**Elastic Stack**

**Partie I : Installation d’Elasticsearch, Kibana et Logstash**

1. **Installation ElasticSearch**

wget -qO - https://artifacts.elastic.co/GPG-KEY-elasticsearch | sudo apt-key add -

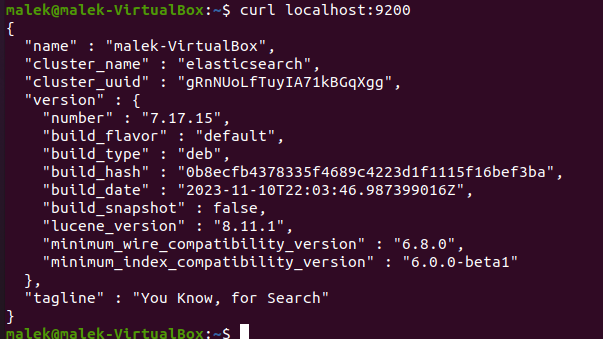
sudo apt-get install apt-transport-https

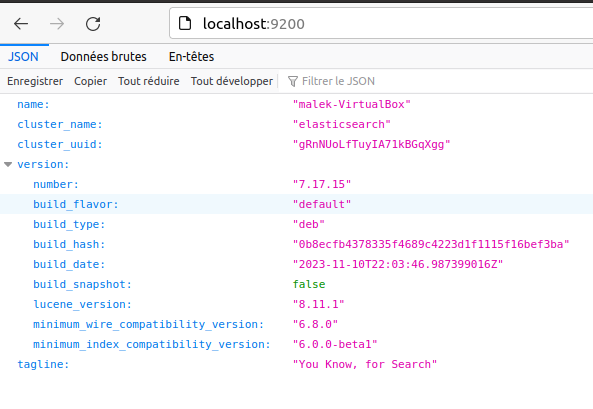
echo "deb https://artifacts.elastic.co/packages/7.x/apt stable main" | sudo tee -a /etc/apt/sources.list.d/elastic-7.x.list

sudo apt-get update -y && sudo apt-get install elasticsearch

sudo systemctl start elasticsearch

curl localhost:9200

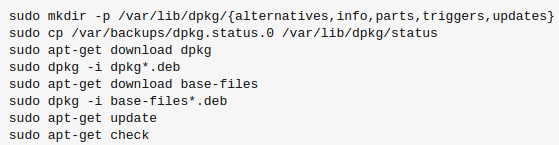




sudo systemctl enable elasticsearch

1. **Installation Kibana**

En cas de problème lock frontend

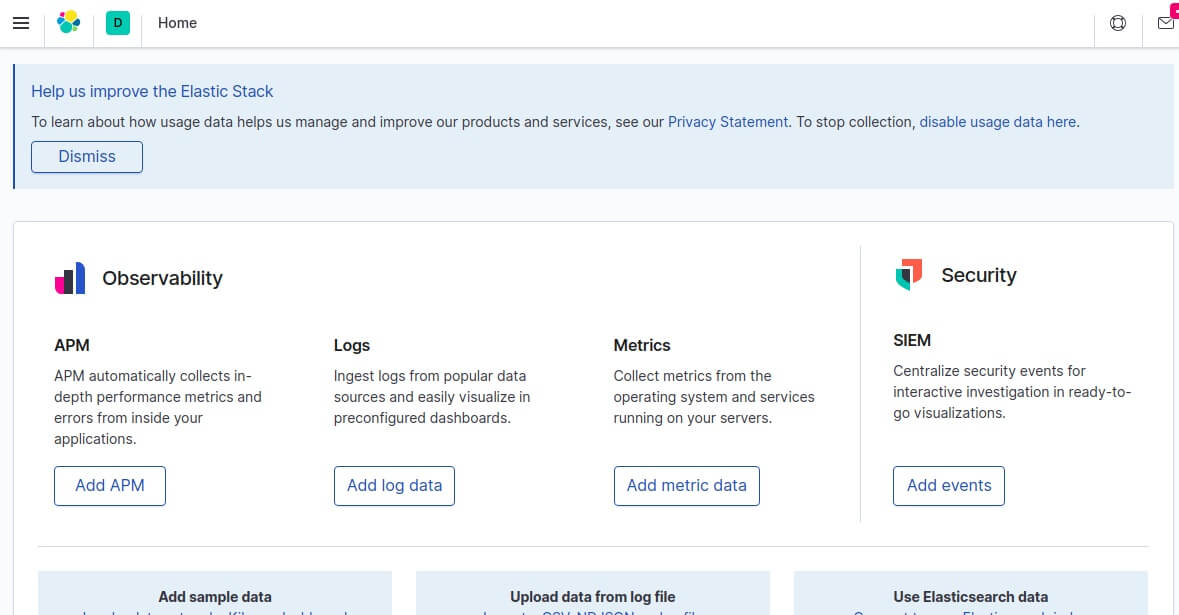


sudo apt-get install kibana

sudo systemctl start kibana

sudo systemctl enable kibana

<http://localhost:5601>



1. **Installation Logstach**

sudo apt-get install default-jre

java -version

sudo apt-get install logstash

sudo systemctl start logstash

sudo systemctl enable logstash

**Partie II : installation apache et configuration des Logs**

1. Installer apache

sudo apt-get install -y apache2

sudo systemctl start apache2

sudo usermod -aG adm logstash

1. création du fichier apache.conf sous */etc/logstash/conf.d/*

input {

file {

path => "/var/log/apache2/access.log"

start\_position => "beginning"

sincedb\_path => "/dev/null"

