# 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.

## Bước 1: Tạo Keystore và Chứng Chỉ CA

**Tạo Keystore cho CA:**

keytool -genkeypair -alias caroot -keyalg RSA -keysize 2048 -sigalg SHA512withRSA -validity 365 -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

-ext "BasicConstraints=ca:true"

keytool(dùng công cụ khóa) -genkeypair(tạo cặp khóa) -alias ( bí danh tên) caroot ( tên Certificate Authority-chứng chỉ ủy quyền) -Dname(Distinguished Name -DN :: tên phân biệt)

**Giải thích**

 **CN (Common Name)**: Thường là tên của chứng chỉ, ở đây là Kafka CA.

 **OU (Organizational Unit)**: Đơn vị tổ chức, ở đây là IT.

 **O (Organization)**: Tên tổ chức, ở đây là MyCompany.

 **L (Locality)**: Tên thành phố, ở đây là MyCity.

 **S (State)**: Tên bang hoặc khu vực, ở đây là MyState.

 **C (Country)**: Mã quốc gia (2 ký tự), ở đây là MyCountry.

 **-keypass abCD@1234**: Mật khẩu bảo vệ cho khóa riêng tư tạo từ -genkeypair

 **-keystore kafka.ca.keystore.jks**: Tên của keystore sẽ lưu trữ cặp khóa này, ở đây là kafka.ca.keystore.jks.

 **-storepass abCD@1234**: Mật khẩu bảo vệ toàn bộ keystore.

## Xuất Chứng Chỉ CA:

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

keytool -exportcert -keystore caroot.jks -alias caroot -file caroot.crt -storepass abCD@1234

Lệnh keytool -exportcert này dùng để **xuất chứng chỉ của CA (Certificate Authority)** từ keystore và lưu vào một tệp chứng chỉ có định dạng PEM. Các thành phần của lệnh này có ý nghĩa như sau:

* -exportcert: Chỉ định thao tác xuất chứng chỉ.
* -alias CARoot: Xác định tên bí danh (alias) của cặp khóa cần xuất, ở đây là "CARoot".
* -keystore kafka.ca.keystore.jks: Đường dẫn đến tệp keystore chứa cặp khóa cần xuất, tên file ở đây là kafka.ca.keystore.jks.
* -file ca-cert.pem: Đường dẫn đến tệp đích mà chứng chỉ sẽ được xuất ra, ở đây là ca-cert.pem.
* -rfc: Định dạng đầu ra là PEM (tức mã hóa bằng Base64), phổ biến để chia sẻ chứng chỉ qua các nền tảng khác nhau.
* -storepass abCD@1234: Mật khẩu truy cập keystore.

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

keytool -genkeypair -alias kafka-broker -keyalg RSA -keysize 2048 -sigalg SHA512withRSA -validity 365 -dname "CN=my-kafka-broker, OU=IT, O=MyCompany, L=MyCity, S=MyState, C=MyCountry" -keypass abCD@1234 -keystore kafka.server.keystore.jks -storepass abCD@1234

**Giải thích**

Lệnh keytool -genkeypair được sử dụng để tạo một cặp khóa mới (key pair) và chứng chỉ tự ký (self-signed certificate) cho Kafka broker. Các tùy chọn trong lệnh này định cấu hình các thông tin cụ thể cho cặp khóa và chứng chỉ đó.

**Phân tích chi tiết các tùy chọn**

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.server.keystore.jks -storepass abCD@1234

