Ateliers
« Elastic Stack»

Pré-requis :

- Bonne Connexion Internet
- 32 Go RAM, espace disque important
- OS : Linux, MacOs, Windows 10
- YAML Editor : (VSCode, Atom, ...)
- Optionnel : Docker, Git

Table des matières

Ateliers 1 : Mise en place du cluster

1.1 Installation premier noeud

1.1.1 Premier démarrage

Vérifier votre espace disque libre (Au moins 20 % de libre)

Télécharger et décompresser la dernière release d' Elastic Search dans un répertoire ***mywork/node1***

Démarrer le serveur avec
`$ES_HOME/bin/elasticsearch`

Si la release est supérieure à 8, Visualiser les traces et sauvegarder toutes les informations relative au password et au jeton d'inscription

Accèder via curl

`export ELASTIC_PASSWORD="your_password"`

`curl --cacert config/certs/http_ca.crt -u elastic:$ELASTIC_PASSWORD https://localhost:9200`

Ou via un navigateur https://localhost:9200

1.1.2 Mode production et bootstrap checks

Éditer le fichier de configuration principal et modifier les propriétés suivantes :

- Nom du Cluster : <***votre-nom***>
- Nom du nœud : ***node-1***
- Adresse d'écoute (Indiquer une de vos adresse IP différent de l'adresse de loopback)

Essayer de démarrer le serveur et observer les vérifications de démarrage (bootstrap-checks), effectuer les corrections si nécessaires.
(voir https://www.elastic.co/guide/en/elasticsearch/reference/current/bootstrap-checks.html)

1.2 Installation de Kibana

Télécharger une distribution de Kibana avec le même numéro de version que celui d'ElasticSearch .

Décompresser l'archive dans le répertoire ***mywork*** et démarrer Kibana (si 8.x +, utiliser l'enrollment token)

Accéder à http://localhost:5601 et retrouver la *Dev Console*.

Exécuter les requêtes suivantes :

Santé du cluster :

```
GET /_cluster/health
```

API cat

```
GET /_cat/nodes
GET /_cat/indices
```

Recherche de tous les documents

```
GET /_search
```

1.3 Démarrage des autres nœuds

Dimensionnement de la JVM

Dans le répertoire ***config/jvm.d*** ajouter un fichier ***ram.options*** avec le contenu suivant :

```
-Xms4g
-Xmx4g
```

Redémarrer le noeud1

Cluster health

- Exécuter :

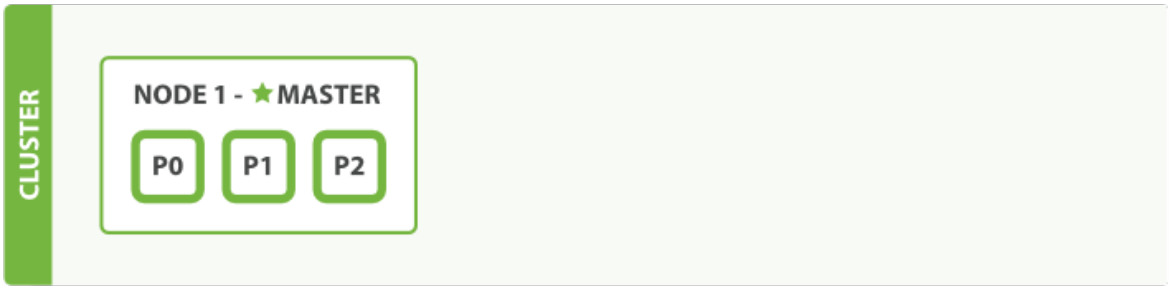
```
GET /_cluster/health?pretty
```

  Combien de nœuds de shards sont disponibles ?