}

}

filter {

grok {

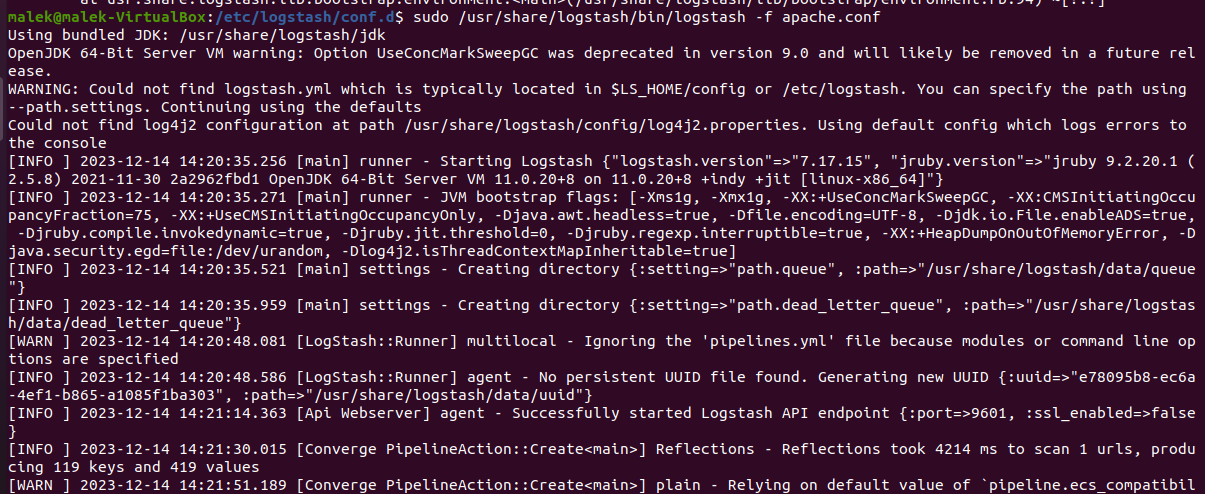
match => { "message" => "%{IP:client} %{WORD:method} %{URIPATHPARAM:request} %{NUMBER:bytes} %{NUMBER:duration}" }

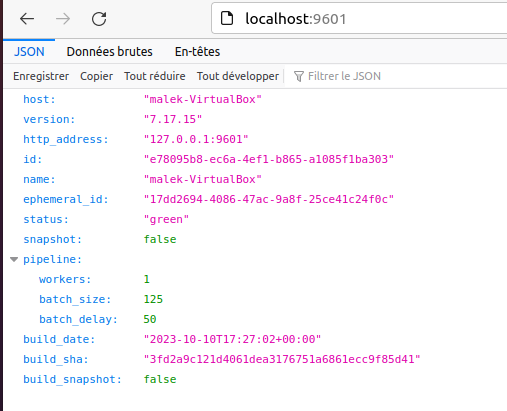
}

}

output {}

1. Exécution





1. Modification du contenu de apache.conf

input {

file { path => "/var/log/apache2/access.log" }

}

filter {

grok {

match => { "message" => "%{COMBINEDAPACHELOG}" }

}

date {

match => [ "timestamp" , "dd/MMM/yyyy:HH:mm:ss Z" ]

}

mutate {

convert => {

"response" => "integer"

"bytes" => "integer"

}

}

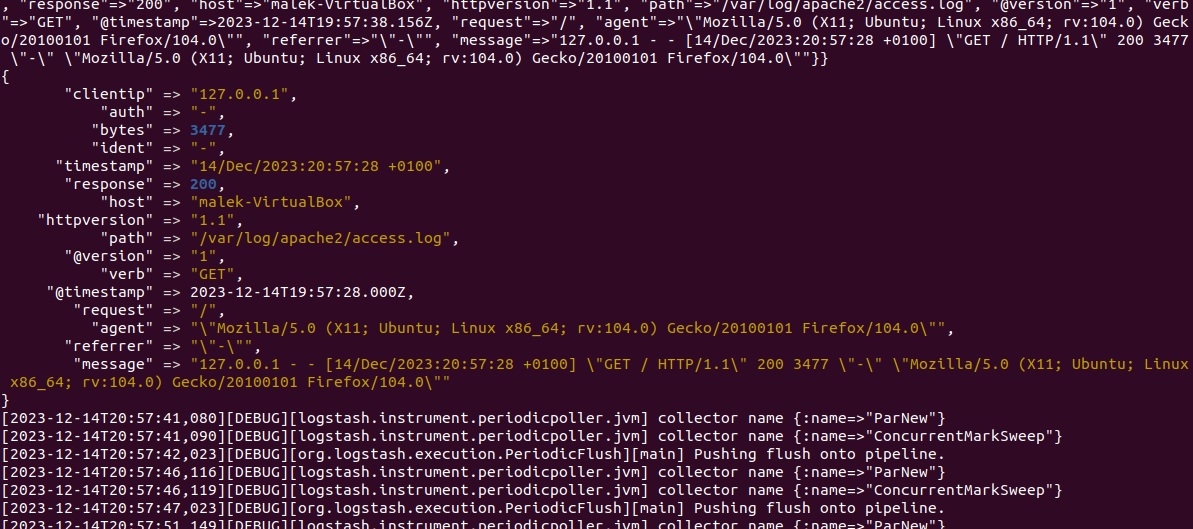
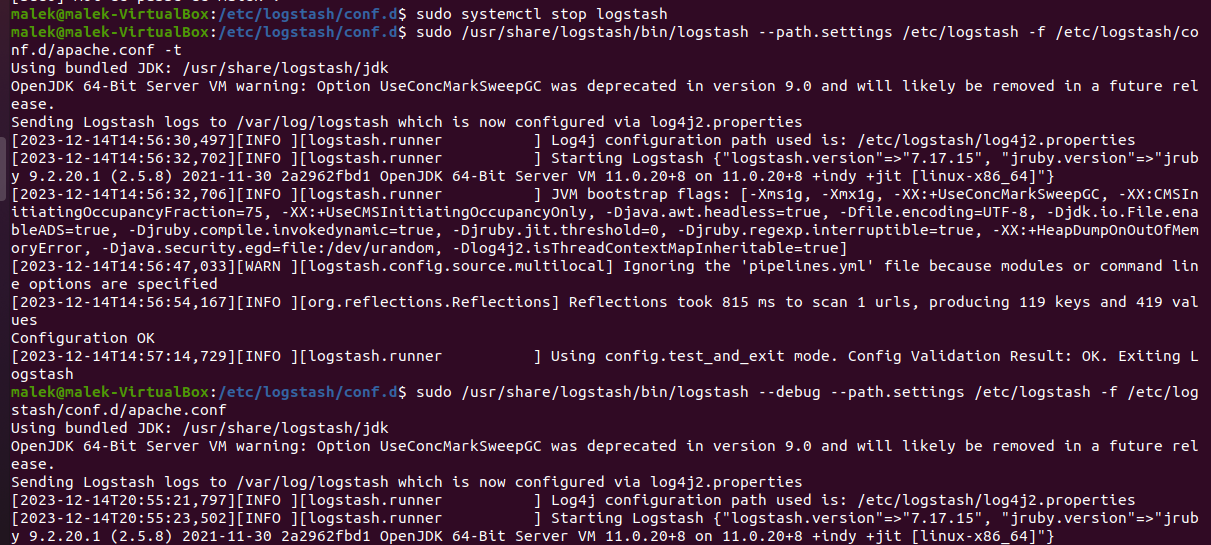
}

