DOROSENCO Nadejda Nadya

AIT IDIR Adem

ROQUE Louis Kinsley

18/07/2023

Analyse de la bourse avec Elasticsearch, Kibana et Logstash

Table des matières

I. Description de l'API

Nous avons d'abord étudié plusieurs API :

- **TMDb (The Movie Database) API** : TMDb est une base de données de films et de séries télévisées. L'API de TMDb permet d'accéder à ces données pour les utiliser dans leurs propres applications. Les fonctionnalités de l'API comprennent l'accès à des données sur les films, les séries télévisées, les acteurs, les réalisateurs, les critiques, les images, les vidéos et d'autres métadonnées.

- **CoinMarketCap API** : CoinMarketCap est l'un des principaux sites web pour le suivi des cAPItalisations boursières des cryptomonnaies, des volumes de trading et des prix. L'API de CoinMarketCap offre un accès aux données du site, sur de la cryptomonnaie, les données historiques, les informations sur les marchés d'échange de cryptomonnaies, et bien plus encore. Nous pouvons obtenir des données sur plus de 2000 cryptomonnaies différentes, y compris leur prix actuel, leur cAPItalisation boursière, leur volume de trading, leur offre et d'autres informations pertinentes pour les traders et les investisseurs en cryptomonnaies.

- **Trefle.io API** : Trefle.io est une API de botanique qui offre un accès à une grande quantité de données sur des plantes. C'est un service RESTful qui renvoie des données au format JSON, ce qui le rend facile à intégrer dans divers types d'applications.

Nous avons décidé de choisir l'API CoinMarketCAP car nous trouvons que l'API offre beaucoup d'analyse intéressent. Nous avons donc décidé d'étudier l'évolution de la cryptomonnaie. Nous verrons qu'elle cryptomonnaie est la plus dominante ou ceux qui on le plus de fluctuations.

Voici quelque information sur les champs de notre API :

- id : Un identifiant unique pour la cryptomonnaie.
- name : Le nom de la cryptomonnaie.
- symbol : Le symbole de la cryptomonnaie.
- slug : Un identifiant en texte lisible pour la cryptomonnaie.
- num_market_pairs : Le nombre de paires de marché disponibles pour la cryptomonnaie.
- date_added : La date à laquelle la cryptomonnaie a été ajoutée à CoinMarketCap.
- tags : Une liste de tags associés à la cryptomonnaie.
- max_supply : L'offre maximale de la cryptomonnaie.
- circulating_supply : L'offre actuellement en circulation de la cryptomonnaie.
- total_supply : L'offre totale de la cryptomonnaie.
- platform : La plateforme sur laquelle la cryptomonnaie a été créée (si applicable).

- cmc_rank : Le rang de la cryptomonnaie sur CoinMarketCap.
- last_updated : La dernière fois que les données de la cryptomonnaie ont été mises à jour.
- quote : Les données de cotation pour la cryptomonnaie, qui comprennent les champs suivants :
- price : Le prix actuel de la cryptomonnaie.
- volume_24h : Le volume de trading de la cryptomonnaie au cours des dernières 24 heures.
- percent_change_1h : Le changement de prix de la cryptomonnaie au cours de la dernière heure.
- percent_change_24h : Le changement de prix de la cryptomonnaie au cours des dernières 24 heures.
- percent_change_7d : Le changement de prix de la cryptomonnaie au cours des 7 derniers jours.
- market_cap : La cAPItalisation boursière de la cryptomonnaie.
- last_updated : La dernière fois que les données de cotation ont été mises à jour.

Pour obtenir l'API il faut se rendre sur le site coinmarketcap.com (vous recevez un mail une fois créer votre compte et pour l'obtention de l'API)

https://coinmarketcap.com/API/

Pour les informations sur le traitement/ la documentation de l'API :

https://coinmarketcap.com/API/documentation/v1/#operation/getV1CryptocurrencyListingsHistorical

II. Installation de logstash

Tous d'abord nous avons dû installer Logstash en plus de Kibana et Elasticsearch que nous avons vu en cours pour cela nous avons intégré dans notre fichier docker-compose :

```
logstash:
  container_name: ls01
  image: docker.elastic.co/logstash/logstash:8.7.1
  environment:
    - ELASTICSEARCH_HOST=http://es01:9200
  networks:
    - elastic
  depends_on:
    - elasticsearch
  volumes:
    - //c/Users/louis/Desktop/Elasticsearch/logstash/pipeline:/usr/share/logstash/pipeline
```

Figure 1: script docker-compose

Nous vous listons ci-dessous à quoi sert les différentes lignes pour les installations

1. **container_name:** ls01 C'est le nom que nous avons donnez au conteneur Docker qui sera créé à partir de ce service.

2. **image:** docker.elastic.co/logstash/logstash:8.7.1 Ceci indique l'image Docker qui sera utilisée pour créer le conteneur. Ici, on utilise l'image de Logstash version 8.7.1 qui est hébergée sur le dépôt docker.elastic.co. on tiens a garder une cohérence avec Kibana et Elasticsearch

   Lien pour récupérer les diffirent version logstash :
   https://www.docker.elastic.co/r/logstash

