Add an Elasticsearch node to Snap4City

Overview

- New node
- Generate new node certificate
- Configure the new node
- Update existing nodes with the new node identity
- Restart the existing cluster nodes
- Start the new node
- Update the Nifi truststores

Create new node

This guide assumes a new node created as a VM clone of one of the pre-existing Elasticsearch nodes in the current Snap4City Elasticsearch cluster.

Node configurations:

- Elasticsearch config file: `/etc/elasticsearch/elasticsearch.yml`
- Elasticsearch data directory: `/var/lib/elasticsearch`

Generate new node certificate

The folder `elastic7-cluster-produzione/cert-elastic7` contains the files needed to create a certificate for the new node. The layout of such folder:

```
elastic7-cluster-produzione/cert-elastic7/
├── keystore
├── pem
└── truststore
```

where:

- `keystore` contains the keystores of the admin, datamanager and all the nodes.
- `pem` contains the certificates in PEM format.
- `truststore` contains `truststore.jks` the truststore (common to all nodes).

Let's assume the new node being `node-8`.

First create a folder for the new node certificate under `pem`:

```
cd elastic7-cluster-produzione/cert-elastic7/pem
mkdir node-8
cd node-8
```

The new node needs a certificate signed with the same key as the other cluster nodes.

```
openssl genrsa -out node-8-key-temp.pem 2048
openssl pkcs8 -inform PEM -outform PEM -in node-8-key-temp.pem -topk8 -nocrypt -v1 PBE-SHA1-3DES -out node-8-
openssl req -new -key node-8-key.pem -days 10950 -out node-8.csr
```

Create a `template.cnf` file to add the IP of the new generated node as SAN, with content:

```
[SAN]
subjectAltName='IP:<NEW_NODE_IP>'
```

Insert the SAN in the generated certificate with:

```
openssl x509 -req -in node-8.csr \
  -CA cert-elastic7/pem/root-ca/root-ca.pem \
  -CAkey cert-elastic7/pem/root-ca/root-ca-key.pem \
  -CAcreateserial -sha256 -days 10950 -extensions SAN -extfile template.cnf \
  -out node-8.crt
```

Convert the generated certificate in PEM:

```
openssl x509 -in node-8.crt -out node-8.pem -outform PEM
```

Keystore

Generate the keystore for the new node

```
openssl pkcs12 -export -in node-8.pem -inkey node-8-key.pem -certfile node-8.pem -out node8_keystore.p12
```

Now the new folder `elastic7-cluster-produzione/cert-elastic7/pem` should contain the following files:

- `node-8.crt`
- `node-8.csr`
- `node-8-key.pem`
- `node-8-key-temp.pem`
- `node-8.pem`
- `node8_keystore.p12`

Now the keystore must be converted to **JKS**.
Create a folder for the new JKS keystore under `elastic7-cluster-produzione/cert-elastic7/keystore`:

```
cd elastic7-cluster-produzione/cert-elastic7/keystore
mkdir node-8
cd node-8
```

Then convert the PKCS12 keystore created in the `elastic7-cluster-produzione/cert-elastic7/pem` folder with:

```
keytool -importkeystore \
  -srckeystore ../pem/node8_keystore.p12 -srcstoretype pkcs12 \
  -destkeystore node8_keystore.jks -deststoretype JKS

keytool -changealias -alias "1" -destalias "elastic7-node-8" -keystore node8_keystore.jks
```

The `node8_keystore.jks` will be created in the current folder (`elastic7-cluster-produzione/cert-elastic7/keystore/node-8`).

Truststore

Finally the identity of the new node contained in the keystore must be added to the truststore (common to all nodes) in `elastic7-cluster-produzione/cert-elastic7/truststore/truststore.jks` :

```
cd elastic7-cluster-produzione/cert-elastic7/truststore
keytool -import -file ../pem/node-8/node-8.crt \
  -alias elastic7-node-8 \
  -keystore truststore.jks
```

To verify the truststore inspect it with:

```
keytool -list -v -keystore truststore.jks
```

Configure the new node

To configure the new node:

- Clear the cloned node data
- Copy the new generated keystore and truststore
- Edit `/etc/elasticsearch/elasticsearch.yml`

IMPORTANT: clear the cloned node data

If the cloned node contains data, these must be cleared to avoid data duplication or problems with the document versioning when the new nodes joins the cluster.

To clear the node's data delete the directory configured as data directory in the `/etc/elasticsearch.yml` configuration file:

```
path.data: /var/lib/elasticsearch
```

```
sudo rm -r /var/lib/elasticsearch
```

Then re-create such directory and set the correct ownership and permissions:

```
sudo mkdir /var/lib/elasticsearch
sudo chown elasticsearch:elasticsearch /var/lib/elasticsearch
sudo chmod o-rx /var/lib/elasticsearch
sudo chmod g+s /var/lib/elasticsearch
```

Copy the new generated keystore and trustore

The keystore generated for the new node in `elastic7-cluster-produzione/cert-elastic7/keystore/node-8/node8_keystore.jks` must be copied to the new node and placed inside the Elasticsearch configuration folder `/etc/elasticsearch` .

Also, the keystore `/etc/elasticsearch/truststore.jks` must be replaced with the updated trustore in `elastic7-cluster-produzione/cert-elastic7/trustore/truststore.jks`.