- Créer un index nommé *blogs*

```
PUT /blogs {
 "settings" : {
 "number_of_shards" : 3,
 "number_of_replicas" : 1
 }
}
```

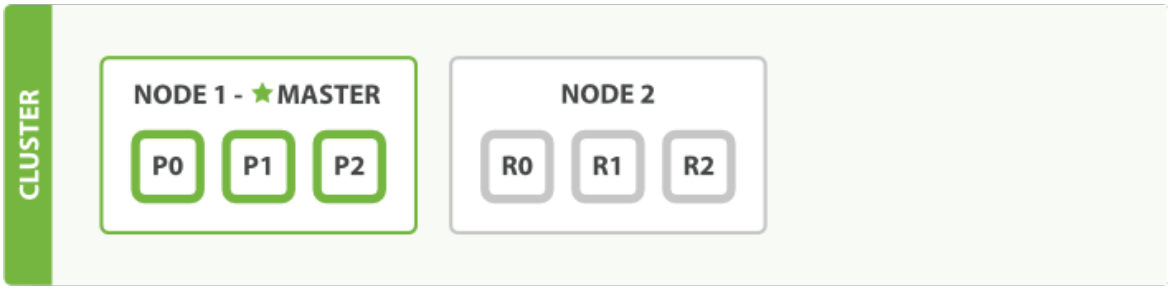
- Re-executer *_cluster/health?pretty*
  Couleur du statut de santé ? Combien de disponibles, actifs ?



Installation second nœud

- Dans le nœud 1 créer un enrollment token avec :
  `bin/elasticsearch-create-enrollment-token -s node`
- Dézipper la distribution dans un autre emplacement : ***mywork/node-2***
- Éditer le fichier de config :
  - ***cluster.name***
  - ***node.name***
  - ***network.host***
- Dimensionner la JVM comme pour le noeud1

- Démarrer le second noeud avec l'enrollment token précédent :
  `bin/elasticsearch --enrollment-token <token>`
- Re-executer `_cluster/health?pretty`
  Couleur du statut de santé ? Combien de disponibles, actifs ?



Installation 3ème nœud

- Démarrer un 3ème nœud
  Couleur du statut de santé ? Combien de disponibles, actifs ?
- Augmenter le nombre de répliques

```
PUT /blogs/_settings
 { "number_of_replicas" : 2 }
```

Fail-over

- Arrêter le premier nœud
- Visualiser les logs, vous devriez voir un changement de master
- Couleur du statut de santé ?
- Redémarrer le premier nœud

1.2 Désactivation de la sécurité

Pour les laboratoires suivants, nous désactivons la sécurité. Dans 8.x+, la sécurité est activée par défaut.

Pour désactiver, il faut :

- Positionner `xpack.security.enabled: false`
- Commenter toutes les propriétés relatives à *xpack.security* dans *elasticsearch.yml*
- Supprimer les propriétés stocker dans le keystore d'elasticsearch.
  `bin/elastisearch-keystore remove <name-of-the-setting>`

Dans *kibana.yml* commenter toutes les propriétés relatives à *xpack.security*

Ateliers 2 : Document API et SearchLite

2.1 Document API

- Indexer les 3 documents fournis via la commande *curl* suivante ou via Kibana
  `curl -XPOST /blogs/_doc/ --data-binary "@entry1.json"`
- Noter les ids et récupérer les document via un ID
- Mettre à jour 1 document en ajoutant de nouveau champs (Remarquer l'incrémentation de la version) :
  - *tags* : Array
  - *views* : Initialisé à 0
- Effectuer des mises à jour avec un script:
  - Incrémenter un champ numérique
  - Ajouter un élément au tableau *tags*
  - Supprimer le champ *tags*
- Supprimer l'index et effectuer toutes les opérations suivantes en une seule commande Bulk API

2.2 Search Lite

Effectuer les recherches suivantes en utilisant la query string et en utilisant les paramètres suivants :

- *size, from*
- *q, df*
- *_source*

Ateliers 3 : Ingestion de données

3.1 Metrics beats

Dans cette partie, nous mettons en place *metricbeats* afin qu'il alimente directement ElasticSearch avec des metrics :

- ***system***
- ***elasticsearch-xpack***

Récupérer une distribution de *metricbeats*

Editer le fichier ***metricbeat.yml*** afin de spécifier la connexion à elasticsearch

```
output.elasticsearch:
  # Array of hosts to connect to.
  hosts: ["192.168.21.196:9200"]

  # Performance preset - one of "balanced", "throughput", "scale",
  # "latency", or "custom".
  preset: balanced

  # Protocol - either `http` (default) or `https`.
  protocol: "https"

  # Authentication credentials - either API key or username/password.
  #api_key: "id:api_key"
  username: "elastic"
  password: <password>
  ssl:
    enabled: true
    ca_trusted_fingerprint: <ca_fingerprint>
```

