# ĐỊnh nghĩa

Để thiết lập xác thực bảo mật cho Kafka bằng cách sử dụng SASL\_SSL với cơ chế PLAIN, bạn cần hiểu rõ về các thành phần chính sau:

1. **Certificate Authority (CA)**
2. **Keystore**
3. **Truststore**

Mỗi thành phần này có vai trò và chức năng riêng, hỗ trợ hệ thống bảo mật thông tin truyền tải giữa các dịch vụ. Dưới đây, tôi sẽ giải thích chi tiết từng thành phần, chức năng, vai trò của chúng, cách chúng hỗ trợ hệ thống, và hướng dẫn cách tạo chúng.

## **1. Certificate Authority (CA)**

### ****1.1. Định Nghĩa****

**Certificate Authority (CA)** là một tổ chức hoặc thực thể đáng tin cậy chịu trách nhiệm phát hành và quản lý các chứng chỉ số. Chứng chỉ số là các tệp tin điện tử xác nhận danh tính của người dùng, máy chủ hoặc dịch vụ và chứa thông tin khóa công khai.

### ****1.2. Chức Năng và Vai Trò****

* **Xác Thực Danh Tính:** CA xác thực danh tính của các thực thể yêu cầu chứng chỉ (ví dụ: máy chủ Kafka) trước khi cấp chứng chỉ.
* **Phát Hành Chứng Chỉ:** Sau khi xác thực, CA phát hành chứng chỉ số chứa khóa công khai và thông tin danh tính.
* **Ký Chữ Ký Số:** CA ký chứng chỉ số bằng khóa riêng của mình, đảm bảo tính toàn vẹn và xác thực của chứng chỉ.
* **Quản Lý Chứng Chỉ:** CA theo dõi, thu hồi và quản lý chứng chỉ đã phát hành để đảm bảo an toàn và tin cậy.

### ****1.3. Cách CA Hỗ Trợ Hệ Thống****

* **Bảo Mật Truyền Thông:** Bằng cách sử dụng chứng chỉ số được CA ký, các giao tiếp giữa các dịch vụ trở nên an toàn và được xác thực.
* **Xác Minh Tính Tin Cậy:** Các thành phần trong hệ thống có thể xác minh tính tin cậy của nhau thông qua chứng chỉ được CA phát hành.

## **2. Keystore**

### ****2.1. Định Nghĩa****

**Keystore** là một kho lưu trữ bảo mật chứa các khóa riêng (private keys) và chứng chỉ số (certificates). Trong ngữ cảnh này, keystore chứa khóa riêng và chứng chỉ số của Kafka Broker.

### ****2.2. Chức Năng và Vai Trò****

* **Lưu Trữ Khóa Riêng:** Keystore lưu trữ khóa riêng mà Kafka Broker sử dụng để ký các giao tiếp SSL/TLS.
* **Lưu Trữ Chứng Chỉ Số:** Keystore cũng chứa chứng chỉ số của Kafka Broker, bao gồm khóa công khai và thông tin danh tính.
* **Bảo Vệ Thông Tin Bảo Mật:** Keystore được bảo vệ bằng mật khẩu để đảm bảo rằng chỉ có các thực thể được ủy quyền mới có thể truy cập nội dung bên trong.

### ****2.3. Cách Keystore Hỗ Trợ Hệ Thống****

* **Thiết Lập SSL/TLS:** Keystore chứa khóa riêng và chứng chỉ số cần thiết để thiết lập giao tiếp SSL/TLS giữa Kafka Broker và các client.
* **Ký Giao Tiếp An Toàn:** Khóa riêng trong keystore được sử dụng để ký các giao tiếp, đảm bảo tính toàn vẹn và bảo mật của dữ liệu truyền tải.

## **3. Truststore**

### ****3.1. Định Nghĩa****

**Truststore** là một kho lưu trữ bảo mật chứa các chứng chỉ CA (Certificate Authorities) mà hệ thống tin cậy. Trong ngữ cảnh này, truststore chứa chứng chỉ CA đã ký chứng chỉ của Kafka Broker.

