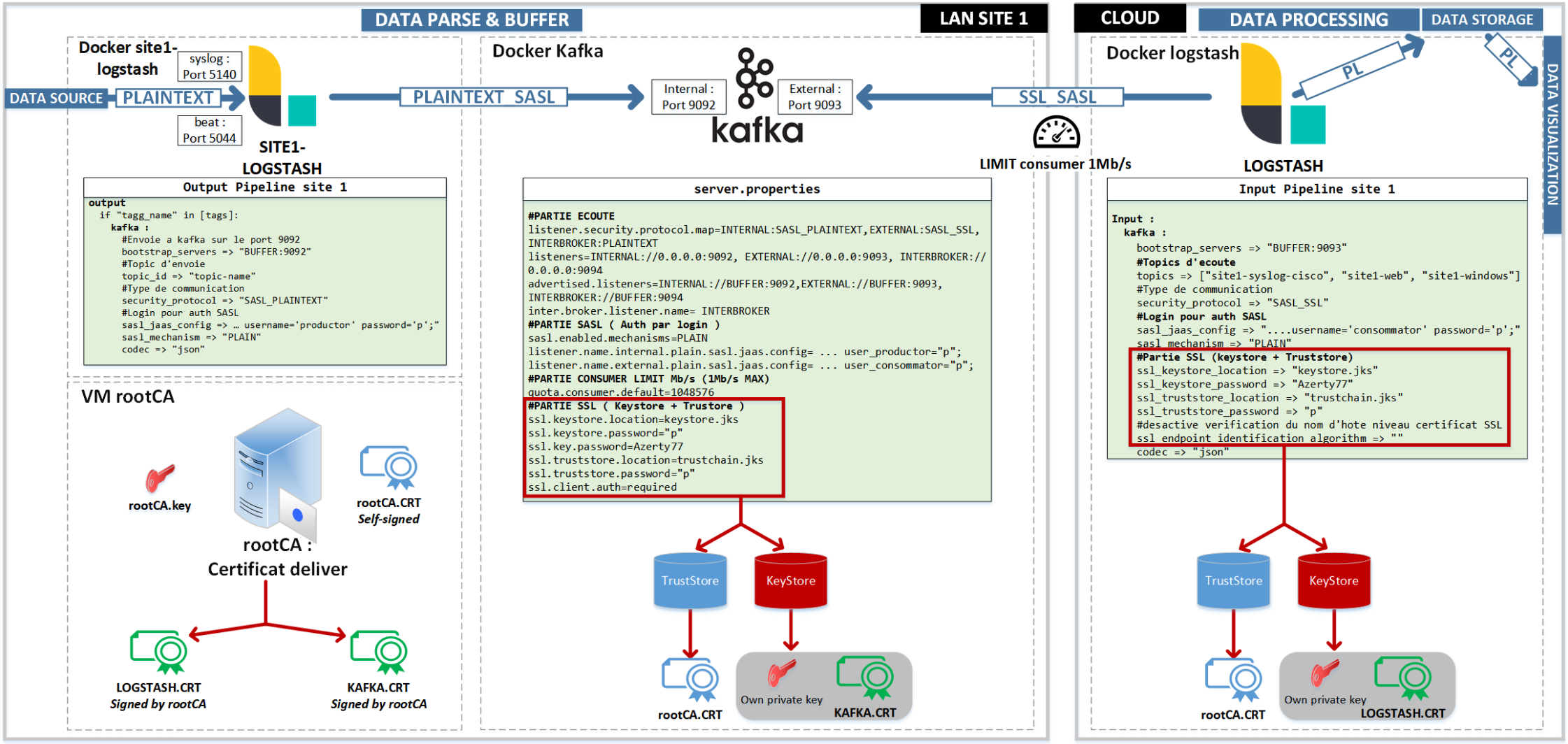
CHIFFREMENT SSL ET AUTHENTIFICATION SASL DANS L'INFRASTRUCTURE

PLAINTEXT_SASL : *no crypted & Auth unidirectionnal per login with kafka*

SSL_SASL : *crypted with Auth bidirectionnal by certificat & Auth unidirectionnal per login with kafka*



SOMMAIRE

A SAVOIR : Pour des raisons pratique j'ai généré l'ensemble de mes certificats sans spécifier le nom d'hôte + nom de domaine.

- Pour la configuration SSL logstash il faut spécifier :
  « ssl_endpoint_identification_algorithm => ""» pour que filebeat ne verifie pas le nom d'hôte du serv kafka du certificat. (il va uniquement vérifier si le kakfa.crt a bien été signé par rootCA)
- Pour le kafka : je n'ai eu aucune modification a faire, il ne check pas le nom d'hote sur certificat (a faire attention peut-etre si nouvelle version ?)

