


Kafka 的认证机制



吃货大米饭

关注

IP属地: 广东

2023.08.31 20:56:43

字数 977

阅读 1,262

Kafka 认证机制

自 0.9.0.0 版本开始，Kafka 正式引入了认证机制，用于实现基础的安全用户认证，这是将 Kafka 上云或进行多租户管理的必要步骤。截止到当前最新的 2.3 版本，Kafka 支持基于 SSL 和基于 SASL 的安全认证机制。

基于 SSL 的认证主要是指 Broker 和客户端的双路认证（2-way authentication）。通常来说，SSL 加密（Encryption）已经启用了单向认证，即客户端认证 Broker 的证书（Certificate）。如果要做 SSL 认证，那么我们要启用双路认证，也就是说 Broker 也要认证客户端的证书。

kafka 还支持通过 SASL 做客户端认证。SASL 是提供认证和数据安全服务的框架。Kafka 支持的 SASL 机制有 5 种，它们分别是在不同版本中被引入的，你需要根据你自己使用的 Kafka 版本，来选择该版本所支持的认证机制。

认证机制	引入版本	推荐理由
SSL	0.9	适用于一般测试场景。
SASL / GSSAPI	0.9	适用于本身已经实现的Kerberos认证的场景。
SASL / PLAIN	0.10.2	适用于中小型公司的Kafka集群。
SASL / SCRAM	0.10.2	适用于中小型公司的Kafka集群，支持认证用户的动态增减。
SASL / OAUTHBEARER	2.0	适用于支持OAuth 2.0框架的场景。
Delegation Token	1.1	适用于Kerberos认证中出现TGT分发性能瓶颈的场景。

建议:你可以使用 SSL 来做通信加密，使用 SASL 来做 Kafka 的认证实现。

SASL/SCRAM-SHA-256 配置实例

第 1 步：创建用户

配置 SASL/SCRAM 的第一步，是创建能否连接 Kafka 集群的用户。在本次测试中，我会创建 3 个用户，分别是 admin 用户、writer 用户和 reader 用户。admin 用户用于实现 Broker 间通信，writer 用户用于生产消息，reader 用户用于消费消息。

```
1 | kafka-configs.sh --zookeeper localhost:2181 --alter --add-config 'SCRAM-SHA-256=[password=admin],SCRAM-SHA-512=[password=admin]' --entity-type users --entity-name admin
```

```
1 | kafka-configs.sh --zookeeper localhost:2181 --alter --add-config 'SCRAM-SHA-256=[password=writer],SCRAM-SHA-512=[password=writer]' --entity-type users --entity-name writer
```

```
1 | kafka-configs.sh --zookeeper localhost:2181 --alter --add-config 'SCRAM-SHA-256=[password=reader],SCRAM-SHA-512=[password=reader]' --entity-type users --entity-name reader
```

查看用户：

```
1 | kafka-configs.sh --zookeeper localhost:2181 --describe --entity-type users --entity-name writer
```

删除用户：

```
1 | kafka-configs.sh --zookeeper localhost:2181 --alter --delete-config 'SCRAM-SHA-512' --entity-type users --entity-name writer
```

第 2 步：创建 JAAS 文件

配置了用户之后，我们需要为每个 Broker 创建一个对应的 JAAS 文件。因为本例中的只有一个 Broker 在一台机器上，所以我只创建了一份 JAAS 文件。但是你要切记，在实际场景中，你需要为每台单独的物理 Broker 机器都创建一份 JAAS 文件。

JAAS 的文件内容如下：

```
1 | KafkaServer {
2 |   org.apache.kafka.common.security.scram.ScramLoginModule required
3 |   username="admin"
4 |   password="admin";
5 | };
```