### ****3.2. Chức Năng và Vai Trò****

* **Lưu Trữ Chứng Chỉ CA Tin Cậy:** Truststore chứa các chứng chỉ CA mà hệ thống sử dụng để xác minh chứng chỉ số của các thực thể khác (ví dụ: Kafka Broker).
* **Xác Minh Tính Tin Cậy:** Khi một client kết nối đến Kafka Broker, truststore giúp xác minh rằng chứng chỉ của Kafka Broker được CA đáng tin cậy ký.

### ****3.3. Cách Truststore Hỗ Trợ Hệ Thống****

* **Xác Thực Client:** Truststore giúp Kafka Broker xác thực các client bằng cách kiểm tra chứng chỉ của client so với các chứng chỉ CA trong truststore.
* **Bảo Vệ Truyền Thông:** Truststore đảm bảo rằng chỉ các client có chứng chỉ được CA tin cậy mới có thể kết nối và giao tiếp với Kafka Broker.

# **Tạo Key**

## Bước 1: Tạo Keystore và Chứng Chỉ CA

**Tạo Keystore cho CA:**

keytool -genkeypair -alias CARoot -keyalg RSA -keysize 2048 -dname "CN=Kafka CA, OU=IT, O=MyCompany, L=MyCity, S=MyState, C=MyCountry" -keypass abCD@1234 -keystore kafka.ca.keystore.jks -storepass abCD@1234

**Xuất Chứng Chỉ CA:**

keytool -exportcert -alias CARoot -keystore kafka.ca.keystore.jks -file ca-cert.pem -rfc -storepass abCD@1234

## Bước 2 Tạo Keystore và Khóa Riêng cho Kafka Broker

keytool -genkeypair -alias kafka-broker -keyalg RSA -keysize 2048 -dname "CN=my-kafka-broker, OU=IT, O=MyCompany, L=MyCity, S=MyState, C=MyCountry" -keypass abCD@1234 -keystore kafka.client.keystore.jks -storepass abCD@1234

## Bước 3: Tạo Yêu Cầu Chữ Ký Chứng Thực (CSR) từ Keystore của Kafka Broker

keytool -certreq -alias kafka-broker -keystore kafka.client.keystore.jks -file kafka-broker.csr -storepass abCD@1234

## Bước 4: Ký CSR bằng CA để Tạo Chứng Chỉ Kafka Broker

keytool -gencert -alias CARoot -keystore kafka.ca.keystore.jks -infile kafka-broker.csr -outfile kafka-broker-cert.pem -rfc -ext SAN=DNS:my-kafka-broker -storepass abCD@1234

## Bước 5: Nhập Chứng Chỉ CA vào Keystore của Kafka Broker

keytool -importcert -alias CARoot -keystore kafka.client.keystore.jks -file ca-cert.pem -storepass abCD@1234 -noprompt

## Bước 6: Nhập Chứng Chỉ Kafka Broker đã Ký vào Keystore của Kafka Broker

keytool -importcert -alias kafka-broker -keystore kafka.client.keystore.jks -file kafka-broker-cert.pem -storepass abCD@1234

## Bước 7 Tạo Truststore cho Kafka Broker

keytool -keystore kafka.client.truststore.jks -alias CARoot -import -file ca-cert.pem -storepass abCD@1234 -noprompt

### ****Tóm Tắt Các Tệp Đã Tạo****

* **CA Keystore:** kafka.ca.keystore.jks
* **CA Certificate:** ca-cert.pem
* **Kafka Broker Keystore:** kafka.client.keystore.jks
* **Kafka Broker Certificate:** kafka-broker-cert.pem
* **Kafka Broker Truststore:** kafka.client.truststore.jks

## **5. Cấu Hình Kafka Broker để Sử Dụng Keystore và Truststore**

Sau khi đã tạo keystore và truststore, bạn cần cấu hình Kafka Broker để sử dụng chúng bằng cách chỉnh sửa tệp server.properties.