I. Création d'un ROOT CA

a. PREQUIS : VM dédié ROOT CA (plus sécurisé) + Ajout répertoire

```
mkdir PROJET-ELK && mkdir primaire
cd PROJET-ELK
```

b. Création root clé privé + self-signature du root certificat

```
openssl req -x509 -newkey rsa:4096 -keyout primaire/rootCA.key -
out primaire/rootCA.crt -days 1024
```

La passphrase de la clé privée est demandé
Ainsi que des informations a ajouté au certificat (j'ai tout laissé vide)
X509 : Dans ce cas permet de faire une self-signature

II. Création d'un certificat (pour les clients)

```
mkdir clients
cd /root/PROJET-ELK
```

a. Création de la clé privée + demande signature certificat

(cette partie là doit normalement se faire du client (dans mon cas je génère la demande sur le rootCA)

```
openssl req -new -keyout clients/logstash.key -
out clients/logstash.csr
```

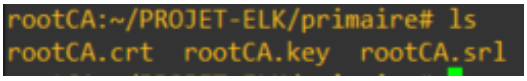
```
openssl req -new -keyout clients/kafka.key -out clients/kafka.csr
```

b. Création du certificat client à partir de la demande (logstash.csr) par rootCA

```
openssl x509 -req -in clients/logstash.csr  -
CA primaire/rootCA.crt -CAkey primaire/rootCA.key -CAcreateserial -
out clients/logstash.crt -days 1024 -sha256
openssl x509 -req -in clients/kafka.csr  -CA primaire/rootCA.crt -
CAkey primaire/rootCA.key -CAcreateserial primaire/rootCA.srl -
out clients/kafka.crt -days 1024 -sha256
```

REFAIRE LA PARTIE 3 POUR LE KAFKA :

c. RESULTAT (explications) :



La partie primaire correspond au rootCA :

RootCA.crt : certificat auto-signé de rootCA

RootCA.key : clé privée de rootCA

(rootCA.srl : fichier serial number créer lors des signatures de certificats (permet d'eviter les doublons de signature) PAS IMPORTANT DANS MON CAS)

La partie clients correspond au clients logstash et kafka :

logtash.crt et kafka.crt : certificats des deux clients signé par rootCA

logstash.key et kafka.key : clé privée des deux clients

logstash.crs et kafka.crs : La demande de signature auprès de rootCA (A SUPPRIMER car plus utile maintenant)

IMPORTANT :

Logstash.crt + logstash.key + rootCA.crt : A envoyer sur docker logstash

kafka.crt + kafka.key + rootCA.crt : A envoyer sur docker kafka

Puis supression de la partie clients sur rootCA (rien a faire ici)

Après export on a : (l'ideal serait de definir les droits sur SSL que pour les personnes concernés)

Pour docker kafka

/opt/kafka/SSL

```
/opt/kafka_2.13-2.7.0/SSL # ls
kafka.crt   kafka.key   rootCA.crt
```

Pour docker logtash :

/opt/logstash/SSL

```
sh-4.2$ ls
logstash.crt  logstash.key  rootCA.crt
```

III. PARTIE KAFKA(SASL_SSL ;SASL_PLAINTEXT)

(apk add openssl) car pas openssl dessus

a. Import du certificat rootCA dans trustchain (=truststore):

```
keytool -importcert -file /opt/kafka/SSL/rootCA.crt -alias ca -
noprompt -keystore /opt/kafka/SSL/trustchain.jks
```

Demande de definir le mot de passe pour le truststore

(install de openssl sur ma docker kafka « apk add openssl »)

b. Conversion du certificat kafka et de la clé privée en « pkcs12 »

```
openssl pkcs12 -export -out /opt/kafka/SSL/kafka.pkcs12 -
in /opt/kafka/SSL/kafka.crt -inkey /opt/kafka/SSL/kafka.key -
passout "pass:Azerty77"
```

Demande de saisir le mot de passe de la clé privée

c. Import de la clé privé et certificat du kafka dans keystore.jks

```
echo "Azerty77" | keytool -importkeystore -
srckeystore /opt/kafka/SSL/kafka.pkcs12 -
destkeystore /opt/kafka/SSL/keystore.jks -srcstoretype pkcs12 -
storepass "Azerty77"
```

