**INSTALLING ELASTICSEARCH USING ANSIBLE – THE QUICK WAY!**

* Get the repository accessible with the following command:

ansible-galaxy install elastic.elasticsearch,7.4.2

* Create Inventory file inventory.yml as below:

[hosts]

esdata1

esdata2

esdata3

escoord1

escoord2

esmaster1

esmaster2

esmaster3

[data]

esdata1

esdata2

esdata3

[masters]

esmaster1

esmaster2

esmaster3

[coordinating]

escoord1

escoord2

* Create the elastic.yml file in order to have our cluster deployed.

- hosts: masters

roles:

- role: elastic.elasticsearch

vars:

es\_heap\_size: "8g"

es\_config:

cluster.name: "esprd"

network.host: 0

cluster.initial\_master\_nodes: "esmaster1,esmaster2,esmaster3"

discovery.seed\_hosts: "esmaster1:9300,esmaster2:9300,esmaster3:9300"

http.port: 9200

node.data: false

node.master: true

node.ingest: false

node.ml: false

cluster.remote.connect: false

bootstrap.memory\_lock: true

- hosts: data

roles:

- role: elastic.elasticsearch

vars:

es\_data\_dirs:

- "/var/lib/elasticsearch"

es\_heap\_size: "30g"

es\_config:

cluster.name: "esprd"

network.host: 0

discovery.seed\_hosts: "esmaster1:9300,esmaster2:9300,esmaster3:9300"

http.port: 9200

node.data: true

node.master: false

node.ml: false

bootstrap.memory\_lock: true

indices.recovery.max\_bytes\_per\_sec: 100mb

- hosts: coordinating

roles:

- role: elastic.elasticsearch

vars:

es\_heap\_size: "16g"

es\_config:

cluster.name: "esprd"

network.host: 0

discovery.seed\_hosts: "esmaster1:9300,esmaster2:9300,esmaster3:9300"

http.port: 9200

node.data: false

node.master: false

node.ingest: false

node.ml: false

cluster.remote.connect: false

bootstrap.memory\_lock: true

* Execute below command:

ansible-playbook elastic.yml -i inventory.yml

* After a few minutes, the above command should get us a nice Elasticsearch cluster up and running. We can verify its status like this:

$ curl -XGET 'https://esmaster1'

{

"name" : "esmaster1",

"cluster\_name" : "esprd",

"cluster\_uuid" : "ABC6pGHgRWGhooEjvIElkA",

"version" : {

"number" : "7.4.2",

"build\_flavor" : "default",

"build\_type" : "rpm",

"build\_hash" : "7a013de",

"build\_date" : "2019-12-07T14:04:00.380842Z",

"build\_snapshot" : false,

"lucene\_version" : "8.0.0",

"minimum\_wire\_compatibility\_version" : "6.8.0",

"minimum\_index\_compatibility\_version" : "6.0.0-beta1"

},

"tagline" : "You Know, for Search"

}

* Furthermore, we can confirm cluster health as well:

$ curl -XGET 'https://esmaster1:9200/\_cluster/health?pretty'

{

"cluster\_name" : "esprd",

"status" : "green",

"timed\_out" : false,

"number\_of\_nodes" : 8,

"number\_of\_data\_nodes" : 3,

"active\_primary\_shards" : 0, "active\_shards" : 0,

"relocating\_shards" : 0,

"initializing\_shards" : 0,

"unassigned\_shards" : 0,

"delayed\_unassigned\_shards" : 0,

"number\_of\_pending\_tasks" : 0,

"number\_of\_in\_flight\_fetch" : 0,

"task\_max\_waiting\_in\_queue\_millis" : 0,

"active\_shards\_percent\_as\_number" : 100.0

}

6 STEPS TO SECURE ELASTICSEARCH:

### 1. Lock Down Open Ports

**Firewall: Close the public ports**

The first action should be to close the relevant ports to the Internet:

iptables -A INPUT -i eth0 -p tcp --destination-port **9200** -s {**PUBLIC**-IP-ADDRESS-HERE} -j DROP

iptables -A INPUT -i eth0 -p tcp --destination-port **9300** -s {**PUBLIC**-IP-ADDRESS-HERE} -j DROP

If you run Kibana note that the Kibana server acts as a proxy to Elasticsearch and thus needs its port closed as well:

iptables -A INPUT -i eth0 -p tcp --destination-port **5601** -s {**PUBLIC**-IP-ADDRESS-HERE} -j DROP

After this you can relax a bit! Elasticsearch won’t not reachable from the Internet anymore.

**Bind Elasticsearch ports only to private IP addresses**

Change the configuration in elasticsearch.yml to bind only to private IP addresses or for single node instances to the loopback interface:

**network.bind\_host**: 127**.0.0.1**

### 2. Add private networking between Elasticsearch and client services

There are several open-source and free solutions that provide Elasticsearch access authentication, but if you want something quick and simple, here is how to do it yourself with just Nginx:

**ssh** **-Nf** **-L** 9200:localhost:9200 **user**@**remote**-**elasticsearch**-**server**

You can then access Elasticsearch via the SSH tunnel with from client machines e.g.

curl http:*//localhost:9200/\_search*

### 3. Set up authentication and SSL/TLS with Nginx

There are several open-source and free solutions that provide Elasticsearch access authentication, but if you want something quick and simple, here is how to do it yourself with just Nginx:

Generate password file

printf **"esuser:$(openssl passwd -crypt MySecret)\n"** > /etc/nginx/passwords

Generate self-signed SSL certificates, if you don’t have official certificates.

sudo **mkdir /etc/nginx/ssl**

sudo **openssl req -x509 -nodes -days 365 -newkey rsa:2048 -keyout**

/etc/nginx/ssl/nginx.key **-out /etc/nginx/ssl/nginx.crt**

Add the proxy configuration with SSL and activate basic authentication to /etc/nginx/nginx.conf (note we expect the SSL certificate and key file in /etc/nginx/ssl/). Example:

