A dark blue vertical bar on the left side of the page, with a blue arrow pointing right from it, containing the date.

24/12/2023

# Projet Docker

A series of thin, dark blue, wavy lines on the left side of the page, resembling stylized grass or reeds.

MESBAHI Lydia & BENMECHICHE Khaled  
YNOV

## Objectif :

Mettre en place une stack applicative composée d'une application frontend WordPress et d'une base de données Maria DB, utilisant Docker et Docker Compose.

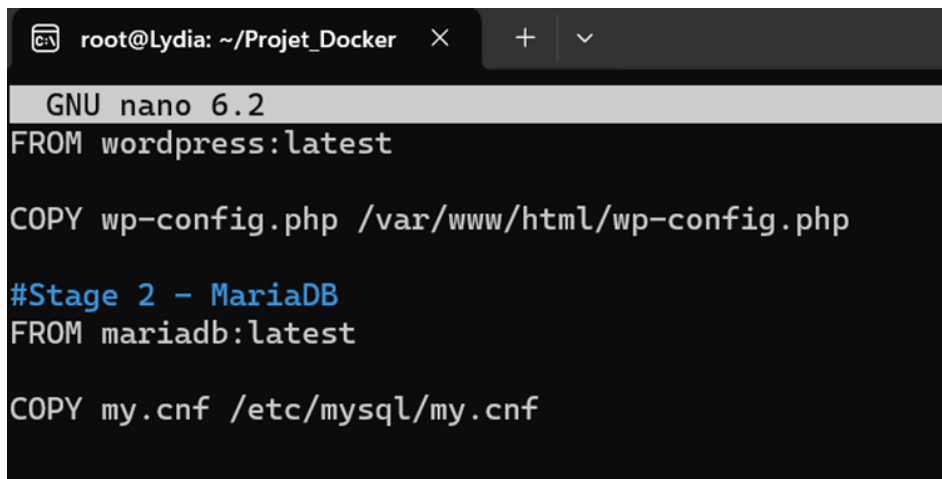
- o Les images de base de système d'exploitation optimisée que nous avons utilisé pour WordPress est : **wordpress:latest & mariadb:latest**

### Justification du choix des images :

Nous avons choisi l'image officielle de WordPress optimisée et qui est basée sur php et Apache.

1. **Prêt à l'emploi** : Ces images sont préconfigurées pour fonctionner immédiatement avec WordPress et Maria DB sans nécessiter une configuration complexe supplémentaire.
2. **Maintenance simplifiée** : Elles sont maintenues par la communauté Docker, assurant des mises à jour régulières et la correction des vulnérabilités de sécurité.
3. **Compatibilité** : Ces images sont bien testées pour fonctionner ensemble, garantissant une intégration fluide entre WordPress et la base de données Maria DB.

### Le fichier Dockerfile :



```
root@Lydia: ~/Projet_Docker
GNU nano 6.2
FROM wordpress:latest

COPY wp-config.php /var/www/html/wp-config.php

#Stage 2 - MariaDB
FROM mariadb:latest

COPY my.cnf /etc/mysql/my.cnf
```

**Variables d'environnement** : WordPress utilise des variables d'environnement pour configurer la base de données. Les principales sont WORDPRESS\_DB\_HOST, WORDPRESS\_DB\_USER, WORDPRESS\_DB\_PASSWORD, et WORDPRESS\_DB\_NAME. Ces variables seront configurées dans le fichier wp-config.php.