* **-genkeypair**: Tạo một cặp khóa mới, gồm một khóa riêng (private key) và một khóa công khai (public key).
* **-alias kafka-broker**: Tên bí danh của cặp khóa trong keystore. Ở đây, bí danh được đặt là "kafka-broker", giúp nhận diện khóa này trong keystore, nhất là khi có nhiều khóa.
* **-keyalg RSA**: Thuật toán mã hóa RSA sẽ được sử dụng để tạo khóa. RSA là một trong những thuật toán phổ biến để mã hóa và tạo khóa công khai.
* **-keysize 2048**: Độ dài khóa RSA là 2048 bit, tạo ra khóa mạnh hơn về mặt bảo mật.
* **-dname "CN=my-kafka-broker, OU=IT, O=MyCompany, L=MyCity, S=MyState, C=MyCountry"**: Chuỗi Distinguished Name (DN) cho chứng chỉ, chứa thông tin định danh về chủ thể của chứng chỉ. Trong trường hợp này:
  + CN=my-kafka-broker: Tên thông dụng của Kafka broker.
  + OU=IT: Bộ phận IT trong tổ chức.
  + O=MyCompany: Tên của công ty.
  + L=MyCity: Thành phố.
  + S=MyState: Bang.
  + C=MyCountry: Quốc gia (mã quốc gia).
* **-keypass abCD@1234**: Mật khẩu để bảo vệ khóa riêng.
* **-keystore kafka.server.keystore.jks**: Tên của file keystore sẽ lưu trữ cặp khóa này. Keystore này sẽ được dùng để lưu chứng chỉ và khóa riêng cho Kafka broker.
* **-storepass abCD@1234**: Mật khẩu bảo vệ keystore, giúp truy cập vào keystore để lấy khóa và chứng chỉ.

**Tóm tắt ý nghĩa**

Lệnh này tạo một cặp khóa và chứng chỉ cho Kafka broker với các thông tin định danh được cung cấp, lưu trữ trong keystore kafka.server.keystore.jks. Cặp khóa và chứng chỉ này sẽ được Kafka broker sử dụng để thiết lập kết nối an toàn với các client hoặc các dịch vụ khác trong hệ thống, đảm bảo bảo mật thông qua mã hóa và xác thực.

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

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

**Giải thích**

Lệnh keytool -certreq dùng để tạo một Certificate Signing Request (CSR) từ một cặp khóa đã có trong keystore. CSR là một yêu cầu chứng thực khóa công khai từ một Certificate Authority (CA) nhằm xác thực tính hợp lệ của chứng chỉ.

Cụ thể, lệnh sau thực hiện các bước như sau:

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

* -certreq: Tạo một CSR từ cặp khóa trong keystore.
* -alias kafka-broker: Chỉ định bí danh của khóa trong keystore từ đó CSR sẽ được tạo. Trong trường hợp này, bí danh là kafka-broker.
* -keystore kafka.server.keystore.jks: Chỉ định keystore chứa cặp khóa (khóa riêng và khóa công khai) dùng để tạo CSR. Keystore này có tên là kafka.server.keystore.jks.
* -file kafka-broker.csr: Đầu ra là tệp CSR có tên kafka-broker.csr, chứa khóa công khai và một số thông tin (như tên miền, thông tin tổ chức) của server yêu cầu.
* -storepass abCD@1234: Mật khẩu để mở khóa keystore.

Ý nghĩa và công dụng của CSR

* CSR chứa thông tin của server, như tên miền và chi tiết tổ chức, cùng với khóa công khai, và sẽ được gửi đến CA để cấp chứng chỉ.
* CA sẽ kiểm tra thông tin, xác thực tính hợp lệ, và ký vào khóa công khai để tạo ra chứng chỉ.
* Sau khi nhận chứng chỉ đã ký từ CA, chứng chỉ này có thể được nhập vào keystore và dùng để chứng thực danh tính của server, ví dụ như với các client Kafka.

## 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

**Giải thích**

Lệnh keytool -gencert được dùng để tạo một chứng chỉ (certificate) từ yêu cầu chứng thực (CSR) của một server và ký chứng chỉ đó bằng khóa của Certificate Authority (CA) có trong keystore. Chứng chỉ được ký này sẽ xác nhận tính hợp lệ của server khi client kết nối đến nó.

