# ELK

## Install ELK Using Docker-compose

1. Docker-compose.yml

services:

setup:

image: docker.elastic.co/elasticsearch/elasticsearch:${STACK\_VERSION}

volumes:

- certs:/usr/share/elasticsearch/config/certs

user: "0"

restart: no # Prevents restart, allowing it to exit without impacting other services

command: >

bash -c '

if [ x${ELASTIC\_PASSWORD} == x ]; then

echo "Set the ELASTIC\_PASSWORD environment variable in the .env file";

exit 1;

elif [ x${KIBANA\_PASSWORD} == x ]; then

echo "Set the KIBANA\_PASSWORD environment variable in the .env file";

exit 1;

fi;

if [ ! -f config/certs/ca.zip ]; then

echo "Creating CA";

bin/elasticsearch-certutil ca --silent --pem -out config/certs/ca.zip;

unzip config/certs/ca.zip -d config/certs;

fi;

if [ ! -f config/certs/certs.zip ]; then

echo "Creating certs";

echo -ne \

"instances:\n"\

" - name: es01\n"\

" dns:\n"\

" - es01\n"\

" - localhost\n"\

" ip:\n"\

" - 127.0.0.1\n"\

" - name: es02\n"\

" dns:\n"\

" - es02\n"\

" - localhost\n"\

" ip:\n"\

" - 127.0.0.1\n"\

" - name: es03\n"\

" dns:\n"\

" - es03\n"\

" - localhost\n"\

" ip:\n"\

" - 127.0.0.1\n"\

> config/certs/instances.yml;

bin/elasticsearch-certutil cert --silent --pem -out config/certs/certs.zip --in config/certs/instances.yml --ca-cert config/certs/ca/ca.crt --ca-key config/certs/ca/ca.key;

unzip config/certs/certs.zip -d config/certs;

fi;

echo "Setting file permissions"

chown -R root:root config/certs;

find . -type d -exec chmod 750 \{\} \;;

find . -type f -exec chmod 640 \{\} \;;

echo "Waiting for Elasticsearch availability";

until curl --cacert config/certs/ca/ca.crt https://es01:9200 | grep -q "missing authentication credentials"; do sleep 30; done;

echo "Setting kibana\_system password";

until curl -X POST --cacert config/certs/ca/ca.crt -u "elastic:${ELASTIC\_PASSWORD}" -H "Content-Type: application/json" https://es01:9200/\_security/user/kibana\_system/\_password -d "{\"password\":\"${KIBANA\_PASSWORD}\"}" | grep -q "^{}"; do sleep 10; done;

echo "All done!";

'

healthcheck:

test: ["CMD-SHELL", "[ -f config/certs/es01/es01.crt ]"]

interval: 1s

timeout: 5s

retries: 120

networks:

- elk\_default

es01:

depends\_on:

setup:

condition: service\_healthy

image: docker.elastic.co/elasticsearch/elasticsearch:${STACK\_VERSION}

volumes:

- certs:/usr/share/elasticsearch/config/certs

- esdata01:/usr/share/elasticsearch/data

ports:

- "9200:9200"

environment:

- node.name=es01

- cluster.name=${CLUSTER\_NAME}

- cluster.initial\_master\_nodes=es01,es02,es03

- discovery.seed\_hosts=es02,es03

- ELASTIC\_PASSWORD=${ELASTIC\_PASSWORD}

- bootstrap.memory\_lock=true

- xpack.security.enabled=true

- xpack.security.http.ssl.enabled=true

- xpack.security.http.ssl.key=certs/es01/es01.key

- xpack.security.http.ssl.certificate=certs/es01/es01.crt

- xpack.security.http.ssl.certificate\_authorities=certs/ca/ca.crt

- xpack.security.transport.ssl.enabled=true

- xpack.security.transport.ssl.key=certs/es01/es01.key

- xpack.security.transport.ssl.certificate=certs/es01/es01.crt

- xpack.security.transport.ssl.certificate\_authorities=certs/ca/ca.crt

- xpack.security.transport.ssl.verification\_mode=certificate

- xpack.license.self\_generated.type=${LICENSE}

- "ES\_JAVA\_OPTS=-Xms2g -Xmx2g"

mem\_limit: 4g

mem\_reservation: 2g

# mem\_limit: ${MEM\_LIMIT}

# sysctls:

# - "vm.max\_map\_count=262144"

ulimits:

memlock:

soft: -1

hard: -1

healthcheck:

test:

[

"CMD-SHELL",

"curl -s --cacert config/certs/ca/ca.crt https://localhost:9200 | grep -q 'missing authentication credentials'",