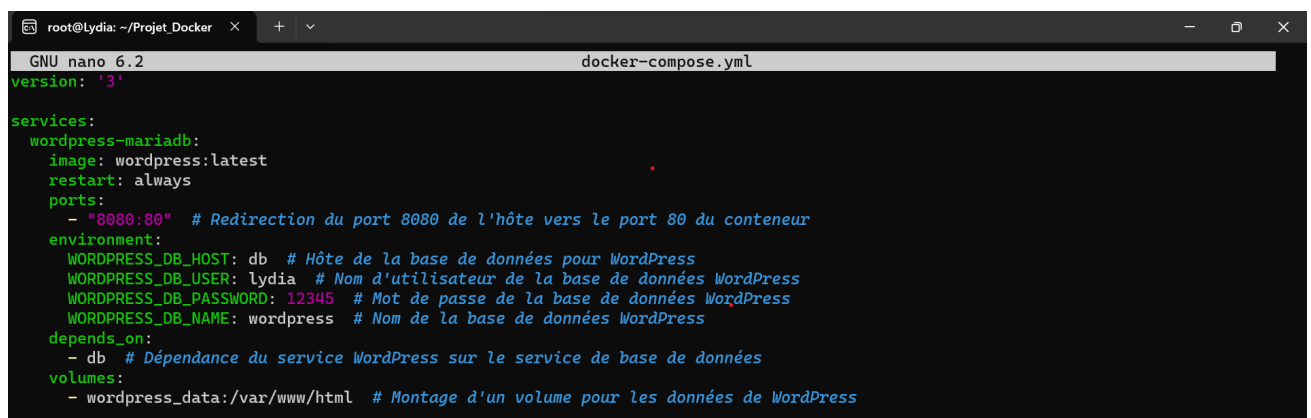
### Explication docker-compose.yml :

**Réseaux virtuels** : Docker Compose crée un réseau virtuel par défaut, ce qui permet aux conteneurs de se référencer les uns les autres par leur nom de service (wordpress et db dans ce cas).

**Variables d'environnement** : Les informations de base de données nécessaires sont fournies en tant que variables d'environnement dans le service WordPress et le service MariaDB.

**Volumes persistants** : Des volumes Docker sont utilisés pour stocker les données persistantes de WordPress et de la base de données MariaDB.

1. **image: wordpress:latest** : Spécifie l'image Docker à utiliser pour ce service, dans ce cas, il s'agit de l'image "latest" de WordPress, ce qui signifie la version la plus récente disponible.
2. **restart: always** : Cette instruction indique à Docker de redémarrer automatiquement le conteneur si celui-ci s'arrête pour une raison quelconque, sauf si l'arrêt a été explicitement demandé.
3. **ports : - "8080:80"** : Cela mappe le port 8080 de votre machine hôte au port 80 à l'intérieur du conteneur WordPress. Cela permet d'accéder à WordPress via le navigateur en utilisant le port 8080.
4. **environment**: Cette section définit les variables d'environnement nécessaires pour que WordPress communique avec la base de données. Ces variables spécifient l'hôte de la base de données (WORDPRESS\_DB\_HOST), le nom d'utilisateur (WORDPRESS\_DB\_USER), le mot de passe (WORDPRESS\_DB\_PASSWORD) et le nom de la base de données (WORDPRESS\_DB\_NAME).
5. **depends\_on**: - db : Cela spécifie la dépendance du service WordPress sur le service de base de données (db). Docker s'assurera que le service de base de données est opérationnel avant de démarrer le service WordPress.
6. **volumes**: - wordpress\_data:/var/www/html : Cela monte un volume nommé wordpress\_data dans le répertoire /var/www/html du conteneur WordPress. Cela permet de stocker les fichiers et données de WordPress de manière persistante, même si le conteneur est arrêté ou supprimé.



```
root@Lydia: ~/Projet_Docker x + v
GNU nano 6.2 docker-compose.yml
version: '3'

services:
  wordpress-mariadb:
    image: wordpress:latest
    restart: always
    ports:
      - "8080:80" # Redirection du port 8080 de l'hôte vers le port 80 du conteneur
    environment:
      WORDPRESS_DB_HOST: db # Hôte de la base de données pour WordPress
      WORDPRESS_DB_USER: lydia # Nom d'utilisateur de la base de données WordPress
      WORDPRESS_DB_PASSWORD: 12345 # Mot de passe de la base de données WordPress
      WORDPRESS_DB_NAME: wordpress # Nom de la base de données WordPress
    depends_on:
      - db # Dépendance du service WordPress sur le service de base de données
    volumes:
      - wordpress_data:/var/www/html # Montage d'un volume pour les données de WordPress
```

## Partie MariaDB :

1. **image: mariadb:latest** : Spécifie l'image Docker à utiliser pour ce service, dans ce cas, l'image "latest" de MariaDB.