d. Modification fichier de conf KAFKA

vi /opt/kafka/config/server.properties :

```
#PARTIE ECOUTE (SASL_PLAINTEXT : Filebeat > Kafka ; SASL_SSL : Kafka < Logstash ; PLAINTEXT : broker > broker (en plaintext car pas de cluster kafka))
listener.security.protocol.map=INTERNAL:SASL_PLAINTEXT,EXTERNAL:SASL_SSL, INTERBROKER:PLAINTEXT
listeners=INTERNAL://0.0.0.0:9092, EXTERNAL://0.0.0.0:9093, INTERBROKER://0.0.0.0:9094
advertised.listeners=INTERNAL://BUFFER:9092,EXTERNAL://BUFFER:9093, INTERBROKER://BUFFER:9094
inter.broker.listener.name= INTERBROKER
#PARTIE SASL ( Auth par login )
sasl.enabled.mechanisms=PLAIN
listener.name.internal.plain.sasl.jaas.config=org.apache.kafka.common.security.plain.PlainLoginModule required username="admin" password="Azerty77" user_productor="Azerty77";
listener.name.external.plain.sasl.jaas.config=org.apache.kafka.common.security.plain.PlainLoginModule required username="admin" password="Azerty77" user_consommator="Azerty77";
#PARTIE SSL ( Keystore + Trustore )
ssl.keystore.location=/opt/kafka/SSL/keystore.jks
ssl.keystore.password=Azerty77
ssl.key.password=Azerty77
ssl.truststore.location=/opt/kafka/SSL/trustchain.jks
ssl.truststore.password=Azerty77
ssl.client.auth=required
#PARTIE CONSOMMATOR LIMIT Mb/s (1Mb MAX)
quota.consumer.default=1048576
```

IV. PARTIE SASL_SSL LOGSTASH

(login as root pour install docker exec -u 0 -it mycontainer bash puis « yum install openss et install de java : «yum install java-1.8.0-openjdk »)

a. Import du certificat rootCA dans trustchain (=truststore):

```
keytool -importcert -file /opt/logstash/SSL/rootCA.crt -alias ca -noprompt -keystore /opt/logstash/SSL/trustchain.jks
```

Demande de definir le mot de passe pour le truststore

(install de openssl sur ma docker kafka « apk add openssl »)

b. Conversion du certificat kafka et de la clé privée en « pkcs12 »

```
openssl pkcs12 -export -out /opt/logstash/SSL/logstash.pkcs12 -
in /opt/logstash/SSL/logstash.crt -
inkey /opt/logstash/SSL/logstash.key -passout "pass:Azerty77"
```

Demande de saisir le mot de passe de la clé privée

c. Import de la clé privé et certificat du kafka dans keystore.jks

```
echo "Azerty77" | keytool -importkeystore -
srckeystore /opt/logstash/SSL/logstash.pkcs12 -
destkeystore /opt/logstash/SSL/keystore.jks -srcstoretype pkcs12 -
storepass "Azerty77"
```

d. Modification fichier du pipeline sur logstash

Pipeline Site1-logstash.conf (input de logstash cloud)

```
input {
    kafka {
      #Ecoute le kafka avec communication en SASL_SSL
      bootstrap_servers => "BUFFER:9093"
      #Topic d'ecoute
      topics => ["site1-syslog-cisco", "site1-web", "site1-windows"]
      #Type de communication
      security_protocol => "SASL_SSL"
      #Login pour auth SASL
      sasl_jaas_config => "org.apache.kafka.common.security.plain.PlainLoginMo
dule required username='consommator' password='Azerty77';"
      sasl_mechanism => "PLAIN"
      #Partie SSL (keystore + Truststore)
      ssl_keystore_location => "/usr/share/logstash/SSL/keystore.jks"
      ssl_keystore_password => "Azerty77"
      ssl_truststore_location => "/usr/share/logstash/SSL/trustchain.jks"
      ssl_truststore_password => "Azerty77"
      #desactive verification du nom d'hote niveau certificat SSL
      ssl_endpoint_identification_algorithm => ""
      codec => "json"
    }
}
#LA SUITE COMPREND LE GROK ET L'OUPUT VERS ELASTIC
```

V. PARTIE SASL_PLAINTEXT SITE1-LOGSTASH

Pipeline Site1-logstash.conf (output de site1-logstash) :

```
    kafka {
      #Envoie a kafka sur le port 9092
      bootstrap_servers => "BUFFER:9092"
      #Topic d'envoie
      topic_id => "site1-windows"
      #Type de communication
      security_protocol => "SASL_PLAINTEXT"
      #Login pour auth SASL
      sasl_jaas_config => "org.apache.kafka.common.security.plain.PlainLoginMo
dule required username='productor' password='Azerty77';"
      sasl_mechanism => "PLAIN"
      codec => "json"
    }
```

COMMANDES UTILES :

Verification handshake SSL : (si se coupe pas c'est bon)

```
openssl s_client -connect BUFFER:9093 -tls1_2 -
CAfile SSL/rootCA.crt -cert SSL/cient.crt -key SSL/client.key
```

Lister les topics kafka :

```
kafka-topics.sh --bootstrap-server BUFFER:9094 --list kafkacat
```

Voir les messages d'un topic kafka :

```
kafka-console-consumer.sh --bootstrap-server localhost:9094 --
topic site1-web --from-beginning
```