### ****5.1. Chỉnh Sửa**** server.properties

Thêm hoặc chỉnh sửa các thuộc tính sau trong server.properties của Kafka Broker:

properties

Sao chép mã

listeners=SASL\_SSL://your-kafka-broker:9093

security.inter.broker.protocol=SASL\_SSL

# SSL Configuration

ssl.keystore.location=/path/to/kafka.client.keystore.jks

ssl.keystore.password=brokerpassword

ssl.key.password=brokerpassword

ssl.truststore.location=/path/to/kafka.client.truststore.jks

ssl.truststore.password=truststorepassword

# SASL Configuration

sasl.enabled.mechanisms=PLAIN

sasl.mechanism.inter.broker.protocol=PLAIN

# JAAS Configuration

sasl.jaas.config=org.apache.kafka.common.security.plain.PlainLoginModule required \

username="admin" \

password="admin-password" \

user\_your\_username="your\_password";

* listeners: Định nghĩa giao thức và cổng mà Kafka Broker sẽ lắng nghe.
* security.inter.broker.protocol: Giao thức bảo mật cho giao tiếp giữa các broker.
* **SSL Configuration:**
  + ssl.keystore.location: Đường dẫn đến keystore của Kafka Broker.
  + ssl.keystore.password: Mật khẩu cho keystore.
  + ssl.key.password: Mật khẩu cho khóa riêng trong keystore.
  + ssl.truststore.location: Đường dẫn đến truststore của Kafka Broker.
  + ssl.truststore.password: Mật khẩu cho truststore.
* **SASL Configuration:**
  + sasl.enabled.mechanisms: Cơ chế SASL được phép (ở đây là PLAIN).
  + sasl.mechanism.inter.broker.protocol: Cơ chế SASL cho giao tiếp giữa các broker.
* **JAAS Configuration:**
  + sasl.jaas.config: Cấu hình JAAS cho Kafka Broker để xác thực các thực thể kết nối.

### ****5.2. Lưu Ý****

* **Đường Dẫn Chính Xác:** Đảm bảo rằng các đường dẫn đến keystore và truststore là chính xác và các tệp có quyền truy cập phù hợp.
* **Chữ Ký và Danh Tính:** Chứng chỉ Kafka Broker phải được ký bởi CA mà truststore của Kafka Broker đã tin cậy.
* **Bảo Mật:** Không chia sẻ keystore và truststore hoặc mật khẩu của chúng với bất kỳ ai không được ủy quyền.

### ****5.3. Khởi Động Lại Kafka Broker****

Sau khi đã cấu hình xong, khởi động lại Kafka Broker để áp dụng các thay đổi:

bash

Sao chép mã

# Trên Windows

bin\windows\kafka-server-start.bat config\server.properties

# Trên Unix/Linux

bin/kafka-server-start.sh config/server.properties

## **6. Cấu Hình Ứng Dụng Spring WebFlux để Kết Nối với Kafka Broker Sử Dụng SASL\_SSL**

### ****6.1. Cập Nhật**** application.yml

Chỉnh sửa tệp application.yml của ứng dụng Spring WebFlux để sử dụng các thông tin bảo mật từ Kubernetes Secrets.

yaml

Sao chép mã

spring:

application:

name: ControlService

main:

allow-bean-definition-overriding: true

web-application-type: reactive

r2dbc:

url: r2dbc:postgresql://localhost:5432/iot

username: postgres

password: abCD@1234

pool:

initial-size: 10

max-size: 20

sql:

init:

mode: always

schema-locations: classpath:schema.sql

data-locations: classpath:data.sql

kafka:

bootstrap-servers: ${SPRING\_KAFKA\_BOOTSTRAP\_SERVERS}

producer:

key-serializer: org.apache.kafka.common.serialization.StringSerializer

value-serializer: org.apache.kafka.common.serialization.StringSerializer

properties:

security.protocol: ${SPRING\_KAFKA\_PRODUCER\_PROPERTIES\_SECURITY\_PROTOCOL}

sasl.mechanism: ${SPRING\_KAFKA\_PRODUCER\_PROPERTIES\_SASL\_MECHANISM}

sasl.jaas.config: ${SPRING\_KAFKA\_PRODUCER\_PROPERTIES\_SASL\_JAAS\_CONFIG}

consumer:

