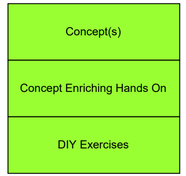
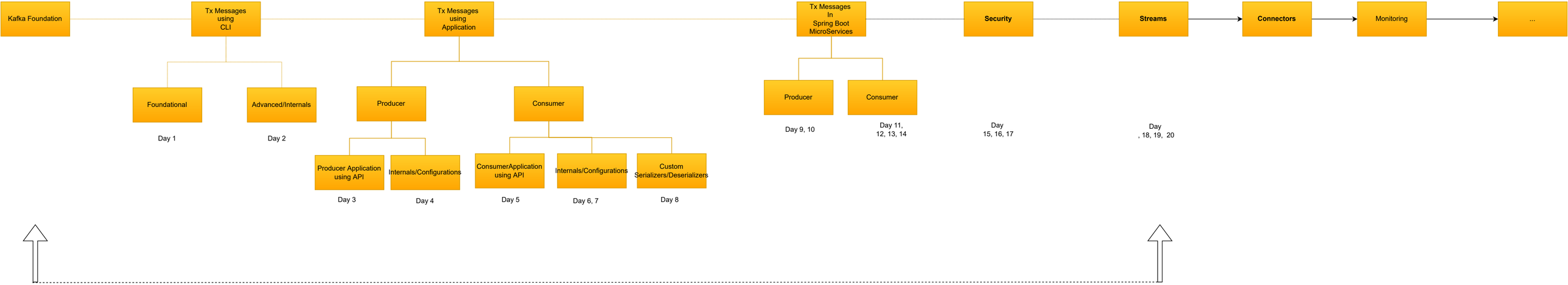
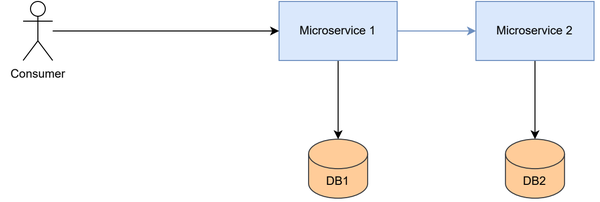
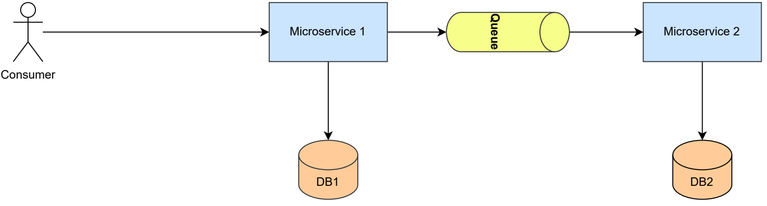
Teaching Principle