Activer les modules elasticsearch-xpack et système

`./metricbeat modules list`

`./metricbeat modules enable elasticsearch-xpack`

Editer le fichier de configuration du module elasticsearch-xpack ***modules.d/elasticsearch-xpack.yml***

```
- module: elasticsearch
  xpack.enabled: true
  period: 10s
  hosts: ["https://<node1-ip>:<port>","https://<node2-
ip>:<port>","https://<node3-ip>:<port>"]
  username: "elastic"
  password: <password>
  #api_key: "foo:bar"
  ssl:
    enabled: true
    ca_trusted_fingerprint: <ca_fingerprint>
```

Changer le propriétaire des fichiers à root.

`sudo chown -R . root`

Initialiser la config et charger les tableaux de bord Kibana
`sudo ./metricbeat setup -e`

Démarrer metricbeats et vérifier les traces
`sudo ./metricbeat -e`

Avec la DevConsole, Visualiser les index créés dans ***elasticsearch*** et que leur nombre de document augmente :

```
GET _cat/indices

GET /.ds-metricbeat*/_count

GET /.ds-metricbeat*/_count

GET .ds-.monitoring-es*/_count
```

Dans Kibana

- Visualiser les métriques système
  Analytics → Dashboard
  Rechercher system
- Visualiser les métriques ElasticSearch
  Management → Stack monitoring

3.2 Elastic Agent

3.2.1 Installation Fleet Server/ElastiAgent

Dans Kibana ***Management → Fleet, activer le bouton Add Fleet Server*** et suivre l'assistant

Référence :
https://www.elastic.co/guide/en/fleet/current/add-fleet-server-on-prem.html

A la fin de l'assistant, vous devriez avoir installer un Fleet Server qui est lui même un Elastic Agent. Un policy « ***Fleet-server policy*** » a été créée avec les intégrations système (logs, métriques, winlog)

3.2.2 Ajout d'une intégration

Dans Kibana ***Management → Integrations***,

Recherche l'intégration ***ElasticSearch*** et l'ajouter, l'associer à ***Fleet-server policy*** et visualiser les assets ramenées par cette intégration

Visualiser également dans ***Observability → Log*** les logs disponibles

Ateliers 4 : Logstash

4.1 Pipeline

4.1.1 Installation logstash

Télécharger la distribution de logstash compatible avec elasticsearch

Placer vous dans *bin* et exécuter :
`./logstash -e 'input { stdin { } } output { stdout {} }'`

Saisir des messages sur l'entrée standard

4.1.2 Traitement de logs Apache

Démarrage filebeat
Récupérer la distribution de *FileBeat* compatible et dézipper dans le répertoire ***mywork***
Récupérer l'archive fournie contenant les logs Apache et dézipper dans ***mywork/logstash-tutorial***

Dans le fichier de configuration ***filebeat.yml***, mettre à jour la configuration avec les lignes suivantes :

```
filebeat.inputs:

- type: filestream

id: my-filestream-id

enabled: true

# Paths that should be crawled and fetched. Glob based paths.
paths:
# - /var/log/*.log
- /path/to/file/logstash-tutorial.log

# Désactiver la sortie elasticsearch et activer la sortie logstash
#output.elasticsearch:
  # Array of hosts to connect to.
  #hosts: ["localhost:9200"]
  # Performance preset - one of "balanced", "throughput", "scale",
  # "latency", or "custom".
  #preset: balanced
output.logstash:
  hosts: ["localhost:5044"]
```

Démarrer *filebeat* avec la commande suivante :
`./filebeat -e -c filebeat.yml -d "publish"`

