3.5 Configuración inicial

Una vez han sido instalados los servicios de **Elasticsearch** y **Kibana**, procederemos a realizar las primeras modificaciones sobre los archivos de configuración de los mismos, realizando algunos cambios como el bloqueo de la memoria para el servicio, deshabilitar la SWAP en el sistema operativo y publicar ambos servicios fuera de la interfaz interna de *localhost*.

Índice de contenidos

Elasticsearch

https://www.elastic.co/guide/en/elasticsearch/reference/7.17/settings.html

Principales rutas de Elasticsearch (`ls` en cada uno)

```
home:     /usr/share/elasticsearch # tenemos los archivos del propio demonio
binaries: /usr/share/elasticsearch/bin  # ejecutables de Elasticsearch
config:   /etc/elasticsearch       # pasaremos un rato con estos ficheros
env:      /etc/default/elasticsearch #configuración del entorno
data:     /var/lib/elasticsearch    # donde se guardará toda la info que indexemos
logs:     /var/log/elasticsearch    # registros del servicio
plugins:  /usr/share/elasticsearch/plugins
```

Configuración de la memoria RAM para el servicio

Recordamos que debemos destinar el 50% de RAM exclusivo para Elasticsearch y el otro 50% para el resto de tareas del SO. Podemos bloquear una cantidad de memoria RAM para que ningún otro servicio del SO.

Comprobamos memoria disponible en el PC:

```
root@elastic01:~# free -h
               total        used        free      shared  buff/cache   available
Mem:           3,8Gi       2,7Gi       490Mi        0,0Ki       714Mi       977Mi
Swap:          974Mi          0B       974Mi
```

Comprobamos la memoria solicitada por el servicio *elasticsearch* al arrancar:

```
cat /var/log/elasticsearch/elasticsearch.log | grep heap
[2021-09-15T11:58:24,982] [elastic01] heap size [1.9gb],
```

Vamos a bloquear una cantidad de memoria RAM para que ningún otro servicio del SO.
Será necesario realizar modificaciones en varios archivos:

1.- El primer fichero será para indicar la memoria que la JVM bloqueará:

```
# cp jvm.options jvm.options_original
# nano /etc/elasticsearch/jvm.options
```

Descomentamos las líneas y damos 2GiB de mínimo y máximo

```
-Xms2g
-Xmx2g
```

2.- El segundo fichero será para indicar en la configuración de *elasticsearch* que solicite bloquear la memoria:

```
# cp elasticsearch.yml elasticsearch.yml_original
# nano /etc/elasticsearch/elasticsearch.yml
```

Descomentamos la línea.
Nota: Con máquinas de 2 GiB o inferior he tenido problemas de OOM killer (Out of Memory). No habría que bloquear memoria o, habría que aumentar la RAM.

```
bootstrap.memory_lock: true
```

En **CORS** (Cross-Origin Resource Sharing), se utilizan varias cabeceras HTTP para **permitir** y controlar el acceso a **recursos** desde orígenes **cruzados**. Estas cabeceras son enviadas por el servidor en las respuestas HTTP a las **solicitudes** realizadas desde otro dominio. Para poder hacer aplicaciones Web que puedan atacar a nuestro servidor elasticsearch tenemos que permitir el acceso CORS. Dentro del aula, tratándose de un entorno “seguro” en RAL, dejaremos paso a cualquier IP que nos quiera consultar. Añadiremos las siguientes dos líneas a nuestro fichero:

```
http.cors.enabled: true
http.cors.allow-origin: "*"
```

3.- El tercer fichero será para indicar al demonio *elasticsearch* que permita bloquear la memoria solicitada:

```
# cp /etc/default/elasticsearch /etc/default/elasticsearch_original
# nano /etc/default/elasticsearch
```

Descomentamos la línea

```
MAX_LOCKED_MEMORY=unlimited
```

4.- El cuarto fichero será para indicar al SO los límites que permitirá al demonio *elasticsearch* que intente bloquear la memoria solicitada:

```
# nano /usr/lib/systemd/system/elasticsearch.service
```

Añadimos en la sección de memoria la instrucción LimitMEMLOCK=infinity

```
# Specifies the maximum size of virtual memory
```

```
LimitAS=infinity
LimitMEMLOCK=infinity
```

Nota: Al modificar un fichero de inicio de demonio, tenemos que recargar el demonio systemctl:

```
# systemctl daemon-reload
```

5.- Comprobamos que se aplican los cambios:

Nota: A partir de este punto hasta acabar el taller práctico de configuración, sería interesante estar monitorizando el *log* de salida de *elasticsearch.log*

```
# tail -f /var/log/elasticsearch/elasticsearch.log
```

Para aplicar los cambios reiniciamos el demonio *elasticsearch*

```
# systemctl restart elasticsearch
```