Lệnh dưới đây chi tiết cách thực hiện quá trình này:

bash

Sao chép mã

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

**Giải thích từng thành phần**

* -gencert: Lệnh tạo chứng chỉ từ một CSR. Khóa CA được sử dụng để ký CSR này.
* -alias CARoot: Chỉ định bí danh của khóa CA (Certificate Authority) trong keystore. Khóa này sẽ được dùng để ký chứng chỉ. Trong trường hợp này, bí danh là CARoot.
* -keystore kafka.ca.keystore.jks: Keystore chứa khóa riêng của CA, được sử dụng để ký chứng chỉ.
* -infile kafka-broker.csr: Tệp CSR từ máy chủ cần được cấp chứng chỉ. Tệp này chứa khóa công khai và thông tin về máy chủ (kafka-broker).
* -outfile kafka-broker-cert.pem: Đầu ra là tệp chứa chứng chỉ đã ký, ở định dạng PEM. Đây sẽ là chứng chỉ của server (kafka-broker), do CA cấp và ký.
* -rfc: Định dạng chứng chỉ đầu ra là PEM (Base64).
* -ext SAN=DNS:my-kafka-broker: Thêm phần mở rộng "Subject Alternative Name" (SAN) vào chứng chỉ, để xác nhận chứng chỉ này là hợp lệ cho tên DNS my-kafka-broker.
* -storepass abCD@1234: Mật khẩu của keystore chứa khóa CA.

**Ý nghĩa và ứng dụng**

1. **Ký chứng chỉ của server**: Lệnh này sử dụng khóa CA để ký CSR của kafka-broker, tạo ra chứng chỉ hợp lệ cho server này. Sau khi ký, chứng chỉ sẽ được tin cậy bởi client vì nó được xác thực bởi CA.
2. **Subject Alternative Name (SAN)**: Việc thêm SAN=DNS:my-kafka-broker vào chứng chỉ giúp xác định rõ ràng rằng chứng chỉ này hợp lệ cho máy chủ với tên DNS là my-kafka-broker. SAN là bắt buộc trong các hệ thống hiện đại, đặc biệt khi dùng HTTPS và các giao thức bảo mật khác.
3. **Phát hành chứng chỉ cho Kafka Broker**: Chứng chỉ đã ký (kafka-broker-cert.pem) có thể được tải lên keystore của broker. Khi client Kafka kết nối tới broker, chứng chỉ này sẽ giúp xác thực broker, đảm bảo rằng nó đã được CA tin cậy xác nhận.

Kết quả cuối cùng là chứng chỉ ký bởi CA, dùng để bảo mật và xác thực danh tính của kafka-broker trong môi trường Kafka.

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

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

**Giải thích**

Lệnh keytool -importcert được sử dụng để nhập một chứng chỉ CA vào keystore của server. Khi một keystore chứa chứng chỉ CA, các chứng chỉ server khác được ký bởi CA đó sẽ được coi là hợp lệ và đáng tin cậy.

Dưới đây là cách lệnh này hoạt động:

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

**Giải thích từng thành phần:**

* -importcert: Thực hiện nhập một chứng chỉ vào keystore.
* -alias CARoot: Chỉ định bí danh cho chứng chỉ CA khi được lưu vào keystore. Trong trường hợp này, bí danh là CARoot.
* -keystore kafka.server.keystore.jks: Keystore của Kafka server, nơi chứng chỉ CA sẽ được nhập vào.
* -file ca-cert.pem: Tệp chứa chứng chỉ của CA, trong trường hợp này là ca-cert.pem. Đây là chứng chỉ tin cậy được sử dụng để xác thực các chứng chỉ server do CA này ký.
* -storepass abCD@1234: Mật khẩu của keystore Kafka server (kafka.server.keystore.jks).
* -noprompt: Bỏ qua các yêu cầu xác nhận trong quá trình nhập chứng chỉ, giúp quá trình thực hiện tự động hơn.

**Ý nghĩa và ứng dụng:**

1. **Thêm chứng chỉ CA vào keystore của server**: Bằng cách nhập chứng chỉ CA (ca-cert.pem) vào keystore của Kafka server, bạn cho phép server xác nhận bất kỳ chứng chỉ nào được CA này ký. Điều này có nghĩa là khi một client kết nối tới server, server sẽ tin tưởng các chứng chỉ do CA này ký.
2. **Hỗ trợ xác thực lẫn nhau (mutual authentication)**: Khi cả client và server có chứng chỉ do CA tin cậy cấp, họ có thể xác thực lẫn nhau. Việc này là cần thiết trong cấu hình bảo mật của Kafka, đảm bảo rằng client chỉ kết nối tới những server đáng tin cậy.
3. **Chứng chỉ trong keystore**: Sau khi thực hiện lệnh này, keystore kafka.server.keystore.jks của server sẽ chứa chứng chỉ CA (CARoot). Điều này cho phép Kafka server nhận diện các client và server khác được CA đó ký.