Edit `/etc/elasticsearch/elasticsearch.yml`

Configure the new node ip:

```
network.host: <NEW_NODE_IP>
```

Configure the new keystore (the truststore name does not need to be updated):

```
opendistro_security.ssl.transport.keystore_filepath: node8_keystore.jks
opendistro_security.ssl.http.keystore_filepath: node8_keystore.jks
```

Add the subject of the new node under the `opendistro_security.nodes_dn` configuration.
The subject of the new node can be obtained inspecting the PEM certificate `node-8.pem` with:

```
openssl x509 -subject -nameopt RFC2253 -noout -in elastic7-cluster-produzione/cert-elastic7/pem/node-8/node-8
```

output:

```
subject=CN=Snap4City-Elastic7-Shard-8,OU=Disit Lab,O=University Of Florence,L=Florence,ST=Tuscany,C=IT
```

after adding the new node subject:

```
opendistro_security.nodes_dn:
  - 'CN=Snap4City-Elastic7-Shard-1,OU=Disit Lab,O=University Of Florence,L=Florence,ST=Tuscany,C=IT'
  - 'CN=Snap4City-Elastic7-Shard-2,OU=Disit Lab,O=University Of Florence,L=Florence,ST=Tuscany,C=IT'
  - 'CN=Snap4City-Elastic7-Shard-3,OU=Disit Lab,O=University Of Florence,L=Florence,ST=Tuscany,C=IT'
  - 'CN=Snap4City-Elastic7-Shard-4,OU=Disit Lab,O=University Of Florence,L=Florence,ST=Tuscany,C=IT'
  - 'CN=Snap4City-Elastic7-Shard-5,OU=Disit Lab,O=University Of Florence,L=Florence,ST=Tuscany,C=IT'
  - 'CN=Snap4City-Elastic7-Shard-6,OU=Disit Lab,O=University Of Florence,L=Florence,ST=Tuscany,C=IT'
  - 'CN=Snap4City-Elastic7-Shard-7,OU=Disit Lab,O=University Of Florence,L=Florence,ST=Tuscany,C=IT'
  - 'CN=Snap4City-Elastic7-Shard-8,OU=Disit Lab,O=University Of Florence,L=Florence,ST=Tuscany,C=IT'
```

Update the existing node with the new node identity

In order let the new node join the cluster, the other nodes must register its identity.
This consists in:

1. Replace the truststore `/etc/elasticsearch/trustore.jks` with the updated truststore in `elastic7-cluster-produzione/cert-elastic7/trustore/trustore.jks` on **all** the old cluster nodes.

2. Update `/etc/elasticsearch/elasticsearch.yml` by adding the subject of the new node certificate under the `opensearch_security.nodes_dn` configuration.

Restart the existing cluster nodes

The identity of the new node will be available to the nodes only after a restart.
This because Elasticsearch will reload the `elasticsearch.yml` file and the trustore.

The general Elasticsearch guidelines suggest to **NOT restart all the master nodes at the same time** .
In general is better to not take out too many master nodes from the cluster.
The **safest procedure is to restart one node at a time** to avoid data rebalance across the cluster.

When all the nodes have been restarted, the new node is able to join the cluster.

Start the new node

Start the new node using

```
sudo service elasticsearch start
```

To keep an eye on the starting process you can inspect the logfile:

```
sudo watch tail -n 30 /var/log/elastisearch/elastic7-cluster.log
```

To ensure the node has joined the cluster, list the cluster nodes by running from the kibana console:

```
GET _cat/nodes?v
```

Once the new node has joined, the cluster will start a shard rebalance process to move some shards from the old nodes to the new node.
This process can be tracked using the endpoint `_cluster/health`, from the kibana console:

```
GET _cluster/health
```

The response will looks like:

```
{
  "cluster_name" : "elastic7-cluster",
  "status" : "yellow",
  "timed_out" : false,
  "number_of_nodes" : 8,
  "number_of_data_nodes" : 8,
  "active_primary_shards" : 659,
  "active_shards" : 1323,
  "relocating_shards" : 2,
  "initializing_shards" : 0,
  "unassigned_shards" : 0,
  "delayed_unassigned_shards" : 0,
  "number_of_pending_tasks" : 1,
  "number_of_in_flight_fetch" : 0,
  "task_max_waiting_in_queue_millis" : 0,
  "active_shards_percent_as_number" : 100.0
}
```

where the `relocating_shards` field reports the number of shards that are being relocating.
Usually elasticsearch relocates 2 shards at a time.

This process will end when the shards are evenly balanced across the cluster.
To view the shards allocation, from the kibana console:

```
GET _cat/shards?v
```

Update the Nifi truststore

To be able to insert documents in the new node, the Nifi cluster needs to have the new node certificate in the truststore used for inserting in Elasticsearch.
On the nifi nodes this truststore is:

```
 /home/debian/elastic_truststores/new_elastic/nodes_keystore.jks
```

Copy the `elastic7-cluster-produzione/cert-elastic7/keystore/node-8/node8_keystore.jks` to `/home/debian/elastic_truststores/new_elastic` on every nifi node.

Similarly to the Elasticsearch nodes truststore import the truststore created for the new node:

```
cd /home/debian/elastic_truststores/new_elastic
keytool -importkeystore \
  -srckeystore  node8_keystore.jks -srcstoretype JKS \
  -destkeystore nodes_keystore.jks -deststoretype JKS
```

Once the truststore has been update on all Nifi nodes, disable and re-enable the controller service `ElasticInsertSSL_New_Cluster` inside Nifi (this will also restart the processors which uses it).

Now the `PutElasticsearchHttp` processors inside the Nifi dataflow should be able to insert documents in the indices by issuing requests to the new node.