Securing Apache Kafka

Outline

Tos

- Kafka and security overview
- Authentication
  - Identify the principal (user) associated with a connection
- Authorization
  - What permission a principal has
- Secure Zookeeper

Security Overview

Tos

- Support since 0.9.0

- Wire encryption btw client and broker
  - For cross data center mirroring

- Access control on resources such as topics
  - Enable sharing Kafka clusters

Authentication Overview

Tos

- Broker support multiple ports
  - plain text (no wire encryption/authentication)
  - SSL (for wire encryption/authentication)
  - SASL (for Kerberos authentication)
  - SSL + SASL (SSL for wire encryption, SASL for authentication)

- Clients choose which port to use
  - need to provide required credentials through configs

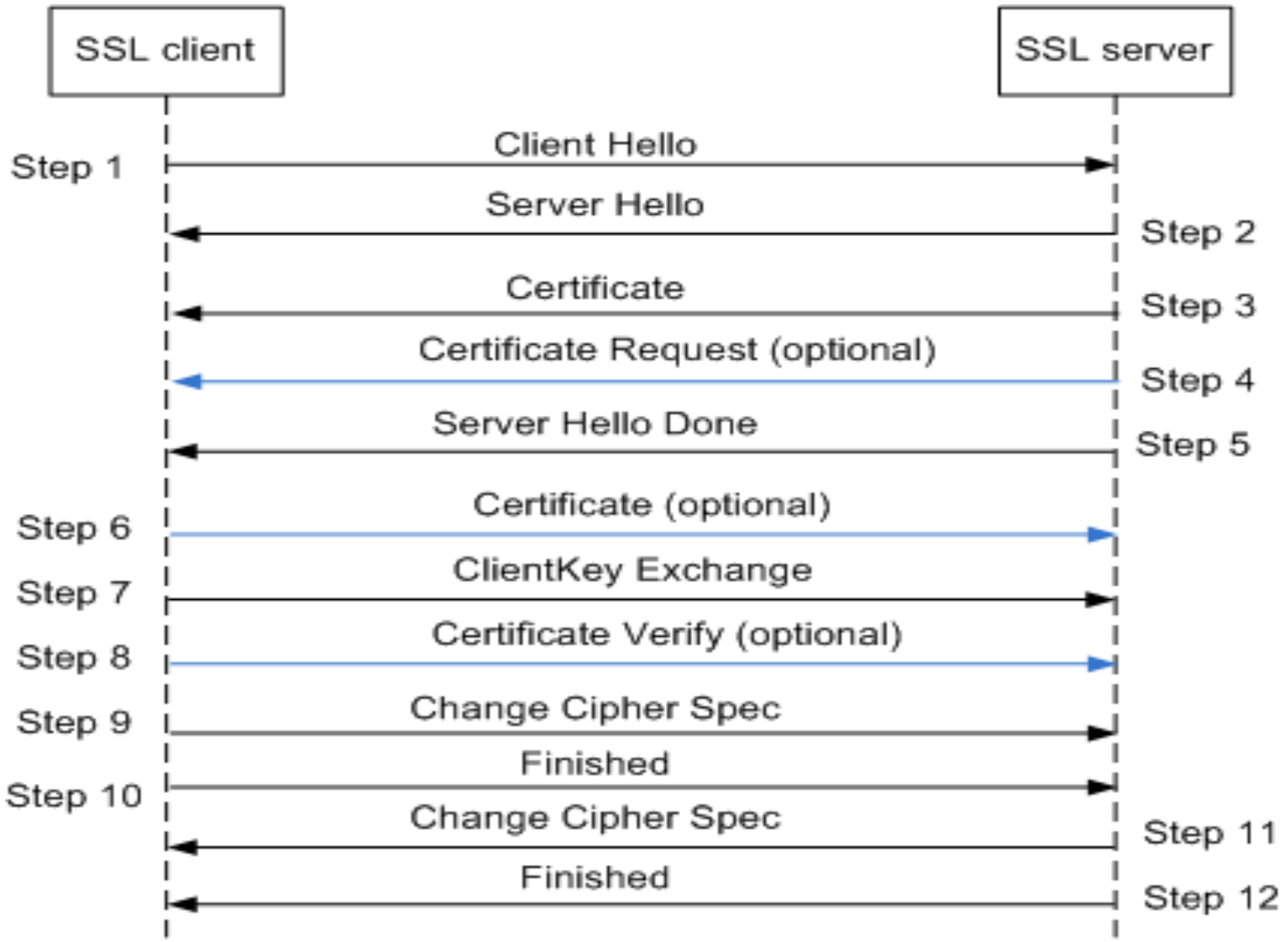
Why is SSL useful

Tos

- 1-way authentication
  - Secure wire transfer through encryption
- 2-way authentication
  - Broker knows the identity of client
- Easy to get started
  - Just involve client and server

SSL handshake

Tos



Subsequent transfer over SSL

Tos

- Data encrypted with agreed upon cipher suite
  - Encryption overhead
  - Losing zero-copy transfer in consumer

Performance impact with SSL

Tos

- **r3.xlarge**
  - **4 core, 30GB ram, 80GB ssd, moderate network (~90MB/s)**

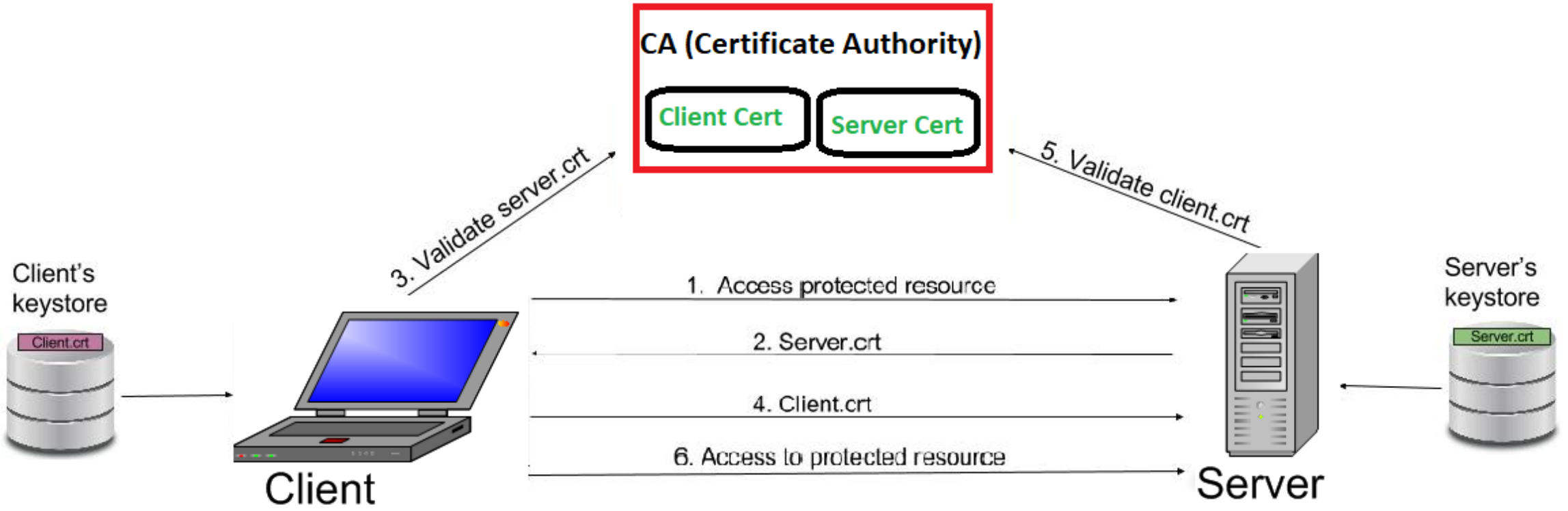
	Throughput(MB/S)	CPU on client	CPU on broker
Producer(plaintext)	83	12%	30%
Producer (SSL)	69	28%	48%
Consumer (plaintext)	83	8%	2%
Consumer (SSL)	69	27%	24%