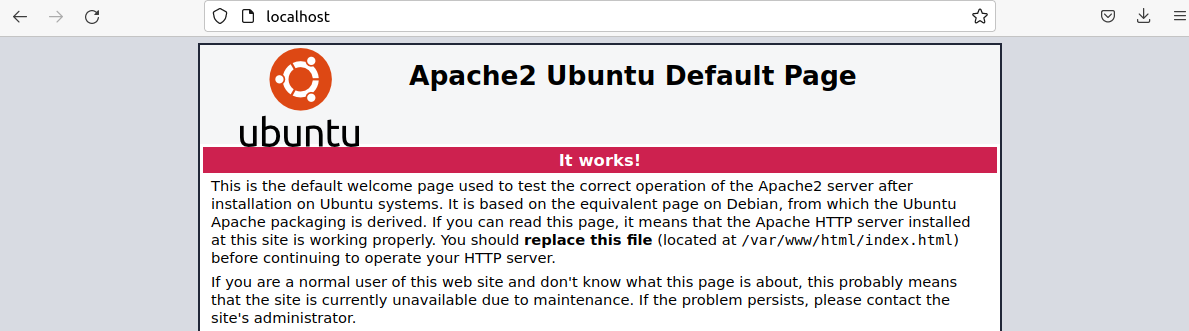
output {

stdout { codec => rubydebug }

}

1. exécution





Actualiser localhost et revoir le Terminal

1. Modifions ensuite notre fichier *apache.conf* pour communiquer avec elasticsearch :

input {

file { path => "/var/log/apache2/access.log" }

}

filter {

grok {

match => { "message" => "%{COMBINEDAPACHELOG}" }

}

date {

match => [ "timestamp", "dd/MMM/yyyy:HH:mm:ss Z" ]

}

mutate {

convert => {

"response" => "integer"

"bytes" => "integer"

}

}

}

output {

elasticsearch {

hosts => "localhost:9200"

index => "apache-%{+YYYY.MM.dd}"

}

}

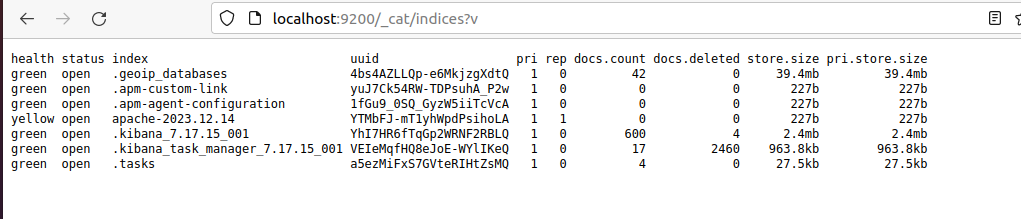
Vous remarquerez que j'utilise un index dans le bloc de code de sortie **elasticsearch** car c'est le moyen pour elasticsearch de structurer ses données et de les gérer plus rapidement. Le nommage utilisé dans notre index actuel est dynamique grâce au pattern **%{+YYYY.MM.dd}**, ce qui aura pour effet de créer un index par jour, permettant à l'administrateur de facilement s'y retrouver en affichant par exemple que les logs d'une date précise dans Kibana, supprimer que les indexs d'une plage de dates , etc ...