group-id: my-group

key-deserializer: org.apache.kafka.common.serialization.StringDeserializer

value-deserializer: org.apache.kafka.common.serialization.StringDeserializer

properties:

security.protocol: SASL\_SSL

sasl.mechanism: PLAIN

sasl.jaas.config: |

org.apache.kafka.common.security.plain.PlainLoginModule required

username="${KAFKA\_USERNAME}"

password="${KAFKA\_PASSWORD}";

logging:

level:

io.r2dbc: DEBUG

io.r2dbc.spi: DEBUG

server:

port: 8060

### ****6.2. Cấu Hình Kubernetes Secrets và ConfigMap****

#### ****Bước 1: Tạo Secret trong Kubernetes****

Giả sử bạn đã có username và password cho Kafka, mã hóa chúng bằng Base64:

bash

Sao chép mã

echo -n 'your\_username' | base64

# Output: eW91cl91c2VybmFtZQ==

echo -n 'your\_password' | base64

# Output: eW91cl9wYXNzd29yZA==

**Tạo tệp kafka-secret.yaml:**

yaml

Sao chép mã

apiVersion: v1

kind: Secret

metadata:

name: kafka-secret

namespace: kafka

type: Opaque

data:

username: eW91cl91c2VybmFtZQ==

password: eW91cl9wYXNzd29yZA==

**Áp Dụng Secret:**

bash

Sao chép mã

kubectl apply -f kafka-secret.yaml -n kafka

#### ****Bước 2: Tạo ConfigMap cho Cấu Hình Kafka****

Nếu bạn muốn chia sẻ cấu hình Kafka giữa nhiều dịch vụ, tạo một ConfigMap.

**Tạo tệp kafka-configmap.yaml:**

yaml

Sao chép mã

apiVersion: v1

kind: ConfigMap

metadata:

name: kafka-config

namespace: kafka

data:

spring.kafka.bootstrap-servers: "your-kafka-broker:9093"

spring.kafka.producer.properties.security.protocol: "SASL\_SSL"

spring.kafka.producer.properties.sasl.mechanism: "PLAIN"

**Áp Dụng ConfigMap:**

bash

Sao chép mã

kubectl apply -f kafka-configmap.yaml -n kafka

### ****6.3. Cập Nhật Deployment để Sử Dụng ConfigMap và Secret****

**Tạo hoặc chỉnh sửa tệp controlservice-deployment.yaml:**

yaml

Sao chép mã

apiVersion: apps/v1

kind: Deployment

metadata:

name: controlservice-deployment

namespace: kafka

spec:

replicas: 2

selector:

matchLabels:

app: controlservice

template:

metadata:

labels:

app: controlservice

spec:

containers:

- name: controlservice

image: your\_dockerhub\_username/controlservice:latest

ports:

- containerPort: 8060

env:

- name: SPRING\_KAFKA\_BOOTSTRAP\_SERVERS

valueFrom:

configMapKeyRef:

name: kafka-config

key: spring.kafka.bootstrap-servers

- name: SPRING\_KAFKA\_PRODUCER\_PROPERTIES\_SECURITY\_PROTOCOL

valueFrom:

configMapKeyRef:

name: kafka-config

key: spring.kafka.producer.properties.security.protocol

- name: SPRING\_KAFKA\_PRODUCER\_PROPERTIES\_SASL\_MECHANISM

valueFrom:

configMapKeyRef:

name: kafka-config

key: spring.kafka.producer.properties.sasl.mechanism

- name: SPRING\_KAFKA\_PRODUCER\_PROPERTIES\_SASL\_JAAS\_CONFIG

value: |

org.apache.kafka.common.security.plain.PlainLoginModule required

username="${KAFKA\_USERNAME}"

password="${KAFKA\_PASSWORD}";

- name: KAFKA\_USERNAME

valueFrom:

secretKeyRef:

name: kafka-secret

key: username

- name: KAFKA\_PASSWORD

valueFrom:

secretKeyRef:

name: kafka-secret

key: password

# Các biến môi trường khác nếu cần