# Lab 8.4: Kafka and SASL SCRAM

Welcome to the session 8 lab 4. The work for this lab is done in `~/kafka-training/labs/lab8.4` . In this lab, you are going to Kafka SASL SCRAM.

Find the latest version of this lab [here](). - Please input the correct link

## Kafka and SASL SCRAM

**SCRAM** is *Salted Challenge Response Authentication Mechanism* (RFC 5802). SCRAM is a **SASL** mechanism that addresses security concerns with traditional mechanisms and is better than PLAIN and DIGEST-MD5.
Kafka supports *SCRAM-SHA-256* and *SCRAM-SHA-512* and can be used with SSL/TLS to perform secure authentication.
Username is used as authenticated Principal for configuration of ACLs.
Default SCRAM implementation stores SCRAM credentials in Zookeeper.

Kafka stores SCRAM credentials in Zookeeper. Zookeeper should be on a private network.
Kafka supports only SHA-256 and SHA-512 with a minimum iteration count of 4096.
Strong hash functions, strong passwords, and high iteration counts protect against brute force attacks.
SCRAM only works with SSL/TLS-encryption to prevent wire snooping.

## Create SCRAM Users

Create users admin, stocks_consumer, stocks_producer store in ZooKeeper

**Create Scram Users**

**~kafka-training/labs/lab8.4/bin/create-scram-users.sh**

```bash
#!/usr/bin/env bash
cd ~/kafka-training
SCRAM_CONFIG='SCRAM-SHA-256=[iterations=8192,password=kafka123]'
SCRAM_CONFIG="$SCRAM_CONFIG,SCRAM-SHA-512=[password=kafka123]"

kafka/bin/kafka-configs.sh \
    --alter --add-config "$SCRAM_CONFIG" \
    --entity-type users --entity-name stocks_consumer
    --zookeeper localhost:2181 \

kafka/bin/kafka-configs.sh \
    --alter --add-config "$SCRAM_CONFIG" \
    --entity-type users --entity-name stocks_producer
    --zookeeper localhost:2181 \

kafka/bin/kafka-configs.sh \
    --alter --add-config "$SCRAM_CONFIG" \
    --entity-type users --entity-name admin
    --zookeeper localhost:2181 \
```

## *ACTION* EDIT bin/create-scram-users.sh and follow instructions in file

## Kafka Broker JAAS Scram Config

Uses Scram for KafkaServer and Plain for ZooKeeper

**~kafka-training/labs/lab8.4/solution/resources/opt/kafka/conf/security/kafka_broker_jaas.conf**

```
KafkaServer {
  org.apache.kafka.common.security.scram.ScramLoginModule required
```

```
  username="admin"
  password="kafka123";
};

// Zookeeper client authentication
Client {
  org.apache.kafka.common.security.plain.PlainLoginModule required
  username="admin"
  password="kafka-123";
};
```

## *ACTION* EDIT solution/resources/opt/kafka/conf/security/kafka_broker_jaas.conf and follow instructions in file

### Kafka Consumer/Producer JAAS Scram Config

Use Scram as login credentials.

**~kafka-training/labs/lab8.4/solution/resources/opt/kafka/conf/security/kafka_consumer_stocks_jaas.conf**

```
KafkaClient {
  org.apache.kafka.common.security.scram.ScramLoginModule required
  username="stocks_consumer"
  password="kafka123";
};
```

## *ACTION* EDIT solution/resources/opt/kafka/conf/security/kafka_consumer_stocks_jaas.conf and follow instructions in file

**~kafka-training/labs/lab8.4/solution/resources/opt/kafka/conf/security/kafka_producer_stocks_jaas.conf**

```
KafkaClient {
  org.apache.kafka.common.security.scram.ScramLoginModule required
  username="stocks_producer"
  password="kafka123";
};
```

## *ACTION* EDIT solution/resources/opt/kafka/conf/security/kafka_producer_stocks_jaas.conf and follow instructions in file