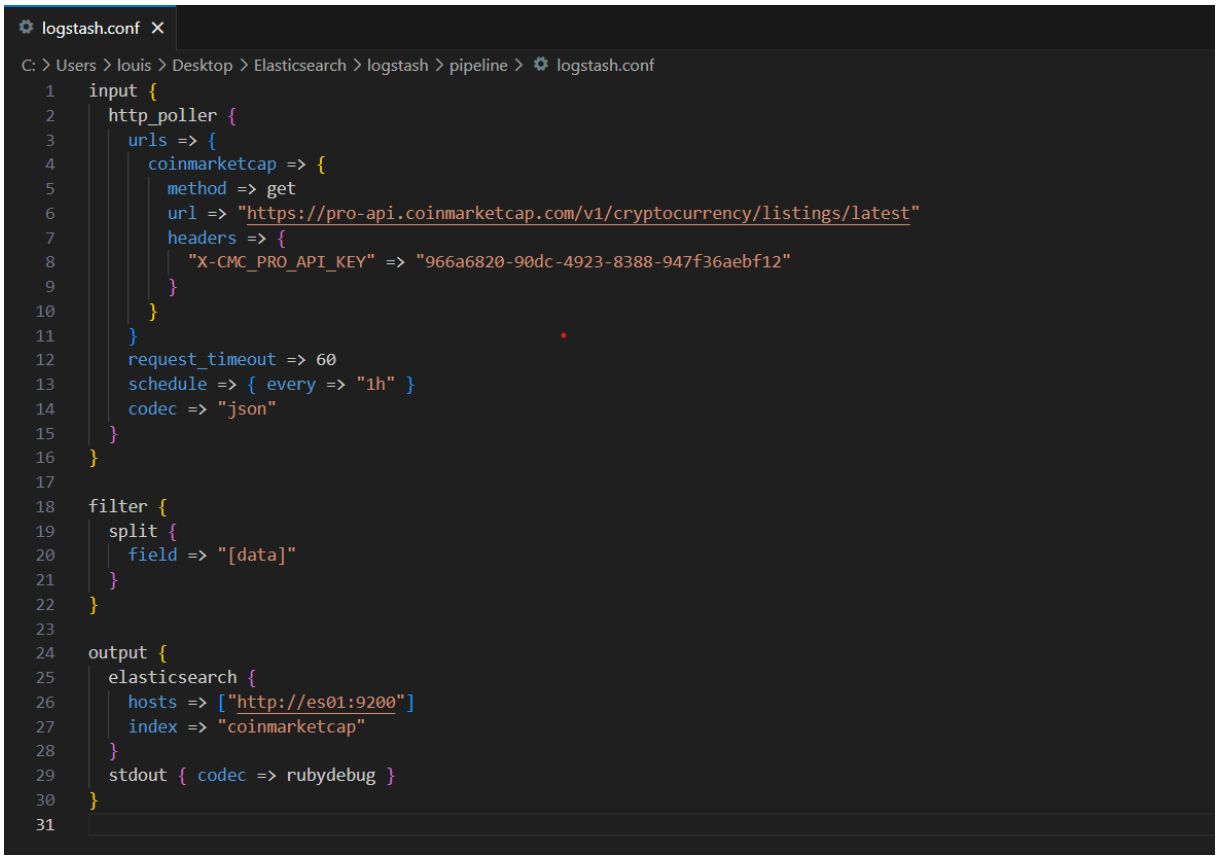
   lien pour les autres composant(kibana etc) :
   https://www.docker.elastic.co/

3. **environment:** Dans cette section, on a définis les variables d'environnement pour le conteneur. Ici, on définit l'adresse du serveur Elasticsearch pour que Logstash sache où envoyer les données.
   - ELASTICSEARCH_HOST=http://es01:9200 indique que le serveur Elasticsearch est accessible via le réseau interne Docker à l'adresse es01 sur le port 9200.
4. **networks :** C'est là que on définis les réseaux auxquels le conteneur doit se connecter. Ici, le conteneur doit se connecter au réseau "elastic".
5. **depends_on:** Cette option permet de définir l'ordre de démarrage des conteneurs. Ici, le conteneur "logstash" ne démarrera pas tant que le conteneur "elasticsearch" ne sera pas en cours d'exécution.
6. **volumes :** Les volumes sont utilisés pour partager des fichiers entre l'hôte et le conteneur. Cela signifie que Logstash aura accès aux fichiers de pipeline situés sur notre bureau dans le répertoire "Elasticsearch/logstash/pipeline". Les pipelines de Logstash sont des fichiers de configuration qui définissent comment les données doivent être traitées. la partie volume n'est pas nécessaire mais nous avons voulus l'ajouter pour voir ce que ça va produire

III. Importation des données

L'importation des données ce fait à l'aide de notre script logstash.conf nous l'avons mis dans un dossier pipeline exemple de chemin :
C:\Users\louis\Desktop\Elasticsearch\logstash\pipeline

Voici le code ci-dessous



1. **Input**: Cette section définit où récupérer les données. On a utilisé le plugin http_poller pour envoyer une requête HTTP GET à l'URL de l'API CoinMarketCap.. La requête est envoyée toutes les heures (comme spécifié par schedule => { every => "1h" }) et on récupère le tous au format JSON (comme spécifié par codec => "json").

2. **Filter**: Cette section est utilisée pour transformer les donnée en ajoutant un filtre. Ici, on utilise le filtre split pour diviser le champ data qui est une liste de cryptomonnaies renvoyée par l'API CoinMarketCap. Après l'opération de split, chaque élément de la liste data est traité comme un événement distinct. Nous pouvons ajouter d'autres filtre dans cette partie tels que le top 10 des prix de la cryptomonnaie les plus hauts ou ce commence par la lettre « B ».

3. **Output**: Cette section définit où les données sont envoyées. Dans ce cas, les données sont envoyées à une instance d'Elasticsearch (http://es01:9200). Les données sont stockées dans l'index coinmarketcap qui est un nom que nous avons choisi. De plus, nous avons décidé que les données sont également affichées à la console (STDOUT) pour le débogage, ce qui est spécifié par stdout { codec => rubydebug }.

Pour résumé notre script, ce fichier de configuration de Logstash récupère les données de l'API CoinMarketCap toute les heures, divise la liste de cryptomonnaies pour quel soit en événements individuels, et envoie ces événements à Elasticsearch et à la console pour stockage et débogage pour pouvoir faire l'analyse de donnée.

Pour aller plus loin nous avons aussi vue comment mettre plusieurs API différent voir annexe.

IV. Vérification des données

Nous allons vérifier que tous marche bien en premier lieu nous avons vérifié l'état de notre cluster.