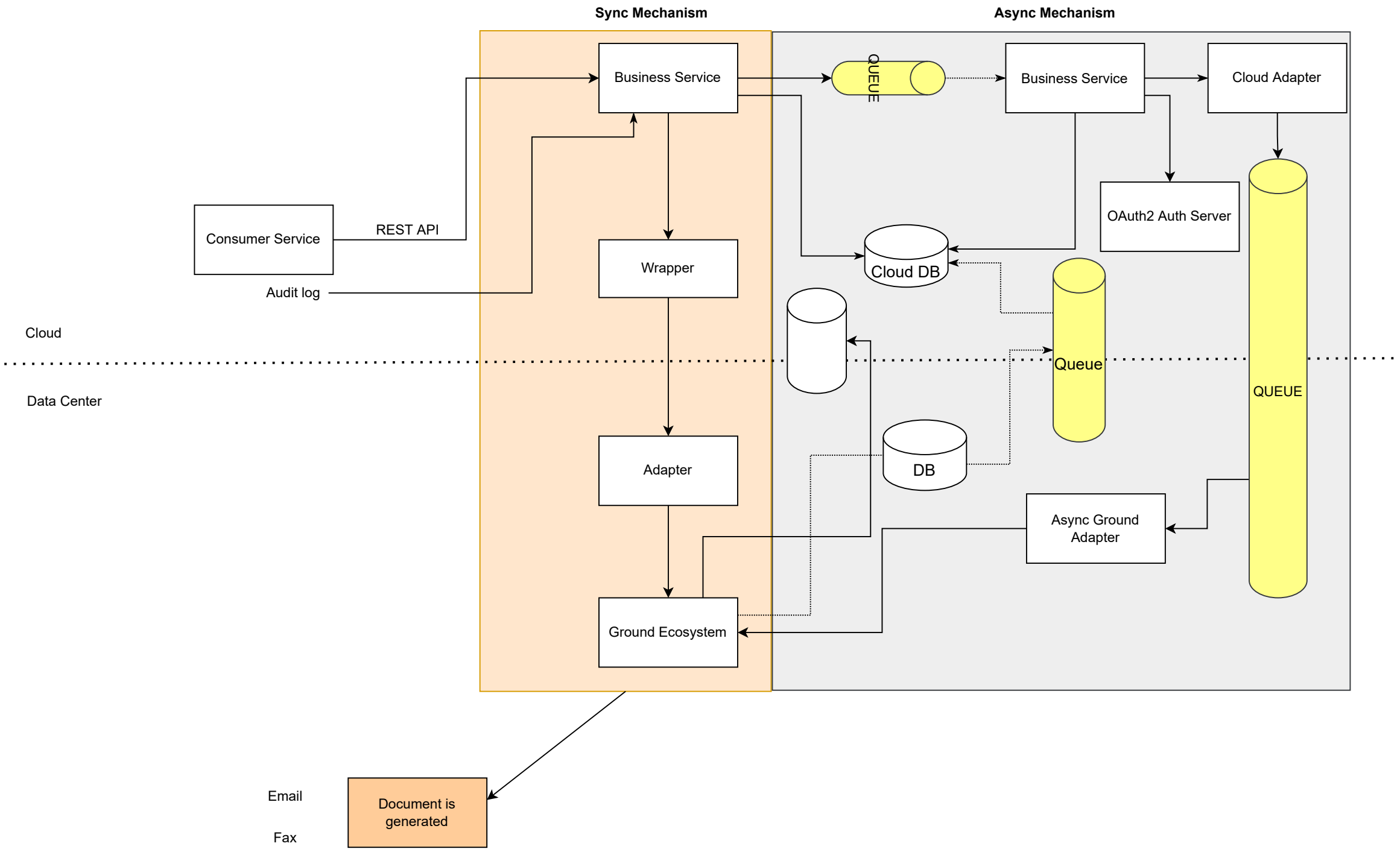
Learning Map



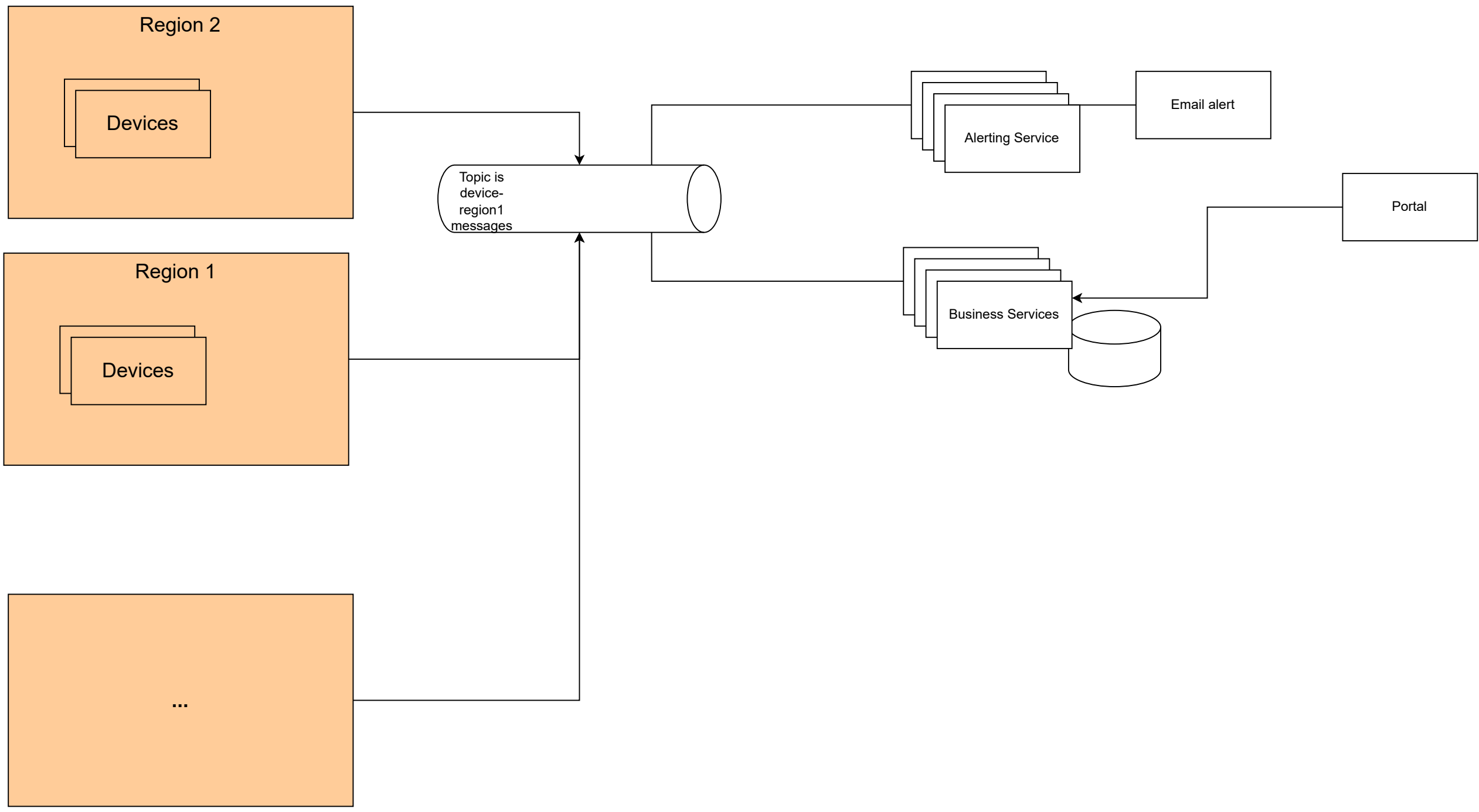

Consumer
Microservice 1
Microservice 2
DB1
DB2


Consumer
Microservice 1
Queue
Microservice 2
DB1
DB2

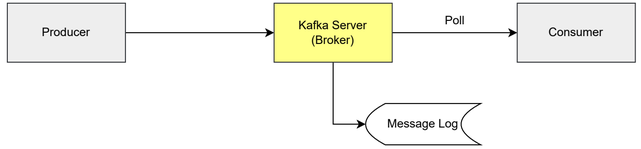
Use case 1



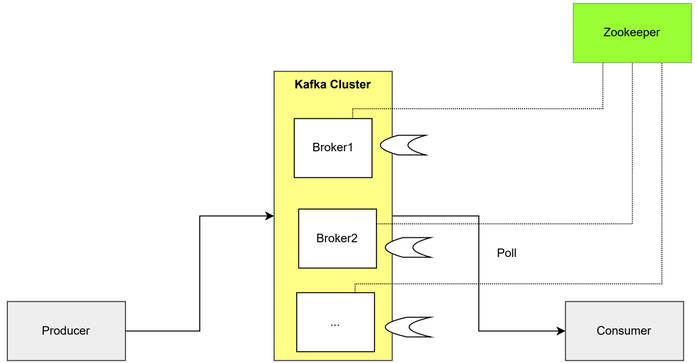
Use case 2



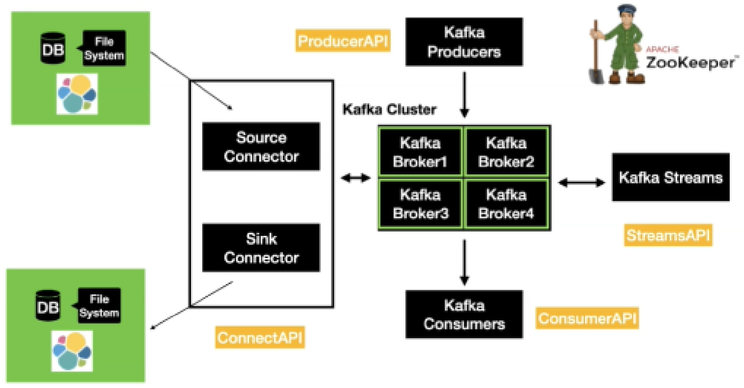
Basic Topology



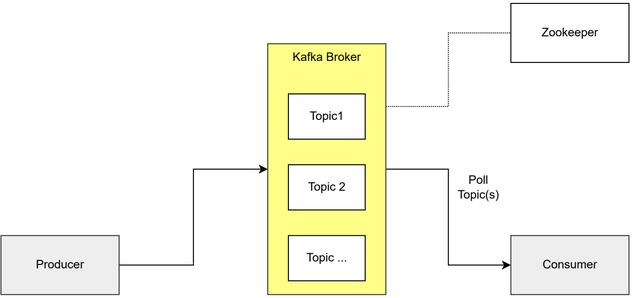
Distributed Topology



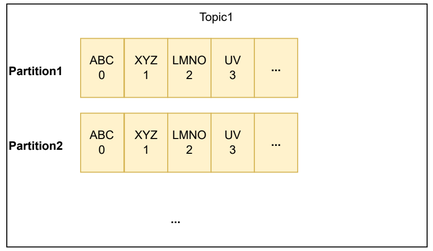
Kafka Landscape



Broker Anatomy

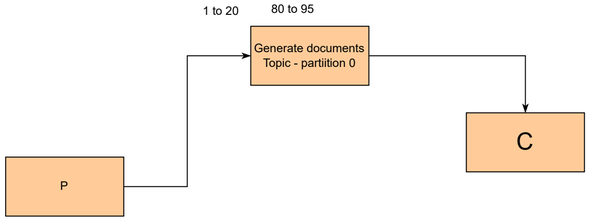


Topic Anatomy

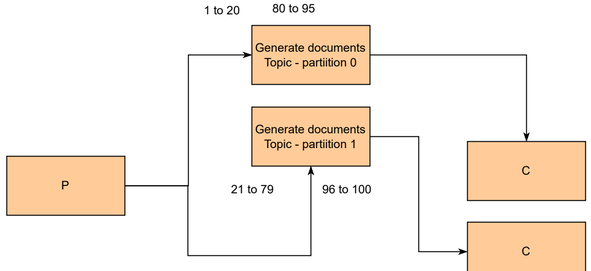


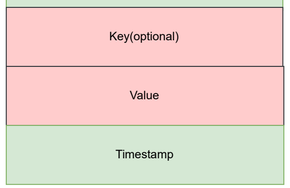
Message Anatomy

Topic-Same

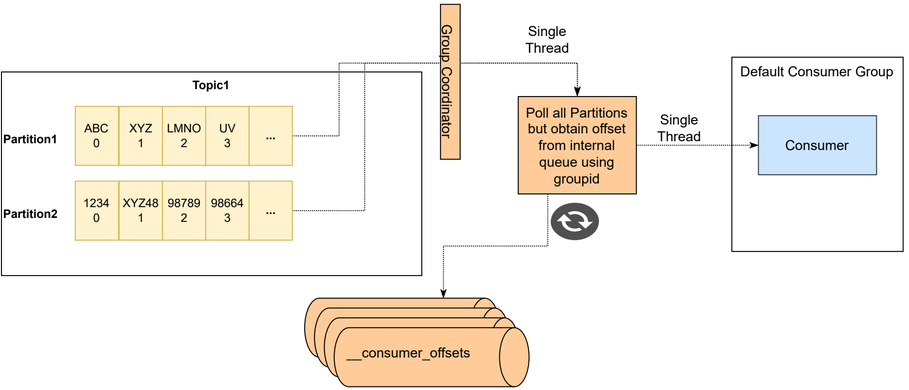


Topic-Same

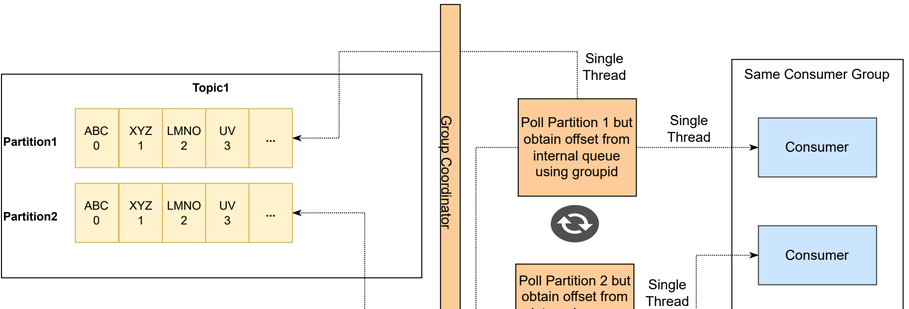


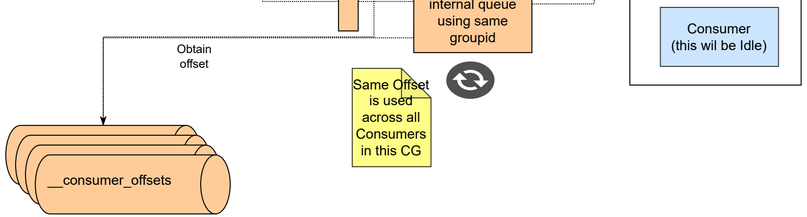

Key(optional)
Value
Timestamp

Default Consumer Group

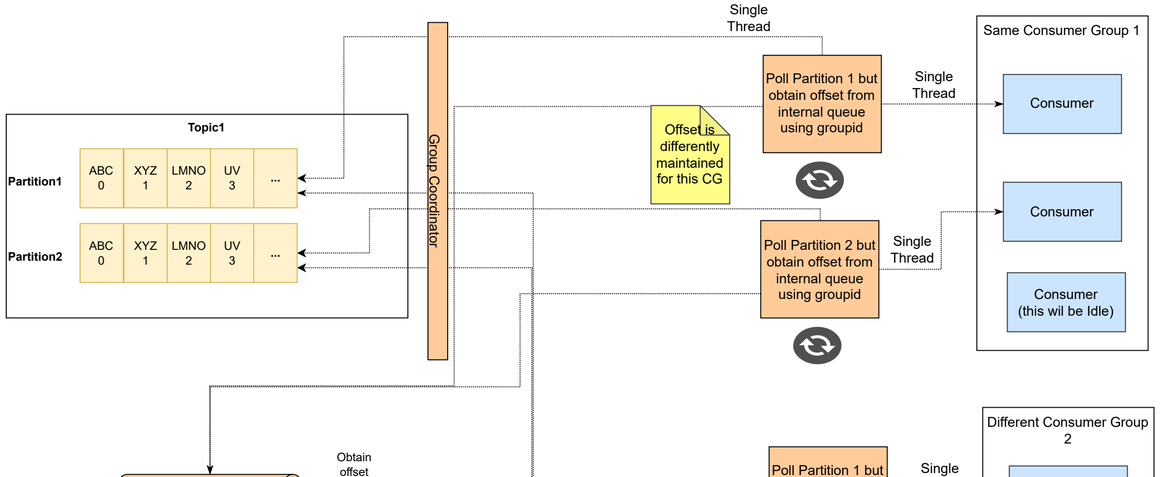


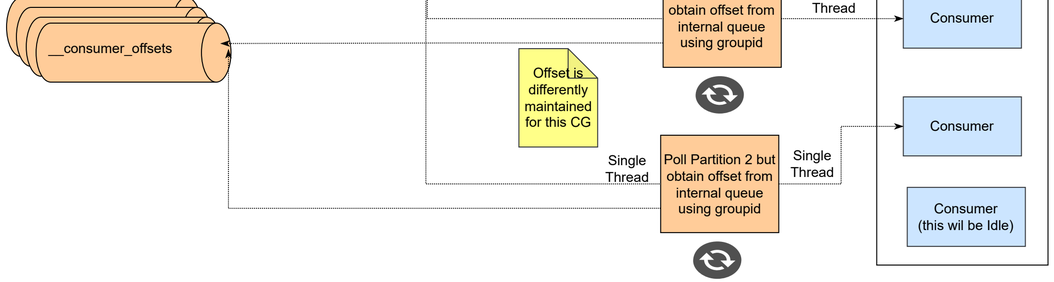
Using same Consumer Group to Read Messages at Scale



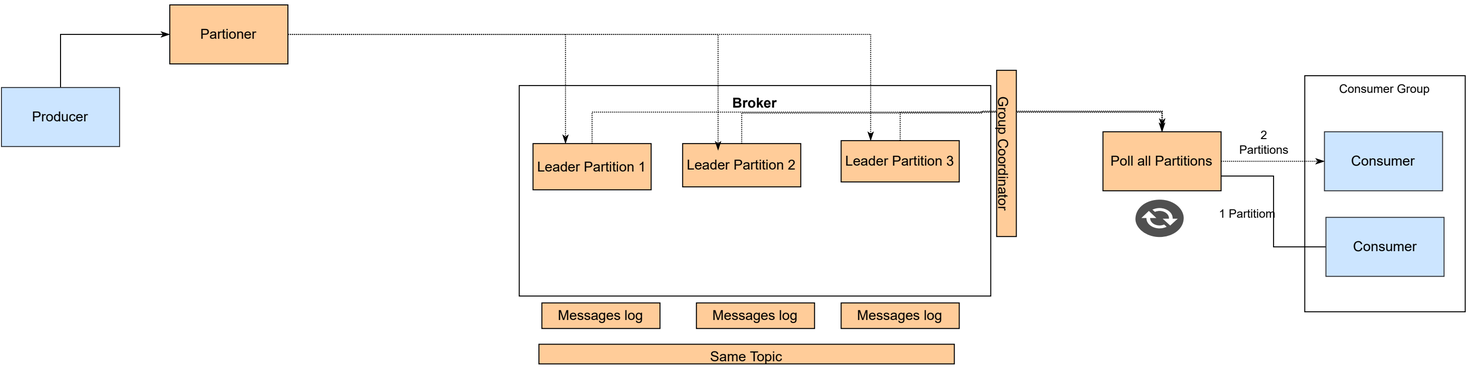


Using Different Consumer Group to Read Messages From Same Topic at Scale

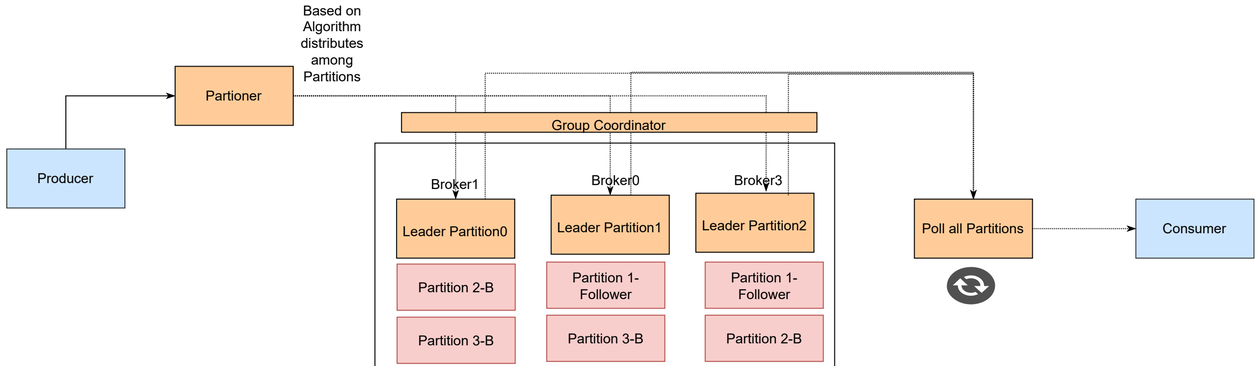


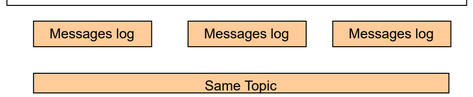

__consumer_offsets
obtain offset from internal queue using groupid
Thread
Consumer
Offset is differently maintained for this CG
Consumer
Single Thread
Poll Partition 2 but obtain offset from internal queue using groupid
Single Thread
Consumer (this wil be Idle)

Standalone Topology

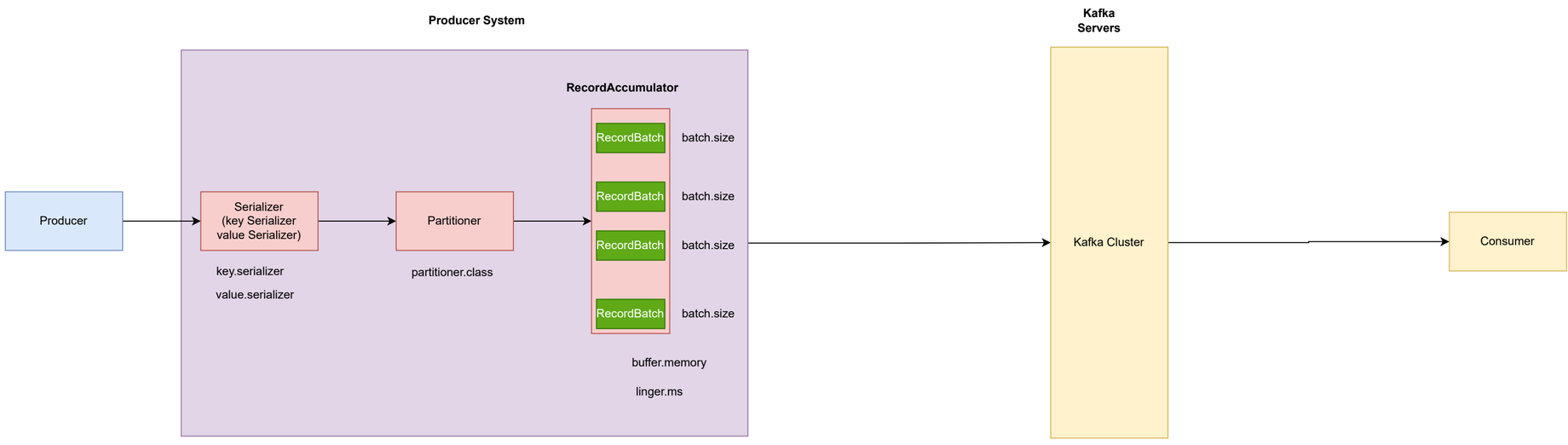


Distributed Topology

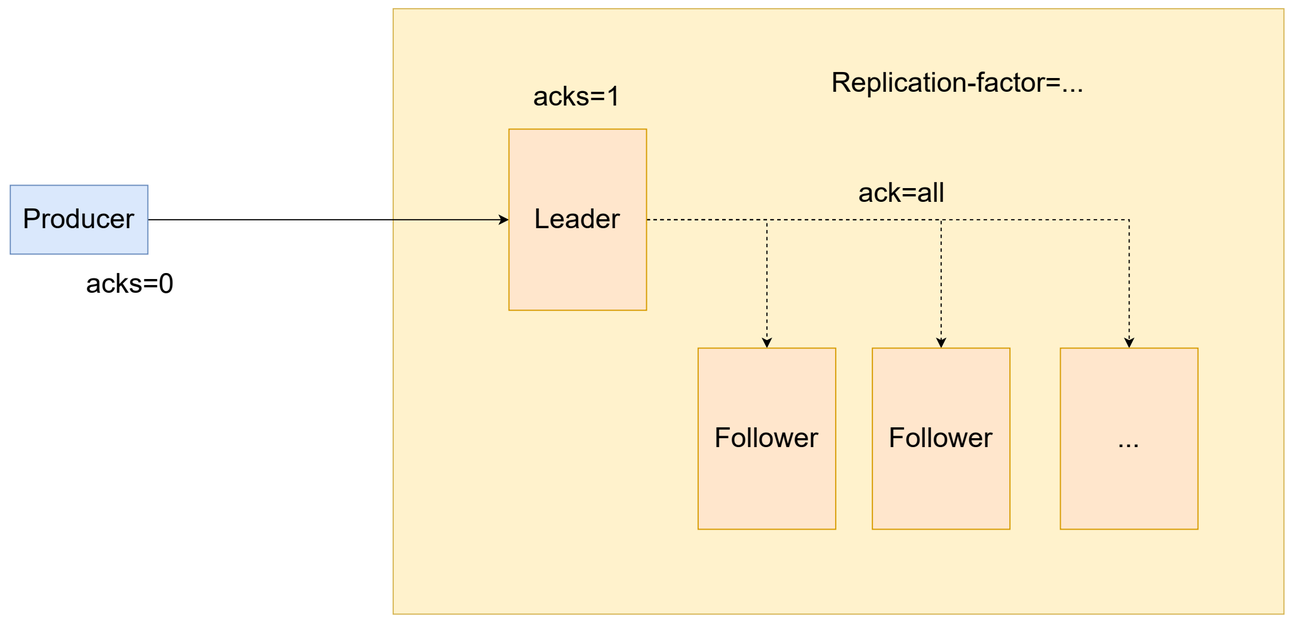



Messages log
Messages log
Messages log
Same Topic

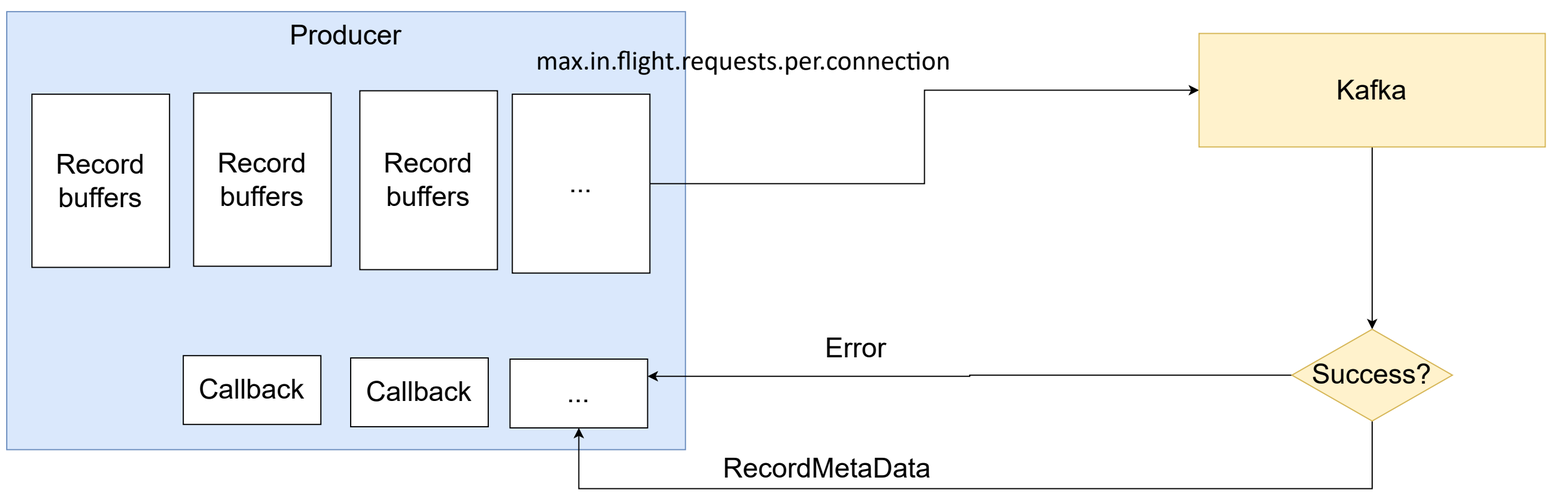
Producer Application Internals & Configurations



