INSTRUCCIONES DE DESPLIEGUE DEL CLUSTER DE ELK

ÍNDICE

PASO 1 (Establecer los servicios)

Poner las carpetas de **kibana** y **elasticsearch** en la máquina 1 y las carpetas de **elasticsearch** en las máquinas 2 y 3. También añadimos la carpeta de **logstash** en la máquina 2.

De forma que queda como se muestra a continuación:

Máquina 1:
├─ Kibana/
└─ Elasticsearch/

Máquina 2:
├─ Elasticsearch/
└─ Logstash/

Máquina 3:
└─ Elasticsearch/

PASO 2 (Configurar primer nodo de elasticsearch)

Iniciamos el elasticsearch del nodo 1 con el comando (***`bin/elasticsearch`***) y al ser la primera vez nos devolverá un usuario y contraseña para iniciar sesión en kibana y un enrollment-token de 30 minutos de validez para conectar el servicio de kibana.

Nos devolverá algo como esto:

✅ Elasticsearch security features have been automatically configured!
✅ Authentication is enabled and cluster connections are encrypted.

ℹ️ Password for the elastic user (reset with `bin/elasticsearch-reset-password -u elastic`):
elastic

ℹ️ HTTP CA certificate SHA-256 fingerprint:
8d16b35eeb0659750f17f5f7495f9c49f6e2ee1d13d3604dc73618ac12b2984b

ℹ️ Configure Kibana to use this cluster:
• Run Kibana and click the configuration link in the terminal when Kibana starts.
• Copy the following enrollment token and paste it into Kibana in your browser (valid for the next 30 minutes):

eyJ2ZXIiOiI4LjE0LjAiLCJhZHIiOlsiMTkyLjE5OS4xLjU5OjkyMDAiXSwiZmdyIjoiOGQxNmIzNWVlYjA2NTk3NTBmMTdmNWY3NDk1ZjljNDlmNmUyZWUxZDEzZDM2MDRkYzczNjE4YWMxMmIyOTg0YiIsImtleSI6IkxsODV1cG9CYUlXQUYweHctdUlSOlRpNGExTEhGQzFrNDM2cC13aTB0cmcifQ==

ℹ️ Configure other nodes to join this cluster:
• Copy the following enrollment token and start new Elasticsearch nodes with

```
`bin/elasticsearch --enrollment-token <token>` (valid for the next 30 minutes):

eyJ2ZXIiOiI4LjE0LjAiLCJhZHIiOlsiMTkyLjE5OS4xLjU5OjkyMDAiXSwiZmdyIjoiOGQxNmIzNWVlYjA2NTk3NTBmMTdmNWY3NDk1ZjljNDlmNmUyZWUxZDEzZDM2MDRkYzczNjE4YWMxMmIyOTg0YiIsImtleSI6IkxGODV1cG9CYUlXQUYweHctdUlNOjVrbkZNLVlQNXVST3pKd2dsb3RwU1EifQ==

  If you're running in Docker, copy the enrollment token and run:
  `docker run -e "ENROLLMENT_TOKEN=<token>"
docker.elastic.co/elasticsearch/elasticsearch:9.2.1`
```

PASO 3 (Iniciar Kibana)

Seguidamente de iniciar el primer nodo de elasticsearch iniciamos kibana con el comando (***bin/kibana***), se nos levantara el servicio de kibana y entramos en la dirección url que nos devuelve. En nuestro caso fue (**http://192.199.1.59:5601/**).

La primera vez al entrar nos pedirá el enrollment-token que nos dio anteriormente elasticsearch, se lo ponemos y ya podemos iniciar sesión con el usuario y contraseña que también nos dio elasticsearch.

Opcionalmente, se puede cambiar la contraseña, nostros la cambiamos con el siguiente comando (***bin/elasticsearch-reset-password -u elastic -p elastic***), con esto la contraseña de nuestro usuario elastic pasa a ser elastic.

PASO 4 (Conectar los otros dos nodos al cluster)

Para ello ejecutamos el siguiente comando en el elasticsearch de la máquina 1 , el cual nos devolverá un enrollment-token para conectar los otros 2 nodos de elasticsearch y generar sus certificados automáticamente.

```
bin/elasticsearch-create-enrollment-token -s node
```

Una vez que tengamos el token, iniciamos los otros 2 nodos restantes de elasticsearch con el siguiente comando.

```
bin/elasticsearch --enrollment-token <TOKEN_GENERADO>
```

Esto generará los certificados necesarios en el elasticsearch.yml de cada máquina.

```
# Enable security features
xpack.security.enabled: true

xpack.security.enrollment.enabled: true
xpack.security.authc.api_key.enabled: true

# Enable encryption for HTTP API client cor
xpack.security.http.ssl:
  enabled: true
  keystore.path: certs/http.p12

# Enable encryption and mutual authenticati
xpack.security.transport.ssl:
  enabled: true
  verification_mode: certificate
  keystore.path: certs/transport.p12
  truststore.path: certs/transport.p12
```

PASO 5 (Configurar los elasticsearch.yml)

Todos los cambios que hay que hacer en los elasticsearch.yml son iguales en las tres máquinas. Hay que añadir las siguientes líneas.