2. **restart: always** : Cette instruction indique à Docker de redémarrer automatiquement le conteneur si celui-ci s'arrête pour une raison quelconque, sauf si l'arrêt a été explicitement demandé.
3. **environment** : Cette section définit les variables d'environnement nécessaires pour la configuration de la base de données MariaDB, telles que le nom de la base de données (MYSQL\_DATABASE), le nom d'utilisateur (MYSQL\_USER), les mots de passe pour cet utilisateur et le mot de passe root (MYSQL\_ROOT\_PASSWORD).
4. **volumes** : Montage de deux volumes Docker. Le premier (db\_data:/var/lib/mysql) est utilisé pour stocker les données de la base de données MariaDB de manière persistante. Le second (./my.cnf:/etc/mysql/my.cnf) lie un fichier de configuration personnalisé pour MariaDB.

```
db:
  image: mariadb:latest
  restart: always
  environment:
    MYSQL_DATABASE: wordpress # Nom de la base de données MariaDB
    MYSQL_USER: lydia # Nom d'utilisateur de la base de données MariaDB
    MYSQL_PASSWORD: 12345 # Mot de passe de la base de données MariaDB
    MYSQL_ROOT_PASSWORD: "12345" # Mot de passe root de la base de données MariaDB
  volumes:
    - db_data:/var/lib/mysql # Montage d'un volume pour les données de la base de données
    - ./my.cnf:/etc/mysql/my.cnf # Liaison d'un fichier de configuration personnalisé

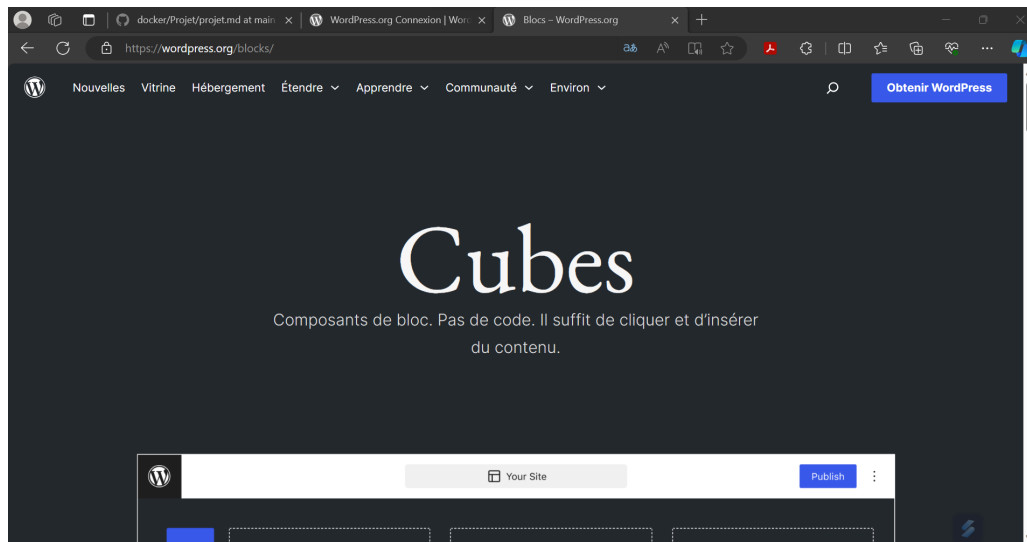
volumes:
  wordpress_data: # Volume pour stocker les données de WordPress
  db_data: # Volume pour stocker les données de la base de données

networks:
  frontend_network: # Définition du réseau frontend (non utilisé dans ce code)
  backend_network: # Définition du réseau backend pour les services
  # Ajout des services à ce réseau
  services:
    - wordpress-mariadb
    - db
```