Kết quả là Kafka server của bạn sẽ tin tưởng vào CA đã được nhập này và có thể xác thực các chứng chỉ của client được CA đó ký.

4o

Top of Form

Bottom of Form

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

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

**Giải thích**

Lệnh keytool -importcert được sử dụng để nhập chứng chỉ số vào keystore (bộ lưu trữ chứng chỉ) trong Java. Đây là một phần quan trọng trong việc quản lý chứng chỉ và bảo mật khi làm việc với các dịch vụ như Kafka. Dưới đây là giải thích chi tiết về từng phần trong lệnh của bạn:

* keytool: Đây là công cụ dòng lệnh được cung cấp bởi Java Development Kit (JDK) để quản lý các keystore và chứng chỉ.
* -importcert: Tham số này chỉ định rằng bạn muốn nhập một chứng chỉ vào keystore.
* -alias kafka-broker: Tùy chọn này chỉ định tên (alias) cho chứng chỉ trong keystore. Tên này sẽ được sử dụng để tham chiếu đến chứng chỉ này trong tương lai.
* -keystore kafka.server.keystore.jks: Đây là tên của keystore mà bạn muốn nhập chứng chỉ vào. Trong trường hợp này, keystore có tên là kafka.server.keystore.jks.
* -file kafka-broker-cert.pem: Đây là tệp chứng chỉ mà bạn muốn nhập vào keystore. Tệp chứng chỉ có định dạng PEM (Privacy Enhanced Mail) và có tên là kafka-broker-cert.pem.
* -storepass abCD@1234: Đây là mật khẩu của keystore, được sử dụng để bảo vệ keystore. Mật khẩu này cần phải được cung cấp để thực hiện các thao tác nhập, xuất chứng chỉ hoặc thay đổi keystore.

**Tóm lại:**

Lệnh trên sẽ nhập chứng chỉ kafka-broker-cert.pem vào keystore kafka.server.keystore.jks với alias là kafka-broker, sử dụng mật khẩu abCD@1234 để xác thực truy cập keystore. Điều này thường được thực hiện để thiết lập một kết nối an toàn (SSL/TLS) cho Kafka broker.

## Bước 7 Tạo Truststore cho Kafka Broker

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

**Giải thích**

Lệnh keytool -keystore kafka.server.truststore.jks -alias CARoot -import -file ca-cert.pem -storepass abCD@1234 -noprompt cũng được sử dụng để quản lý chứng chỉ trong Java, nhưng lệnh này có một số điểm khác so với lệnh trước đó. Dưới đây là giải thích chi tiết về từng phần trong lệnh của bạn:

* **keytool**: Công cụ dòng lệnh của Java để quản lý keystore và chứng chỉ.
* **-keystore kafka.server.truststore.jks**: Tham số này chỉ định keystore mà bạn muốn nhập chứng chỉ vào. Trong trường hợp này, keystore có tên là kafka.server.truststore.jks. Truststore được sử dụng để lưu trữ các chứng chỉ mà bạn tin tưởng (các chứng chỉ của CA - Certificate Authority).
* **-alias CARoot**: Tùy chọn này chỉ định tên (alias) cho chứng chỉ mà bạn đang nhập vào truststore. Ở đây, chứng chỉ được đặt tên là CARoot, thường là chứng chỉ gốc (CA root certificate).
* **-import**: Tham số này chỉ định rằng bạn muốn nhập một chứng chỉ vào truststore.
* **-file ca-cert.pem**: Đây là tệp chứng chỉ mà bạn muốn nhập vào truststore. Tệp chứng chỉ có định dạng PEM và có tên là ca-cert.pem. Đây thường là chứng chỉ gốc của CA mà bạn tin tưởng.
* **-storepass abCD@1234**: Đây là mật khẩu của truststore, cần phải cung cấp để thực hiện các thao tác nhập chứng chỉ hoặc thay đổi truststore.
* **-noprompt**: Tham số này chỉ định rằng công cụ sẽ không yêu cầu xác nhận trước khi nhập chứng chỉ. Điều này hữu ích cho các kịch bản tự động hóa, vì nó cho phép lệnh chạy mà không cần sự can thiệp của người dùng.