### Configure SCRAM in Producer

Configure SCRAM_SHA_256

**~kafka-training/labs/lab8.4/src/main/java/com/fenago/kafka/producer/support/StockPriceProducerUtils.java**

```
package com.fenago.kafka.producer.support;

import com.fenago.kafka.model.StockPrice;
import io.advantageous.boon.core.Lists;
import org.apache.kafka.clients.CommonClientConfigs;
import org.apache.kafka.clients.producer.KafkaProducer;
```

```java
import org.apache.kafka.clients.producer.Producer;
import org.apache.kafka.clients.producer.ProducerConfig;
import org.apache.kafka.common.serialization.StringSerializer;
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;

import java.util.List;
import java.util.Properties;
import java.util.concurrent.ExecutorService;
import java.util.concurrent.Executors;
import java.util.concurrent.TimeUnit;

public class StockPriceProducerUtils {

    private static Producer<String, StockPrice> createProducer() {

        System.setProperty("java.security.auth.login.config",
                "/opt/kafka/conf/security/kafka_producer_stocks_jaas.conf");
        final Properties props = new Properties();
        props.put(ProducerConfig.BOOTSTRAP_SERVERS_CONFIG,
                "localhost:10092,localhost:10093");
        props.put(CommonClientConfigs.SECURITY_PROTOCOL_CONFIG, "SASL_SSL");

        props.put("sasl.mechanism", "SCRAM-SHA-256");

        props.put("ssl.keystore.location",
                "/opt/kafka/conf/certs/kafka.keystore");
        props.put("ssl.keystore.password", "kafka123");
        props.put("ssl.truststore.location",
                "/opt/kafka/conf/certs/kafka.truststore");
        props.put("ssl.truststore.password", "kafka123");
        props.put(ProducerConfig.CLIENT_ID_CONFIG, "StockPriceProducerUtils");
        props.put(ProducerConfig.KEY_SERIALIZER_CLASS_CONFIG,
                StringSerializer.class.getName());
        props.put(ProducerConfig.VALUE_SERIALIZER_CLASS_CONFIG,
                StockPriceSerializer.class.getName());
        props.put(ProducerConfig.LINGER_MS_CONFIG, 100);
        props.put(ProducerConfig.BATCH_SIZE_CONFIG, 16_384 * 4);
        props.put(ProducerConfig.COMPRESSION_TYPE_CONFIG, "snappy");
        return new KafkaProducer<>(props);
    }
...
}
```

## *ACTION* - EDIT src/main/java/com/fenago/kafka/producer/support/StockPriceProducerUtils.java and follow directions

## Configure SCRAM in Consumer

Configure SCRAM_SHA_256.

**~kafka-training/labs/lab8.4/src/main/java/com/fenago/kafka/consumer/ConsumerUtil.java**

```java
package com.fenago.kafka.consumer;

import com.fenago.kafka.model.StockPrice;
```

```java
import org.apache.kafka.clients.CommonClientConfigs;
import org.apache.kafka.clients.consumer.Consumer;
import org.apache.kafka.clients.consumer.ConsumerConfig;
import org.apache.kafka.clients.consumer.KafkaConsumer;
import org.apache.kafka.common.serialization.StringDeserializer;
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;

import java.util.ArrayList;
import java.util.List;
import java.util.Properties;
import java.util.concurrent.ExecutorService;
import java.util.concurrent.TimeUnit;
import java.util.stream.IntStream;

import static java.util.concurrent.Executors.newFixedThreadPool;

public class ConsumerUtil {
    public static final String BROKERS = "localhost:10092,localhost:10093";

    private static Consumer<String, StockPrice> createConsumer(
            final String bootstrapServers, final String clientId ) {
        System.setProperty("java.security.auth.login.config",
                "/opt/kafka/conf/security/kafka_consumer_stocks_jaas.conf");
        final Properties props = new Properties();
        props.put(ConsumerConfig.BOOTSTRAP_SERVERS_CONFIG,
                bootstrapServers);
        props.put(CommonClientConfigs.SECURITY_PROTOCOL_CONFIG, "SASL_SSL");

        props.put("sasl.mechanism", "SCRAM-SHA-256");

        props.put("ssl.keystore.location",
                "/opt/kafka/conf/certs/kafka.keystore");
        props.put("ssl.keystore.password", "kafka123");
        props.put("ssl.truststore.location",
                "/opt/kafka/conf/certs/kafka.truststore");
        props.put("ssl.truststore.password", "kafka123");
        props.put(ConsumerConfig.ENABLE_AUTO_COMMIT_CONFIG, false);
        props.put(ConsumerConfig.CLIENT_ID_CONFIG, clientId);
        props.put(ConsumerConfig.GROUP_ID_CONFIG,
                "StockPriceConsumer");
        props.put(ConsumerConfig.KEY_DESERIALIZER_CLASS_CONFIG,
                StringDeserializer.class.getName());
        props.put(ConsumerConfig.VALUE_DESERIALIZER_CLASS_CONFIG,
                StockDeserializer.class.getName());
        props.put(ConsumerConfig.MAX_POLL_RECORDS_CONFIG, 500);
        return new KafkaConsumer<>(props);
    }
...
}
```

*ACTION* - **EDIT src/main/java/com/fenago/kafka/consumer/ConsumerUtil.java and follow directions**

**Modify Kafka Brokers Config properties file add SCRAM config**

We will need to edit config files `config/server-0.properties`, `config/server-1.properties`, `config/server-2.properties`.

Enabled **SASL** support to use **PLAIN SASL**.

Inter-broker communication is using **SASL_SSL** and config producers and consumers to use **10092, 10093, 10094** with **SASL_SSL** protocol.