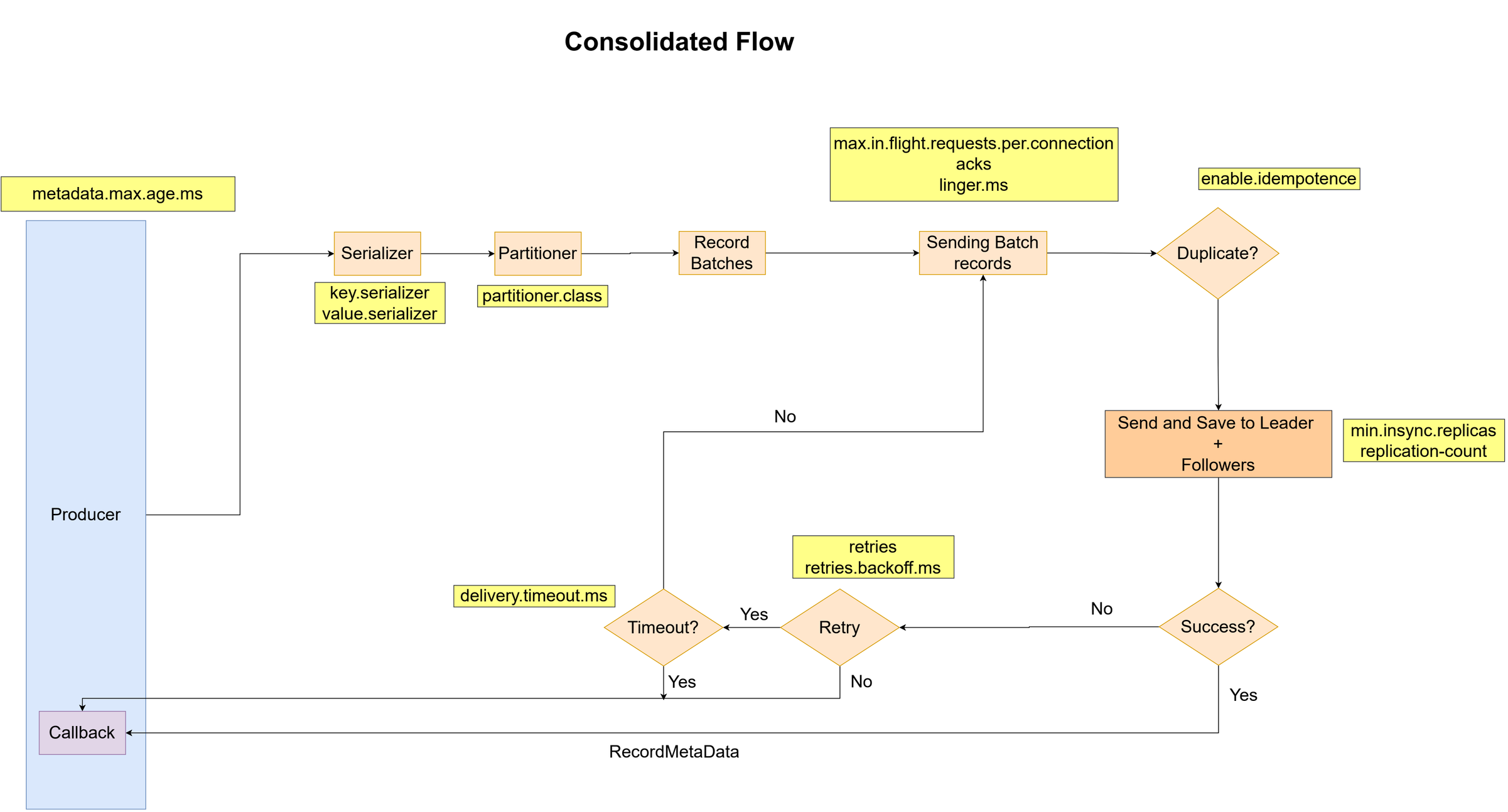
ack configuration



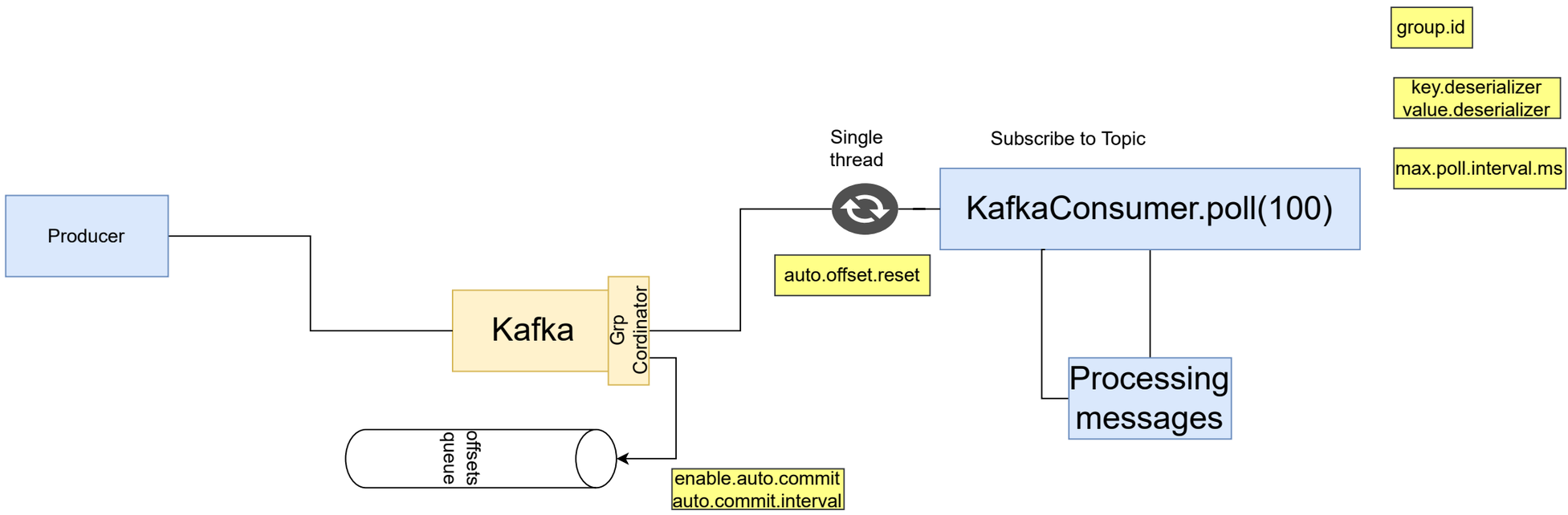
max.in.flight.requests.per.connection Configuration