- **GET /_cluster/health** : Cette requête renvoie des informations sur l'état du cluster Elasticsearch, y compris le nombre de nœuds et de shards, si le cluster est actuellement en train de réallouer des shards, etc.


```
{
  "cluster_name": "docker-cluster",
  "status": "yellow",
  "timed_out": false,
  "number_of_nodes": 1,
  "number_of_data_nodes": 1,
  "active_primary_shards": 20,
  "active_shards": 20,
  "relocating_shards": 0,
  "initializing_shards": 0,
  "unassigned_shards": 14,
  "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": 58.82352941176471
}
```


- **GET /_cat/nodes?v** : Cette requête renvoie des informations sur les nœuds dans le cluster Elasticsearch, y compris l'adresse IP, le nom du nœud, le rôle, le statut, la quantité de CPU utilisée, la mémoire utilisée, etc.

```
ip         heap.percent ram.percent cpu load_1m load_5m load_15m node.role   master name
172.18.0.2           40          98   0    0.01    0.14     0.14 cdfhilmrstw *      ea0f9207da06
```

- **GET /_cat/indices?v** : Cette requête renvoie des informations sur les indices dans le cluster Elasticsearch, y compris le nom de l'indice, le nombre de documents, la taille de l'indice, l'état, etc. nous remarquons que nous avons bien notre base coinmarketcap

```
1 health status index           uuid                   pri rep docs.count docs.deleted store.size pri.store.size
2 yellow open   coinmarketcap   JqDZ9UQwRryspEXrbuYWLg   1   1       2501            0      138mb          138mb
3 yellow open   new_reviews     O_U38n_JTcCkz00k9eU75A   1   1     145191            0     71.2mb         71.2mb
4 yellow open   new_voitures    oQKgC2WXTMOP9hOhiHxxaQ   1   1      50849            0      3.4mb          3.4mb
5 yellow open   magasin3        CRRRAXEMQeC44LBr1d5aEg   2   2          2            0       11kb           11kb
6 yellow open   magasin4        1CJZg_roSv2t8mY1X13V1Q   2   2          0            0       450b           450b
7 yellow open   weather         cvEpwnJOSgSIOq6gyL1pPQ   1   1         38            0     85.3kb         85.3kb
8 yellow open   voitures        FJtGAWEPT8WG8JULLFy5Wg   1   1      50849            0      3.7mb          3.7mb
9 yellow open   nouveau_voitures EigpFSx-R3O1z8Xkbki7hA   1   1      50849            0      3.7mb          3.7mb
```

- **GET /_cat/shards?v** : Cette requête renvoie des informations sur les shards dans le cluster Elasticsearch, y compris l'indice auquel ils appartiennent, l'état du shard, le nœud sur lequel le shard est situé, etc.

```
index                                                    shard prirep state        docs   store ip         node
magasin4                                                 0     p      STARTED         0    225b 172.18.0.2 ea0f9207da06
magasin4                                                 0     r      UNASSIGNED
magasin4                                                 0     r      UNASSIGNED
magasin4                                                 1     p      STARTED         0    225b 172.18.0.2 ea0f9207da06
magasin4                                                 1     r      UNASSIGNED
magasin4                                                 1     r      UNASSIGNED
new_voitures                                             0     p      STARTED     50849   3.4mb 172.18.0.2 ea0f9207da06
new_voitures                                             0     r      UNASSIGNED
magasin3                                                 0     p      STARTED         2  10.8kb 172.18.0.2 ea0f9207da06
magasin3                                                 0     r      UNASSIGNED
magasin3                                                 0     r      UNASSIGNED
magasin3                                                 1     p      STARTED         0    225b 172.18.0.2 ea0f9207da06
magasin3                                                 1     r      UNASSIGNED
magasin3                                                 1     r      UNASSIGNED
.apm-custom-link                                         0     p      STARTED         0    225b 172.18.0.2 ea0f9207da06
.kibana_8.7.1_001                                        0     p      STARTED       797   2.6mb 172.18.0.2 ea0f9207da06
.async-search                                            0     p      STARTED         0   9.6kb 172.18.0.2 ea0f9207da06
nouveau_voitures                                         0     p      STARTED     50849   3.7mb 172.18.0.2 ea0f9207da06
nouveau_voitures                                         0     r      UNASSIGNED
voitures                                                 0     p      STARTED     50849   3.7mb 172.18.0.2 ea0f9207da06
voitures                                                 0     r      UNASSIGNED
.kibana-event-log-8.7.1-000001                           0     p      STARTED       193   103kb 172.18.0.2 ea0f9207da06
.apm-source-map                                          0     p      STARTED         0    225b 172.18.0.2 ea0f9207da06
new_reviews                                              0     p      STARTED    145191  71.2mb 172.18.0.2 ea0f9207da06
new_reviews                                              0     r      UNASSIGNED
.ds-.logs-deprecation.elasticsearch-default-2023.06.18-000001 0     p      STARTED        86 130.3kb 172.18.0.2 ea0f9207da06
.ds-ilm-history-5-2023.06.18-000001                      0     p      STARTED         9  20.1kb 172.18.0.2 ea0f9207da06
weather                                                  0     p      STARTED        38  85.3kb 172.18.0.2 ea0f9207da06
weather                                                  0     r      UNASSIGNED
.apm-agent-configuration                                 0     p      STARTED         0    225b 172.18.0.2 ea0f9207da06
.tasks                                                   0     p      STARTED         1   7.9kb 172.18.0.2 ea0f9207da06
coinmarketcap                                            0     p      STARTED      2501   138mb 172.18.0.2 ea0f9207da06
coinmarketcap                                            0     r      UNASSIGNED
.kibana_task_manager_8.7.1_001                           0     p      STARTED        24 306.1kb 172.18.0.2 ea0f9207da06
```

Puis nous passons sur l'analyse de notre base de données pour voir si il n'y a pas eu de soucie lors de l'importation