N'oubliez pas de démarrer votre service logstash avec la commande suivante :

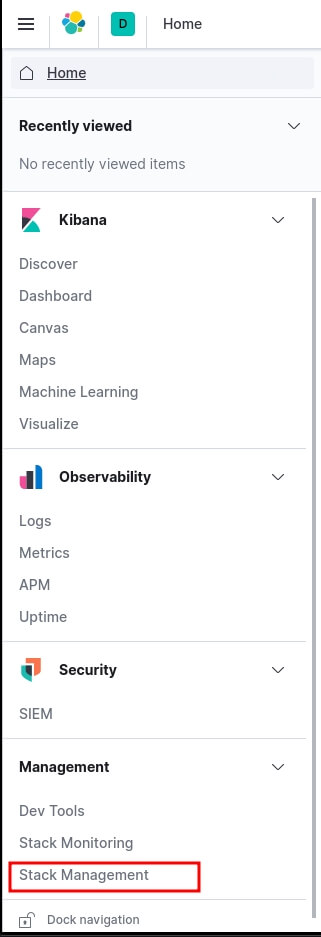
sudo systemctl start logstash

Après démarrage de votre service, logstash va lancer votre pipeline. Ensuite vous pouvez vérifier si l'index s'est correctement créé, pour cela appelez l'API REST fournie par elasticsearch, comme suit :

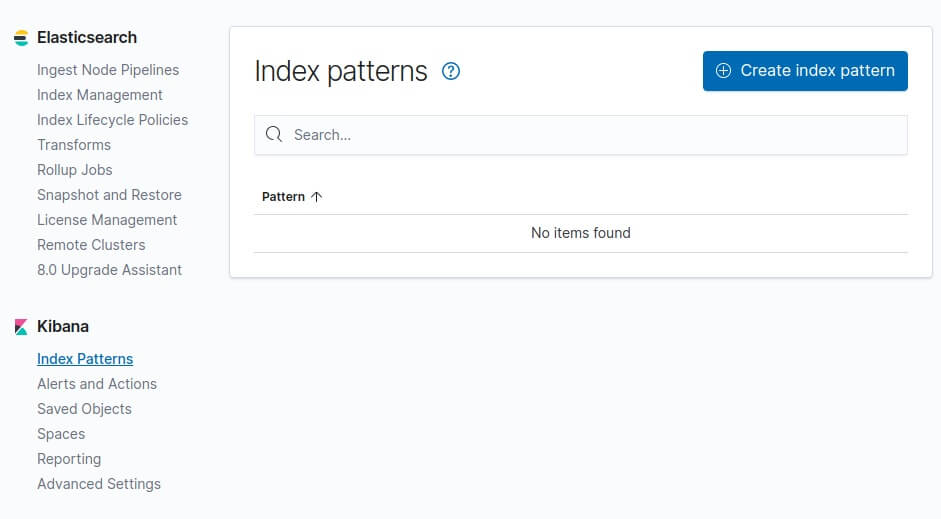
curl "localhost:9200/\_cat/indices?v"



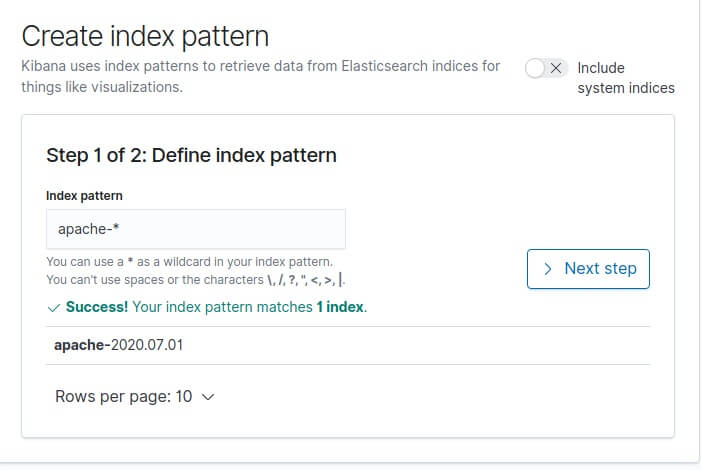
Exemple : "response : 404" pour n'afficher que les requêtes en erreur 404.



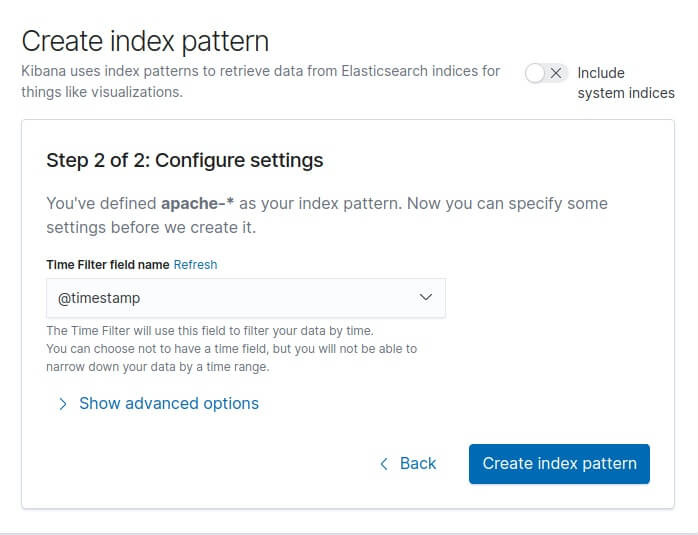
Par la suite, vous devez **ajouter un pattern index dans kibana** afin de prendre en considération vos indexs quotidiens Apache récupérés par elasticsearch. Cliquez sur "Index Patterns" sur le volet à gauche et cliquez sur le bouton "Create index pattern" :