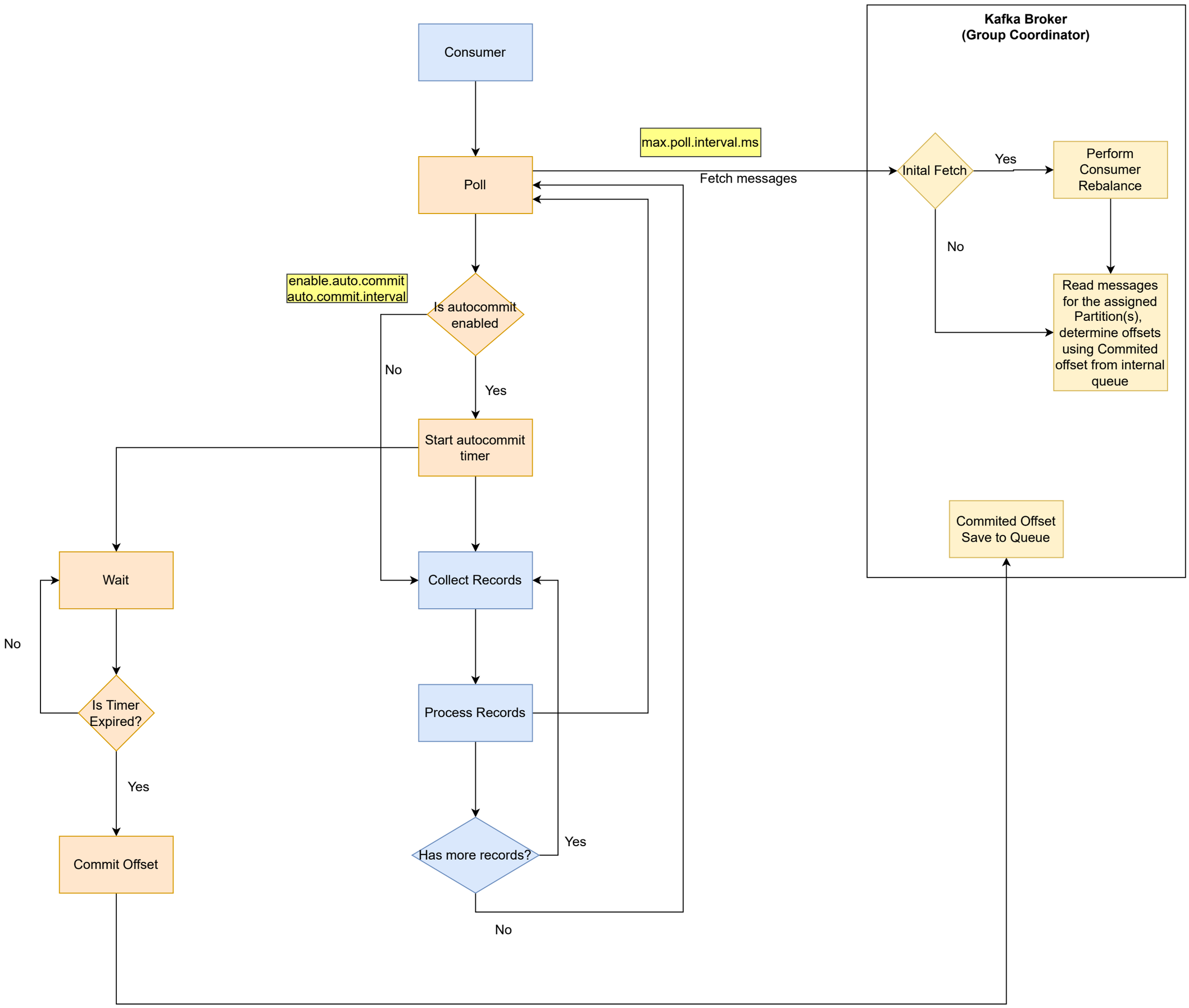
Consolidated Flow



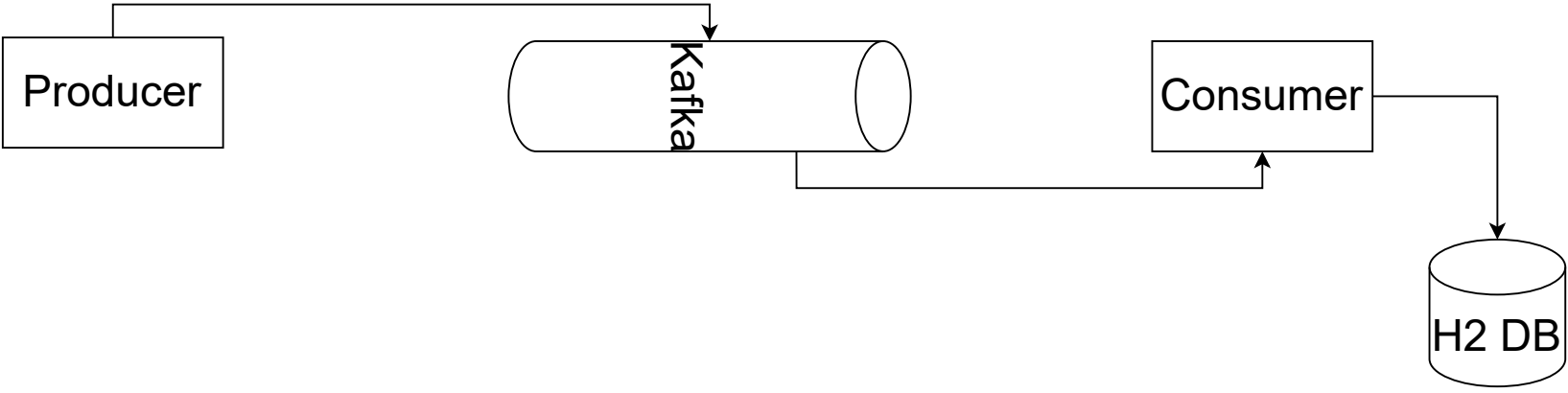
Consumer Pooling

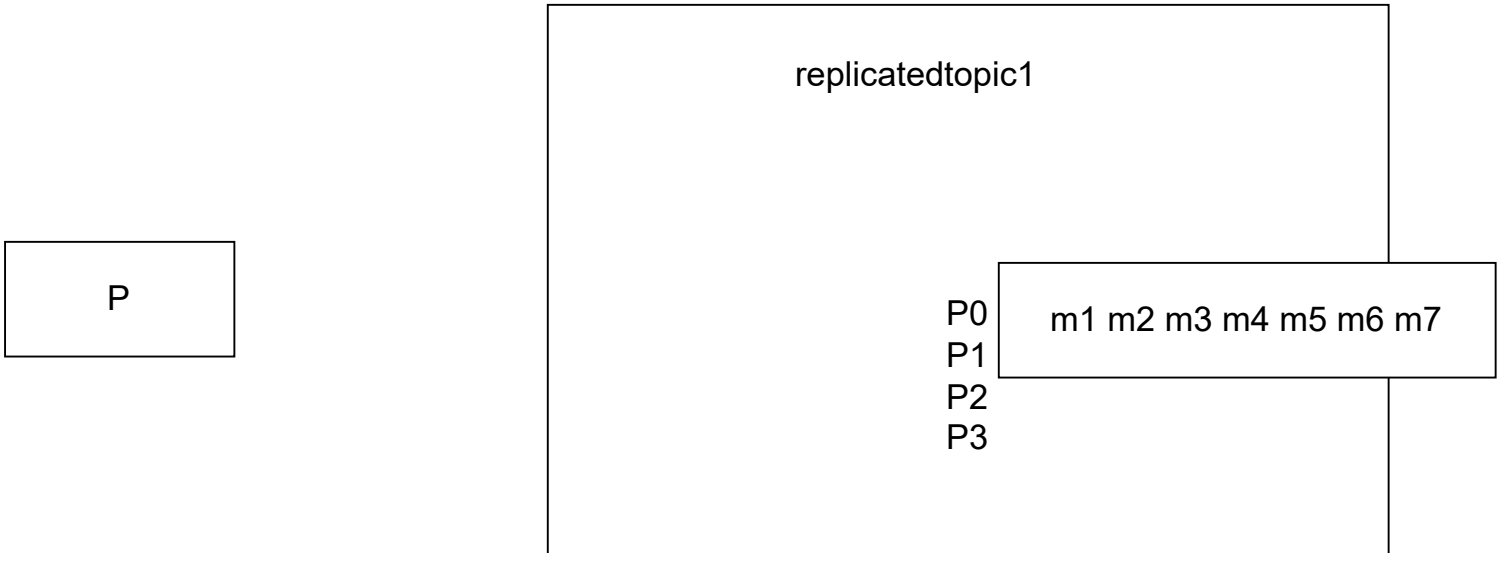


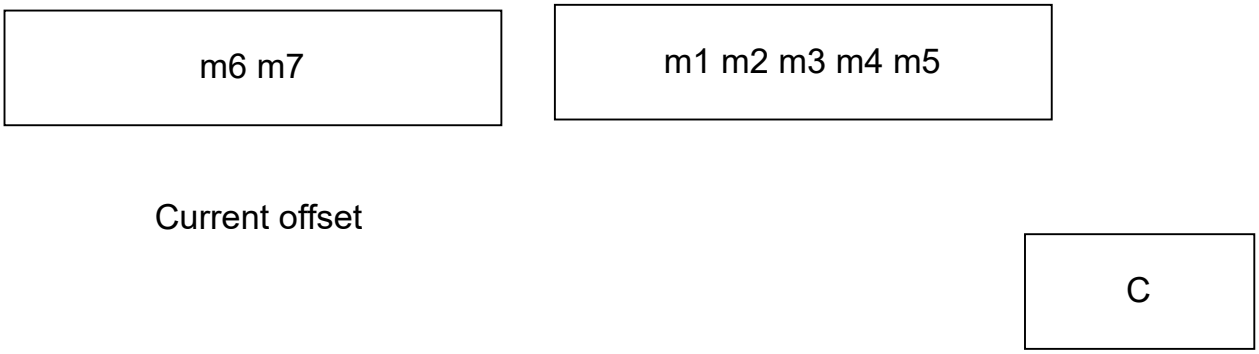
Consumer Internal Working

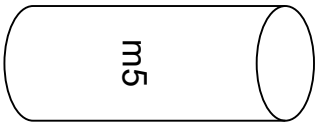


Persistency of read messages




replicatedtopic1
P
P0
P1
P2
P3
m1 m2 m3 m4 m5 m6 m7


m6 m7
m1 m2 m3 m4 m5
Current offset
C



Commited offset

Config Kafkaconsu

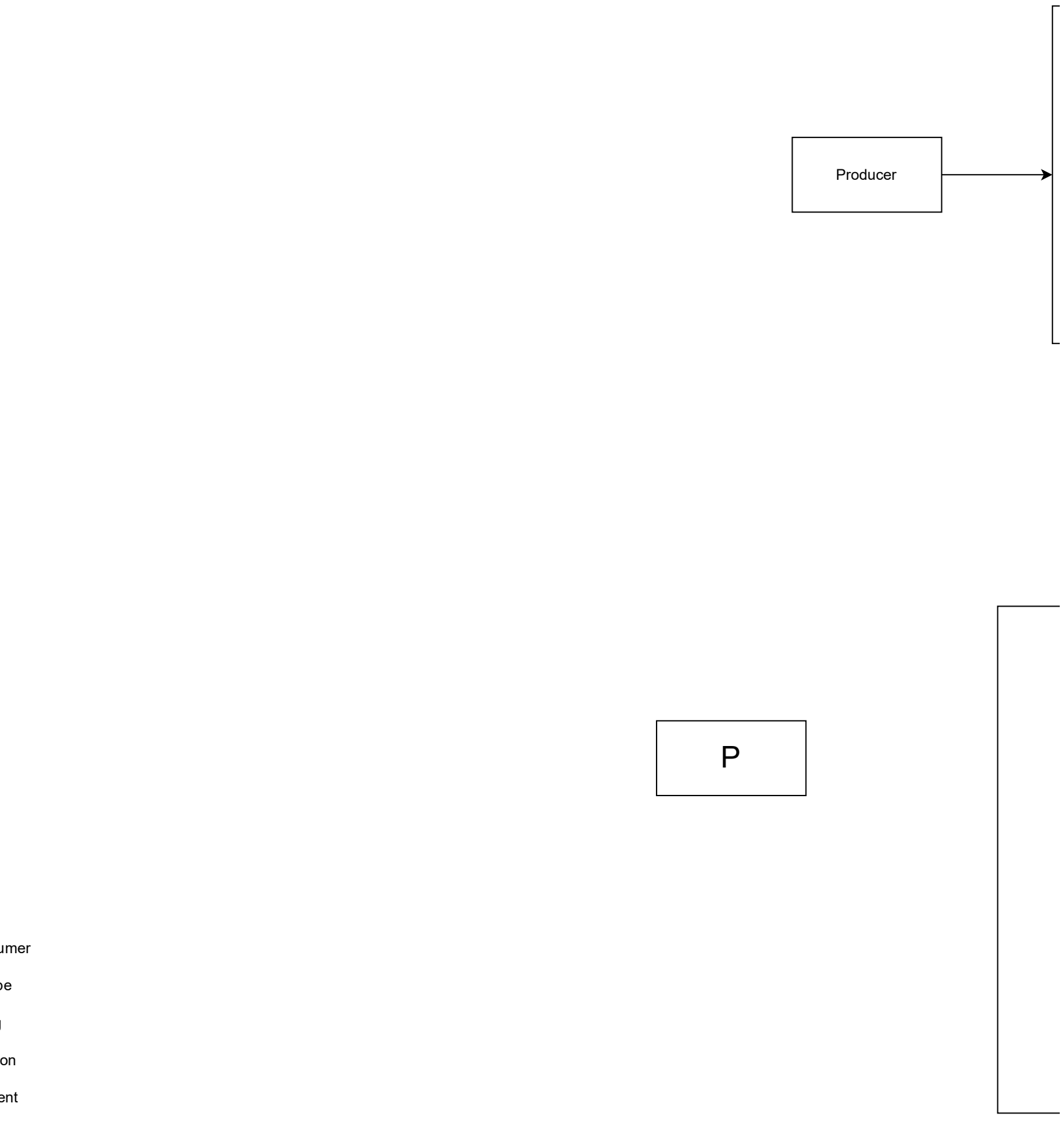
Subscrib

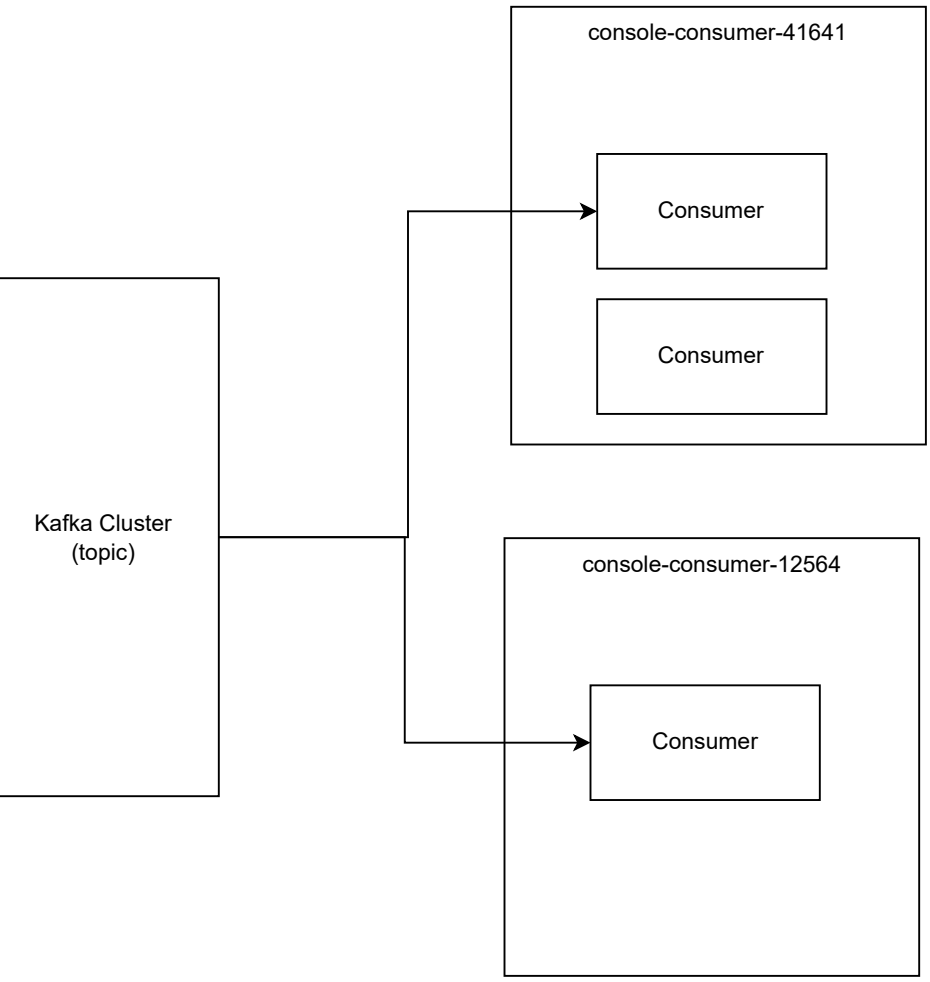
Polling

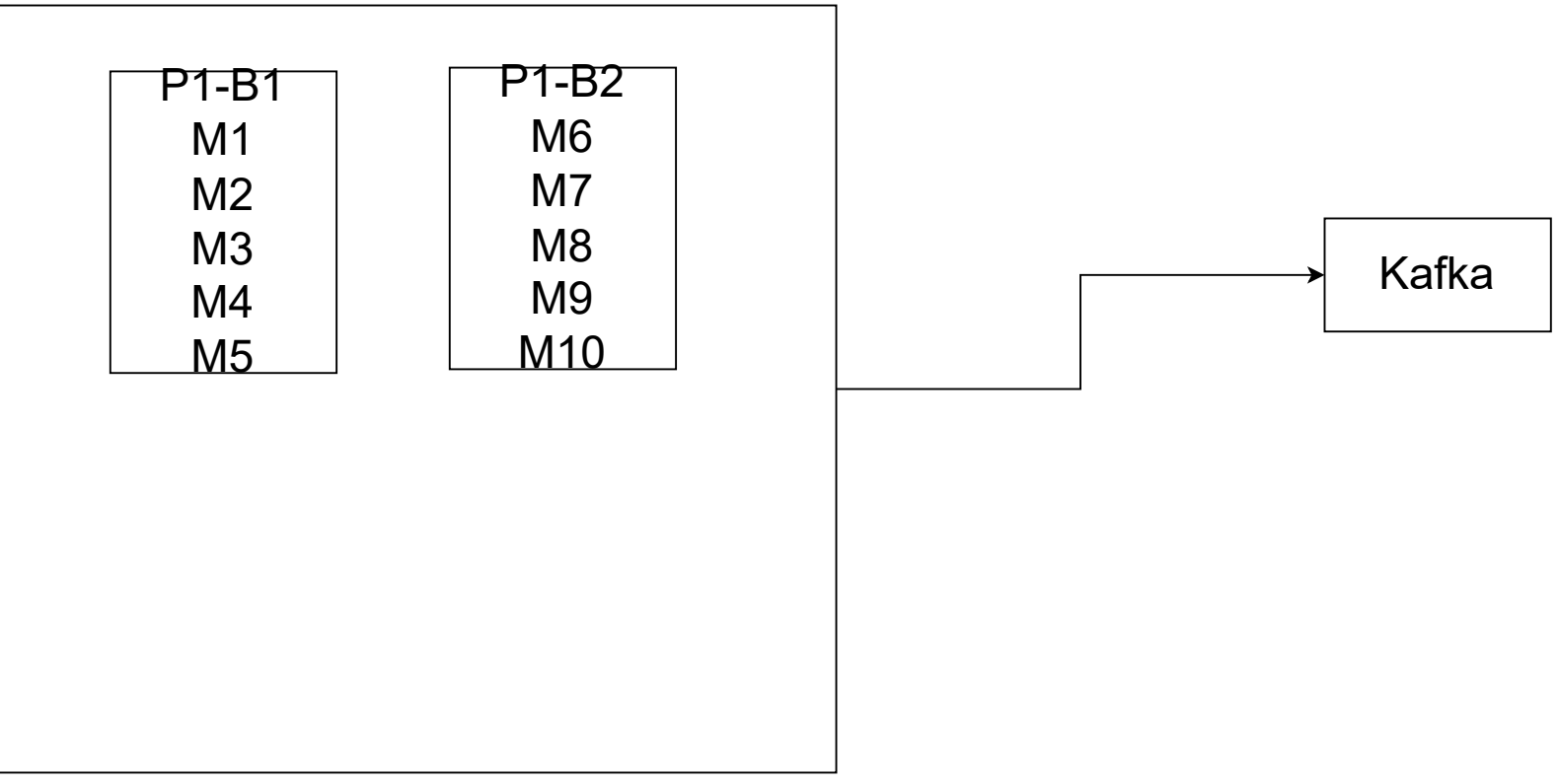
Partitions

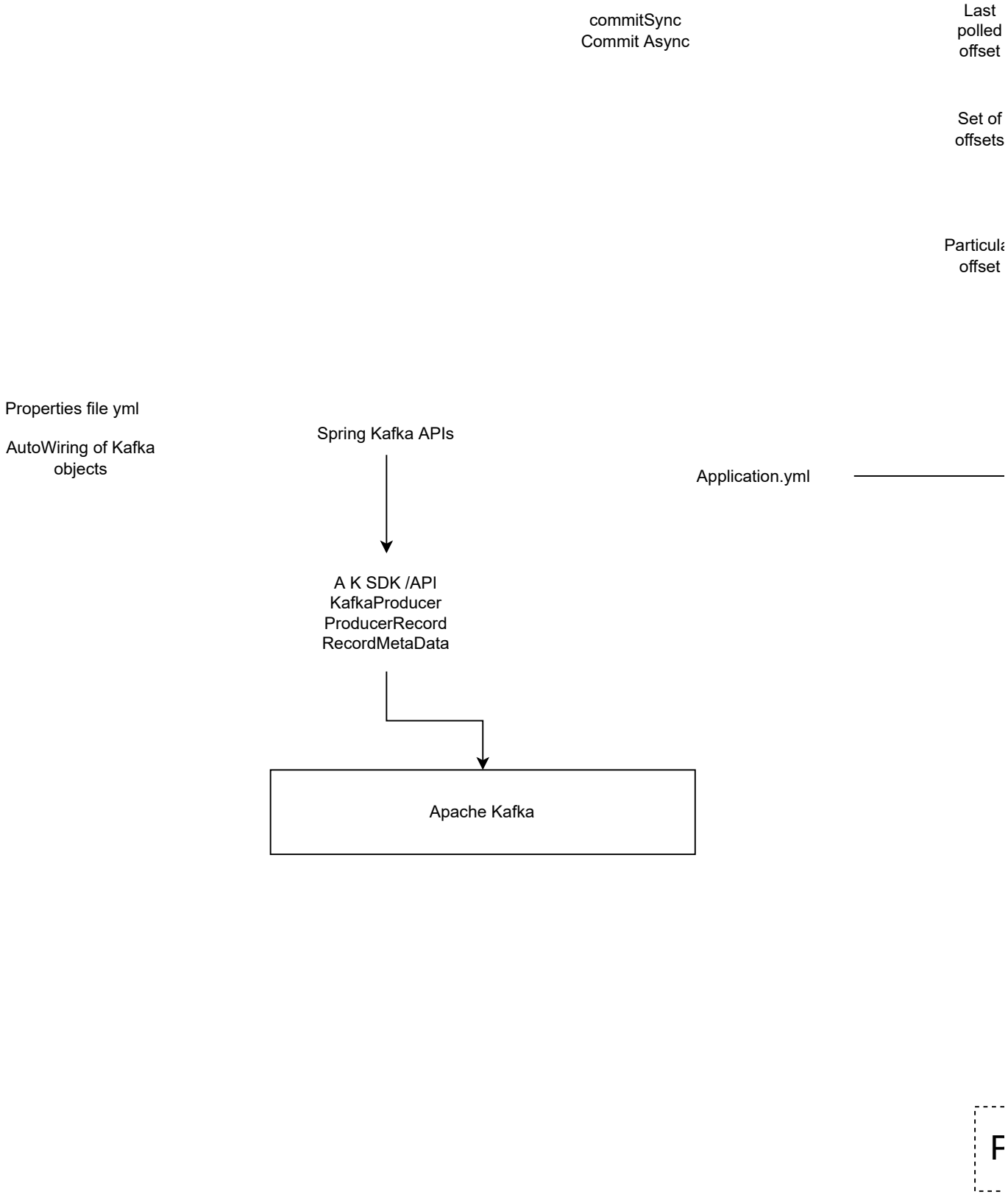
Revocati

Assignme

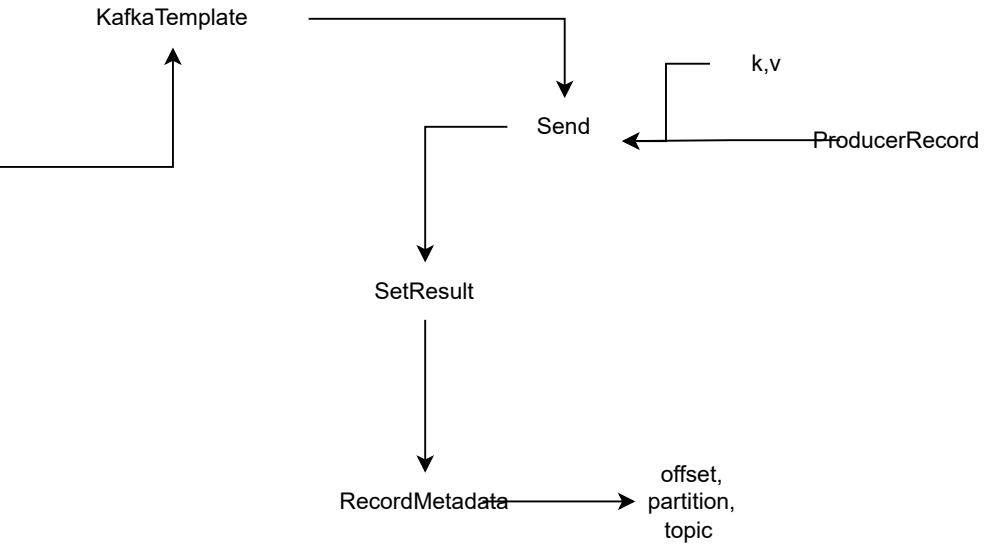

Producer
P
umer
pe
on
ent

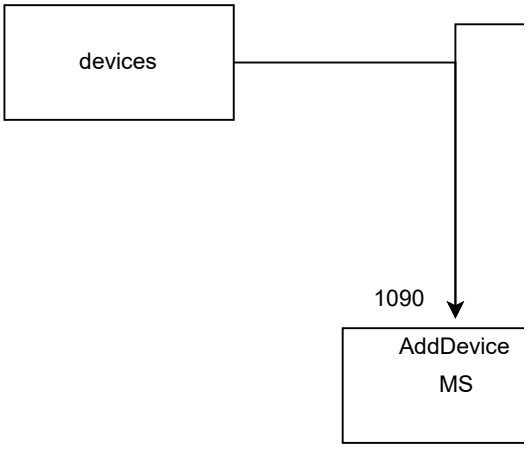

console-consumer-41641
Consumer
Consumer
Kafka Cluster
(topic)
console-consumer-12564
Consumer