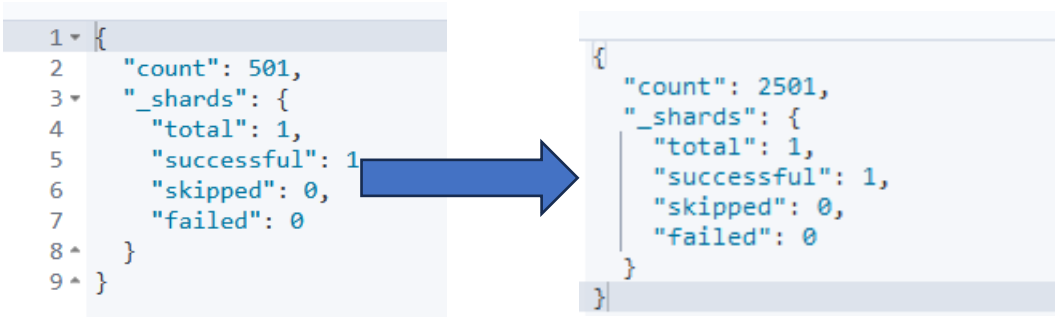
- **GET /coinmarketcap/_search** : Cette requête renvoie les documents de l'index coinmarketcap qui correspondent aux critères de recherche par défaut (c'est-à-dire tous les documents, car aucun critère spécifique n'est donné).


```
{
  "took": 11,
  "timed_out": false,
  "_shards": {
    "total": 1,
    "successful": 1,
    "skipped": 0,
    "failed": 0
  },
  "hits": {
    "total": {
      "value": 501,
      "relation": "eq"
    },
    "max_score": 1,
    "hits": [
      {
        "_index": "coinmarketcap",
        "_id": "yFjARYkBjapCpJ4rv8Vz",
        "_score": 1,
        "_ignored": [
          "event.original.keyword"
        ],
        "_source": {
          "@timestamp": "2023-07-11T16:20:02.312651764Z",
          "status": {
            "total_count": 10261,
            "elapsed": 178,
            "error_message": null,
            "error_code": 0,
            "notice": null,
            "credit_count": 1,
            "timestamp": "2023-07-11T16:20:03.789Z"
          },
          "@version": "1",
```

- **GET /coinmarketcap/_mapping** : Cette requête renvoie les définitions de mapping pour l'index coinmarketcap. Les mappings définissent comment les documents et leurs champs sont stockés et indexés. On peut voir par exemple que le cmc_rank est de type long

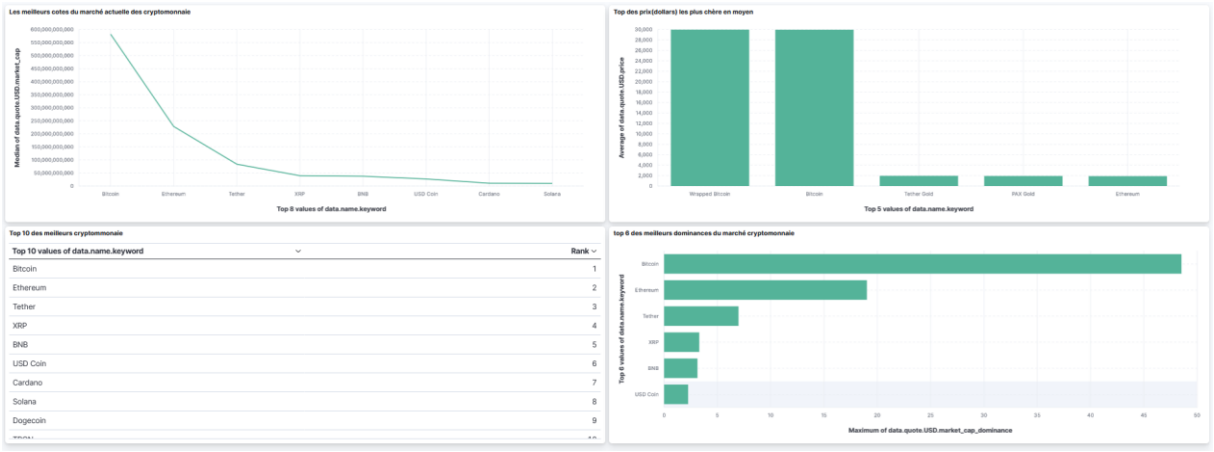

```
{
  "coinmarketcap": {
    "mappings": {
      "properties": {
        "@timestamp": {
          "type": "date"
        },
        "@version": {
          "type": "text",
          "fields": {
            "keyword": {
              "type": "keyword",
              "ignore_above": 256
            }
          }
        },
        "data": {
          "properties": {
            "circulating_supply": {
              "type": "long"
            },
            "cmc_rank": {
              "type": "long"
            },
            "date_added": {
              "type": "date"
            },
            "id": {
              "type": "long"
            },
            "infinite_supply": {
              "type": "boolean"
            },
            "last_updated": {
              "type": "date"
```

- **GET /coinmarketcap/_count** : Cette requête renvoie le nombre total de documents dans l'index coinmarketcap. Cette requête changera au fur et à mesure du temps car plus les jours passe plus on a de données