]

interval: 10s

timeout: 10s

retries: 120

networks:

- elk\_default

es02:

depends\_on:

- es01

image: docker.elastic.co/elasticsearch/elasticsearch:${STACK\_VERSION}

volumes:

- certs:/usr/share/elasticsearch/config/certs

- esdata02:/usr/share/elasticsearch/data

environment:

- node.name=es02

- cluster.name=${CLUSTER\_NAME}

- cluster.initial\_master\_nodes=es01,es02,es03

- discovery.seed\_hosts=es01,es03

- bootstrap.memory\_lock=true

- xpack.security.enabled=true

- xpack.security.http.ssl.enabled=true

- xpack.security.http.ssl.key=certs/es02/es02.key

- xpack.security.http.ssl.certificate=certs/es02/es02.crt

- xpack.security.http.ssl.certificate\_authorities=certs/ca/ca.crt

- xpack.security.transport.ssl.enabled=true

- xpack.security.transport.ssl.key=certs/es02/es02.key

- xpack.security.transport.ssl.certificate=certs/es02/es02.crt

- xpack.security.transport.ssl.certificate\_authorities=certs/ca/ca.crt

- xpack.security.transport.ssl.verification\_mode=certificate

- xpack.license.self\_generated.type=${LICENSE}

- "ES\_JAVA\_OPTS=-Xms2g -Xmx2g"

mem\_limit: 4g

mem\_reservation: 2g

# mem\_limit: ${MEM\_LIMIT}

# sysctls:

# - "vm.max\_map\_count=262144"

ulimits:

memlock:

soft: -1

hard: -1

healthcheck:

test:

[

"CMD-SHELL",

"curl -s --cacert config/certs/ca/ca.crt https://localhost:9200 | grep -q 'missing authentication credentials'",

]

interval: 10s

timeout: 10s

retries: 120

networks:

- elk\_default

es03:

depends\_on:

- es02

image: docker.elastic.co/elasticsearch/elasticsearch:${STACK\_VERSION}

volumes:

- certs:/usr/share/elasticsearch/config/certs

- esdata03:/usr/share/elasticsearch/data

environment:

- node.name=es03

- cluster.name=${CLUSTER\_NAME}

- cluster.initial\_master\_nodes=es01,es02,es03

- discovery.seed\_hosts=es01,es02

- bootstrap.memory\_lock=true

- xpack.security.enabled=true

- xpack.security.http.ssl.enabled=true

- xpack.security.http.ssl.key=certs/es03/es03.key

- xpack.security.http.ssl.certificate=certs/es03/es03.crt

- xpack.security.http.ssl.certificate\_authorities=certs/ca/ca.crt

- xpack.security.transport.ssl.enabled=true

- xpack.security.transport.ssl.key=certs/es03/es03.key

- xpack.security.transport.ssl.certificate=certs/es03/es03.crt

- xpack.security.transport.ssl.certificate\_authorities=certs/ca/ca.crt

- xpack.security.transport.ssl.verification\_mode=certificate

- xpack.license.self\_generated.type=${LICENSE}

- "ES\_JAVA\_OPTS=-Xms2g -Xmx2g"

mem\_limit: 4g

mem\_reservation: 2g

# mem\_limit: ${MEM\_LIMIT}

# sysctls:

# - "vm.max\_map\_count=262144"

ulimits:

memlock:

soft: -1

hard: -1

healthcheck:

test:

[

"CMD-SHELL",

"curl -s --cacert config/certs/ca/ca.crt https://localhost:9200 | grep -q 'missing authentication credentials'",

]

interval: 10s

timeout: 10s

retries: 120

networks:

- elk\_default

kibana:

depends\_on:

es01:

condition: service\_healthy

es02:

condition: service\_healthy

es03:

condition: service\_healthy

image: docker.elastic.co/kibana/kibana:${STACK\_VERSION}

volumes:

- certs:/usr/share/kibana/config/certs

- kibanadata:/usr/share/kibana/data

# - ./kibana/kibana.yml:/usr/share/kibana/config/kibana.yml:ro

ports:

- ${KIBANA\_PORT}:5601

environment:

- SERVERNAME=kibana

- ELASTICSEARCH\_HOSTS=https://es01:9200

- ELASTICSEARCH\_USERNAME=kibana\_system

- ELASTICSEARCH\_PASSWORD=${KIBANA\_PASSWORD}

- ELASTICSEARCH\_SSL\_CERTIFICATEAUTHORITIES=config/certs/ca/ca.crt

mem\_limit: 4g

healthcheck:

test:

[

"CMD-SHELL",

"curl -s -I http://localhost:5601 | grep -q 'HTTP/1.1 302 Found'",

]

interval: 10s

timeout: 10s

retries: 120

networks:

- elk\_default

logstash:

user: "0" # Run as root for debugging purposes

image: docker.elastic.co/logstash/logstash:${STACK\_VERSION}

volumes:

# - ./logstash.yml:/usr/share/logstash/config/logstash.yml

- ./logstash:/usr/share/logstash/pipeline

- certs:/usr/share/logstash/config/certs

# - ./batch-logstash-local.conf:/usr/share/logstash/pipeline/logstash.conf

environment:

- ELASTICSEARCH\_URL=${ELASTICSEARCH\_URL}

- ELASTIC\_USERNAME=${ELASTIC\_USERNAME}

- ELASTIC\_PASSWORD=${ELASTIC\_PASSWORD}

- ELASTIC\_TIMEOUT=${ELASTIC\_TIMEOUT}

- xpack.monitoring.elasticsearch.username=${ELASTIC\_USERNAME}

- xpack.monitoring.elasticsearch.password=${ELASTIC\_PASSWORD}

- ELASTICSEARCH\_HOST=${ELASTICSEARCH\_URL}

# - LOGSTASH\_CONFIG\_FILE=/usr/share/logstash/pipeline/logstash.conf

ports:

- "9600:9600"

command: >

bash -c "

logstash -f /usr/share/logstash/pipeline/${LOGSTASH\_CONFIG\_FILE}

"

networks:

- elk\_default

networks:

elk\_default:

driver: bridge

volumes:

certs:

driver: local

esdata01:

driver: local

esdata02:

driver: local

esdata03:

driver: local

kibanadata:

driver: local

1. Logstash-enriched.conf

input {

file {

path => "/usr/share/logstash/pipeline/customers-1000.csv"

start\_position => "beginning"

sincedb\_path => "/dev/null"

codec => plain {

charset => "UTF-8"

}

}

file {

path => "/usr/share/logstash/pipeline/people-1000.csv"

start\_position => "beginning"

sincedb\_path => "/dev/null"

codec => plain {

charset => "UTF-8"

}

}

}

filter {

csv {

separator => ","

skip\_header => "true"

columns => [

"Index", "User Id", "First Name", "Last Name", "Company", "City", "Country",

"Subscription Date", "Website"

]

add\_field => { "source" => "customers" }

}

if [source] == "customers" {

mutate {

convert => { "Index" => "integer" }

}

# Ensure the 'Subscription Date' field is parsed into a date format

date {

match => ["Subscription Date", "dd/MM/yyyy"]

target => "Subscription Date"

tag\_on\_failure => ["date\_parse\_failure"]

}

}

csv {

separator => ","

skip\_header => "true"

columns => [

"Index", "User Id", "First Name", "Last Name", "Sex", "Email", "Phone",

"Date of birth", "Job Title"

]

add\_field => { "source" => "people" }

}

if [source] == "people" {

mutate {

convert => { "Index" => "integer" }

}

# Ensure the 'Date of birth' field is parsed into a date format

date {

match => ["Date of birth", "dd/MM/yyyy"]

target => "Date of birth"

tag\_on\_failure => ["date\_parse\_failure"]

}

}

# Enrichment: Join customers and people data

aggregate {

task\_id => "%{[User Id]}"

code => "

map['User Id'] = event.get('User Id')

map['First Name'] ||= event.get('First Name')

map['Last Name'] ||= event.get('Last Name')

map['Company'] ||= event.get('Company')

map['City'] ||= event.get('City')

map['Country'] ||= event.get('Country')

map['Subscription Date'] ||= event.get('Subscription Date')

map['Website'] ||= event.get('Website')

map['Sex'] ||= event.get('Sex')

map['Email'] ||= event.get('Email')

map['Phone'] ||= event.get('Phone')

map['Date of birth'] ||= event.get('Date of birth')

map['Job Title'] ||= event.get('Job Title')

event.cancel()

"

push\_previous\_map\_as\_event => true

timeout => 3

}

# Enrichment: Calculate Age

ruby {

code => "

require 'date'

if event.get('Date of birth')

dob = Date.parse(event.get('Date of birth'))

today = Date.today

age = today.year - dob.year - ((today.month > dob.month || (today.month == dob.month && today.day >= dob.day)) ? 0 : 1)

event.set('Age', age)

end

"

}

}

output {

elasticsearch {

hosts => ["https://es01:9200"]

index => "enriched\_users"

user => "${ELASTIC\_USERNAME}"

password => "${ELASTIC\_PASSWORD}"

ssl\_enabled => true

ssl\_certificate\_verification => true

cacert => "/usr/share/logstash/config/certs/ca/ca.crt"