- **Most overhead from encryption**

Preparing SSL

Tos

1. Generate certificate (X509) in broker key store
2. Generate certificate authority (CA) for signing
3. Sign broker certificate with CA
4. Import signed certificate and CA to broker key store
5. Import CA to client trust store
6. 2-way authentication: generate client certificate in a similar way



Configuring SSL

Tos

- No client code change; just configuration change.

Client/Broker

```
ssl.keystore.location =
/var/private/ssl/kafka.server.keystore.jks
ssl.keystore.password = test1234
ssl.key.password = test1234
ssl.truststore.location =
/var/private/ssl/kafka.server.truststore.jks
ssl.truststore.password = test1234
```

Broker

```
listeners = SSL://host.name:port
security.inter.broker.protocol = SSL
ssl.client.auth = required
```

Client

```
security.protocol = SSL
```

What is SASL

Tos

- Simple Authentication and Security Layer
  - Challenge/response protocols
  - Server issues challenge and client sends response
  - Continue until server is satisfied
- Different mechanisms
  - Plain: cleartext username/password
  - Digest MD5
  - GSSAPI: Kerberos
- Kafka 0.9.0 only supports Kerberos

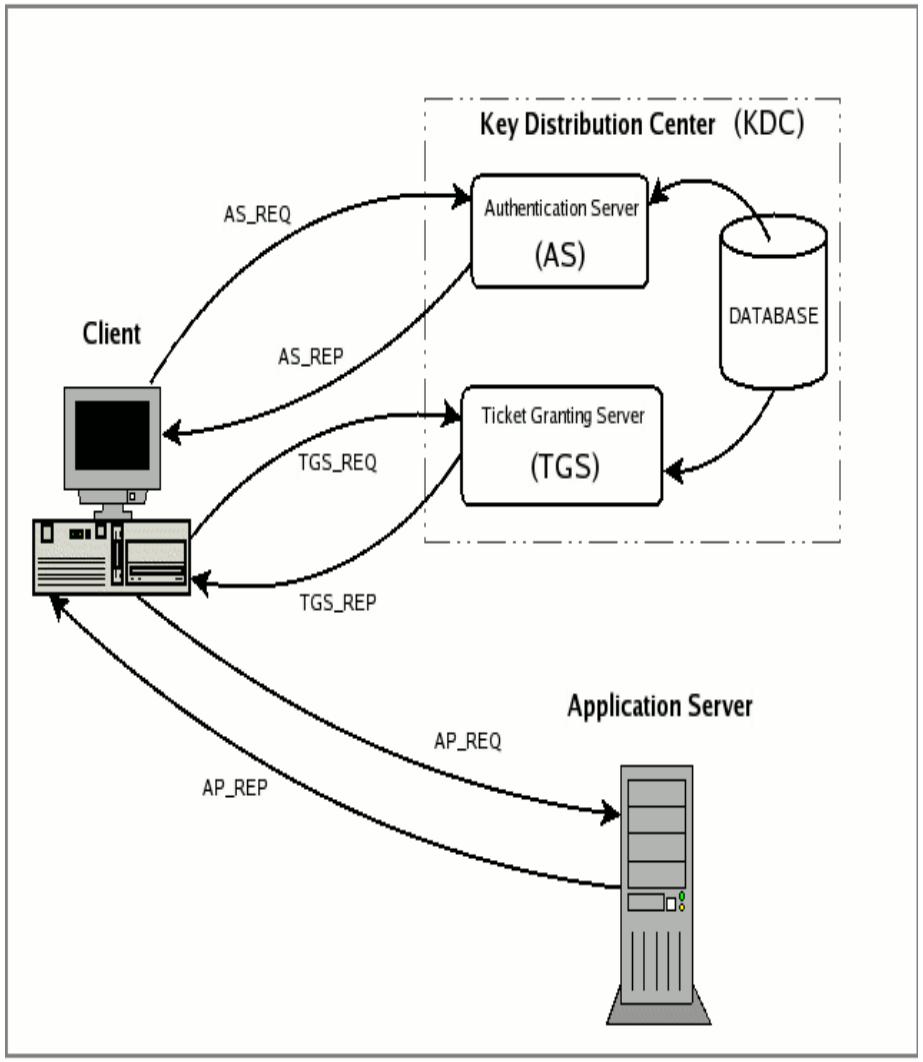
Why Kerberos

Tos

- Secure single sign-on
    - An organization may provide multiple services
    - User just remember a single Kerberos password to use all services
- More convenient when there are many users
- Need Key Distribution Center (KDC)
    - Each service/user need a Kerberos principal in KDC