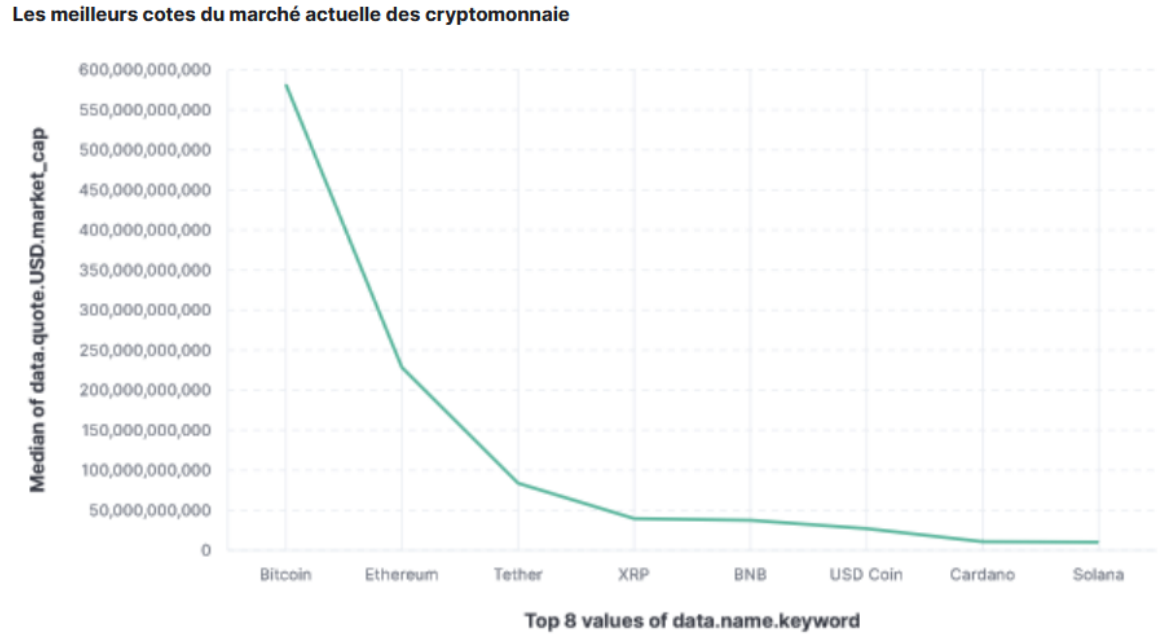


V. Réalisation des Dashboard

1. Nous commençons par répondre à la problématique quelle cryptomonnaie est la plus présente dans le marché avec la plus grosse capitalisation boursière, son rang, sa dominance et son prix



Pour voir plus claire on détaille tous les graphiques

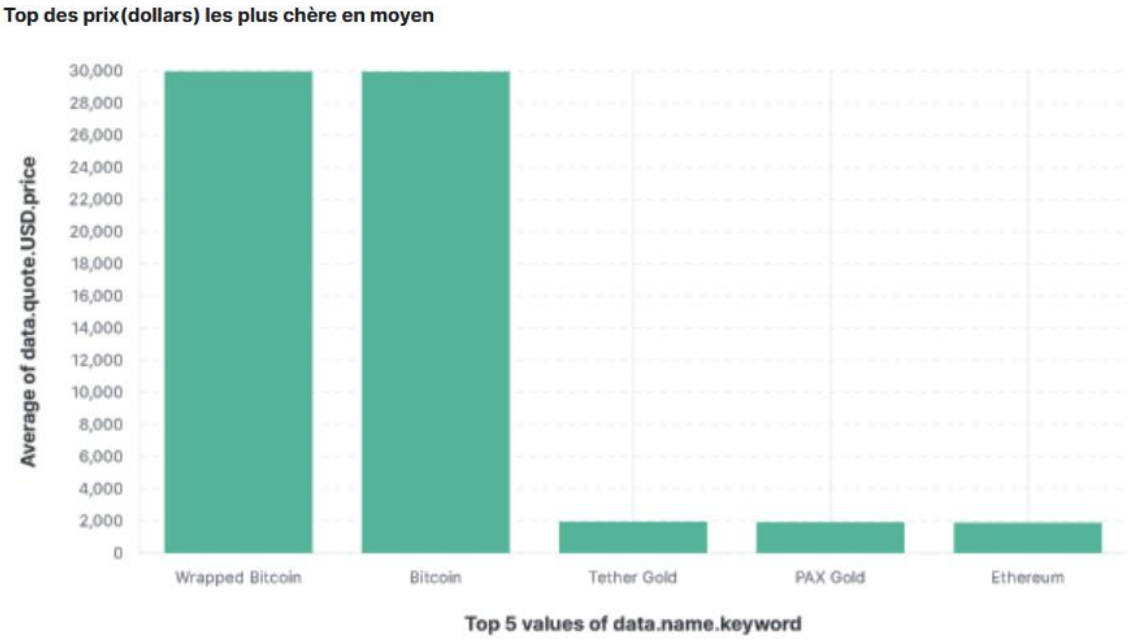


Le champ data.quote.USD.market_cap dans l'API CoinMarketCap représente la capitalisation boursière de la cryptomonnaie en dollars américains (USD).

Voici une définition de la capitalisation boursière :

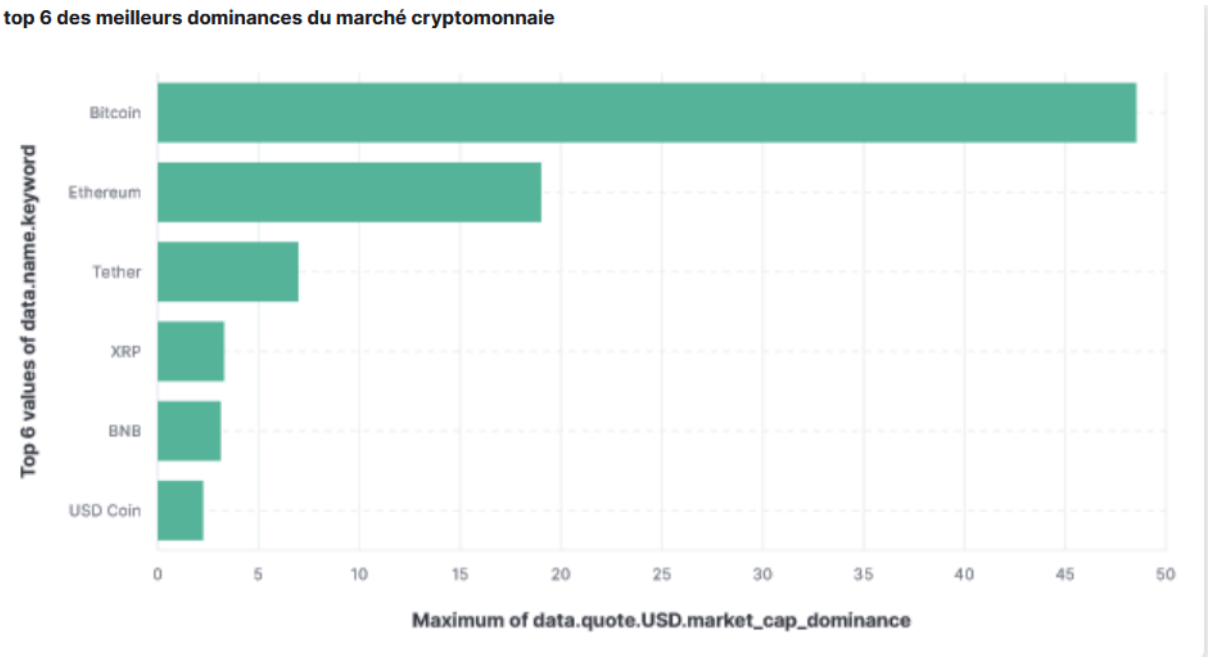
La capitalisation boursière est une mesure de la valeur totale d'une cryptomonnaie. Elle est calculée en multipliant le prix actuel de la cryptomonnaie par le nombre total de pièces en circulation. Par exemple, si une cryptomonnaie a 1 000 000 de pièces en circulation et que le prix actuel de chaque pièce est de 1 $, la capitalisation boursière de cette cryptomonnaie serait de 1 000 000 $.