1. **frontend\_network** : Il s'agit d'une déclaration de réseau qui pourrait être utilisée pour connecter des services spécifiques, mais dans ce fichier, ce réseau n'est pas utilisé. Il est simplement défini mais n'a pas de services associés.
2. **backend\_network** : C'est la définition du réseau dans lequel les services wordpress-mariadb et db sont ajoutés. Ce réseau est destiné à être utilisé pour la communication entre ces services.
3. **services** : Cela spécifie les services Docker qui seront connectés à ce réseau (backend\_network). Dans ce cas, les services wordpress-mariadb et db sont ajoutés à ce réseau, leur permettant de communiquer entre eux via ce réseau défini.

### Démonstration :

En lançant le "localhost:80:80" on obtient ce résultat:



## Project 02: Cluster elasticsearch avec kabana

**Objectif :** Mettre en place un cluster Elasticsearch composé de trois nœuds et un conteneur Kibana qui a accès au cluster.

**Demande :**

### 1. Structure du Projet :

- Créez un répertoire nommé `votre-nom-de-binome`
- À l'intérieur de ce répertoire, créez une structure de fichiers et de répertoires selon le schéma suivant :

```
votre-nom-de-binome/  
├── docker-compose.yml  
├── .env  
└── es-cluster/  
    ├── elasticsearch.yml  
    └── kibana.yml
```

**Reponse :**

```
root@Lydia:~# cd Projet_Docker/  
root@Lydia:~/Projet_Docker# cd Lydia_khaled/  
root@Lydia:~/Projet_Docker/Lydia_khaled# ls  
docker-compose.yml  es-cluster  
root@Lydia:~/Projet_Docker/Lydia_khaled# ls -a  
.  ..  .env  docker-compose.yml  es-cluster  
root@Lydia:~/Projet_Docker/Lydia_khaled# cd es-cluster/  
root@Lydia:~/Projet_Docker/Lydia_khaled/es-cluster# ls  
elasticsearch.yml  es01  es02  es03  kibana.yml  
root@Lydia:~/Projet_Docker/Lydia_khaled/es-cluster# ls -a  
.  ..  .env  elasticsearch.yml  es01  es02  es03  kibana.yml  
root@Lydia:~/Projet_Docker/Lydia_khaled/es-cluster#
```

## Elasticsearch.yml:

```
GNU nano 6.2
cluster.name: my-cluster
node.name: es01
network.host: 0.0.0.0
discovery.seed_hosts: ["es02", "es03"]
cluster.initial_master_nodes: ["es01", "es02", "es03"]
http.port: 9200
```

1. **cluster.name**: Nom du cluster. Il s'agit de "my-cluster" dans ce cas.
2. **node.name**: Nom du nœud Elasticsearch. Il est défini comme "es01".
3. **network.host**: Spécifie à quelle adresse IP le nœud Elasticsearch doit écouter les connexions. "0.0.0.0" signifie qu'il écouterait sur toutes les interfaces réseau.
4. **discovery.seed\_hosts**: Liste des nœuds qui peuvent agir comme points de découverte pour les autres nœuds lors de la formation du cluster. Dans ce cas, ce sont "es02" et "es03".
5. **cluster.initial\_master\_nodes**: Liste des nœuds qui doivent être considérés comme les premiers maîtres du cluster. Dans ce cas, ce sont "es01", "es02", et "es03".
6. **http.port**: Le port sur lequel le nœud Elasticsearch écouterait les requêtes HTTP. Ici, c'est le port standard 9200.

