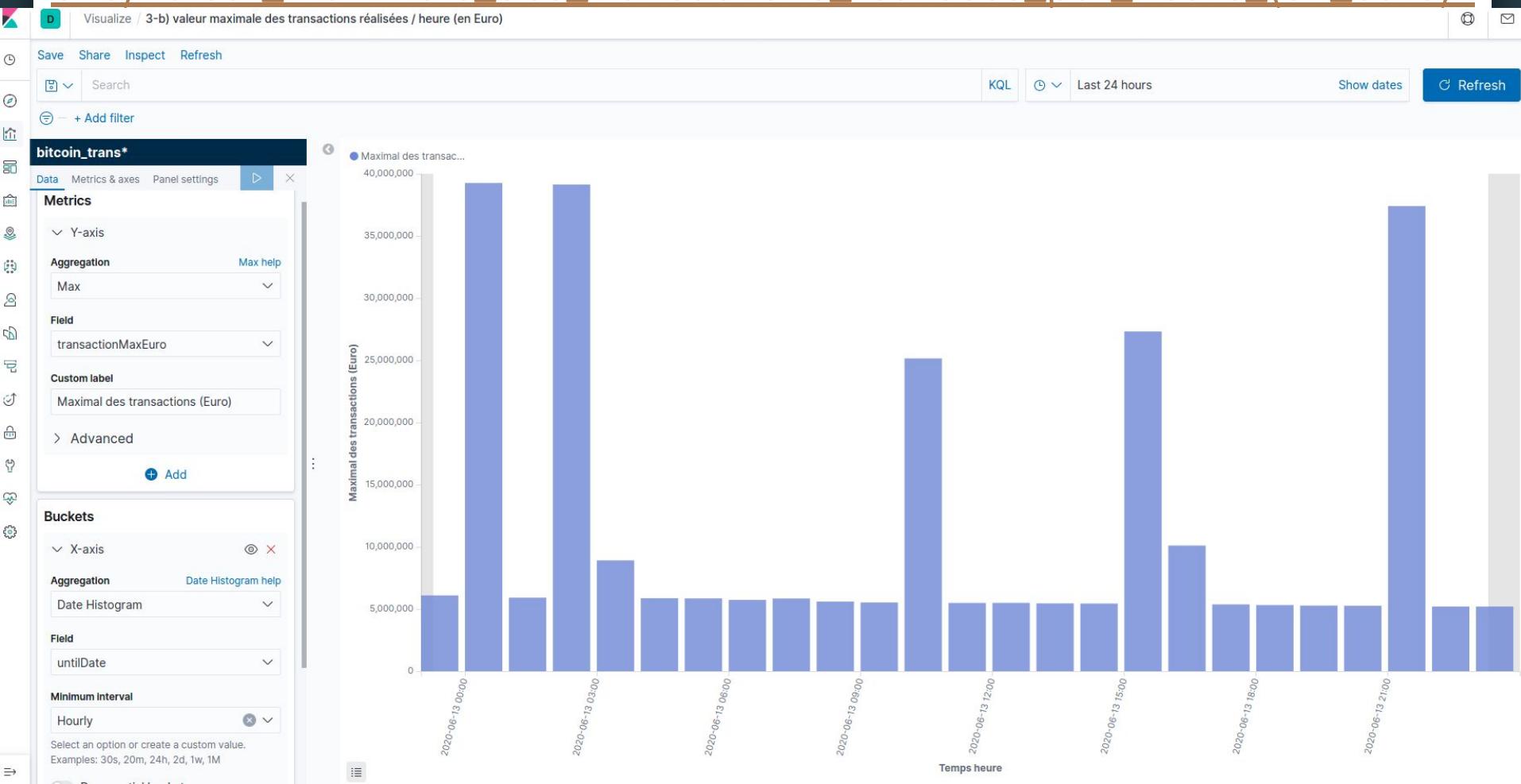
2-b)Volume_bitcoin_échangés_par_heure_(en_euro) :