Nous pouvons voir que la capitalisation boursière la plus élevée est celle du Bitcoin puis de l'Ethereum et ainsi que du Tether. De plus on remarque qu'il y a une grosse différence entre le 1er bitcoin et le 2 Ethereum. Nous remarquons qu'il y a que 3 cryptomonnaie qui dépassent les 100 000 000 000 en capitalisation boursière par exemple le Bitcoin est à 600 000 000 000 alors que le Solana le dernier est à environ 49 000 000 000



Dans cet histogramme nous pouvons voir que le Bitcoin est toujours le numéro 1 sur le prix en moyen, 1 bitcoin vaut 30 000 dollars, cependant on voit que l'Ethereum malgré sa capitalisation boursière élever ne vaut qu'en moyen 4000 dollars on voie que le Tether est plus élevé que l'Ethereum alors que cela est l'inverse dans le premier graphique. Nous apercevons aussi le PAX Gold qui n'était pas présent avant.

Avec les 2 graphiques nous pouvons dire que par exemple il y plus d'Ethereum que de Tether mais le Tether aura un prix plus élevé



On peut voir que celui avec la plus grosse dominance est ceux avec la meilleure capitalisation boursière ce qui veut dire que c'est une variable très importante nous retrouvons donc le Bitcoin, l'Ethereum, Tether, XRP, BNB, USD Coin

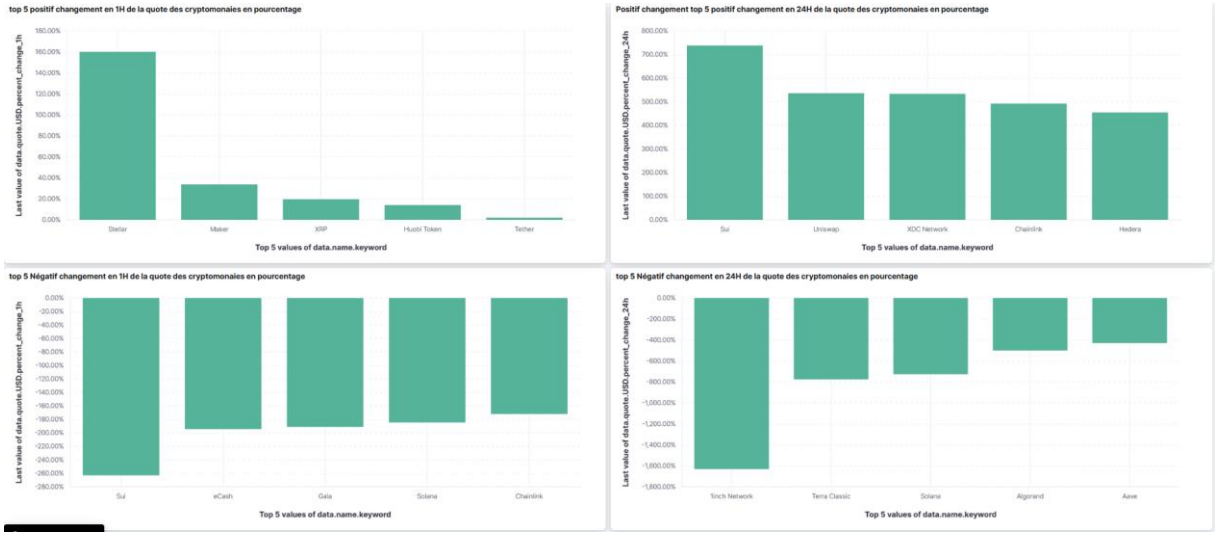
Top 10 des meilleurs cryptommonaie

Top 10 values of data.name.keyword	Rank
Bitcoin	1
Ethereum	2
Tether	3
XRP	4
BNB	5
USD Coin	6
Cardano	7
Solana	8
Dogecoin	9
TRON	10

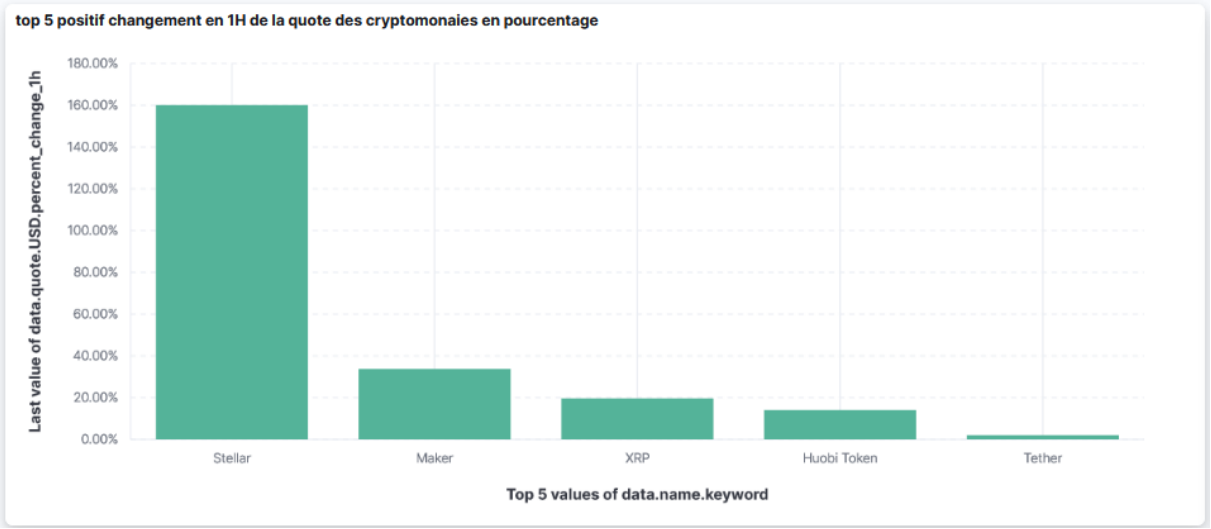
Nous avons donc le rang de chaque cryptomonnaie nous avons prix le top 10 et cela confirme que la capitalisation boursière joue un gros rôle en effet l'Ethereum aura un meilleur rang car une meilleure capitalisation boursière que le Tether. Certes le Tether coute plus chère mais il y en a moins en circulation.