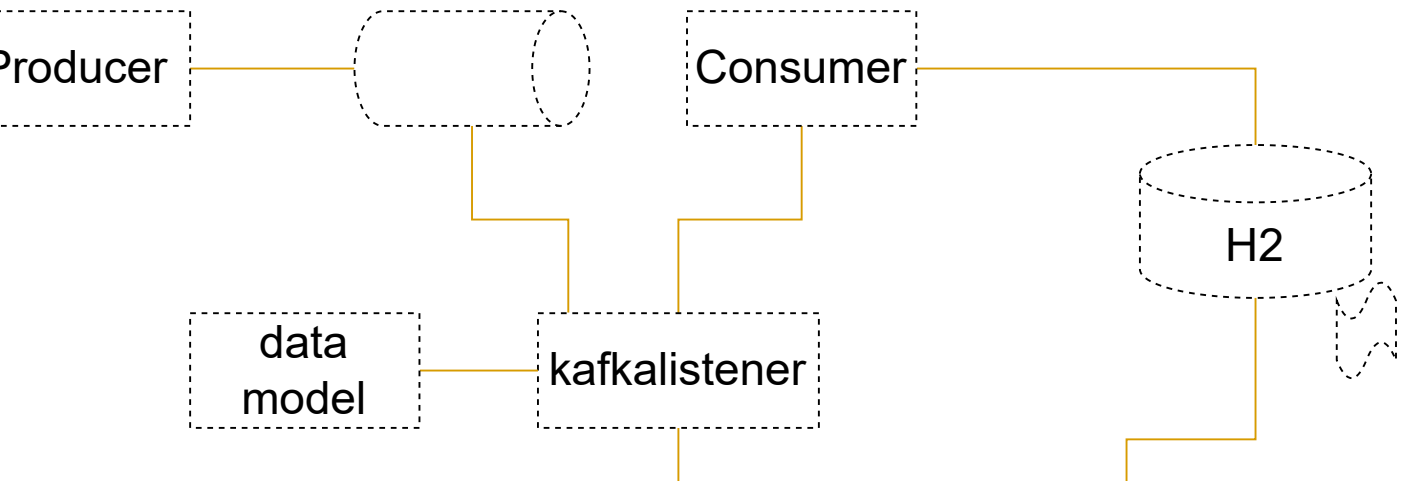

P1-B1
M1
M2
M3
M4
M5
P1-B2
M6
M7
M8
M9
M10
Kafka

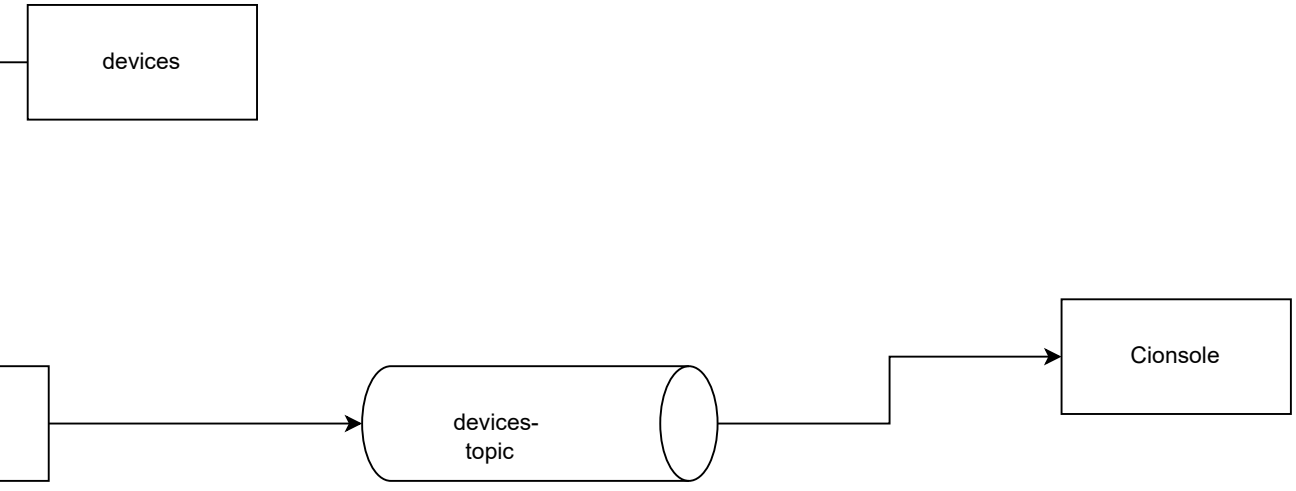

commitSync
Commit Async
Last
polled
offset
Set of
offsets
Particula
offset
Properties file yml
AutoWiring of Kafka
objects
Spring Kafka APIs
Application.yml
A K SDK /API
KafkaProducer
ProducerRecord
RecordMetaData
Apache Kafka
F

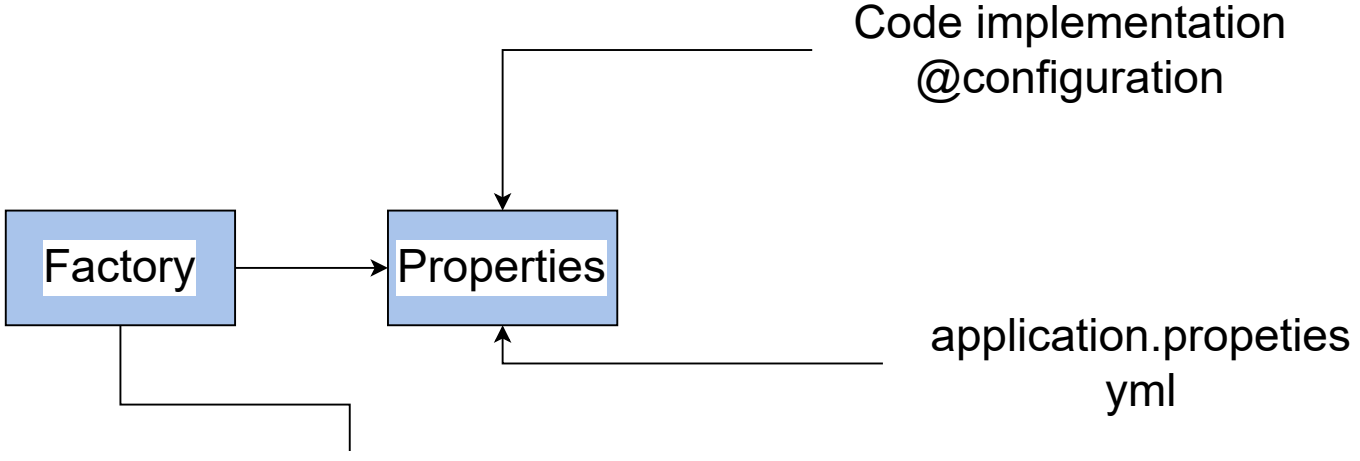
ar


KafkaTemplate
k,v
Send
ProducerRecord
SetResult
RecordMetadata
offset,
partition,
topic