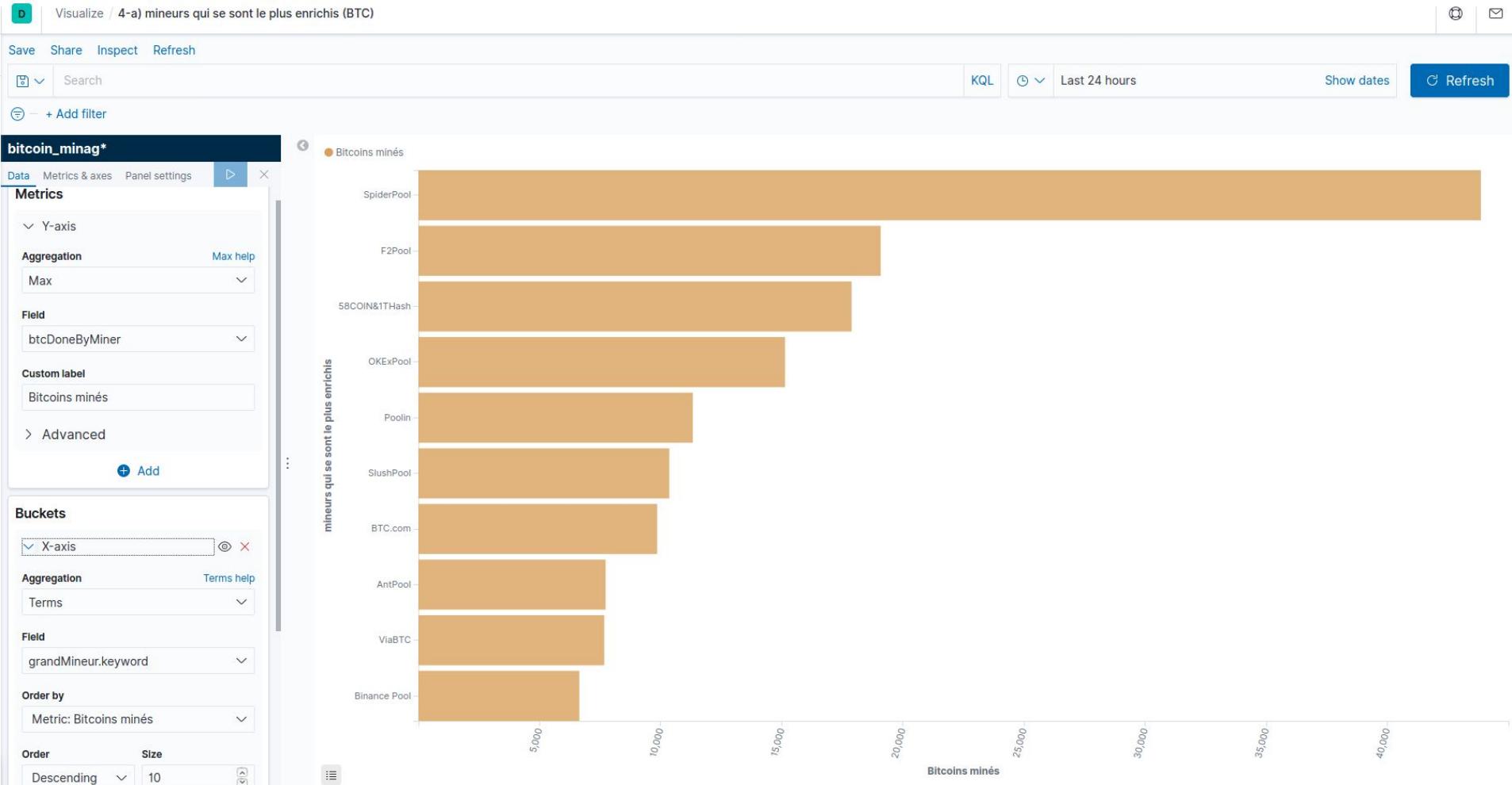
3-a)Valeur_maximale_des_transactions_réalisées_par_heure_(en_BTC) :