On voit aussi que le Bitcoin est la cryptomonnaie la plus dominante et celle qui circule le plus.

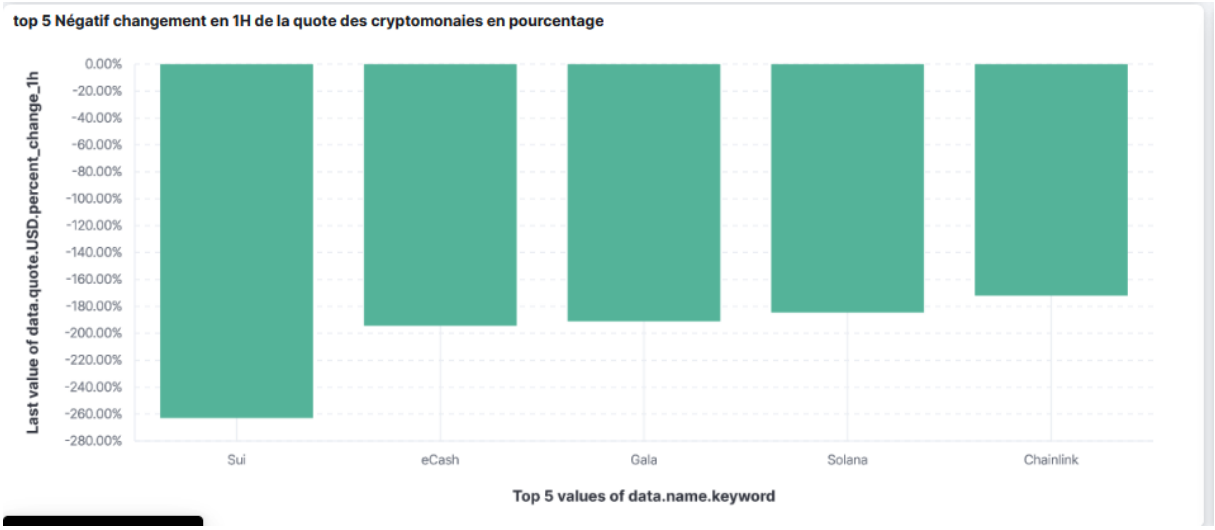
1. Nous avons aussi fait un Dashboard sur les cryptomonnaies avec le plus de changement en fonction du temps pour répondre à la problématique quelle cryptomonnaie a le plus de fluctuation sur le temps



Comme précédemment ont réparti les différents graphiques pour les expliquer en détaille

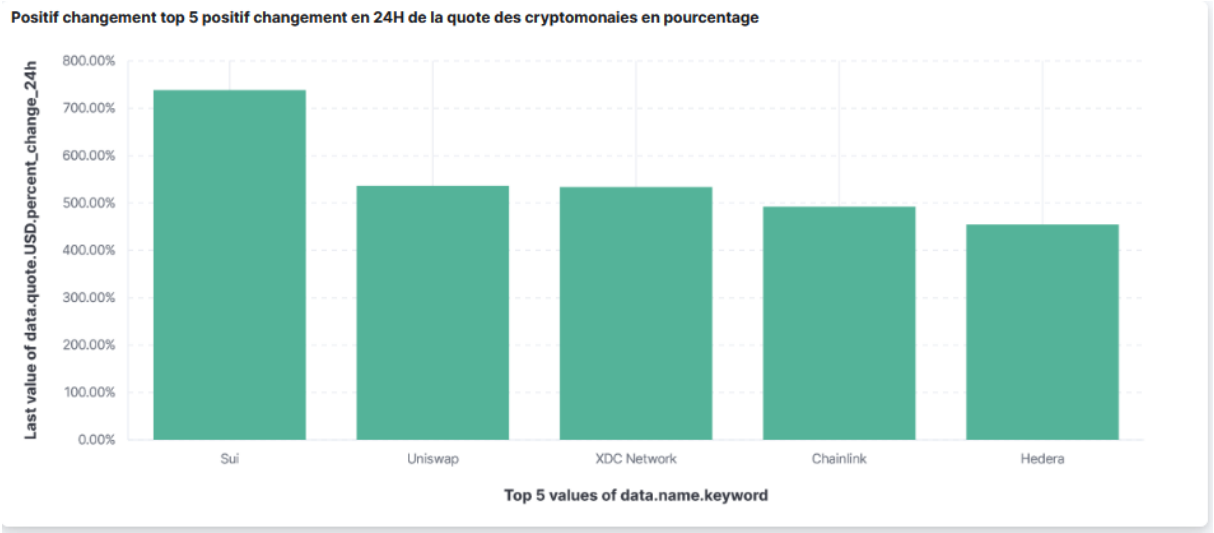


On peut voir que le Stellar a eu une fluctuation positive donc le cours est monté de 160% en 1H. Nous remarquons que le Maker quant à lui à augmenter de 35% en 1H. Cela montre une grosse différence entre le Maker et le Stellar qui a eu un pic à l'heure actuelle