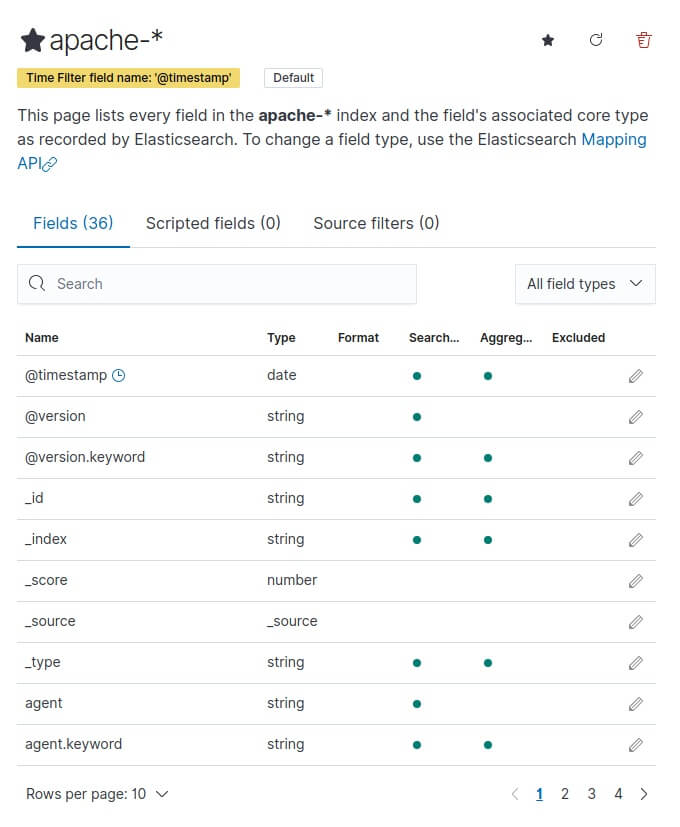
Dans notre cas le préfix de nos indexs apache est "apache-", donc le pattern index à créer dans kibana sera **apache-\*** :



Cliquez ensuite sur "Next Step". Ensuite il nous demande par quel moyen il doit gérer le filtre temporel (timestamp). Ce filtre est utile pour filtrer et affiner nos données par plage horaire. Nous avions prévu le coup sur notre configuration logstash en créant un champ **@timestamp**, et c'est celui là qu'on sélectionnera :

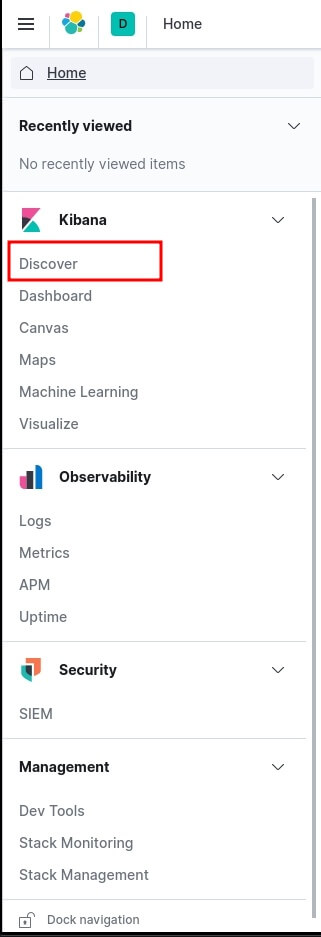


Enfin, cliquez sur le bouton "Create index pattern" et vous verrez apparaître tous vos champs :