**Tóm lại:**

Lệnh trên sẽ nhập chứng chỉ gốc ca-cert.pem vào truststore kafka.server.truststore.jks với alias là CARoot, sử dụng mật khẩu abCD@1234 để xác thực. Việc này thường được thực hiện để thiết lập một kết nối an toàn (SSL/TLS) cho Kafka broker, cho phép nó tin tưởng các chứng chỉ được ký bởi CA gốc này.

4

### ****Tóm Tắt Các Tệp Đã Tạo****

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

## **Tạo lại cấu hình**

openssl req -new -x509 -keyout caroot.key -out caroot.crt -days 365 -subj "/CN=Kafka CA/OU=IT/O=MyCompany/L=MyCity/ST=MyState/C=MyCountry" -addext "basicConstraints=CA:TRUE"

keytool -genkeypair -v -keystore caroot.jks -keyalg RSA -keysize 2048 -validity 365 -alias caroot -dname "CN=Kafka CA, OU=IT, O=MyCompany, L=MyCity, ST=MyState, C=US" -storepass abCD@1234 -keypass abCD@1234

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

1. **Tạo CA (Caroot)**: Bước này vẫn giữ nguyên, bạn đã tạo chứng chỉ CA cho Kafka với thuật toán SHA256withRSA cho ký.

keytool -genkeypair -v -keystore caroot.jks -keyalg RSA -keysize 2048 -validity 365 -alias caroot -dname "CN=Kafka CA, OU=IT, O=MyCompany, L=MyCity, ST=MyState, C=US" -storepass abCD@1234 -keypass abCD@1234 -ext "BC=CA:TRUE"

**2.Xuất chứng chỉ CA**: Xuất chứng chỉ của CA để sử dụng sau này trong các bước khác.

keytool -exportcert -keystore caroot.jks -alias caroot -file caroot.crt -storepass abCD@1234

**3.Tạo chứng chỉ Kafka broker với SHA256withRSA**: **Cập nhật thuật toán chữ ký từ SHA512withRSA thành SHA256withRSA** trong bước tạo chứng chỉ cho Kafka broker.

keytool -genkeypair -alias kafka-broker -keyalg RSA -keysize 2048 -sigalg SHA256withRSA -validity 365 -dname "CN=my-kafka-broker, OU=IT, O=MyCompany, L=MyCity, S=MyState, C=MyCountry" -keypass abCD@1234 -keystore kafka.server.keystore.jks -storepass abCD@1234

**4.Tạo yêu cầu chứng chỉ (CSR)**: Sau khi tạo chứng chỉ cho Kafka broker, tạo yêu cầu chứng chỉ (CSR) để gửi tới CA (hoặc ký chứng chỉ tự cấp).

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

5. **Ký chứng chỉ cho Kafka broker bằng CA**: CA ký yêu cầu chứng chỉ và tạo ra chứng chỉ đã ký.

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

6. **Nhập chứng chỉ CA vào keystore của Kafka broker**: Nhập chứng chỉ CA vào keystore của Kafka broker để nó có thể xác thực với CA trong quá trình thiết lập SSL.

keytool -importcert -alias caroot -keystore kafka.server.keystore.jks -file caroot.crt -storepass abCD@1234 -noprompt

7. **Nhập chứng chỉ CA vào truststore của Kafka broker**: Đảm bảo rằng Kafka broker có thể xác thực chứng chỉ CA bằng cách nhập chứng chỉ CA vào truststore.