devices
1090
AddDevice
MS


Producer
Consumer
H2
data
model
kafkalistener


devices
devices-
topic
Cionsole


Code implementation
@configuration
Factory
Properties
application.propeties
yml

me

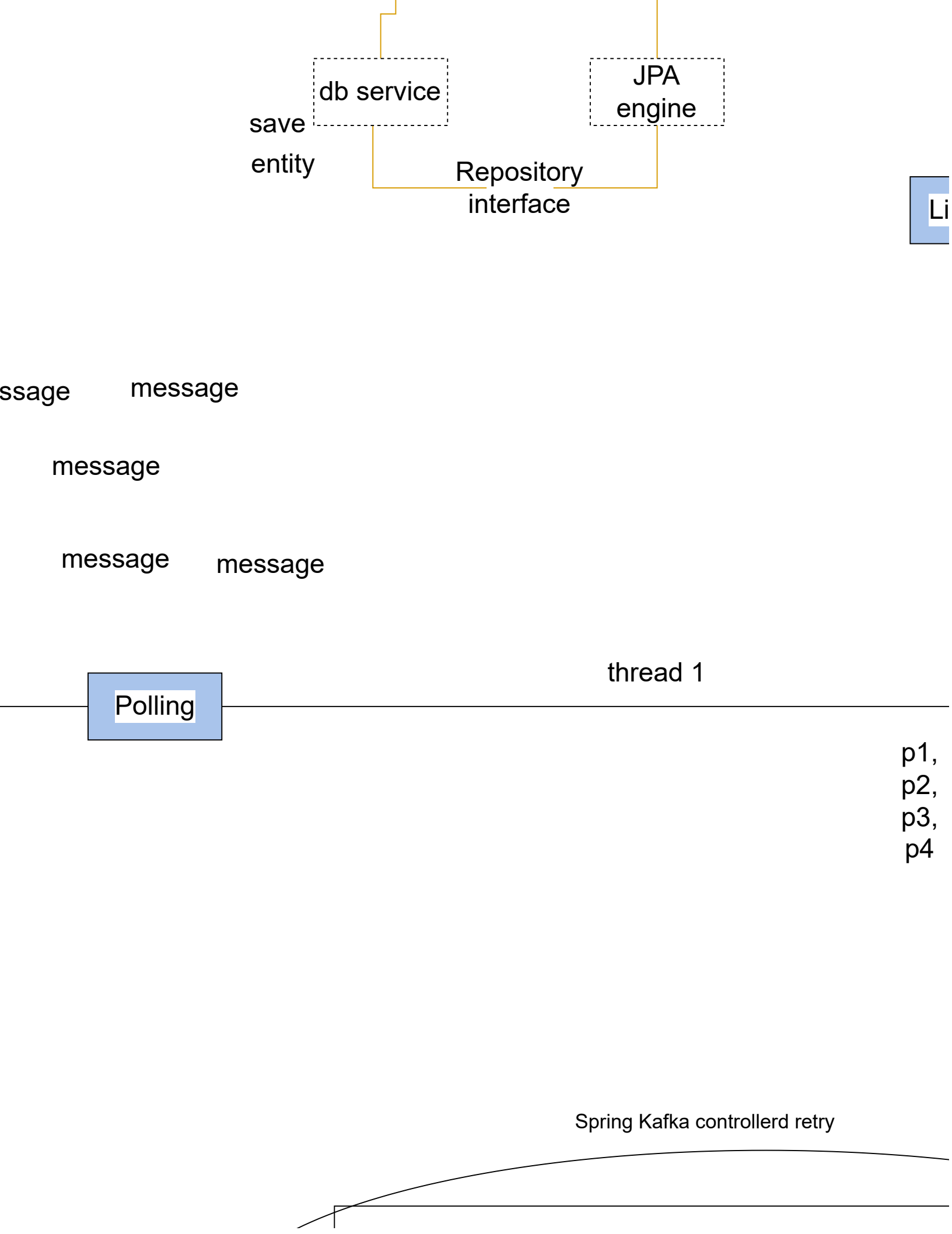
message

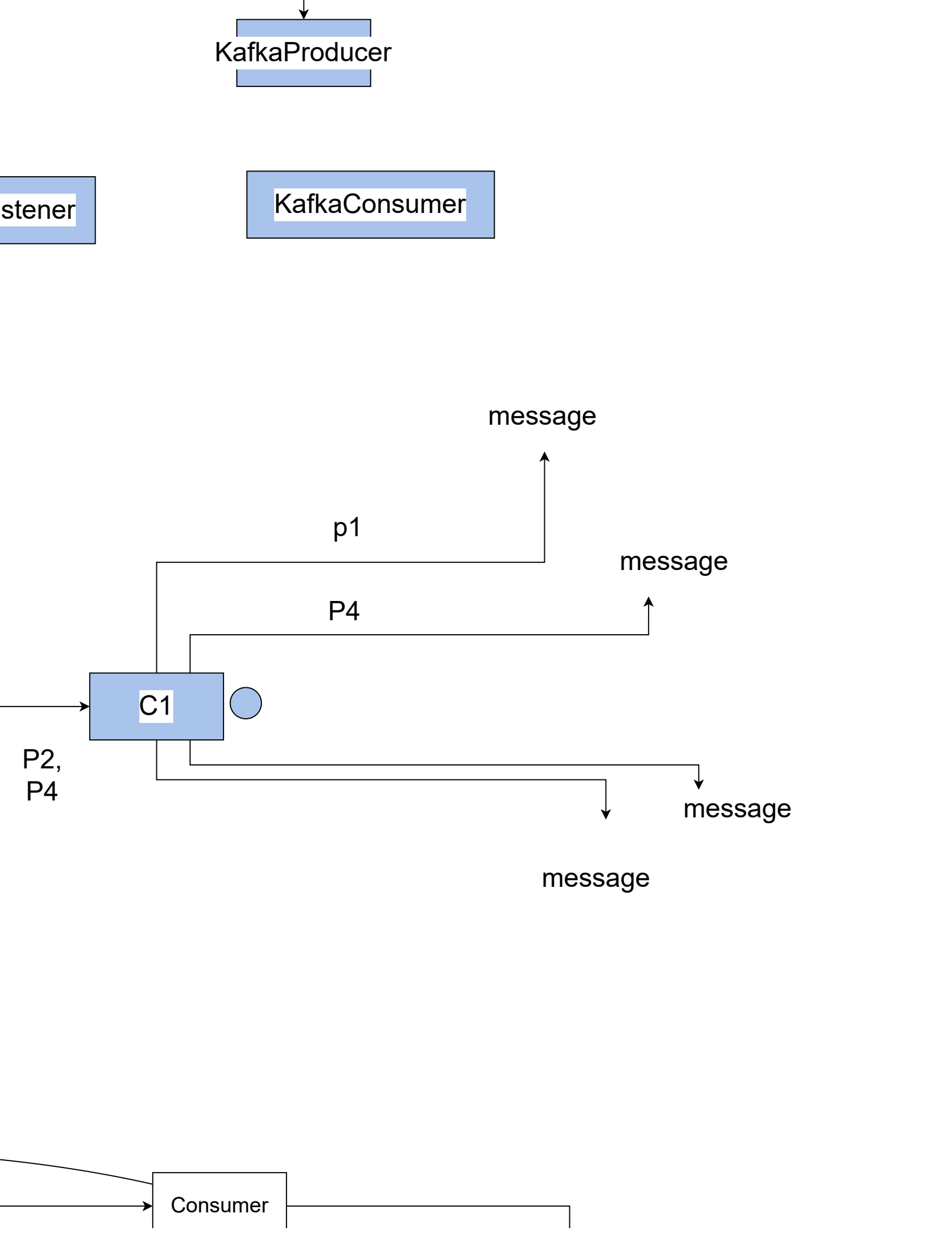
Queue

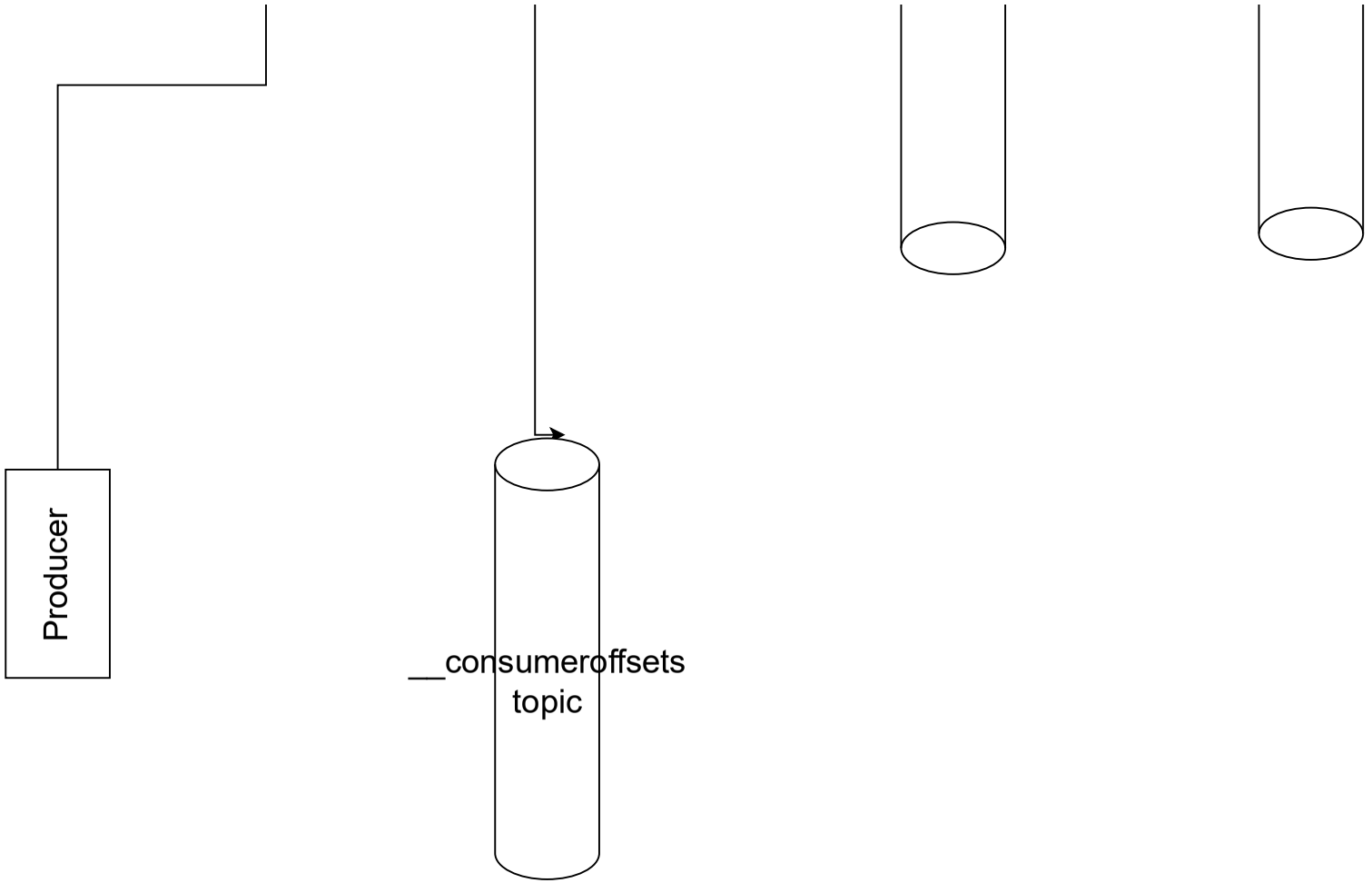
Partion

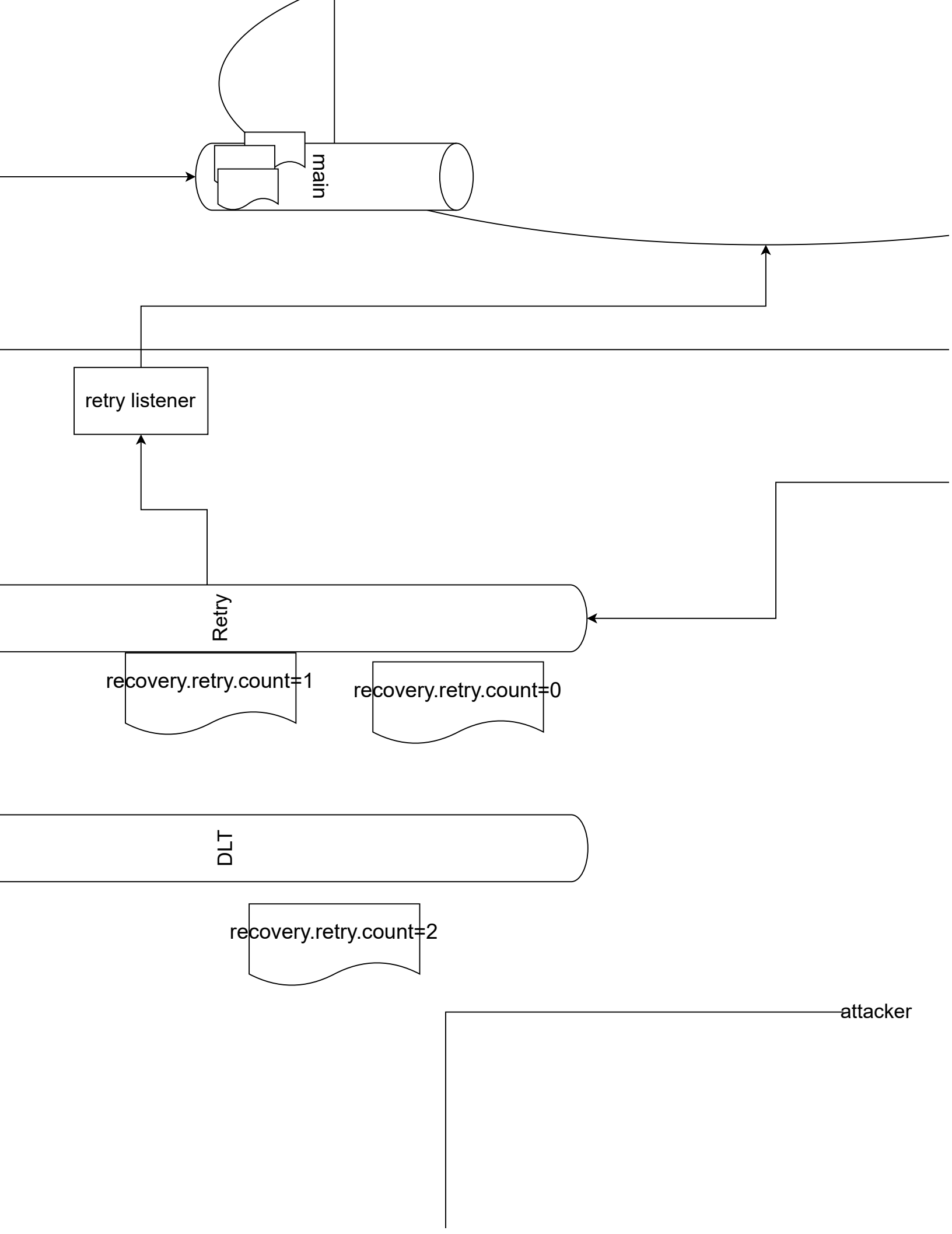
Partion

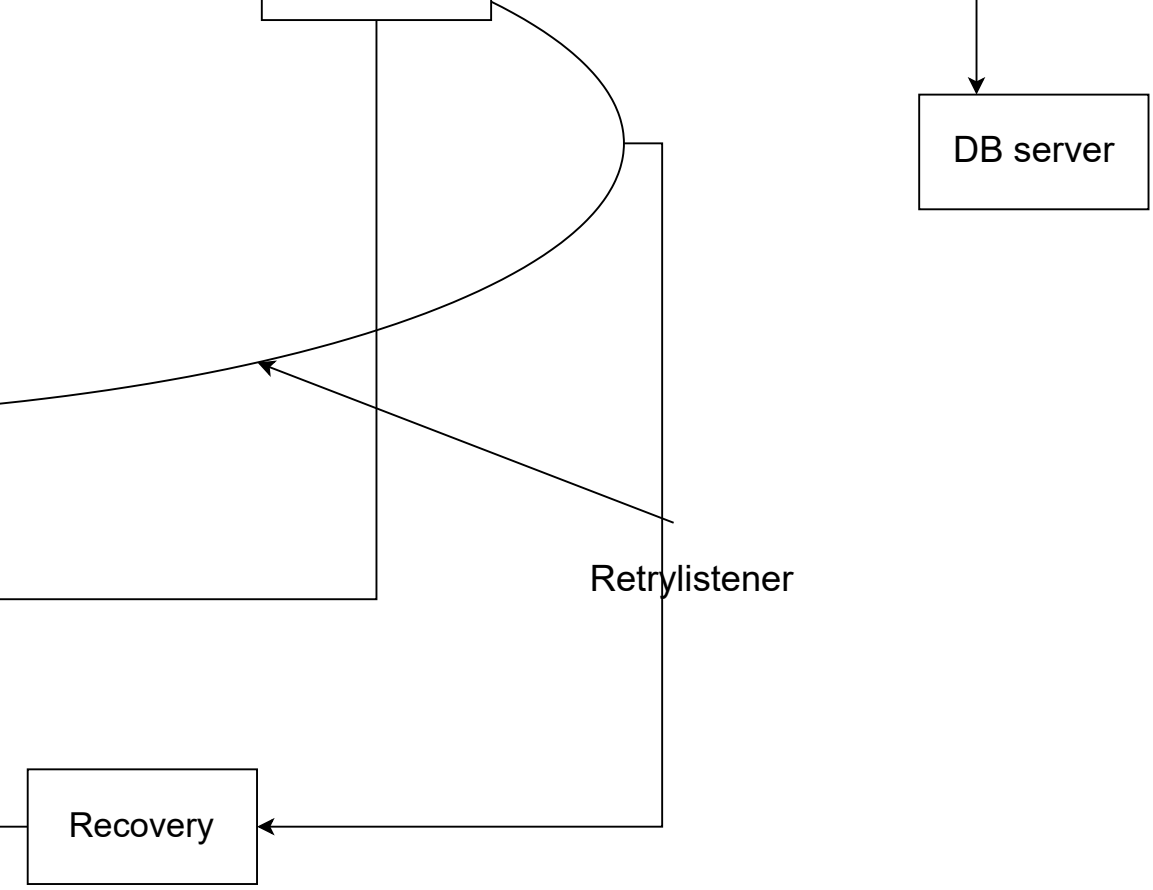
Partion

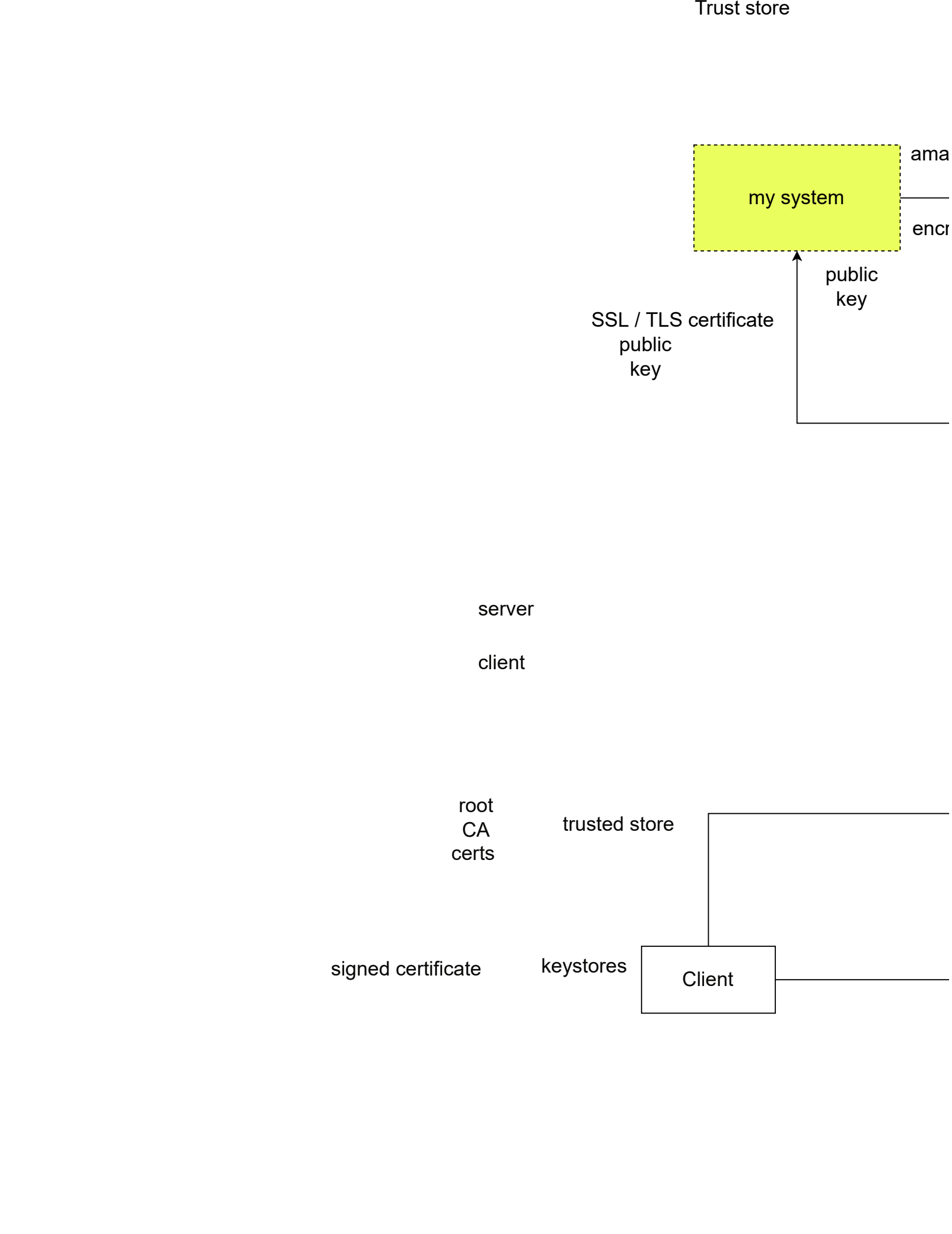

db service
JPA engine
save
entity
Repository interface
Li
ssage
message
message
message
message
Polling
thread 1
p1,
p2,
p3,
p4
Spring Kafka controllerd retry

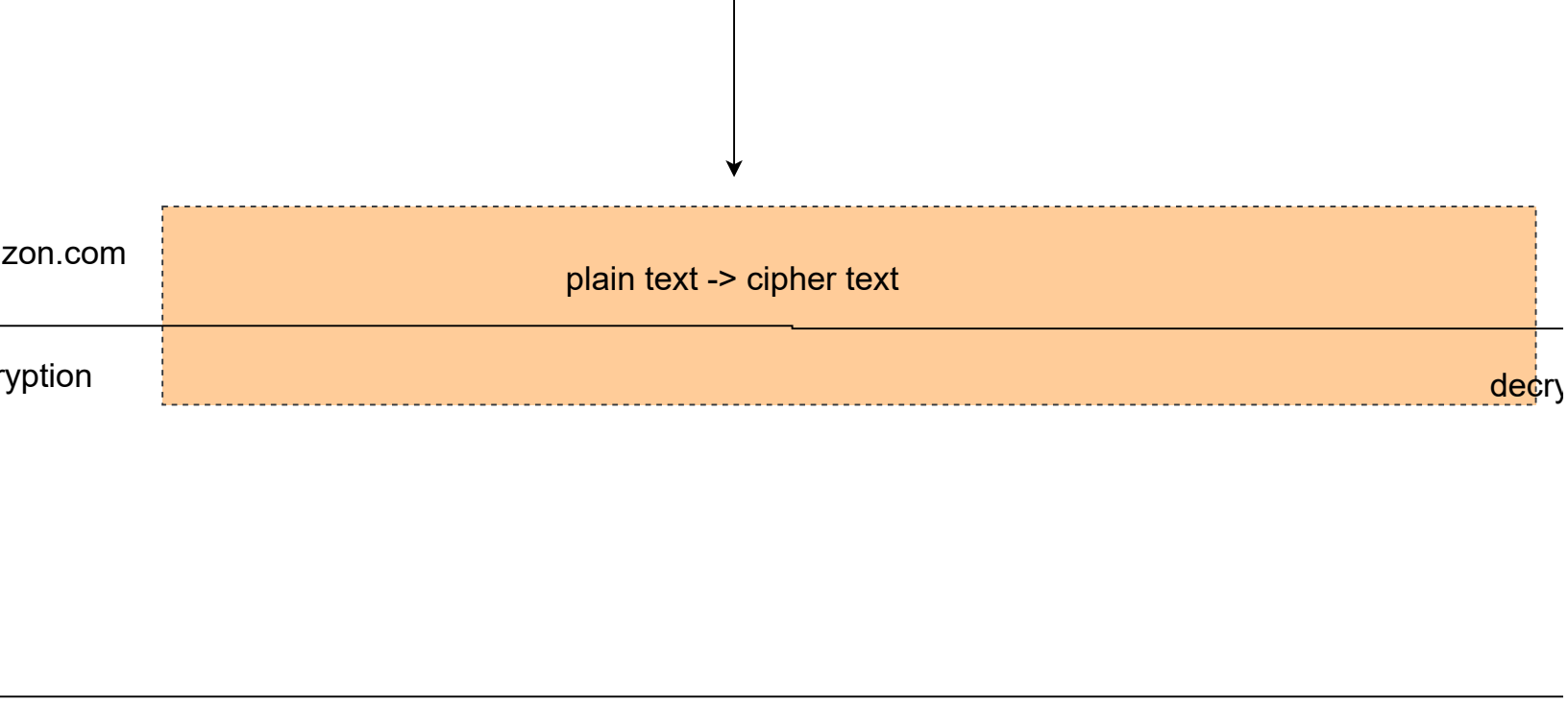

KafkaProducer
stener
KafkaConsumer
message
p1
message
P4
C1
P2,
P4
message
message
Consumer