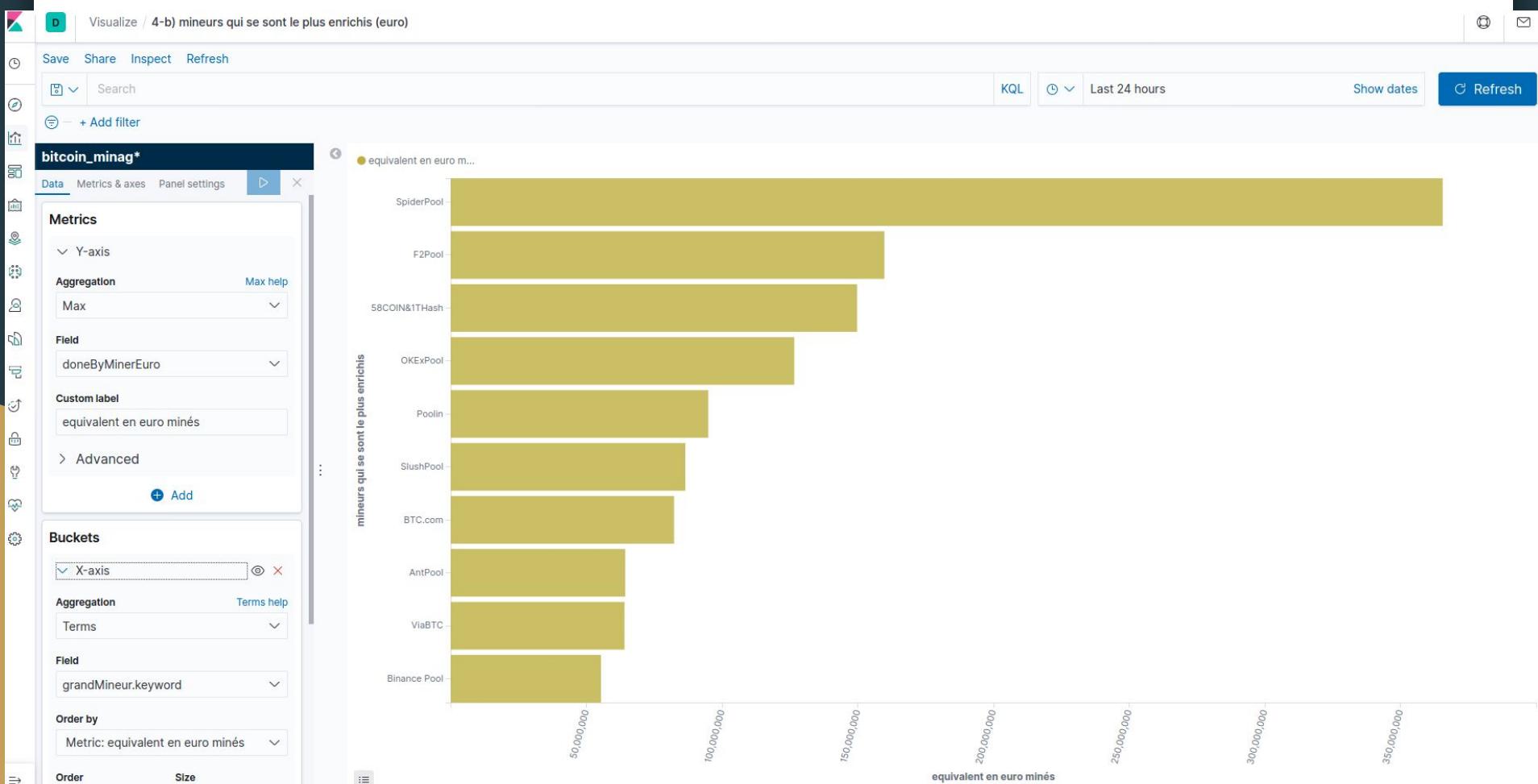
3-b)Valeur_maximale_des_transactions_réalisées_par_heure_(en_euro) :