## Kibana.yml:

```
GNU nano 6.2
server.name: kibana
server.host: "0.0.0.0"
elasticsearch.hosts: ["http://es01:9200"]
```

1. **server.name**: Nom du serveur Kibana
2. **server.host**: Spécifie à quelle adresse IP Kibana doit écouter les connexions. "0.0.0.0" signifie qu'il écouterait sur toutes les interfaces réseau.
3. **elasticsearch.hosts**: Liste des nœuds Elasticsearch auxquels Kibana se connecterait. Ici, il se connecte à "es01" sur le port 9200.

## Note :

**Network.host** à "0.0.0.0": Permet aux nœuds Elasticsearch d'accepter les connexions provenant de n'importe quelle interface réseau, ce qui est utile pour une configuration sur un réseau local.

**discovery.seed\_hosts** et **cluster.initial\_master\_nodes** : Ils sont configurés pour permettre la découverte et l'initialisation correctes du cluster, assurant une formation de cluster réussie.

**server.host** à "0.0.0.0" dans **kibana.yml** : Permet à Kibana d'accepter les connexions depuis n'importe quelle interface réseau, rendant l'accès à l'interface utilisateur de Kibana plus flexible.

**elasticsearch.hosts** dans **kibana.yml**: Indique à Kibana où se trouve le cluster Elasticsearch auquel il doit se connecter.

Le but est de créer une configuration robuste et flexible pour un cluster Elasticsearch avec Kibana, en s'assurant que les différents composants peuvent se découvrir et se connecter correctement.

## docker-compose.yml:

```
GNU nano 6.2                                     docker-compose.yml *
version: '3' # Version du format Docker Compose utilisé

services: # Section des services Docker
  es01: # Service Elasticsearch 01
    image: docker.elastic.co/elasticsearch/elasticsearch:${ES_VERSION} # Image Docker d'Elasticsearch avec la version spécifiée
    container_name: es01 # Nom du conteneur
    mem_limit: 1500m # Limite mémoire pour le conteneur Elasticsearch
    volumes: # Configuration des volumes Docker
      - esdata01:/usr/share/elasticsearch/data # Montage du volume pour la persistance des données
    ports: # Configuration des ports
      - "9200:9200" # Mappage du port 9200 du conteneur vers le port 9200 de l'hôte
    networks: # Configuration des réseaux
      - es-net # Utilisation du réseau personnalisé 'es-net'
    environment: # Configuration des variables d'environnement pour Elasticsearch
      - node.name=es01 # Nom du nœud Elasticsearch
      - discovery.seed_hosts=es02,es03 # Nœuds de découverte pour le clustering
      - cluster.initial_master_nodes=es01,es02,es03 # Nœuds maîtres initiaux pour le clustering
      - cluster.name=my-cluster # Nom du cluster

  es02: # Service Elasticsearch 02
    image: docker.elastic.co/elasticsearch/elasticsearch:${ES_VERSION}
    container_name: es02
    mem_limit: 1500m
    volumes:
      - esdata02:/usr/share/elasticsearch/data
    networks:
      - es-net
    environment:
      - node.name=es02
      - discovery.seed_hosts=es01,es03
      - cluster.initial_master_nodes=es01,es02,es03
      - cluster.name=my-cluster

  es03: # Service Elasticsearch 03
    image: docker.elastic.co/elasticsearch/elasticsearch:${ES_VERSION}
    container_name: es03
    mem_limit: 1500m
    volumes:
      - esdata03:/usr/share/elasticsearch/data
    networks:
      - es-net
    environment:
      - node.name=es03
      - discovery.seed_hosts=es01,es02
      - cluster.initial_master_nodes=es01,es02,es03
      - cluster.name=my-cluster

  kibana: # Service Kibana
    image: docker.elastic.co/kibana/kibana:${ES_VERSION} # Image Docker de Kibana avec la version spécifiée
    container_name: kibana
    ports:
      - "5601:5601" # Mappage du port 5601 du conteneur vers le port 5601 de l'hôte
    networks:
      - es-net
    environment:
      - ELASTICSEARCH_URL=http://es01:9200 # URL Elasticsearch pour Kibana

volumes: # Configuration des volumes Docker
  esdata01:
  esdata02:
  esdata03:

networks: # Configuration des réseaux Docker
  es-net:
```

### Version Docker Compose :

La version 3 est une version stable largement utilisée et prend en charge les fonctionnalités nécessaires sans complications.