How Kerberos Works

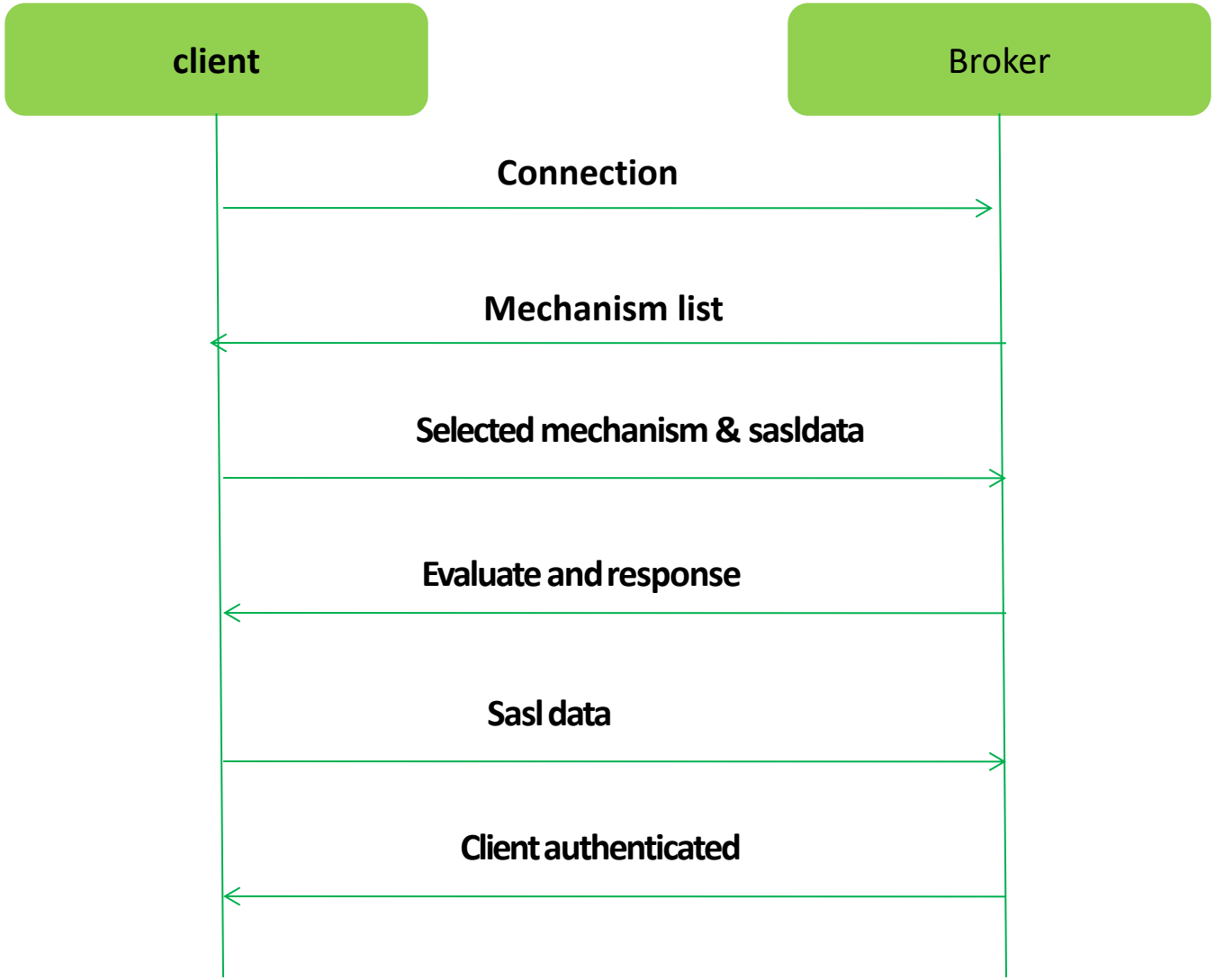
Tos



- Create service and client principal in KDC
- Client authenticate with AS on startup
- Client obtain service ticket from TGS
- Client authenticate with service using service ticket

SASL handshake

Tos



Data transfer

Tos

- SASL_PLAINTEXT
  - No wire encryption
- SASL_SSL
  - Wire encryption over SSL

Preparing Kerberos

Tos

- Create Kafka service principal in KDC
- Create a keytab for Kafka principal
    - Keytab includes principal and encrypted Kerberos password
    - Allow authentication w/o typing password
- Create an application principal for client KDC
- Create a keytab for application principal

Configuring Kerberos

Tos

No client code change; just configuration change

Broker JAAS file

```
KafkaServer {
    com.sun.security.auth.module.
    Krb5LoginModule required
    useKeyTab=true
    storeKey=true
    keyTab="/etc/security/keyt
    abs/kafka_server.keytab"
    principal="kafka/kafka1.ho
    stname.com@EXAMPLE.COM";
};
```

Client JAAS file

```
KafkaClient {
    com.sun.security.auth.module.
    Krb5LoginModule required
    useKeyTab=true
    storeKey=true
    keyTab="/etc/security/keyt
    abs/kafka_client.keytab"
    principal="kafka-client-
    1@EXAMPLE.COM";
};
```

Broker JVM

```
Djava.security.auth.lo
gin.config=/etc/kafka/
kafka_server_jaas.conf
```

ClientJVM

```
-
Djava.security.auth.log
in.config=/etc/kafka/
kafka_client_jaas.conf
```

Broker config

```
security.inter.broker.protocol=
SASL_PLAINTEXT(SASL_SSL)
sasl.kerberos.service.name=kafka
```

Client config

```
security.protocol=SA
SL_PLAINTEXT(SASL_SSL)
sasl.kerberos.servic
e.name=kafka
```

Kerberos principal name

Tos

- Kerberos principal
  - Primary[/Instance]@REALM
  - kafka/kafka1.hostname.com@EXAMPLE.COM
  - kafka-client-1@EXAMPLE.COM
- Primary extracted as the default principal name
- Can customize principal name through

  sasl.kerberos.principal.to.local.rules

Authentication Caveat

Tos

- Authentication (SSL or SASL) happens once during socket connection
  - No re-authentication
- If a certificate needs to be revoked, use authorization to remove permission