关于这个文件内容，你需要注意以下两点：

- 不要忘记最后一行和倒数第二行结尾处的分号；
- JAAS 文件中不需要任何空格键。

这里，我们使用 admin 用户实现 Broker 之间的通信。接下来，我们来配置 Broker 的 server.properties 文件，下面这些内容，是需要单独配置的：

```
1 | # 认证配置
2 | sasl.enabled.mechanisms=SCRAM-SHA-256
3 | sasl.mechanism.inter.broker.protocol=SCRAM-SHA-256
4 | security.inter.broker.protocol=SASL_PLAINTEXT
5 | listeners=SASL_PLAINTEXT://localhost:9092
```

第 1 项内容表明开启 SCRAM 认证机制，并启用 SHA-256 算法；
第 2 项的意思是为 Broker 间通信也开启 SCRAM 认证，同样使用 SHA-256 算法；
第 3 项表示 Broker 间通信不配置 SSL，本例中我们不演示 SSL 的配置；
最后 1 项是设置 listeners 使用 SASL_PLAINTEXT，依然是不使用 SSL。

第 3 步：启动 Broker

修改 /usr/local/kafka/bin/kafka-server-start.sh

```
1 | # exec $base_dir/kafka-run-class.sh $EXTRA_ARGS com.cloudera.kafka.wrap.Kafka "$@"
2 | exec $base_dir/kafka-run-class.sh $EXTRA_ARGS -Djava.security.auth.login.config=$base_dir/./config/kafka_server_jaas.conf com.cloudera.kafka.wrap.Kafka "$@"
```

第 4 步：发送消息

先创建一个topic:

```
1 | kafka-topics.sh --create --zookeeper localhost:2181 --partitions 1 --replication-factor 1 --topic test
```

在创建好测试主题之后，我们使用 kafka-console-producer 脚本来尝试发送消息。由于启用了认证，客户端需要做一些相应的配置。我们创建一个名为 producer.conf 的配置文件，内容如下：

```
1 |
2 | security.protocol=SASL_PLAINTEXT
3 | sasl.mechanism=SCRAM-SHA-256
4 | sasl.jaas.config=org.apache.kafka.common.security.scram.ScramLoginModule required username="writer" password="writer";
```

之后运行 Console Producer 程序：

```
1 | kafka-console-producer.sh --broker-list localhost:9092 --topic test --producer.config <your_path>/producer.conf
2 | >hello, world
3 | >
```

第 5 步：消费消息

接下来，我们使用 Console Consumer 程序来消费一下刚刚生产的消息。同样地，我们需要为 kafka-console-consumer 脚本创建一个名为 consumer.conf 的脚本，内容如下：

```
1 |
2 | security.protocol=SASL_PLAINTEXT
3 | sasl.mechanism=SCRAM-SHA-256
4 | sasl.jaas.config=org.apache.kafka.common.security.scram.ScramLoginModule required username="reader" password="reader";
```

之后运行 Console Consumer 程序：

```
1 | kafka-console-consumer.sh --bootstrap-server localhost:9092 --topic test --from-beginning --consumer.config <your_path>/consumer.conf
2 | hello, world
```

第 6 步：动态增减用户

最后，我们来演示 SASL/SCRAM 动态增减用户的场景。假设我删除了 writer 用户，同时又添加了一个新用户：new_writer，那么，我们需要执行的命令如下：

```
1 |
2 | kafka-configs.sh --zookeeper localhost:2181 --alter --delete-config 'SCRAM-SHA-256' --entity-type users --entity-name writer
3 | Completed Updating config for entity: user-principal 'writer'.
4 |
5 | kafka-configs.sh --zookeeper localhost:2181 --alter --delete-config 'SCRAM-SHA-512' --entity-type users --entity-name writer
6 | Completed Updating config for entity: user-principal 'writer'.
7 |
8 | kafka-configs.sh --zookeeper localhost:2181 --alter --add-config 'SCRAM-SHA-256=[iterations=8192,password=new_writer]' --entity-type users --entity-name new_writer
9 | Completed Updating config for entity: user-principal 'new_writer'.
```