keytool -keystore kafka.server.truststore.jks -alias caroot -import -file caroot.crt -storepass abCD@1234 -noprompt

keytool -delete -alias caroot -keystore "%JAVA\_HOME%/lib/security/cacerts" -storepass changeit

keytool -importcert -file caroot.crt -keystore %JAVA\_HOME%/lib/security/cacerts -alias caroot -storepass changeit

keytool -genkeypair -alias kafka-broker -keyalg RSA -keysize 2048 -sigalg SHA256withRSA -validity 365 -dname "CN=my-kafka-broker, OU=IT, O=MyCompany, L=MyCity, ST=MyState, C=MyCountry" -keypass abCD@1234 -keystore kafka.server.keystore.jks -storepass abCD@1234

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

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

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

keytool -importcert -alias caroot -keystore kafka.server.keystore.jks -file caroot.crt -storepass abCD@1234 -noprompt

keytool -list -v -keystore kafka.server.keystore.jks -storepass abCD@1234

openssl s\_client -connect localhost:9092 -tls1\_2

## **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

## Cli kafka

keytool -list -v -keystore kafka.server.keystore.jks -storepass abCD@1234

keytool -list -v -keystore kafka.server.truststore.jks -storepass abCD@1234

## 2) Cấu hình kafka ssl/sasl(security (cấu hình tin tưởng thông tin trong quá trình giao tiếp)/simple authen security layer(cấu hình tin tưởng thiết lập giao tiếp))

# Tạo keystore cho **broker**(các service có các producer/consumer gọi là client và broker là server nơi xử lý giao tiếp cho các client kafka)

keytool -keystore kafka.server.keystore.jks -alias localhost -validity 365 -genkey

# Tạo chứng chỉ và xuất public certificate

keytool -keystore kafka.server.keystore.jks -alias localhost -certreq -file cert-file

# Ký chứng chỉ bằng CA hoặc tự ký chứng chỉ và nhập lại vào keystore

openssl x509 -req -CA ca-cert -CAkey ca-key -in cert-file -out cert-signed -days 365

# Nhập CA vào truststore của Kafka

keytool -keystore kafka.server.truststore.jks -alias CARoot -import -file ca-cert

# Nhập chứng chỉ đã ký vào keystore của Kafka

keytool -keystore kafka.server.keystore.jks -alias localhost -import -file cert-signed

### server.properties

# Cấu hình SSL cho Kafka Broker

listeners=SSL://localhost:9093

advertised.listeners=SSL://localhost:9093

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

ssl.keystore.password=keystore\_password

ssl.key.password=key\_password

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

ssl.truststore.password=truststore\_password

ssl.endpoint.identification.algorithm=

security.inter.broker.protocol=SSL

ssl.client.auth=required

### Cấu hình SASL cho Kafka Broker

# Cấu hình SASL/SSL cho Kafka Broker

listeners=SASL\_SSL://localhost:9094

advertised.listeners=SASL\_SSL://localhost:9094

security.inter.broker.protocol=SASL\_SSL

sasl.mechanism.inter.broker.protocol=SCRAM-SHA-512

sasl.enabled.mechanisms=PLAIN,SCRAM-SHA-512

sasl.jaas.config=org.apache.kafka.common.security.scram.ScramLoginModule required \

username="broker\_username" \

password="broker\_password";

### Cấu hình SSL cho Kafka Client

security.protocol=SSL

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

ssl.truststore.password=truststore\_password

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

ssl.keystore.password=keystore\_password

ssl.key.password=key\_password

ssl.endpoint.identification.algorithm=

### Cấu hình SASL cho Kafka Client

security.protocol=SASL\_SSL

sasl.mechanism=SCRAM-SHA-512

sasl.jaas.config=org.apache.kafka.common.security.scram.ScramLoginModule required \

username="client\_username" \

password="client\_password";

### Cấu hình Kafka Consumer/Producer

Properties props = new Properties();

props.put("bootstrap.servers", "localhost:9093");

props.put("security.protocol", "SSL");

props.put("ssl.truststore.location", "/path/to/client.truststore.jks");

props.put("ssl.truststore.password", "truststore\_password");

props.put("ssl.keystore.location", "/path/to/client.keystore.jks");

props.put("ssl.keystore.password", "keystore\_password");

props.put("ssl.key.password", "key\_password");