Y comprobamos de nuevo el *log* de *elasticsearch*, desde la monitorización o haciendo un *cat*:

```
#  cat  /var/log/elasticsearch/elasticsearch.log  |  grep  heap
[2021-09-15T11:58:24,982]    [elastic01]    heap    size    [1.9gb]
[2021-09-15T13:34:45,211]    [elastic01]    heap    size    [2gb]
```

Y memoria disponible en el PC:

```
root@elastic01:~# free -h
      total     used      free     shared    buff/cache  available
Mem:  3,8Gi    2,8Gi     189Mi       0,0Ki       872Mi       655Mi
Swap: 974Mi       0B     974Mi
```

6.- El último de los cambios a aplicar a la configuración de la memoria es deshabilitar el uso de memoria SWAP ya que puede afectar muy negativamente al rendimiento del clúster debido a que es mucho más lenta que la memoria RAM. Para ello será necesario comentar la línea de configuración de la SWAP en el archivo `/etc/fstab` y después reiniciar la máquina:

```
# nano /etc/fstab
# reboot
```

Comprobamos la memoria ahora disponible:

```
root@elastic01:~# free -h
      total     used      free     shared    buff/cache  available
Mem:  3,8Gi    2,8Gi     255Mi      0,0Ki       771Mi       633Mi
Swap:    0B       0B       0B
```

Configuración del demonio

Veamos los parámetros a configurar en el servicio para que pertenezca el nodo a un clúster compuesto únicamente por él (de momento), cambiar el nombre del nodo dentro del servicio y publicarlo fuera de la interfaz interna.

Fichero a modificar `/etc/elasticsearch/elasticsearch.yml`

```
# cp elasticsearch.yml elasticsearch.yml_original
# nano elasticsearch.yml
```

```
# --------- Cluster ---------
cluster.name: elasticBDA #utilizad el nombre de vuestro equipo de alumnado

# --------- Node ---------
# Indicaremos el siguiente parámetro aunque la versión que trabajamos ya obtiene por defecto el hostname. Se puede comprobar que así lo hace en el log de elasticsearch.
node.name: ${HOSTNAME} #ojo que cambiará el fichero de log
#node.attr.rack: rack1 # Es una etiqueta que podemos poner cuando tenemos muchos servidores en diferentes racks. Es meramente un identificador para orientar su ubicación física.

# --------- Memory ---------
bootstrap.memory_lock: true

# --------- Network ---------
#Aquí tenéis que poner la IP estática del servidor donde escucha las solicitudes de la red.
network.host: [“127.0.0.1”, “10.0.0.10”]

# --------- Discovery ---------
#discovery.seed_hosts: ["host1", "host2"]
#Proporciona una lista de las direcciones de los nodos elegibles como maestros en el clúster.
discovery.seed_hosts: ${HOSTNAME}
#cluster.initial_master_nodes: ["node-1", "node-2"]
#Establece el conjunto inicial de nodos elegibles como maestros en un clúster.
cluster.initial_master_nodes: ${HOSTNAME}

# --------- Various ---------
# Para evitar el borrado de todos los índices (BD) por error. Se verá en la unidad siguiente en la sección: Delete Index  API
action.destructive_requires_name: true
```

Para aplicar los cambios reiniciamos el demonio *elasticsearch*

```
# systemctl restart elasticsearch
```

Para comprobar el estado del clúster se usará la siguiente consulta de la API desde la terminal, la cual ya se estudiará en módulos posteriores:

```
curl -XGET http://10.0.0.10:9200/_cluster/health?pretty
```

Kibana

Principales rutas de Kibana

```
home:     /usr/share/kibana        # tenemos los archivos del propio demonio
binaries: /usr/share/kibana/bin    # ejecutables de Kibana
config:   /etc/kibana              # pasaremos un rato con estos ficheros
data:     /var/lib/kibana
```

```
optimize: /usr/share/kibana/optimize
logs:     /var/log/kibana          # registros del servicio
plugins:  /usr/share/kibana/plugins
```

Configuración de los parámetros de Kibana

Veamos ahora los parámetros a configurar en el servicio de *Kibana* para que se conecte al nodo de *Elasticsearch*, publique el servicio a la interfaz externa y escriba sus *logs* en el correspondiente directorio.

Fichero a modificar `/etc/kibana/kibana.yml`

```
# cp kibana.yml kibana.yml_original
# nano kibana.yml
```

```
# Interfaz donde atiende peticiones. (No he podido ponerle más de una interfaz)
server.host: "10.0.0.10"

# En el siguiente parámetro especificaremos la interfaz de conexión con elasticsearch. En primera instancia no tuve que indicarlo pero al trasladar las máquinas virtuales a otras instalaciones tuve que especificar la IP del servidor elasticsearch.
elasticsearch.url: "http://10.0.0.10:9200"

kibana.index: ".kibana"

#En versiones anteriores había que especificar el fichero de logs:
#(El directorio /var/log/kibana debe de existir con permisos rwx r-s ---)
# logging.dest: /var/log/kibana/kibana.log
```

Para aplicar los cambios reiniciamos el demonio *Kibana*

```
# systemctl restart kibana
```