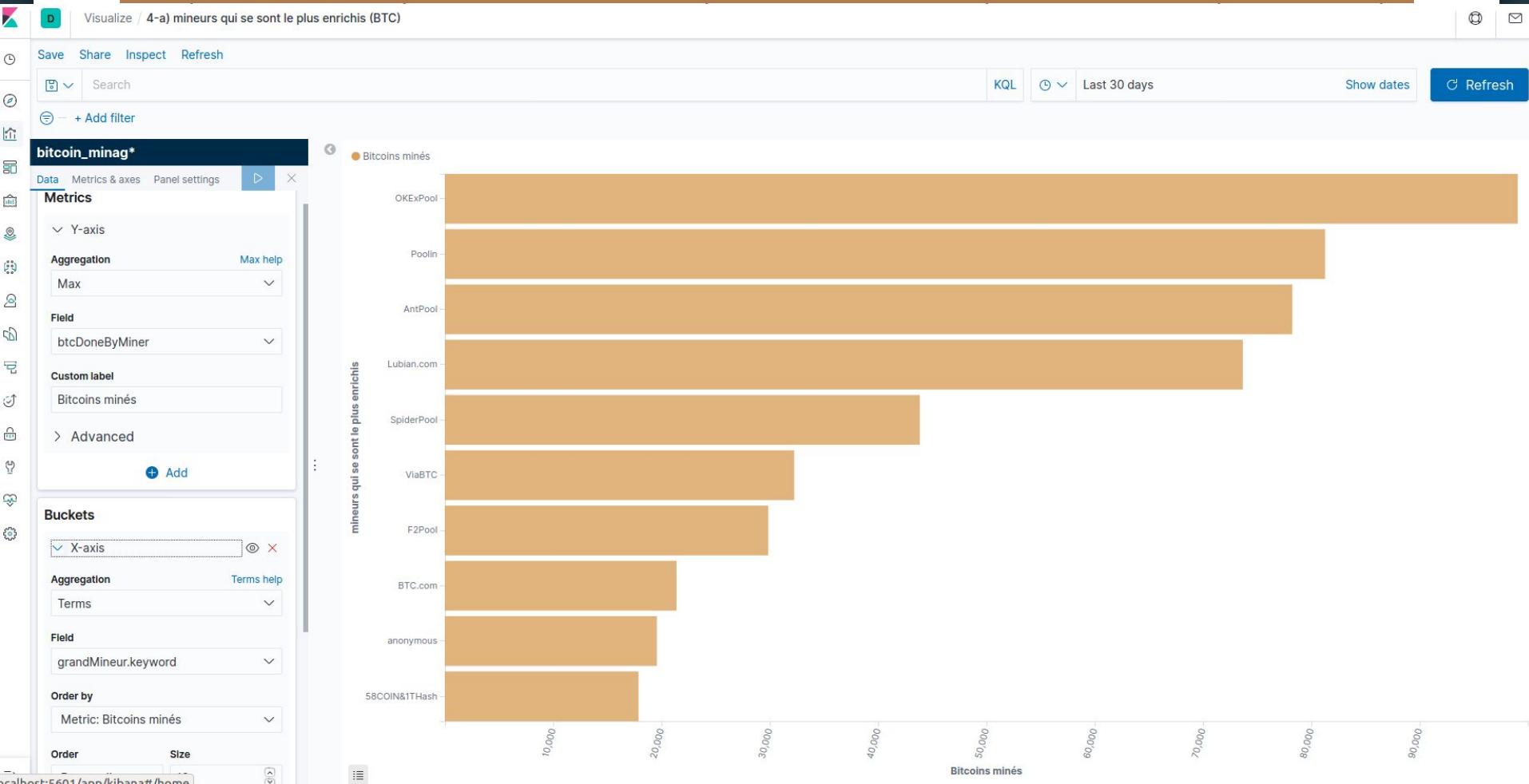
4-a)mineurs_qui_se_sont_le_plus_enrichis_par_jour_(en_BTC):