Visualiser les logs

Mise au point de la pipeline
Créer un fichier de configuration de pipeline ***mywork/pipeline.conf*** comme suit :

```
input {
    beats {
        port => "5044"
    }
}
# The filter part of this file is commented out to indicate that it is
# optional.
 filter {
    grok {
        match => { "message" => "%{COMBINEDAPACHELOG}"}
    }
    geoip {
        source => "[source][address]"
        ecs_compatibility => "v1"
        target => "geoip"
    }
 }
output {
   elasticsearch {
        hosts => [ "https://localhost:9200" ]
        data_stream => "true"
        data_stream_dataset => "logstash-tutorial"
        data_stream_namespace => "logstash-tutorial"
        data_stream_type => "logs"
        user => "elastic"
        password => "elastic"
        ssl_enabled => true
        ca_trusted_fingerprint =>
"8f1c11a376cfb58a1bffe1812095857c6cbd6fe830f8a04b7c25b20b68e23132"
    }
    stdout { codec => rubydebug }
}
```

Tester la configuration :

`bin/logstash -t -f ./pipeline.conf`

Ingestion des données

Arrêter filebeat et le réinitialiser avec

`sudo rm -rf data/registry`

Démarrer logstash et observer la console :

`bin/logstash -r -f ./pipeline .conf`

Démarrer *filebeat*

Vérification

Dans la dev console, visualiser les nouveaux index et visualiser les documents

- Recommencer le traitement, vérifier la console d'ElasticSearch et trouver l'index créé.
- Effectuer une recherche afin de trouver des documents ElasticSearch

4.2 Syntaxe logstash, Traitement des logs applicatifs

Récupérer les données sources ***plbsi.zip*** et dézipper dans le répertoire ***mywork***
A l'intérieur du répertoire dézipper tous les fichiers .gz puis les supprimmer.

Mettre en place une pipeline traitant ces fichiers applicatifs.

Input
Utiliser l'input ***file***
Utiliser le codec mulitine afin que les stack trace java soit rassemblées en 1 évènement

Filtres
Commencer par un filtre ***grok*** qui permettent d'extraire :

- le timestamp
- Le niveau de log
- Le process id
- Le nom de la thread Java
- Le logger Java
- Le message

Vous pouvez vous aider du Grok Debugger présent dans la console kibana
Remplacer le champ message par le message java

Créer un champ timestamp via le filtre ***date***, le format initial de la date est ISO8601

Si le niveau de log est ***ERROR*** ajouté un tag « error »

Passer le niveau de log en minuscule pour s'adapter à *Elastic Common Schema*

Sorties
Utiliser une sortie elastic search vers un datastream springboot
Et un ***stdout*** pour debugger

4.3 Exécution simultanée des 2 pipelines

Mettre au point le fichier ***pipelines.yml*** qui configure les 2 pipelines précédentes.

Utiliser un modèle d’exécution différent pour chaque pipeline

Ateliers 5 : Mapping, gabarits d'index et datastream

5.1 Mapping

Créer un alias pour tous le datastream SpringBoot

```
POST /_aliases
{
    "actions": [
    {
      "add": {
        "index": "logs-springboot-app-springboot",
        "alias": "springboot"
      }
    }
  ]
}
```

Visualiser le mapping en utilisant l'alias créé. Quels sont les champs de texte intégral et quels analyseurs sont utilisés ?

Effectuer des recherches full text sur un champ indexé. Par exemple :

```
GET springboot/_search?q=message:(+Created%20+sessions)
GET springboot/_search?q=message:(+Created%20+session)
```

Conserver le mapping pour l'atelier suivant

5.2 Mise en place de datastream

Arrêter logstash et le réinitialiser

sudo rm -rf data/plugins/inputs/file

Supprimer le datastream

```
DELETE /_data_stream/logs-springboot-app-springboot
```

Créer une ILM

```
PUT _ilm/policy/springboot_policy
{
  "policy": {
    "phases": {
      "hot": {
        "actions": {
          "rollover": {
            "max_size": "1GB",
            "max_age": "1d"
          }
        }
      },
      "delete": {
        "min_age": "2d",
```

```
        "actions": {
          "delete": {}
        }
      }
    }
  }
}
```

Créer un gabarit de composant en reprenant le mapping précédent mais en ajoutant l'analyseur english aux champs text

PUT _component_template/springboot_component_template

{

...