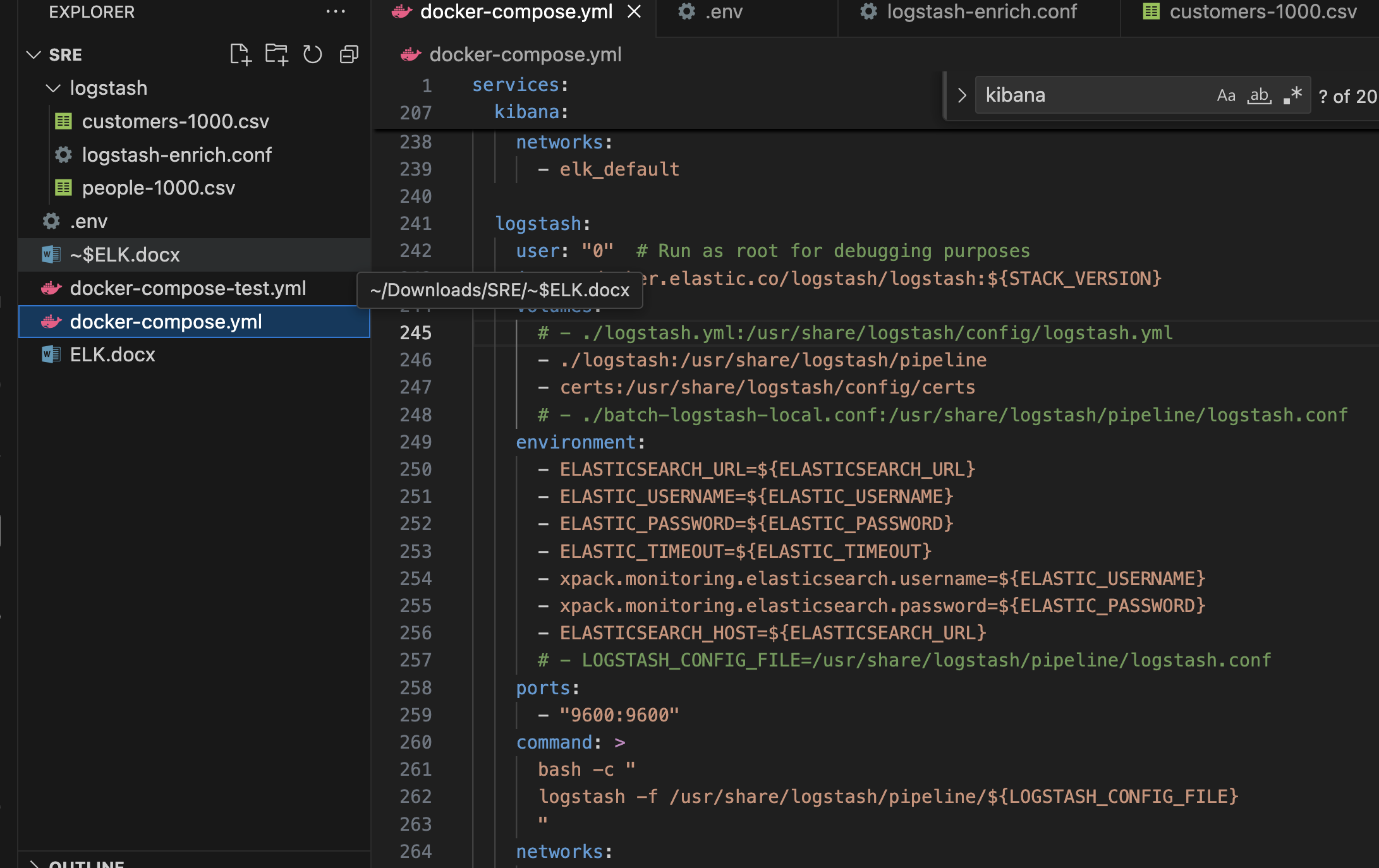
timeout => "${ELASTIC\_TIMEOUT}"

}

stdout { codec => rubydebug }

}

File hierarchy would look like below:



## Query ElasticSearch

1. To query the elasticsearch

curl -u elastic:new123 https://localhost:9200/ -k

## Password Rest for Kibana user

1. To reset kibana\_system user’s password in elastic search, run below command’s from one of the container or the host where es’s API is accessible.

curl -X POST "https://<your\_elasticsearch\_host>:9200/\_security/user/kibana\_system/\_password" -H "Content-Type: application/json" -u elastic:<elastic\_user\_password> -d '{

"password": "new\_password\_for\_kibana\_system"

}'