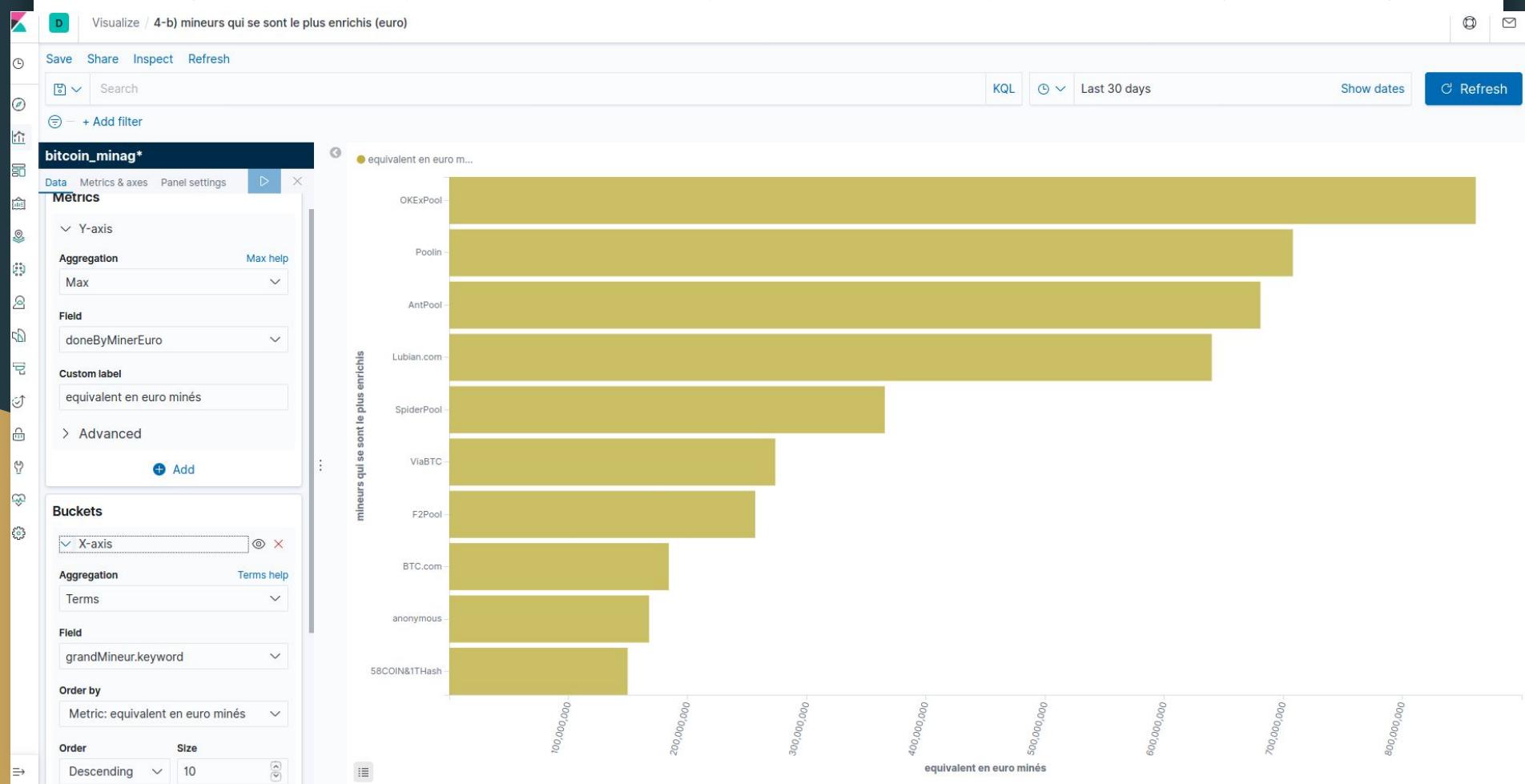
4-b)mineurs qui se sont le plus enrichis par jour (en euro) :



4-c)mineurs_qui_se_sont_le_plus_enrichis_par_mois_(en_BTC) :



4-d)mineurs qui se sont le plus enrichis par mois (en euro):



3. Métriques demandées:

1	Cours BTC (€)		OK
2	volume BTC échangés / h	(BTC)	OK
		(€)	OK
3	valeur maximale des transactions réalisées / heure	(BTC)	OK
		(€)	OK
4	mineurs qui se sont le plus enrichis	/ jour	(BTC)
			(€)
	/ mois	(BTC)	OK
		(€)	OK

4- scénarios: Kafka

Cluster	Scénario	Exigences	Solutions proposées
Kafka	Panne d'une ou plusieurs machines (serveur kafka)	Pas de single point of failure	répliquer les données de manière redondantes sur différentes serveurs
	MAJ nécessitant redémarrage des machines du cluster	Pas d'interruption de service	avoir plusieurs machines et les redémarrer une par une
	Augmentation quantité de données	Passer à l'échelle de données élevées massives	distribuer les tâches sur plusieurs serveurs Kafka
Zookeeper	Erreur-plantage Zookeeper⇒ s'éteint brusquement	Tolérant au fail-fast	Utiliser des outils de supervision (supervisor) pour : redémarrage automatique

4- scénarios: Kafka

Pour ajouter un serveur dans un cluster Kafka:

- Serveur Kafka communique par zookeeper --> ajouter une ou plusieurs machine dans Zookeeper
- Définir le “**broker.id**” (int) des nouveaux server à partir de leur fichiers de configuration “[config/server.properties](#)”. Cet identifiant doit être unique pour chaque serveur.
- Lancer le nouveau serveur Kafka

Répliquer les données:

- L'option **--replication-factor** permet d'augmenter le taux de réPLICATION d'un topic, ici :
`$./bin/kafka-topics.sh --create --zookeeper localhost:2181 --replication-factor 2 --partitions 1 --topic bitcoin-price`
- Si taux de réPLICATION est de **N**, l'architecture permettra de SUPPORTER la panne de **N-1** serveurs.