}

Créer un gabarit d'index

```
PUT _index_template/springboot_template
{
  "index_patterns": [
    ".ds-logs-springboot-app-springboot-*"
  ],

  "template": {
    "settings": {
      "number_of_shards": 3,
      "number_of_replicas": 2,
      "index.lifecycle.name": "springboot_policy",
      "index.lifecycle.rollover_alias": "springboot"
    }
  },
   "composed_of": ["springboot_component_template"],
   "priority": 1000,
   "data_stream": {
     "allow_custom_routing": false,
     "hidden": false
   }
}
```

Redémarrer logstash, visualiser les shards des indices associés au datastream springboot, le template et l'ilm associé au datastream

Effectuer des recherches full text sur un champ indexé. Par exemple :

```
GET logs-springboot-app-springboot/_search?q=message:(+Created%20+sessions)
GET logs-springboot-app-springboot/_search?q=message:(+Created%20+session)
```

Ateliers 6 : Recherche via DSL

6.1 Syntaxe DSL

Effectuez des requêtes DSL basées sur le datastream springboot:

- Traces de niveau ERROR triés par date
- Traces dont le champ de message répond à "exception"
- Traces dont le champ logger se termine par Service
- Traces de niveau ERROR pour un intervalle de date
- Traces dont le champ message contient starting et le champ thread Schedul*
- Traces dont le champ message répond à "formation found"

6.2 Agrégations

Sur l'index Apache

- Effectuer des agrégations de comptage
  - par type de code retour
  - par verbe
  - par les 2

- Effectuer des regroupements par intervalles de taille
- Trouver la taille moyenne d'une requête
- Trouver la taille moyenne des requêtes groupées par code retour
- Trouver la somme des octets transférés par trimestre
- Trouver le client ayant un taux anormal de code réponse 400

6.3 Géo-localisation

Sur l'index Apache

- Effectuer des agrégation de type *geo_hash_grid* sur les localisation des clients en augmentant progressivement la précision

Ateliers 7: Kibana et tableaux de bord

7.1 Démarrer avec Kibana

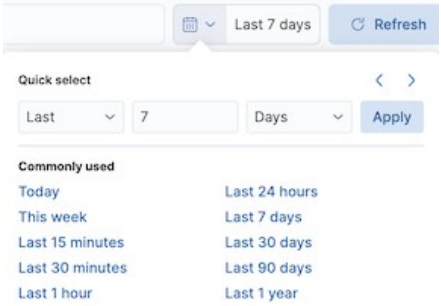
Ajouter les données

1. Sur la page d'accueil, cliquez sur ***Try sample Data***
2. Cliquez sur ***Other sample data sets***.
3. Choisir ***Sample eCommerce orders***

Explorer les données

Aller sur ***Discover***

Sélectionner la fenêtre temporel au 7 derniers jours :



Filtrer les données afin de ne voir : « *les commandes de vêtements pour femmes d'une valeur de 60 $ ou plus* »

Dans le champ de recherche saisir :

```
products.taxless_price >= 60 and category : Women's Clothing
```

Ajouter le champ ***category***

Visualiser les données

Aller dans ***Dashboard*** et visualiser le tableau de bord : [eCommerce] Revenue Dashboard.

Dans la barre d'outils, cliquez sur ***Edit***.

Sur le tableau de bord, cliquez sur ***Create Visualization***

Dans l'éditeur de visualisation ouvrez la liste déroulante ***Visualization Type*** puis sélectionnez ***Treemap***.

Dans la liste des champs disponibles, faites glisser les champs suivants vers l'espace de travail :

- ***geoip.city_name***

- ***manufacturer.keyword***

Sauvegarder

Interactivité

Utiliser en suite le panel [eCommerce] Controls pour sélectionner un fabricant.

Filtre

1. Cliquer sur ***Add Filter***
2. Dans la liste déroulante ***Field***, sélectionnez ***day_of_week***
3. Dans la liste déroulante Opérateur, sélectionnez ***is***.
4. Dans la liste déroulante Valeur, sélectionnez ***Wednesday***.