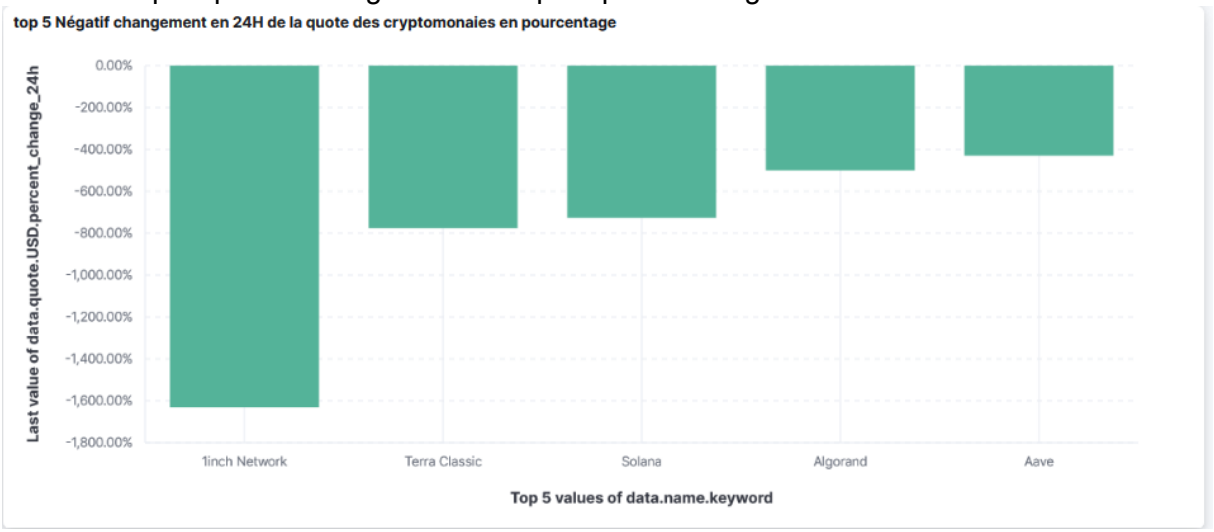


Attention pour comprendre ce calcule, il faut savoir que pour comprendre comment une telle valeur a été obtenue, on pourrait envisager une interprétation théorique. Supposons que la valeur "précédente" soit considérée comme une valeur minimale atteinte précédemment plutôt que le prix précédent directement. Par exemple, si une cryptomonnaie a chuté à un creux de 100 $ à un certain moment, puis a augmenté à 360 $, mais est maintenant retombée à 100 $, cela pourrait être interprété comme une baisse de 260% par rapport au pic de 360 $.

Ici on peut voir que le Sui a eu une fluctuation négative donc le cours est monté descendu de 260% en 1H. Nous remarquons que le Chainlink quant à lui à diminuer de 160% en 1H. On peut voir qu'il y a eu un grand crash à ce moment-là de c'est différentes cryptomonnaies qu'elle on prix plus de valeurs est d'un coup elle ont chuté.



On peut voir que dans ce graphique cela confirme notre théorie en effet le Sui qui a eu un gros crash est remonté d'un coup en 24H ce qui peut arriver quand la cryptomonnaie a de grosse fluctuation en effet vu qu'elle a eu un prix au plus bas beaucoup ont dû investir dessus ce qui a permis d'augmenter son prix qui a été augmenter de 700% en 24 H



On peut voir ici des gros crashes dus à un prix qui est monté trop haut au cours de la journée surtout le 1inch Network

VI. Conclusion

La cryptomonnaie évolue tout le temps on peut voir qu'il y a 2 façons d'investir.

A faible revenue vaux mieux viser ceux avec le plus de fluctuations qui auront plus de chances d'augmenter leur prix mais il y a un risque de gros crash. On peut exemple aussi prendre une cryptomonnaie qui subit une grosse chute comme le Sui qui a eu des moments de fluctuation élever et qui a diminué d'un coup en 1H puis en 24H est remontée alors que Solana lui a continué à diminuer malgré des augmentations.

Si on a un plus gros revenu le meilleur serait d'investir dans des cryptomonnaies plus chères avec des cours élever cependant cela prendra plus de temps a monté car moins de fluctuation mais générera plus de revenue. En effet le Bitcoin n'est pas dans le top 5 des fluctuations.

VII. Annexe

Code pour supprimer des bases :
curl -XDELETE http://localhost:9200/ « Nom de la base »

Code pour mettre plusieurs API (Exemple avec la base TDMB):

```
INPUT {
  HTTP_POLLER {
    URLS => {
      NOW_PLAYING =>
"HTTPS://API.THEMOVIEDB.ORG/3/MOVIE/NOW_PLAYING?API_KEY=FD2EE8B3ACE8046B4D2C3B46ADBF19A1"
      TOP_RATED =>
"HTTPS://API.THEMOVIEDB.ORG/3/MOVIE/TOP_RATED?API_KEY=FD2EE8B3ACE8046B4D2C3B46ADBF19A1"
      UPCOMING =>
"HTTPS://API.THEMOVIEDB.ORG/3/MOVIE/UPCOMING?API_KEY=FD2EE8B3ACE8046B4D2C3B46ADBF19A1"
    }
    REQUEST_TIMEOUT => 60
    SCHEDULE => { CRON => "* * * * * UTC"}
    CODEC => "JSON"
  }
}

FILTER {
  JSON {
    SOURCE => "MESSAGE"
  }
}

OUTPUT {
  IF [HTTP_POLLER_METADATA][NAME] == "NOW_PLAYING" {
    ELASTICSEARCH {
      HOSTS => ["ES01:9200"]
      INDEX => "NOWPLAYING"
    }
  }
  IF [HTTP_POLLER_METADATA][NAME] == "TOP_RATED" {
    ELASTICSEARCH {
      HOSTS => ["ES01:9200"]
      INDEX => "TOPRATED"
    }
  }
  IF [HTTP_POLLER_METADATA][NAME] == "UPCOMING" {
    ELASTICSEARCH {
      HOSTS => ["ES01:9200"]
      INDEX => "UPCOMING"
    }
  }
}
```