### Services Elasticsearch (es01, es02, es03) :

- Utilisation de l'image Elasticsearch avec la version spécifiée par la variable d'environnement ES\_VERSION.
- Configuration de trois nœuds Elasticsearch pour former un cluster.
- Utilisation de volumes Docker pour stocker les données Elasticsearch de manière persistante.
- Configuration des ports pour permettre l'accès à Elasticsearch depuis l'hôte.
- Configuration du réseau pour permettre la communication entre les conteneurs.
- Configuration des variables d'environnement pour les paramètres Elasticsearch.

**Justification :**

L'utilisation de la version spécifiée dans ES\_VERSION permet de rendre le cluster flexible et conforme à la demande du projet.

Les volumes Docker garantissent la persistance des données même si les conteneurs sont détruits.

La configuration des ports et du réseau facilite la communication entre les services et l'accès à Elasticsearch depuis l'hôte.

**Service Kibana :**

- Utilisation de l'image Kibana avec la même version spécifiée pour Elasticsearch.
- Configuration des ports pour permettre l'accès à Kibana depuis l'hôte.
- Configuration du réseau pour permettre la communication avec Elasticsearch.
- Configuration de l'URL Elasticsearch pour Kibana.

**Justification :**

- L'utilisation de la même version pour Kibana assure la compatibilité avec Elasticsearch.
- La configuration des ports et du réseau facilite l'accès à Kibana depuis l'hôte et la communication avec Elasticsearch.
- La configuration de l'URL permet à Kibana de se connecter correctement au cluster Elasticsearch.

**Volumes :**

Définition de volumes pour la persistance des données Elasticsearch.

**Justification :**

Les volumes Docker garantissent la persistance des données entre les redémarrages des conteneurs, assurant ainsi la durabilité du cluster.

**Réseau personnalisé (es-net) :**

Configuration d'un réseau personnalisé pour les services.

**Justification :**

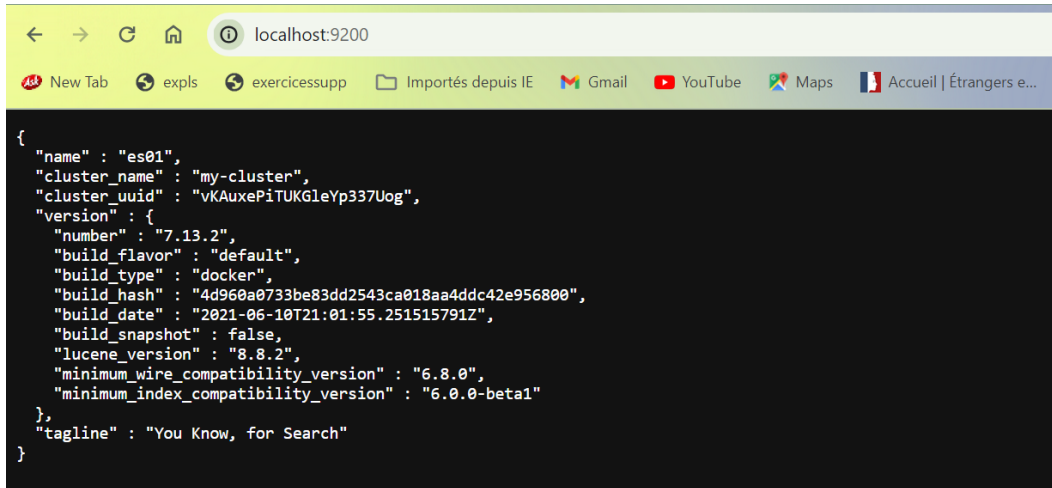
La création d'un réseau personnalisé facilite la communication entre les services tout en les isolant du réseau par défaut, améliorant ainsi la sécurité et la gestion du trafic.