7.2 Fonctionnalités de Discover

Toujours sur la dataview ***eCommerce***

Ajouter le champ ***manufacturer*** dans le tableau de données, visualiser les 10 valeurs les plus courantes

Trouver les champs ***customer_first_name*** et ***customer_last_name*** et les ajouter

Ajouter un champ customer et indiquer le script Painless suivant :

```
String str = doc['customer_first_name.keyword'].value;
char ch1 = str.charAt(0);
emit(doc['customer_last_name.keyword'].value + ", " + ch1);
```

Supprimer ***customer_first_name*** et ***customer_last_name***

Rechercher des documents avec la requête KQL suivante :
geoip.country_iso_code : US and products.taxless_price >= 75

Ajouter un filtre afin d'exclure les documents dont le champ ***day_of_week*** n'est pas égal à ***Wednesday***

Accéder au détail d'un document

Visualiser les boutons associés à un champ

Essayer les 2 liens actions

7.3 La syntaxe KQL

Commencer par créer une dataview ***logstash-apache*** sur les index ***logstash-apache****, la positionner comme dataview par défaut.

Visualiser les informations de Mapping dans Kibana

Accéder ensuite à la page Discover

- Quel est le pays le plus présent ?
- Faire une recherche sur le mot google dans quel champ est présent ce terme
- Rechercher dans le champ url.original le mot clé microsoft
- Recherche dans le champ message la valeur exacte /formation/microsoft/
- Faire une recherche isolant les requêtes POST PUT DELETE
- Toutes les requêtes ont elle un champ client.ip ?

Effectuer une requête KQL qui isole les requêtes en erreur qui ne provienne pas de machine ayant une IP commençant par 192

Sauvegarder la requêtes sous le noms « Requêtes en erreur »

7.4 La syntaxe ES|QL

Écrire une requête ES|QL qui affiche le total d'octets transférés par pays, limiter au 10 premier pays les plus importants

Éditer la visualisation pour la convertir en camembert

Ajouter le camembert à un nouveau tableau de bord

A l'intérieur du tableau de bord, revenir sur la requête pour afficher 20 pays

7.5 Tableau de bord

Créer un dataview pour les logs SpringBoot.

Créer un dashboard qui permet de visualiser :

- Sur un graphique ligne montrant l'évolution du nombre de messages pour chaque level
- Un camembert par thread
- Les tops messages d'erreur
- Les tops loggers

Ateliers 8 : Architecture

8.1 Cluster logstash

Définir un cluster logstash 3 noeuds, utiliser persistent queue

Activer des beats (MetricBeats, Packetbeats) sur toutes les machines qui load-balanced vers les nœuds logstash,

8.2 Cluster ElasticSearch

Un cluster ElasticSearch avec des spécialisations de nœud :

- Data node
- Master nodes
- Ingstion nodes

8.3 Exploitation

Utiliser les APIs de monitoring d'ELS

Sans redémarrer le serveur, augmenter le niveau de log à DEBUG

Activer le slowlog et effectuer des requêtes provoquant des écritures dans le fichier

Désactiver la suppression d’index par wildcards

Effectuer un backup d’index puis restore

Réindexation d’un index existant vers un autre index avec un nombre de shards différents

Atelier 4-b : Ingestion via JDBC

Alimenter la BD :

Récupérer l'archive fournie et la dézipper.

Démarrer mysql

`docker run --name mysql -e MYSQL_ROOT_PASSWORD=root -v .:/ -d mysql:latest`

`docker exec -it mysql bash`

`mysql -u root -p`

Importer les données via :

`source mysqlsampledatabase.sql`

Mise au point de la pipeline

Mettre au point une pipeline utilisant le plugin d'input jdbc permettant d'ingérer tous les enregistrements fournis par la requête suivante :

```
select * from customers,orders,orderdetails,products where
orders.customernumber=customers.customernumber and
orderdetails.orderNumber=orders.orderNumber and
orderdetails.productcode=products.productcode;
```

Les données seront ingérer dans un index nommé ***classicmodels*** et l'ID des documents correspondra au champ ***orderLineNimber***