// SASL

props.put("security.protocol", "SASL\_SSL");

props.put("sasl.mechanism", "SCRAM-SHA-512");

props.put("sasl.jaas.config", "org.apache.kafka.common.security.scram.ScramLoginModule required username=\"client\_username\" password=\"client\_password\";");

KafkaProducer<String, String> producer = new KafkaProducer<>(props);

### Lý giải

Để hiểu rõ vai trò của keystore, truststore, username, password trong cơ chế SSL/SASL của Kafka, chúng ta cần xem xét cả hai phần chính: **Kafka Broker** (máy chủ Kafka) và **Kafka Client** (các ứng dụng hoặc dịch vụ kết nối tới Kafka).

### 1. ****Keystore và Truststore của Kafka****

#### ****Keystore:****

* **Keystore** chứa khóa riêng (private key) và chứng chỉ (certificate) công khai của Kafka. Nó giúp chứng minh danh tính của Kafka Broker (hoặc Kafka Client) khi thực hiện giao tiếp qua mạng. Keystore cần được bảo mật vì nó chứa thông tin nhạy cảm (khóa riêng).
* Đối với **Kafka Broker**, keystore chứa chứng chỉ mà Broker sử dụng để mã hóa dữ liệu truyền qua SSL. Khi một Kafka Client kết nối, Broker sẽ cung cấp chứng chỉ công khai (certificate) từ keystore để client có thể xác minh danh tính của broker.
* Đối với **Kafka Client**, nếu client cũng yêu cầu xác thực (mutual TLS), keystore của client sẽ chứa khóa riêng và chứng chỉ công khai mà nó sẽ cung cấp cho Kafka Broker để xác minh danh tính của client.

#### ****Truststore:****

* **Truststore** chứa các chứng chỉ của các tổ chức tin cậy (CA - Certificate Authority). Truststore giúp xác thực tính hợp lệ của chứng chỉ được cung cấp bởi bên đối diện (Kafka Broker hoặc Kafka Client).
* **Kafka Broker** sử dụng truststore để xác minh rằng chứng chỉ của Kafka Client là hợp lệ (nếu có mutual TLS, tức là client cũng cung cấp chứng chỉ).
* **Kafka Client** sử dụng truststore để xác minh rằng chứng chỉ của Kafka Broker là hợp lệ và đáng tin cậy.

### 2. ****Username và Password trong JAAS (Java Authentication and Authorization Service)****

JAAS là một framework cho phép xác thực dựa trên cơ chế bảo mật như **SASL**. Kafka sử dụng JAAS để cấu hình các cơ chế xác thực như SCRAM (Simple Authentication and Security Layer).

#### ****Kafka Broker (JAAS Configuration)****

* **Username và password của broker** trong cấu hình JAAS là thông tin để xác thực giữa các Kafka Broker với nhau, và khi Kafka Broker cần tương tác với các thành phần khác (ví dụ Zookeeper hoặc các brokers khác trong cụm).
* Ví dụ, khi Kafka Broker sử dụng cơ chế **SCRAM-SHA-512** để xác thực nội bộ giữa các broker, các thông tin sau trong cấu hình JAAS sẽ được sử dụng:

properties

Sao chép mã

sasl.jaas.config=org.apache.kafka.common.security.scram.ScramLoginModule required \

username="broker\_username" \

password="broker\_password";

* Username và password này được dùng để xác minh tính hợp lệ giữa các broker khi giao tiếp nội bộ, hoặc giữa broker và Zookeeper.