Cluster	Scénario	Exigences	Solutions proposées
Storm	échec d'un traitement de tuples	Pas de perte de message	Gestion des erreurs: réémettre les tuples dont le traitement a échoué
	un worker s'éteint	Pas de single point of failure	Supervisor le redémarre et si répétitif, nimbus ré-affecte ses tâches à un autre worker
	Redémarrage de bolts ou de la topologie	Pas de pertes de données	stocker les tuples en cours de traitement dans une base de données à laquelle pourront accéder tous les bolts
	Augmentation quantité de données	Passer à l'échelle du Big data	distribuer les tâches d'un bolt sur plusieurs --> tuple grouping +  nb de executors
Zookeeper ou Storm	Zookeeper ou nimbus s'éteint	Pas de single point of failure	<ul style="list-style-type: none"> - Supervisor les redémarre - Nimbus arrêté : worker continuent traitement des tâches mais impossible de réaffecter les tâches des workers ou de soumettre de nouvelles topologies - mettre second nimbus en backup

4- scénarios: Storm

Pour re-émettre les tuples dont le traitement a échoué :

- Tracer le spout ayant émis le tuple:
 - en ajoutant un **identifiant aux tuples** émis par les spout ==> Ceci se fait à l'appel de la méthode **emit()** dans le spout
- Les bolts indiquent le succès ou l'échec du traitement de chaque tuple à l'aide des méthodes **ack()** et **fail()**.
- La méthode **fail()** du spout ayant émis un tuple sera appelée dès que le tuple ou l'un de ses descendants sera en échec.

```
public class KafkaSpout<K, V > extends BaseRichSpout {  
    ...  
    @Override  
    public void fail(Object msgId) {  
        Values tuple = (Values)msgId;  
  
        // En cas d'erreur, on ré-émet le tuple lui-même  
        outputCollector.emit(tuple, msgId);  
    }  
}
```

Parallélisations des tâches:

- augmenter le nb d'executors et optimiser le paramètre **parallelism_hint** aux méthodes **setBolt()** et **setSpout()**:

```
// Sauvegarde du current price dans elasticSearch  
builder.setBolt("savingES-currencyPrice", new EsBolt("current_price/devises", conf), 1)  
    .shuffleGrouping("parseDeviseBolt");
```
- Optimiser les tuples grouping: shuffle, fields

4- scénarios: ElasticSearch

Cluster	Scénario	Exigences	Solutions proposées
Elasticsearch	Panne d'une ou plusieurs machines du un cluster ES	résistance au pannes	réplica shards sur différents noeuds
	MAJ nécessitant redémarrage des machines du cluster	Pas d'interruption de service	avoir plusieurs machines et les redémarrer une par une
	Augmentation quantité de données	Passer à l'échelle de données élevées massives	répartir les charges sur des serveurs ES par les shards: $nb_noeuds = nb_shards_prim$
Zookeeper	Erreur-plantage Zookeeper⇒ s'éteint brusquement	Tolérant au fail-fast	Gérer zookeeper sous supervisior (supervisord): redemarrage automatique

5. Conclusion

- On a vu le cours du bitcoin fluctue avec des tendances imprévisible: pendant 24 h
- Beaucoup de transactions observés avec des montants très élevé
- Mieux vaut observer les pendant au minimum un an avant de songer à investir
- Traitement en temps réel: suivant le flux de données entrants, nécessite une optimisation des machines utilisées

Question ?

4. Storm topology

Spout:
current-prices

JSON

Bolt:
Parser-price

updated:String
euro: double
usd; double
gbp: double

Bolt:
save-Price

Bolt:
filter-transa
ctions

timestamp:Long
hash: String
somme; double

date:String
transactionsMax: double
montantTotalTransx: double
tupleCount: Int

Bolt:
save-transa
ctions

Spout:
operations

JSON

Bolt:
Parser-
op

op: String
JSON

Bolt:
filter-block

timestamp:Long
hash: String
montantTotalBTC: double
mineur: String

Bolt:
highest-min
eur

date:String
grandMineur:String
sommeMinée; double
Bolt:
save-min
eur



2.a) Architecture fonctionnelle