**Résumé :**

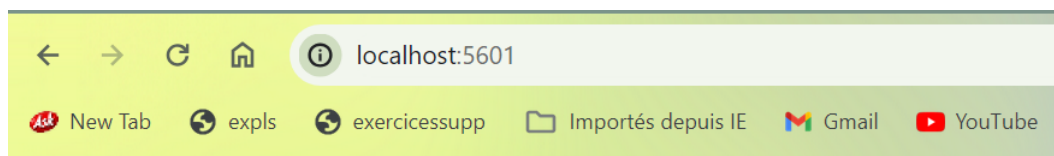
Les choix techniques visent à créer un environnement Elasticsearch stable, persistant et sécurisé, conforme aux exigences du projet. L'utilisation de variables d'environnement permet une flexibilité accrue, tandis que la configuration des ports, des volumes et du réseau garantit la communication efficace entre les services.

**Résultat :**





```
{
  "name" : "es01",
  "cluster_name" : "my-cluster",
  "cluster_uuid" : "vKAuxePiTUKGleYp337Uog",
  "version" : {
    "number" : "7.13.2",
    "build_flavor" : "default",
    "build_type" : "docker",
    "build_hash" : "4d960a0733be83dd2543ca018aa4ddc42e956800",
    "build_date" : "2021-06-10T21:01:55.251515791Z",
    "build_snapshot" : false,
    "lucene_version" : "8.8.2",
    "minimum_wire_compatibility_version" : "6.8.0",
    "minimum_index_compatibility_version" : "6.0.0-beta1"
  },
  "tagline" : "You Know, for Search"
}
```



## Approche 02: Cluster elasticsearch avec kabana

Cette approche qui est fonctionnelle consiste à déployer un seul elasticsearch avec kabana.

Cette approche consiste à configurer un environnement Elasticsearch et Kibana, où Elasticsearch fonctionne en mode autonome et Kibana dépend d'Elasticsearch. L'utilisation de la version 7.13.2 assure la compatibilité entre les deux services.

docker-compose.yml:

```
GNU nano 0.2
Version: '3'
services:
  elasticsearch:
    image: docker.elastic.co/elasticsearch/elasticsearch:7.13.2
    environment:
      - discovery.type=single-node
    ports:
      - "9200:9200"
  kibana:
    image: docker.elastic.co/kibana/kibana:7.13.2
    ports:
      - "5601:5601"
    depends_on:
      - elasticsearch
```

Résultat d'exécution:

localhost:9200

The screenshot shows a web browser window with the address bar set to `localhost:9200`. The page content is a JSON response from the Elasticsearch API. The JSON is displayed in a light blue theme with a dark background. The data includes cluster information such as name, UUID, and version details.

Field	Value
<code>name</code>	<code>"e8df8f246062"</code>
<code>cluster_name</code>	<code>"docker-cluster"</code>
<code>cluster_uuid</code>	<code>"7imtJ9D_TSKMzkZkr9pnhQ"</code>
<code>version</code>	<ul style="list-style-type: none"><li><code>number</code>: <code>"7.13.2"</code></li><li><code>build_flavor</code>: <code>"default"</code></li><li><code>build_type</code>: <code>"docker"</code></li><li><code>build_hash</code>: <code>"4d960a0733be83dd2543ca018aa4ddc42e956800"</code></li><li><code>build_date</code>: <code>"2021-06-10T21:01:55.251515791Z"</code></li><li><code>build_snapshot</code>: <code>false</code></li><li><code>lucene_version</code>: <code>"8.8.2"</code></li><li><code>minimum_wire_compatibility_version</code>: <code>"6.8.0"</code></li><li><code>minimum_index_compatibility_version</code>: <code>"6.0.0-beta1"</code></li></ul>
<code>tagline</code>	<code>"You Know, for Search"</code>

localhost:5601