Producer
__consumeroffsets
topic

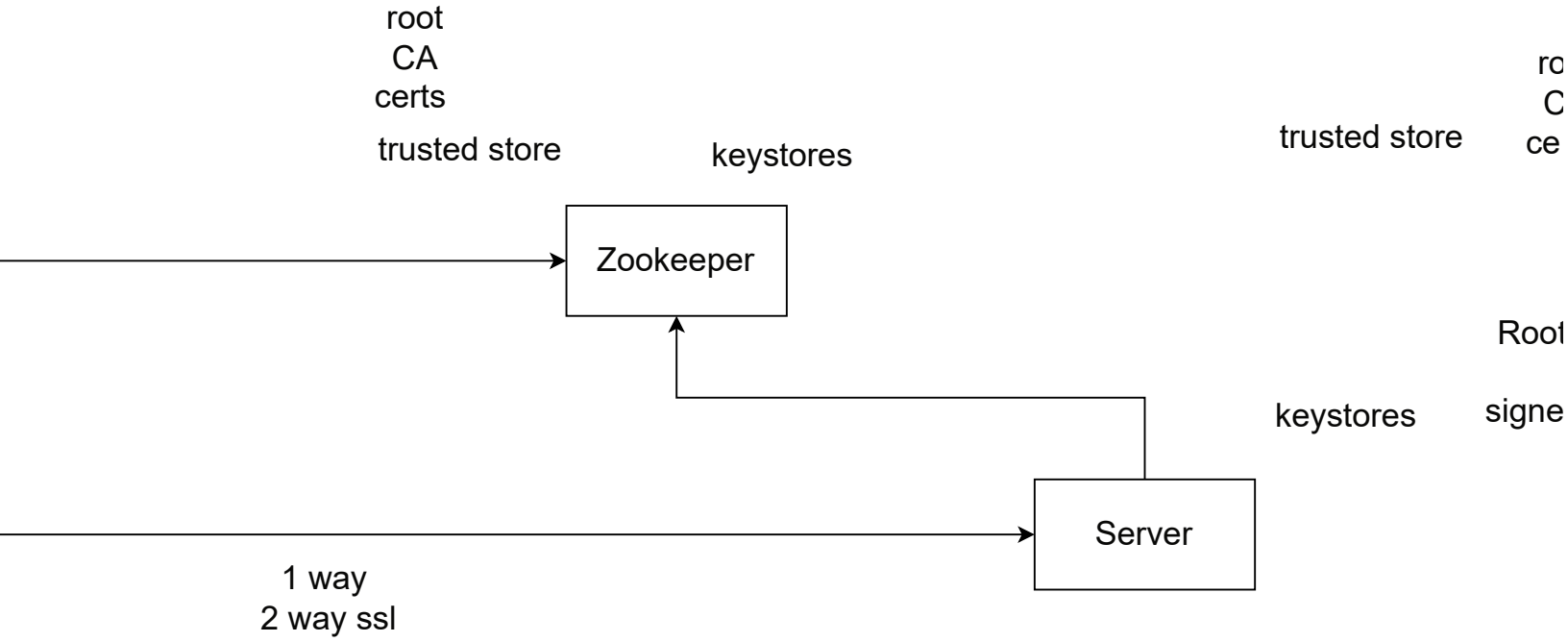

main
retry listener
Retry
recovery.retry.count=1
recovery.retry.count=0
DLT
recovery.retry.count=2
attacker

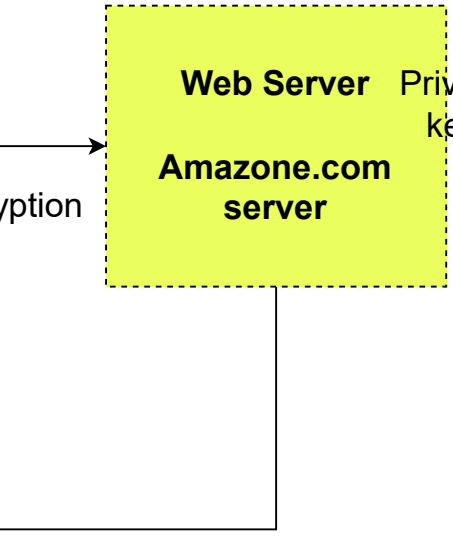

DB server
Retrylistener
Recovery


Trust store
my system
ama
encr
public
key
SSL / TLS certificate
public
key
server
client
root
CA
certs
trusted store
signed certificate
keystores
Client