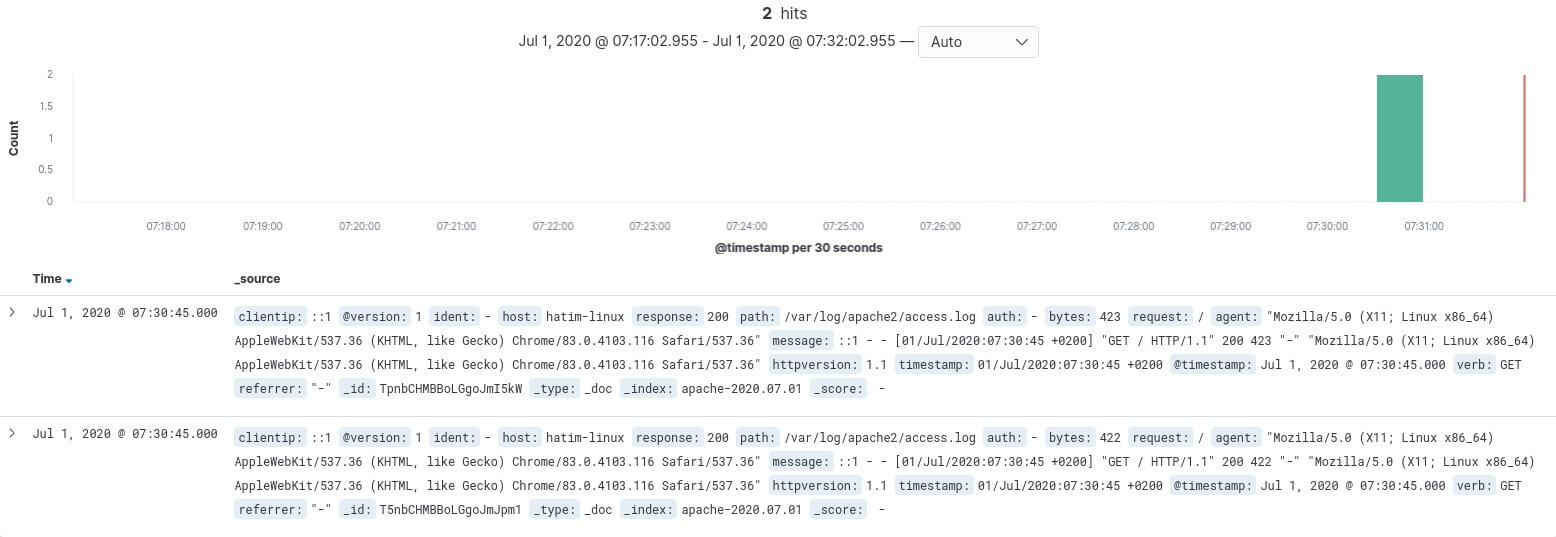
>

Découvrir vos logs

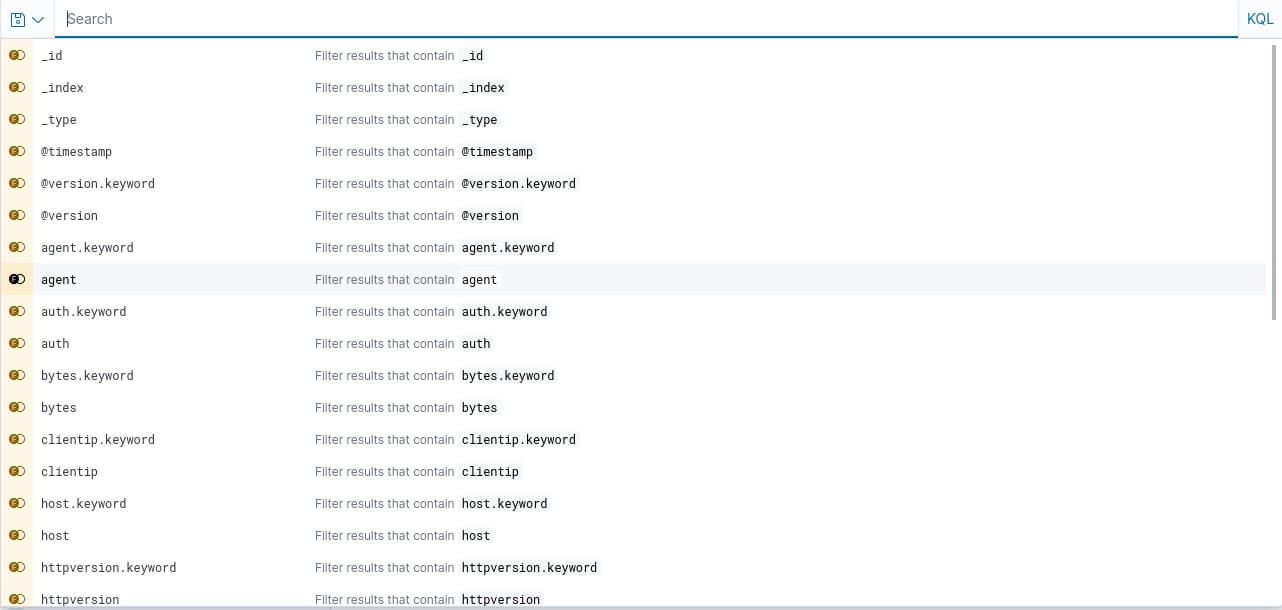
Pour découvrir vos logs, sur le menu à gauche cliquez sur Discover :



Ensuite, Faites quelques visites depuis votre navigateur sur la page d'accueil d'apache <http://localhost/> et revenez sur la page de discover :



Vous avez également la possibilité de filtrer vos logs par champ depuis la barre de recherche du Discover :



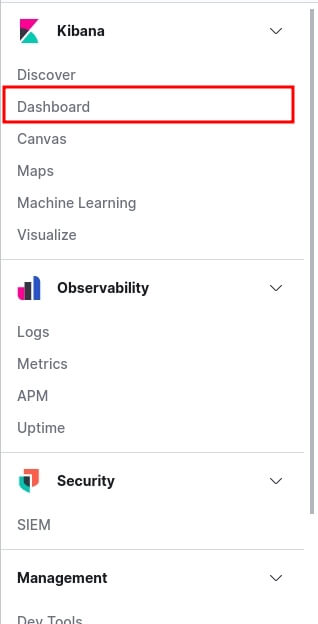
Exemple : "response : 404" pour n'afficher que les requêtes en erreur 404.

Dashboard

L'étape suivante est de **créer un tableau de bord afin de visualiser une collection de visualisations en temps réel**. Pour commencer, ouvrez le menu, accédez à Dashboard , puis cliquez sur "Create dashboard" :