Authorization

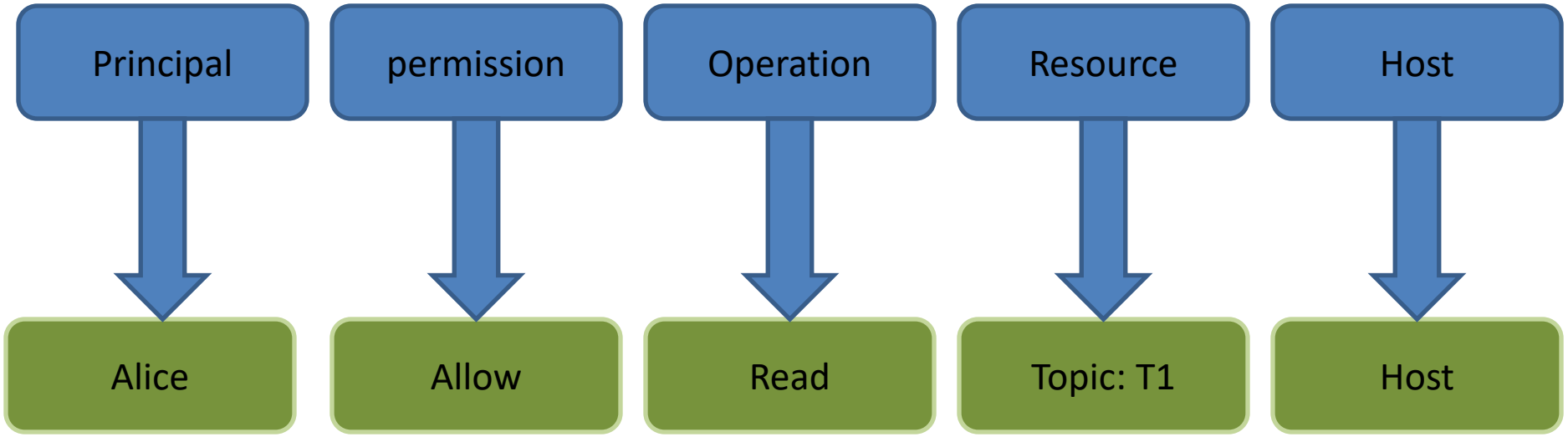
Tos

- Control which permission each authenticated principal has
- Pluggable with a default implementation

ACL

Tos

Alice is Allowed to Read from topic T1 from Host1



Operations and Resources

Tos

- Operations
  - Read, Write, Create, Describe, ClusterAction, All
- Resources
  - Topic, Cluster and ConsumerGroup

Operations	Resources
Read, write, Describe (Read, Write implies Describe)	Topic
Read	Consumer Group
Create, ClusterAction(communication between controller and brokers)	Cluster