**~kafka-training/labs/lab8.4/config/server-0.properties**

```
broker.id=0

listeners=PLAINTEXT://localhost:9092,SASL_SSL://localhost:10092
sasl.mechanism.inter.broker.protocol=SCRAM-SHA-256
sasl.enabled.mechanisms=SCRAM-SHA-256
security.inter.broker.protocol=SASL_SSL

ssl.keystore.location=/opt/kafka/conf/certs/kafka.keystore
ssl.keystore.password=kafka123
ssl.key.password=kafka123
ssl.truststore.location=/opt/kafka/conf/certs/kafka.truststore
ssl.truststore.password=kafka123
ssl.client.auth=required

log.dirs=./logs/kafka-0
default.replication.factor=3
num.partitions=8
min.insync.replicas=2
auto.create.topics.enable=false
broker.rack=us-west2-a
queued.max.requests=1000
auto.leader.rebalance.enable=true
zookeeper.connect=localhost:2181
delete.topic.enable=true
compression.type=producer
message.max.bytes=65536
replica.lag.time.max.ms=5000
num.network.threads=3
num.io.threads=8
socket.send.buffer.bytes=102400
socket.receive.buffer.bytes=102400
socket.request.max.bytes=104857600
num.recovery.threads.per.data.dir=1
log.retention.hours=168
log.segment.bytes=1073741824
log.retention.check.interval.ms=300000
zookeeper.connection.timeout.ms=6000
```

## *ACTION* - EDIT config/server-0.properties and follow directions

**~kafka-training/labs/lab8.4/config/server-1.properties**

```
broker.id=1
listeners=PLAINTEXT://localhost:9093,SASL_SSL://localhost:10093
sasl.mechanism.inter.broker.protocol=SCRAM-SHA-256
sasl.enabled.mechanisms=SCRAM-SHA-256
security.inter.broker.protocol=SASL_SSL

ssl.keystore.location=/opt/kafka/conf/certs/kafka.keystore
ssl.keystore.password=kafka123
```

```
ssl.key.password=kafka123
ssl.truststore.location=/opt/kafka/conf/certs/kafka.truststore
ssl.truststore.password=kafka123
ssl.client.auth=required

log.dirs=./logs/kafka-1
min.insync.replicas=1
auto.create.topics.enable=false
zookeeper.connect=localhost:2181
num.partitions=1
delete.topic.enable=true
broker.rack=rack1
auto.leader.rebalance.enable=true
compression.type=producer
message.max.bytes=65536
replica.lag.time.max.ms=5000
num.network.threads=3
num.io.threads=8
socket.send.buffer.bytes=102400
socket.receive.buffer.bytes=102400
socket.request.max.bytes=104857600
num.recovery.threads.per.data.dir=1
log.retention.hours=168
log.segment.bytes=1073741824
log.retention.check.interval.ms=300000
zookeeper.connection.timeout.ms=6000
```

### *ACTION* - EDIT config/server-1.properties and follow directions

**~kafka-training/labs/lab8.4/config/server-2.properties**

```
broker.id=2

listeners=PLAINTEXT://localhost:9094,SASL_SSL://localhost:10094
sasl.mechanism.inter.broker.protocol=SCRAM-SHA-256
sasl.enabled.mechanisms=SCRAM-SHA-256
security.inter.broker.protocol=SASL_SSL

ssl.keystore.location=/opt/kafka/conf/certs/kafka.keystore
ssl.keystore.password=kafka123
ssl.key.password=kafka123
ssl.truststore.location=/opt/kafka/conf/certs/kafka.truststore
ssl.truststore.password=kafka123
ssl.client.auth=required

log.dirs=./logs/kafka-2
min.insync.replicas=1
auto.create.topics.enable=true
zookeeper.connect=localhost:2181
num.partitions=1
delete.topic.enable=true
broker.rack=rack2
auto.leader.rebalance.enable=true
compression.type=producer
message.max.bytes=65536
replica.lag.time.max.ms=5000
num.network.threads=3
num.io.threads=8
```

```
socket.send.buffer.bytes=102400
socket.receive.buffer.bytes=102400
socket.request.max.bytes=104857600
num.recovery.threads.per.data.dir=1
log.retention.hours=168
log.segment.bytes=1073741824
log.retention.check.interval.ms=300000

zookeeper.connection.timeout.ms=6000
```

*ACTION* - **EDIT config/server-2.properties and follow directions**

**Run the lab**

*ACTION* - **RUN ZooKeeper and three Kafka Brokers (scripts are under bin for ZooKeeper and Kafka Brokers).**

*ACTION* - **RUN ConsumerBlueMain from the IDE**

*ACTION* - **RUN StockPriceProducer from the IDE**

**Expected results**

You should be able to send records from the producer to the broker and read records from the consumer to the broker using SASL SCRAM auth.