elasticsearch.yml	Nodo-1	Nodo-2	Nodo-3
cluster.name:	g2-reto1		
node.name:	nodo-1	nodo-2	nodo-3
node.roles:	[master, data, ingest]		
network.host:	0.0.0.0		
http.host:		0.0.0.0	0.0.0.0
transport.host:	0.0.0.0		
discovery.seed_hosts:	["192.199.1.59:9300","192.199.1.60:9300","192.199.1.61:9300"]		

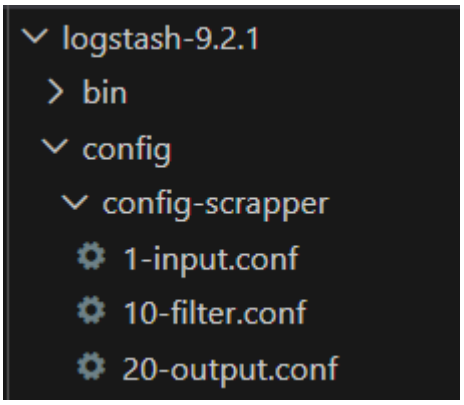
PASO 6 (Limitar el uso de RAM)

En cada nodo de elasticsearch entramos en el archivo (***jvm.options***) y descomentamos y modificamos las siguientes lineas.

```
-Xms2g
-Xmx2g
```

PASO 7 (Configurar logstash)

Lo que hemos hecho es crear una carpeta (***/config-scrapper***) dentro de la carpeta de configuración de losgasth donde creamos los 3 archivos de configuración que vamos a usar como se puede ver en la siguiente foto.



Input

En el archivo de configuración (***1-input.conf***) establecemos de donde cogerá los datos y que reglas tiene que seguir sobre el archivo.

```
input {
  file {
    path =>
"/home/g2/g2r12025scrapper/storage/datasets/default/articles.jsonl"
    start_position => "beginning"
    sincedb_path => "/home/g2/ELK/logstash-sincedb"
    mode => "tail"
    codec => json
  }
}
```

`path`: ruta al archivo que se quiere procesar. En este caso, `articles.jsonl` (JSON Lines).

`start_position => "beginning"`: si es la primera vez que se lee este archivo, empieza desde el **principio** del archivo y no desde el final.

`sincedb_path`: ruta al archivo donde Logstash guarda **el estado de lectura**, para no procesar los mismos datos más de una vez.

`mode => "tail"`: Logstash seguirá leyendo nuevas líneas que se agreguen al archivo.

`codec => json`: indica que cada línea del archivo está en formato JSON y debe ser decodificada como tal.

Filter

En el archivo de configuración (***10-filter.conf***) establecemos los filtros para mejorar la ingesta de datos.

```
filter {
  ruby {
    code => '
      require "net/http"
      require "uri"
      require "json"

      begin
        uri = URI.parse("http://192.199.1.61:8000/api/embed")
        http = Net::HTTP.new(uri.host, uri.port)
        req = Net::HTTP::Post.new(uri.path, { "Content-Type" =>
"application/json" })
        req.body = { text: event.get("body") }.to_json
        res = http.request(req)
        if res.code.to_i == 200
          embedding = JSON.parse(res.body)["embedding"]
          event.set("embedding", embedding)
        else
          event.tag("embedding_failed")
        end
      rescue => e
        event.tag("embedding_error")
        event.set("embedding_error_msg", e.message)
      end
    '
```

```
  }

  # Extraer la parte de la URL
  grok {
    match => { "url" => "https://%{DATA:fuente}/" }
  }

  # Eliminamos campos innecesarios y ponermos fuente en minuscula
  mutate {
    remove_field => ["host", "log", "event", "@version", "message",
"path", "tags"]
    lowercase => ["fuente"]
  }

  # Convertimos date a timestamp
  date {
    match => ["date", "ISO8601"]
    target => "date"
  }
}
```

Este filtro hace cuatro cosas:

Ruby: Este bloque **ejecuta código Ruby dentro del filtro de Logstash** para enviar el campo `body` de cada evento al endpoint de la API donde está el modelo de embeddings. La API devuelve un vector (`embedding`) que Logstash añade al evento; si falla, marca el evento con tags de error.

Grok: toma la URL y extrae solo el dominio, guardándolo en el campo `fuente`.

Mutate: elimina campos que no necesitas y convierte el valor de `fuente` a minúsculas.

Date: transforma el campo `date` de texto a un timestamp que Elasticsearch puede usar.

Output

En el archivo de configuración (***20-output.conf***) establecemos a donde mandará los datos.

```
output {
  elasticsearch {
    hosts =>
["https://192.199.1.59:9200","https://192.199.1.60:9200","https://192.1
99.1.61:9200"]
    api_key => "jAbx3ZoB8_buqiHcbWPg:3ynGrlurs3CySNbLdXXDRg"
```

```
    index => "noticias-%{+YYYY.MM.dd}"
    ssl_enabled => true
    ssl_certificate_authorities =>
"/home/g2/ELK/elasticsearch-9.2.1/config/certs/http_ca.crt"
  }

  stdout {
    codec => rubydebug
  }
}
```

Este bloque `output` de Logstash indica a dónde enviar los datos procesados:

1. **Elasticsearch**:
    - Se conecta a los tres hosts que pones.
    - Usa `api_key` para autenticarse.
    - Guarda los documentos en un índice diario llamado `noticias-AAAA.MM.DD`.
    - Usa SSL y el certificado indicado para la conexión segura.
2. **stdout**: imprime los datos en la consola en formato legible (`rubydebug`) para depuración.