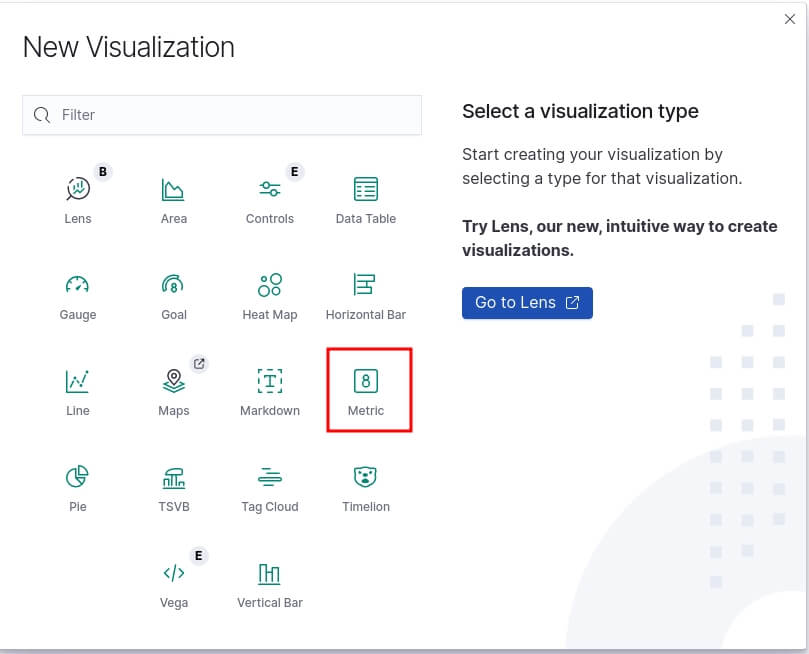
Dashboard

L'étape suivante est de **créer un tableau de bord afin de visualiser une collection de visualisations en temps réel**. Pour commencer, ouvrez le menu, accédez à Dashboard , puis cliquez sur "Create dashboard" :

>

Pour ajouter des éléments à votre dashboard vous devez **créer des visualisations Kibana** que vous pouvez déplacer et redimensionner dans votre dashboard. Vous pouvez ajouter des visualisations à partir de plusieurs indexs patterns et la même visualisation peut apparaître dans plusieurs tableaux de bord.

Dans notre exemple nous allons commencer par créer une visualisation qui permet d'afficher le nombre total d'utilisateurs unique. Pour ce faire, créez une visualisation en cliquant sur "Create new" et dans la fenêtre de "New visualisation" nous allons choisir le type de visualisation "Metric" :



Pour information, voici la **liste des types de visualisations Kibana les plus fréquemment utilisées** :

* **Line**, **area**, et **bar charts** : compare différentes métriques sur l'axe X et Y.
* **Pie chart** : graphique circulaire.
* **Data table** : données en format de tableau.
* **Metric** : affiche une seule métrique.
* **Goal and gauge** : affiche un nombre avec des indicateurs de progression.
* **Tag cloud** : affiche les mots dans un nuage, où la taille du mot correspond à son importance.

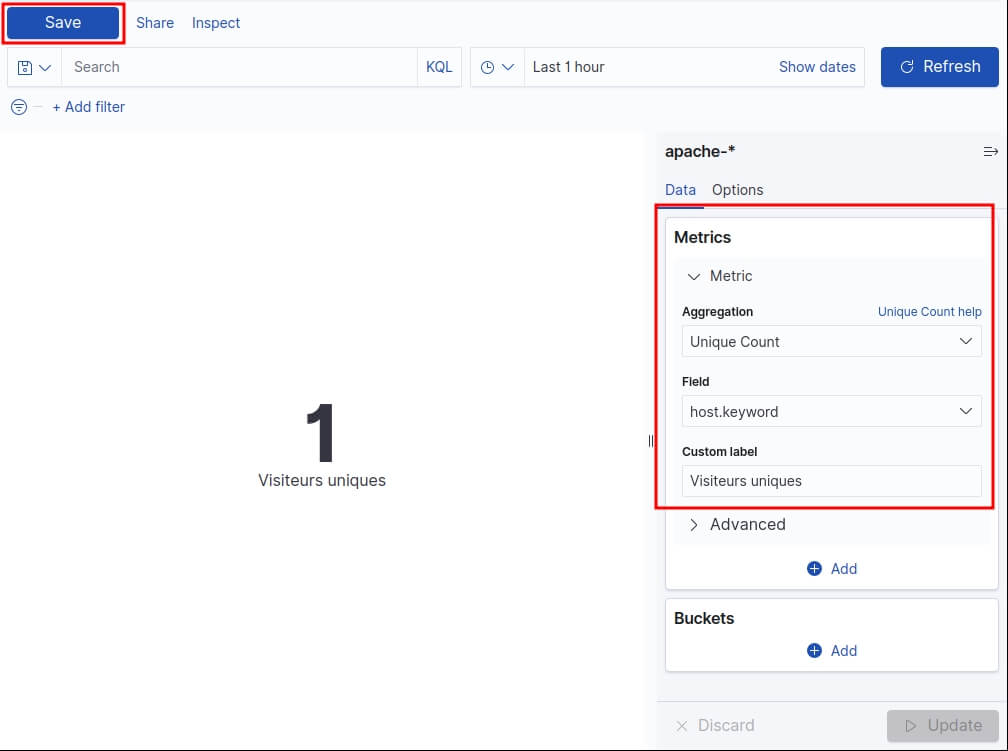
On vous demande ensuite de **paramétrer votre visualisation**, vous devez choisir d'abord votre agrégation qui correspond aux métriques extraient pour générer des valeurs de données. Voici les valeurs les plus communes :

* **Average** : valeur moyenne.
* **Count** : nombre total de documents correspondant à une requête.
* **Max** : la valeur la plus élevée.
* **Median** : médiane.
* **Min** : la valeur la plus basse.
* **Sum** : La valeur totale.
* **Unique Count** : nombre unique d'une métrique.

**Information**

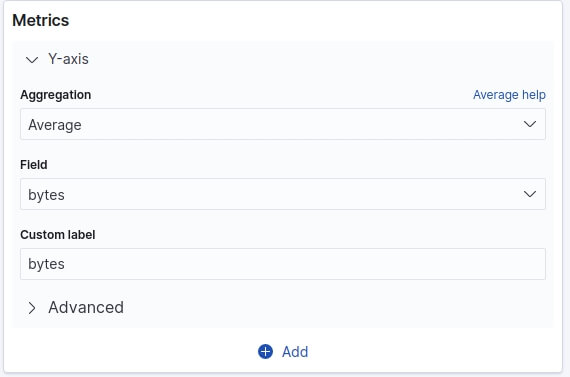
Dans le langage Elasticsearch, un document correspond aux données JSON sérialisées.