#### ****Kafka Client (JAAS Configuration)****

* **Username và password của client** trong JAAS là thông tin xác thực để Kafka Client (producer/consumer) kết nối và tương tác với Kafka Broker. Khi Kafka Broker được cấu hình để sử dụng cơ chế SASL (ví dụ: SASL\_SSL hoặc SASL\_PLAINTEXT), Kafka Client sẽ cần cung cấp thông tin xác thực của mình (username/password) để chứng minh danh tính với Kafka Broker.
* Ví dụ, cấu hình JAAS của Kafka Client khi sử dụng SCRAM-SHA-512:

properties

Sao chép mã

sasl.jaas.config=org.apache.kafka.common.security.scram.ScramLoginModule required \

username="client\_username" \

password="client\_password";

* Khi Kafka Client gửi yêu cầu kết nối tới Kafka Broker, nó sẽ gửi kèm username và password này để xác thực. Kafka Broker sẽ kiểm tra username/password này so với dữ liệu người dùng đã được cấu hình trong Kafka Broker (thông qua Zookeeper hoặc cơ sở dữ liệu bảo mật của Kafka).

### 3. ****Cơ chế Hoạt động giữa Keystore, Truststore, Username và Password****

#### ****SSL (TLS)****

* **Keystore và Truststore** đảm bảo bảo mật cho việc truyền tải dữ liệu giữa client và broker. Khi Kafka Broker và Kafka Client giao tiếp qua SSL:
  + **Broker cung cấp chứng chỉ** (certificate từ keystore) cho client để client có thể xác minh danh tính của broker thông qua **truststore**.
  + Nếu broker yêu cầu client cung cấp chứng chỉ (mutual TLS), **client cung cấp chứng chỉ** (certificate từ keystore của client) để broker có thể xác minh qua truststore của broker.

#### ****SASL (Simple Authentication and Security Layer)****

* **Username và Password** cung cấp một lớp xác thực bổ sung cho SSL. Cơ chế SASL sẽ xác minh danh tính của các Kafka Client với broker thông qua việc gửi username và password từ client tới broker. Kafka Broker kiểm tra thông tin này dựa trên cấu hình JAAS của nó.
* Các cơ chế SASL phổ biến trong Kafka:
  + **PLAIN:** Username và password được truyền dưới dạng văn bản (dùng cùng với SSL để mã hóa).
  + **SCRAM-SHA-512:** Đây là cơ chế sử dụng thuật toán băm SHA-512 cho quá trình xác thực, giúp tăng cường bảo mật.

### 4. ****Tóm Tắt Vai Trò của Các Thành Phần****

| **Thành phần** | **Vai trò đối với Kafka Broker** | **Vai trò đối với Kafka Client** |
| --- | --- | --- |
| **Keystore** | Chứa khóa riêng và chứng chỉ của broker, dùng để chứng minh danh tính với client và mã hóa dữ liệu qua SSL. | Chứa khóa riêng và chứng chỉ của client, nếu client yêu cầu mutual TLS. |
| **Truststore** | Chứa chứng chỉ của CA đáng tin cậy, dùng để xác minh tính hợp lệ của chứng chỉ client (nếu có). | Chứa chứng chỉ của CA, dùng để xác minh tính hợp lệ của chứng chỉ broker. |
| **Username/Password (JAAS)** | Cung cấp thông tin xác thực để broker tương tác với các broker khác hoặc Zookeeper. | Cung cấp thông tin xác thực để client chứng minh danh tính với broker. |

### 5. ****Tích hợp SSL/SASL vào Kafka****

SSL giúp bảo mật dữ liệu trong khi SASL bổ sung một lớp xác thực mạnh mẽ. Trong môi trường Kafka, bạn thường sử dụng **SASL\_SSL** để tận dụng cả hai lợi ích này:

* **SSL**: Đảm bảo tính bảo mật của dữ liệu qua mạng.
* **SASL**: Đảm bảo rằng các Kafka Client và Kafka Broker đều được xác thực trước khi có thể giao tiếp.

### 6. ****Các bước cấu hình chính****

* **Bước 1**: Tạo keystore/truststore cho broker và client.
* **Bước 2**: Cấu hình SSL cho Kafka Broker và Kafka Client.
* **Bước 3**: Cấu hình SASL (JAAS) cho Kafka Broker và Kafka Client.
* **Bước 4**: Kiểm tra và xác thực kết nối.

Việc kết hợp đúng các thành phần này giúp bạn đảm bảo an ninh mạng và xác thực mạnh mẽ khi triển khai Kafka trong môi trường sản xuất.