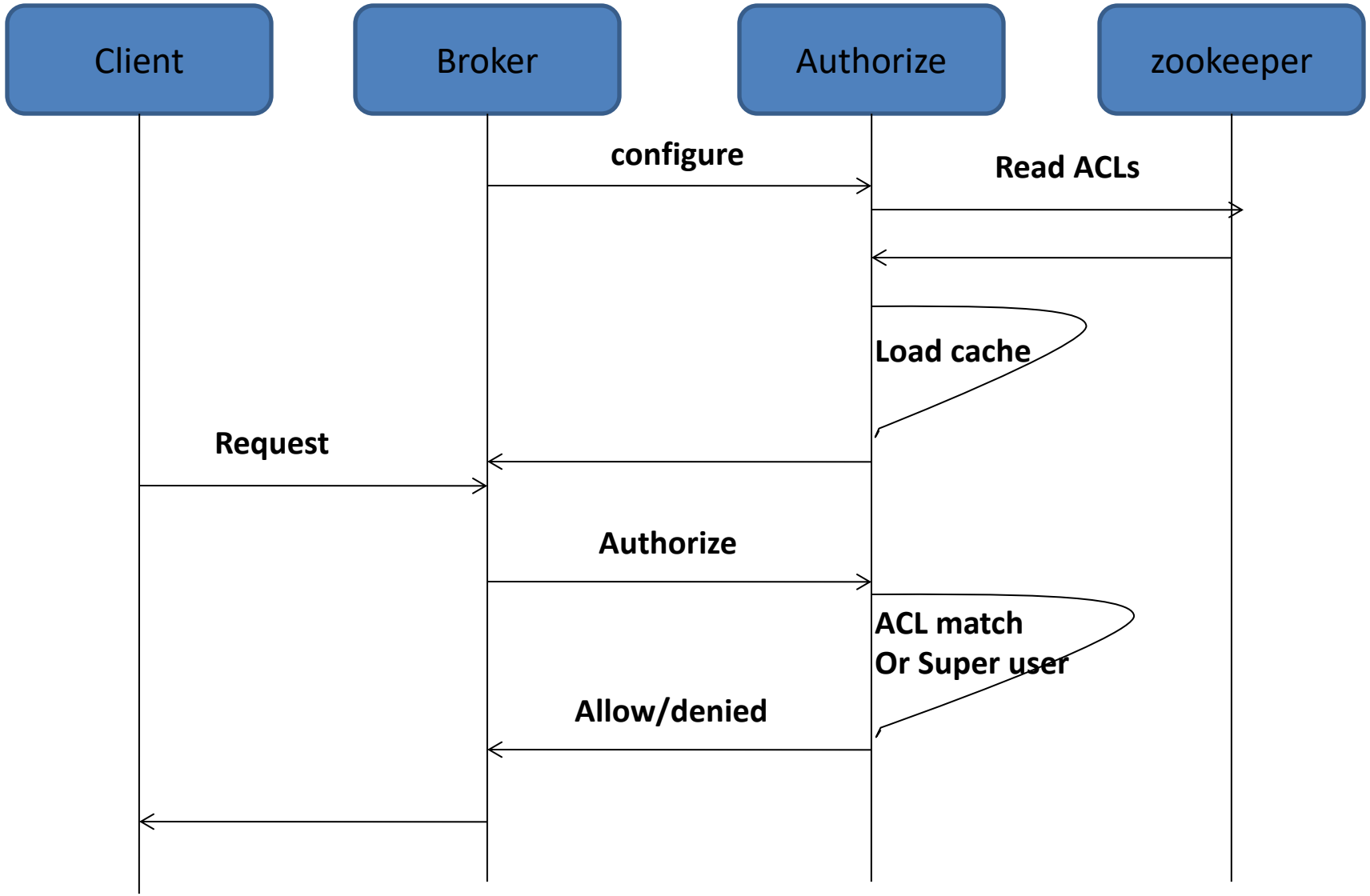
SimpleAclAuthorizer

Tos

- Out of box authorizer implementation.
- CLI tool for adding/removing acls
- ACLs stored in zookeeper and propagated to brokers asynchronously
- ACL cache in broker for better performance

Authorizer Flow

Tos



Configure broker ACL

Tos

- authorizer.class.name=kafka.security.auth.SimpleAclAuthorizer

- Make Kafka principal super users
  - Or grant ClusterAction and Read all topics to Kafka principal

Configure client ACL

Tos

- Producer
  - Grant Write on topic, Create on cluster (auto creation)
  - Or use --producer option in CLI

```
bin/kafka-acls --authorizer-properties zookeeper.connect=localhost:2181 \
    --add --allow-principal User:Bob --producer --topic t1
```

- Consumer
  - Grant Read on topic, Read on consumer group
  - Or use --consumer option in CLI

```
bin/kafka-acls --authorizer-properties zookeeper.connect=localhost:2181 \
    --add --allow-principal User:Bob --consumer --topic t1 --group group1
```

Secure Zookeeper

Tos

- Zookeeper stores
  - critical Kafka metadata
  - ACLs
- Need to prevent untrusted users from modifying

Zookeeper Security Integration

Tos

- ZK supports authentication through SASL
  - Kerberos or Digest MD5
- Set zookeeper.set.acl to true on every broker
- Configure ZK user through JAAS config file
  - Each ZK path writable by creator, readable by all

Migrating from non-secure to secure Kafka

Tos

- Configure brokers with multiple ports
    - listeners=PLAINTEXT://host.name:port,SSL://host.name:port
- Gradually migrate clients to secure port
- When done
    - Turn off PLAINTEXT port on brokers

Lab : - Securing Kafka