*# define proxy upstream to Elasticsearch via loopback interface in*

**http** {

**upstream** elasticsearch {

**server** **127.0.0.1:9200**;

}

}

**server** {

*# enable TLS*

**listen** **0.0.0.0:443** ssl;

**ssl\_certificate** /etc/nginx/ssl/nginx.crt;

**ssl\_certificate\_key** /etc/nginx/ssl/nginx.key

ssl\_protocols TLSv1.**2**;

**ssl\_prefer\_server\_ciphers** on;

**ssl\_session\_timeout** **5m**;

**ssl\_ciphers** **"HIGH:!aNULL:!MD5 or HIGH:!aNULL:!MD5:!3DES"**;

*# Proxy for Elasticsearch*

**location** / {

**auth\_basic** **"Login"**;

**auth\_basic\_user\_file** passwords;

**proxy\_set\_header** X-Real-IP $remote\_addr;

**proxy\_set\_header** X-Forwarded-For $proxy\_add\_x\_forwarded\_for;

**proxy\_set\_header** Host $http\_host;

**proxy\_set\_header** X-NginX-Proxy true;

*# use defined upstream with the name "elasticsearch"*

**proxy\_pass** http://elasticsearch/;

**proxy\_redirect** off;

**if** ($request\_method = OPTIONS ) {

**add\_header** Access-Control-Allow-Origin **"\*"**;

**add\_header** Access-Control-Allow-Methods **"GET, POST, , PUT, OPTIONS"**;

**add\_header** Access-Control-Allow-Headers **"Content-Type,Accept,Authorization, x-requested-with"**;

**add\_header** Access-Control-Allow-Credentials **"true"**;

**add\_header** Content-Length **0**;

**add\_header** Content-Type application/json;

**return** **200**;

}

}

Restart Nginx and try to access Elasticsearch via [https://localhost/\_search.](https://your-public-ip/)

### 4. Install Free Security Plugins for Elasticsearch

Alternatively, you could install and configure one of the several free security plugins for Elasticsearch to enable authentication:

* [ReadonlyREST plugin](https://github.com/sscarduzio/elasticsearch-readonlyrest-plugin) for Elasticsearch is available on Github. It provides different types of authentication, from basic to LDAP, as well as index- and operation-level access control.
* [SearchGuard](https://github.com/floragunncom/search-guard/wiki) is a [free security plugin for Elasticsearch](https://sematext.com/blog/search-guard-security-for-elasticsearch/) including role-based access control and SSL/TLS encrypted node-to-node communication. Additional enterprise features like LDAP authentication or JSON Web Token authentication are available and licensed per Elasticsearch cluster. See how simple it is to install and configure [SearchGuard to secure an Elasticsearch and Kibana](https://sematext.com/blog/elasticsearch-kibana-security-search-guard/) setup. Note that SearchGuard support is also included in some [Sematext Elasticsearch Support Subscriptions](https://sematext.com/support/elasticsearch-production-support/).
* [Open Distro for Elasticsearch](https://github.com/opendistro-for-elasticsearch) from AWS has free security plugins for Elasticsearch and Kibana. Open Distro provides a rich set of features to ensure data security and compliance with regulations such as GDPR, HIPAA, PCI, and ISO. Some of its main capabilities include using Kerberos or JSON tokens for single sign-on (SSO), encrypting data-in-transit, authenticating users against Active Directory, or monitoring and logging any malicious attempts. Read our [complete guide to Open Ditro for Elasticsearch](https://sematext.com/blog/open-distro-elasticsearch-review) and learn more.

### 5. Maintain an audit trail and set up alerts

As with any type of system holding sensitive data, you have to monitor it very closely.  This means not only monitoring its various metrics (whose sudden changes could be an early sign of trouble), but also watching its logs.  Concretely, in the recent Elasticsearch attacks, anyone who had alert rules that trigger when the number of documents in an index suddenly drops would have immediately been notified that something was going on. A number of monitoring vendors have Elasticsearch support, including Sematext (see [Elasticsearch monitoring](https://sematext.com/integrations/elasticsearch-monitoring/)integration).

Logs should be collected and shipped to a [log management](https://sematext.com/guides/log-management/) service in real time, where alerting needs to be set up to watch for any anomalous or suspicious activity, among other things. The log management service can be on premises or it can be a 3rd party SaaS, like [Sematext Cloud](https://sematext.com/logsene). Shipping logs off site has the advantage of preventing attackers from covering their tracks by changing the logs.  Once logs are off site attackers won’t be able to get to them. Alerting on metrics and logs means you will become aware of a security compromise early and take appropriate actions to, hopefully, prevent further damage.

### 6. Backup and restore data

A very handy tool to backup/restore or re-index data based on Elasticsearch queries is [Elasticdump](https://www.npmjs.com/package/elasticdump" \t "_blank).

To backup complete indices, the Elasticsearch snapshot API is the right tool. The snapshot API provides operations to create and restore snapshots of whole indices, stored in files, or in Amazon S3 buckets.

Let’s have a look at a few examples for Elasticdump and snapshot backups and recovery.

1. Install elasticdump with the node package manager

npm i elasticdump -g

1. Backup by query to a zip file:

**elasticdump** --input='http://username:password@localhost:9200/myindex' --searchBody '{**"query"** : {**"range"** :{**"timestamp"** : {**"lte"**: 1483228800000}}}}' --output=$ --limit=1000 | gzip > /backups/myindex.gz

1. Restore from a zip file:

zcat /backups/myindex.gz | elasticdump --input=$ --output=http://username:password@localhost:**9200**/index\_name