Dans notre cas, nous utiliserons l'agrégation **Unique Count** en utilisant le champ **host** :



Cliquez ensuite sur "Save" et choisissez le nom de votre visualisation.

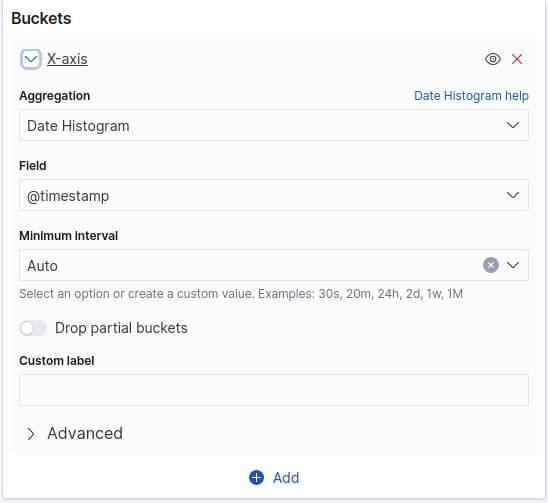
Rajoutons un autre graphique qui permet d'afficher la taille moyenne des requêtes temporellement. Pour ce type de besoin nous aurons besoin d'une visualisation de type "area". Pour l'axe Y nous allons utiliser une agrégation de type "Average" sur le champ **byte** :



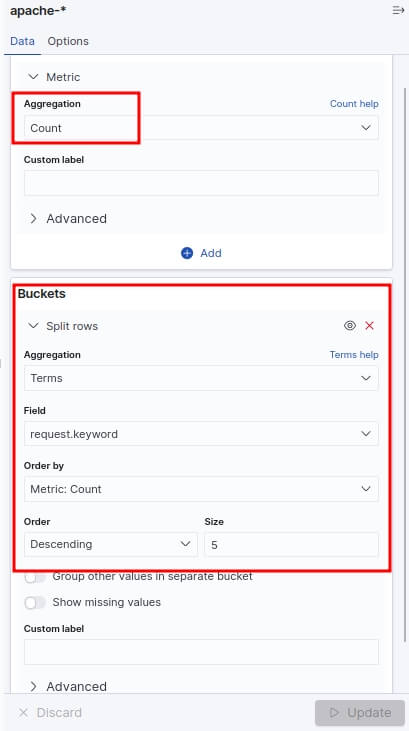
Pour l'axe X ça sera un peu différent car nous utiliserons les Bucket aggregations qui trient les documents en compartiments selon le contenu du document. Voici les valeurs les plus communes :

* **Date histogram** : fractionne un champ de date en compartiments par intervalle.
* **Date range** : valeurs comprises dans une plage de dates que vous spécifiez.
* **Filter** : filtre les données récupérées (ex : traiter que les erreurs 404).
* **IPv4 range** : plages d'adresses IPv4.
* **Range** : plages de valeurs pour un champ numérique.
* **Terms** : Spécifiez le nombre d'éléments supérieurs ou inférieurs d'un champ donné à afficher, classés par nombre ou par une métrique personnalisée.

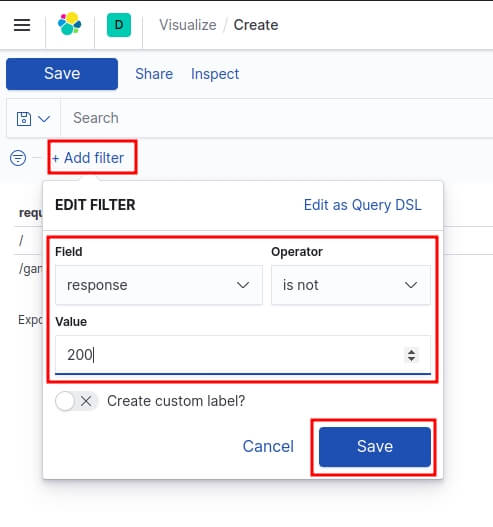
Pour notre cas nous utiliserons le type "Date histogram" :



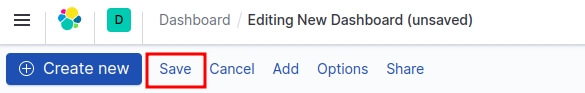
Un dernier graphique avant de clôturer cet article. Nous allons cette fois-ci afficher le top des requêtes en erreur sous forme d'un tableau. Créez une nouvelle visualisation de type "Data table" avec comme configuration une agrégation de type "Count" (par défaut) et une Bucket aggregation de type "Terms" sur le champ "request" et trier par ordre décroissant par l'agrégation "Count", ce qui nous affichera pour le moment que les pages web les plus visitées. Cette partie de la configuration ressemblera à ceci :



Ensuite pour récupérer que les requêtes en erreur, nous filtrerons ces requêtes si elles ont une réponse différente au code HTTP 200. Pour cela, vous devez cliquer sur le bouton situé en haut à gauche nommée "+ Add filter" et ajouter le filtre suivant :



Lorsque vous avez terminé d'ajouter et d'organiser les panneaux, enregistrez le tableau de bord. Dans la barre d'outils Kibana, cliquez sur "Save" et saisissez ensuite le titre du tableau de bord et la description facultative , puis enregistrez votre tableau de bord.



Le dashboard final ressemble à ceci :