SSL rough work area





oot
CA
rts

SSL
- Encryption/ Decruption

- Authentication

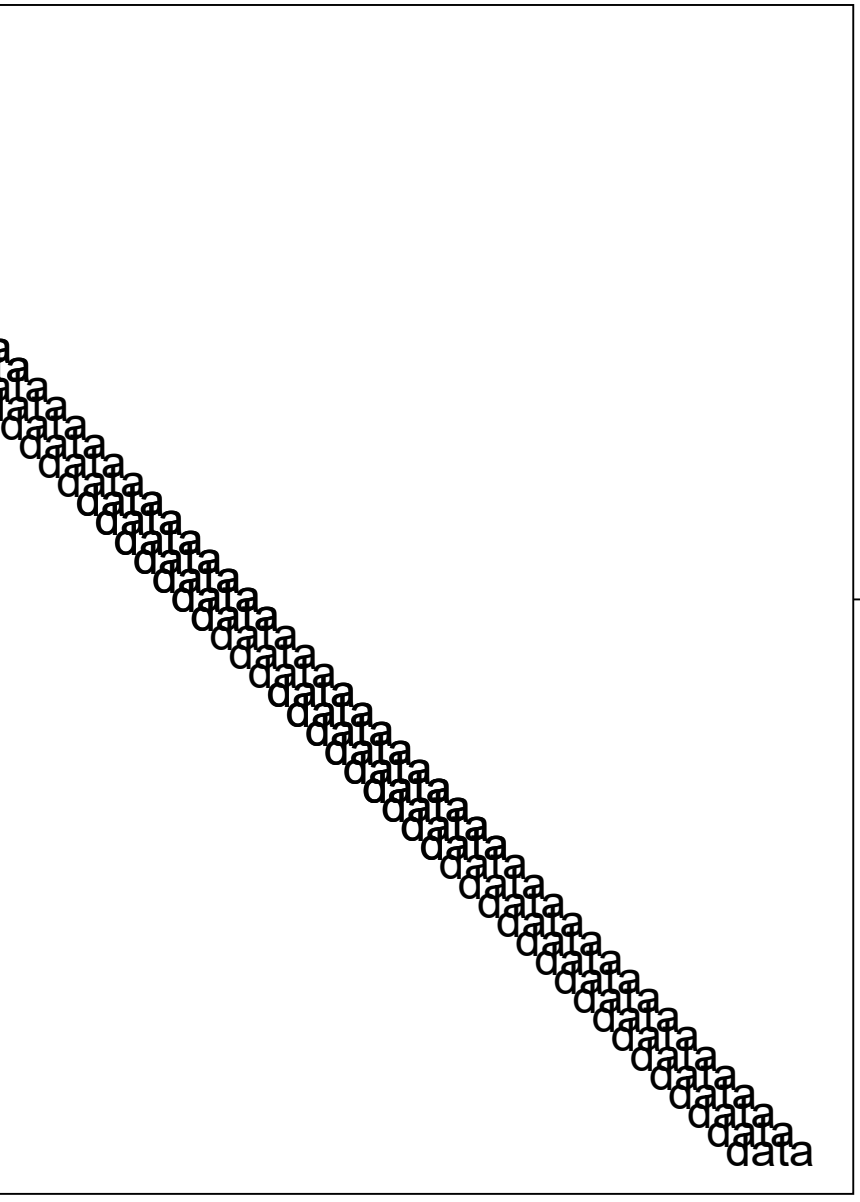
SSL certificates

t CA cert

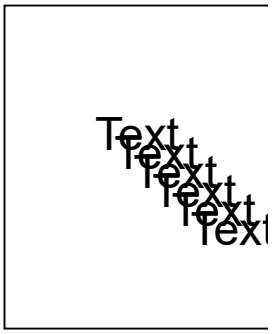
d certificate

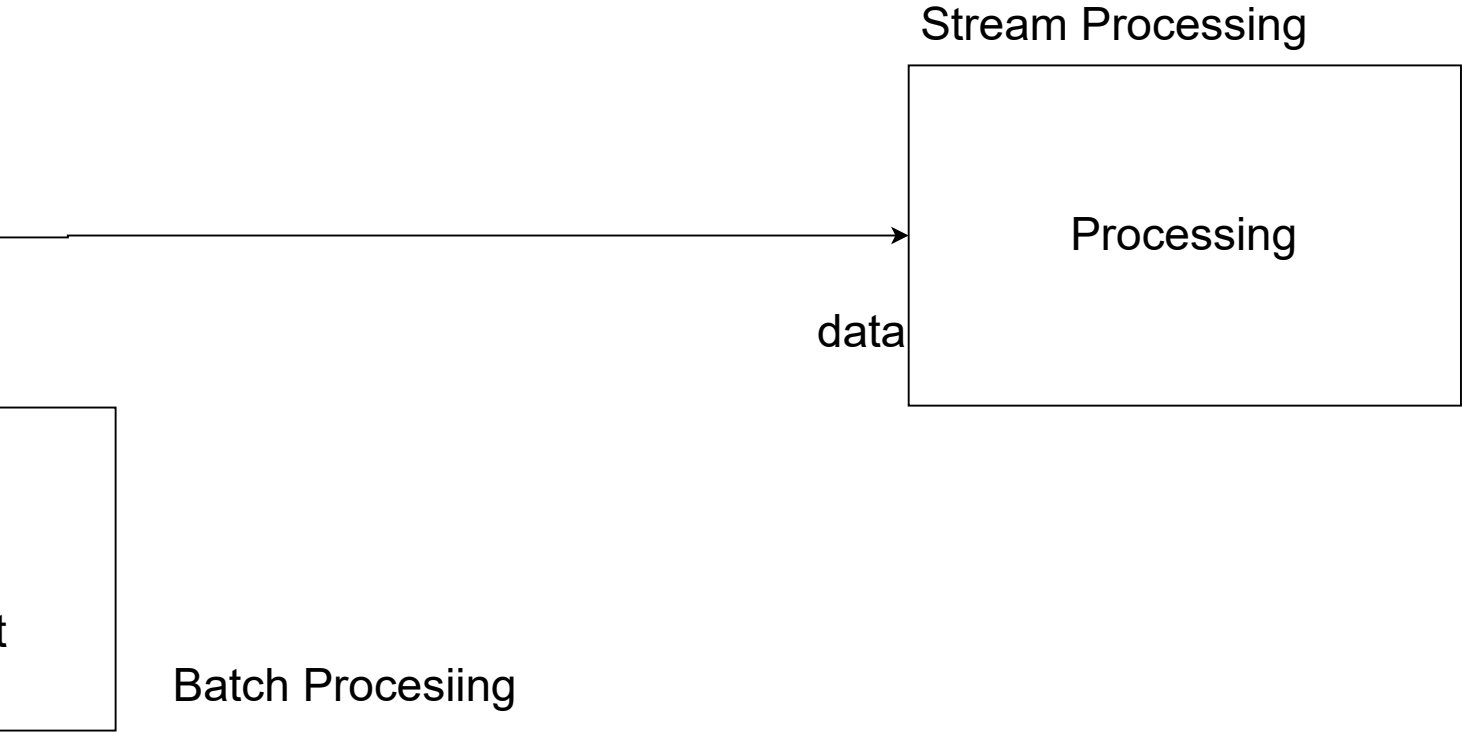
CA

data

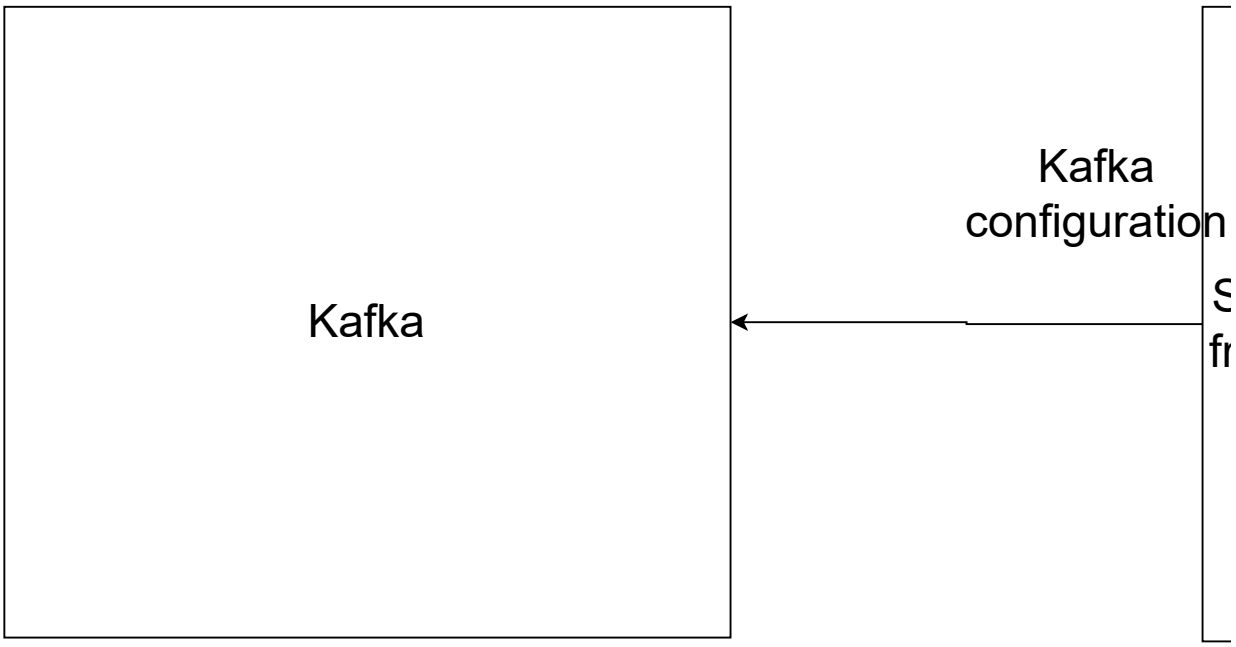


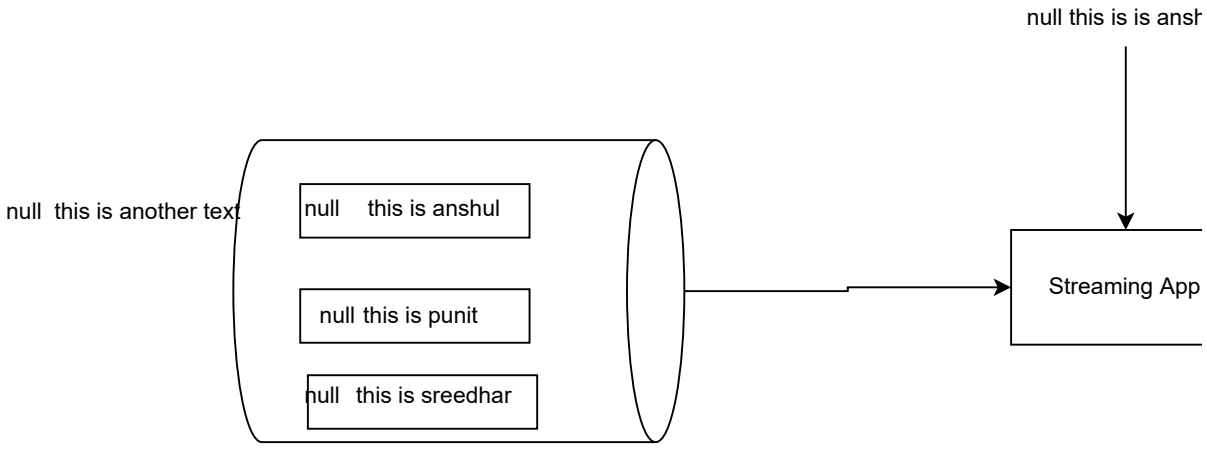
StrreamBuilder

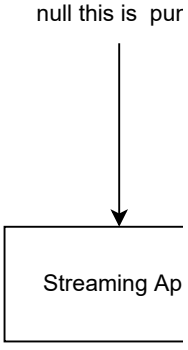




Topology


Kafka
Kafka
configuration

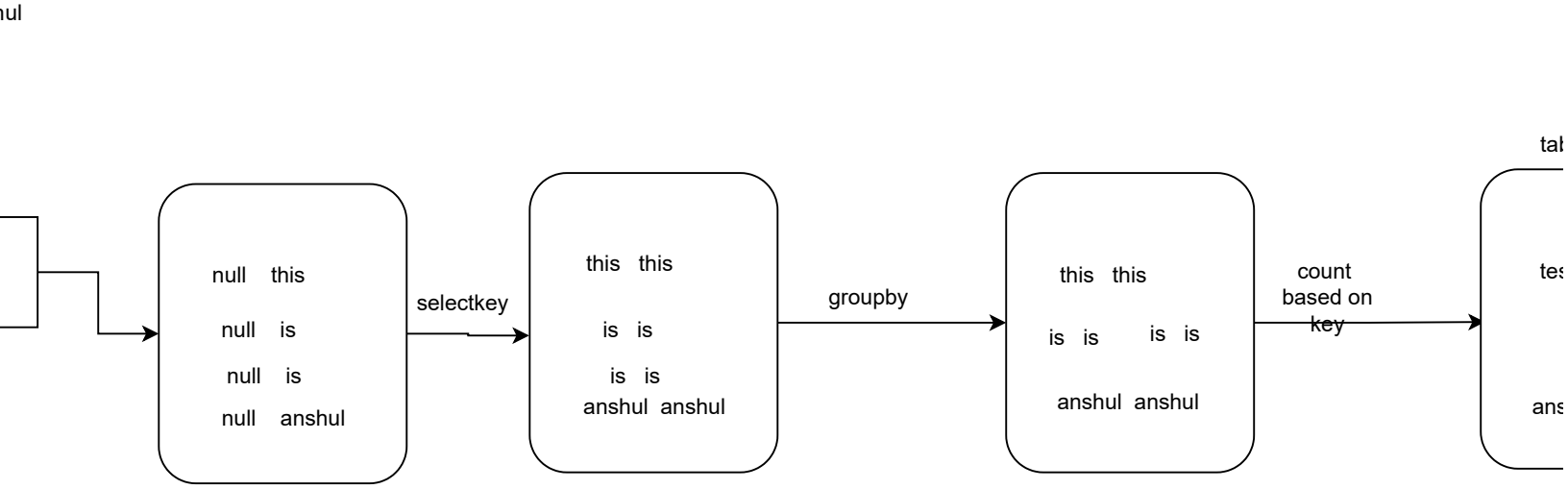

null this is is ansh
null this is another text
null this is anshul
null this is punit
null this is sreedhar
Streaming App

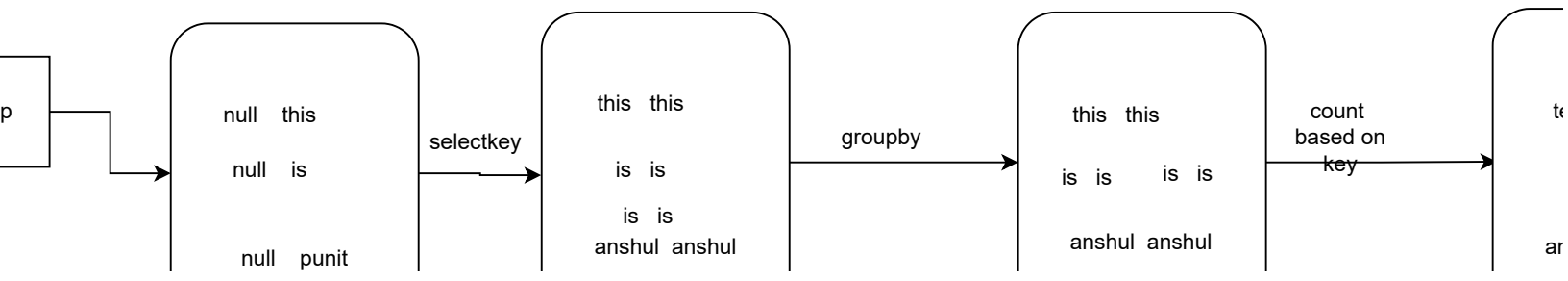

null this is pur
Streaming Ap

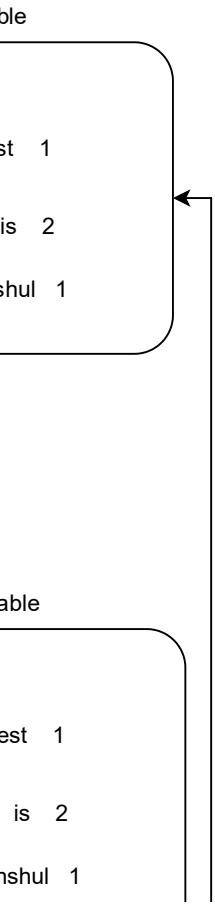
Topology

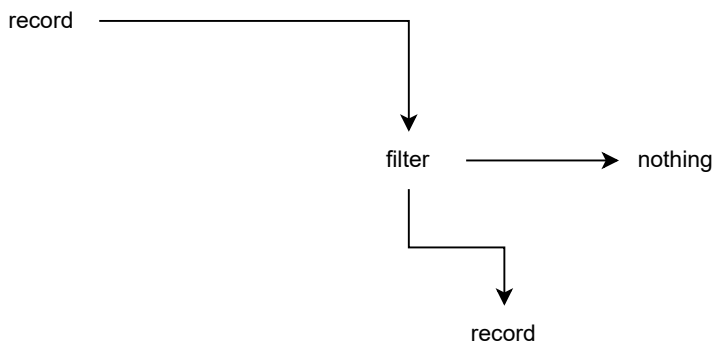
Processors
(Source, Sink, processor)

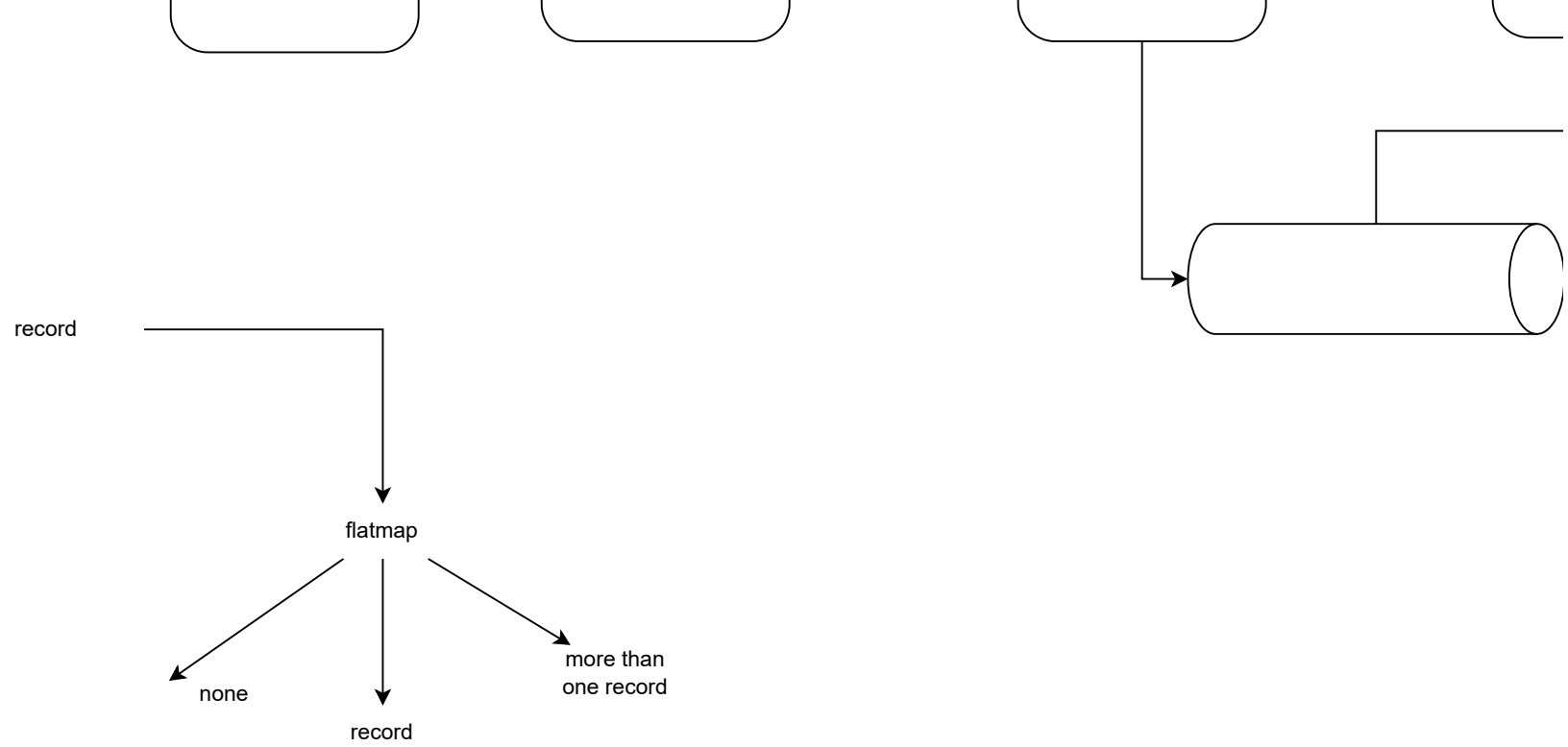
Streaming
ramework

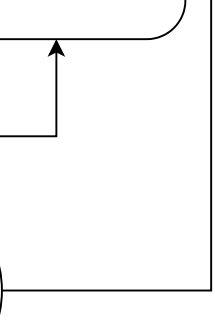




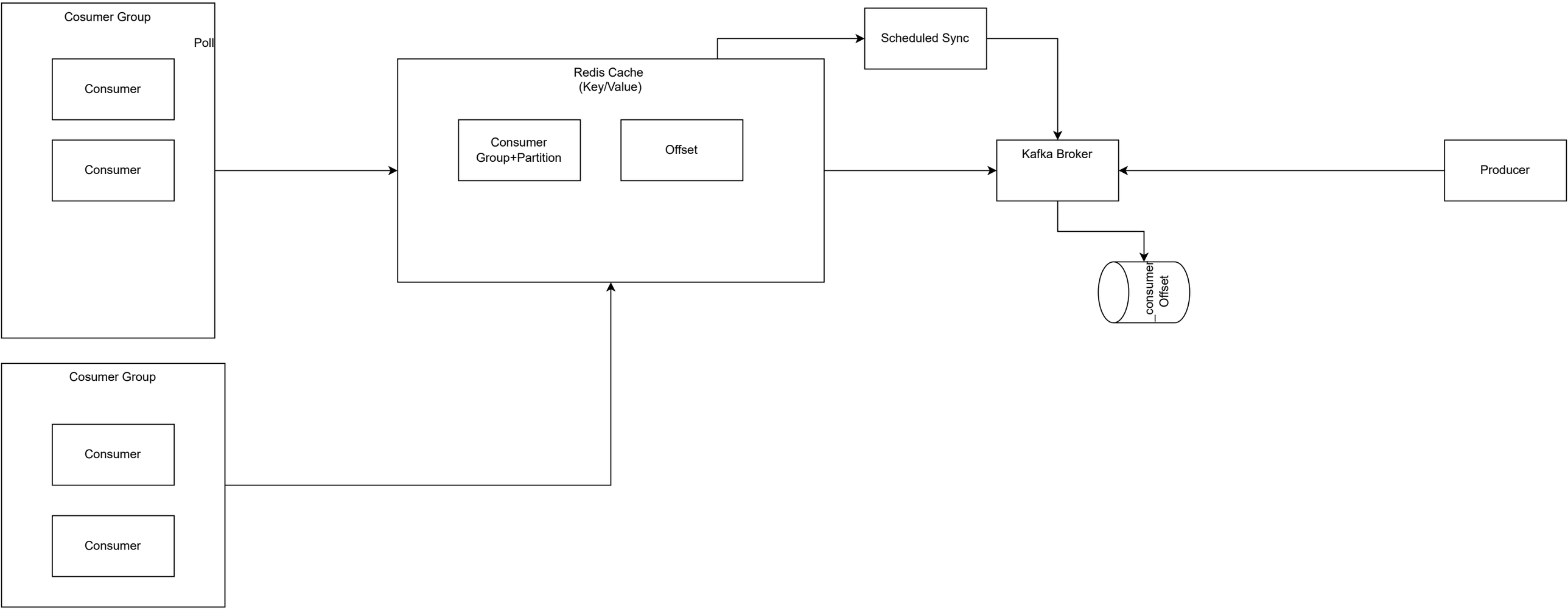

ble
st 1
is 2
shul 1
able
est 1
is 2
nshul 1


record
filter
nothing
record

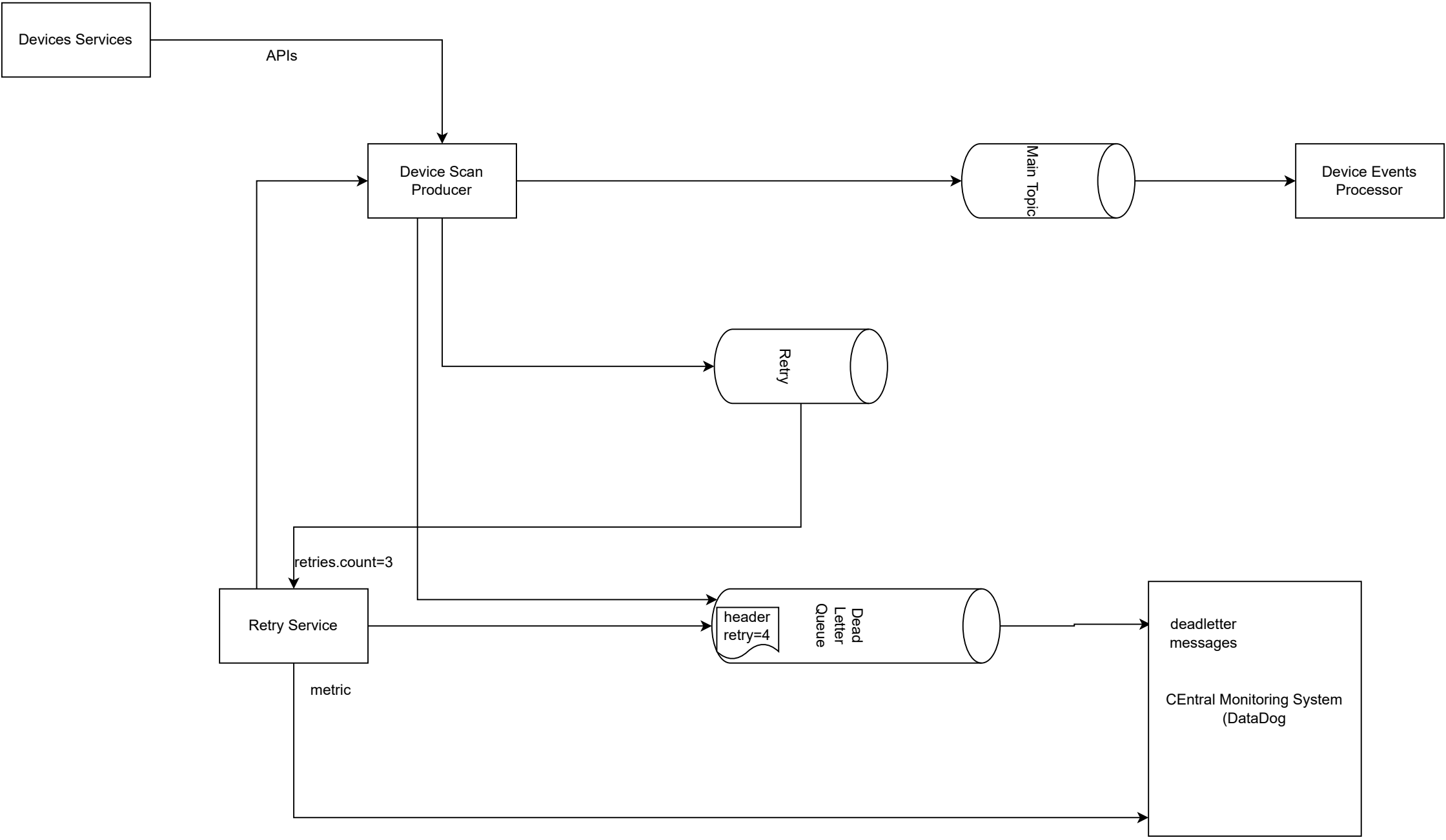

record
flatmap
none
record
more than
one record



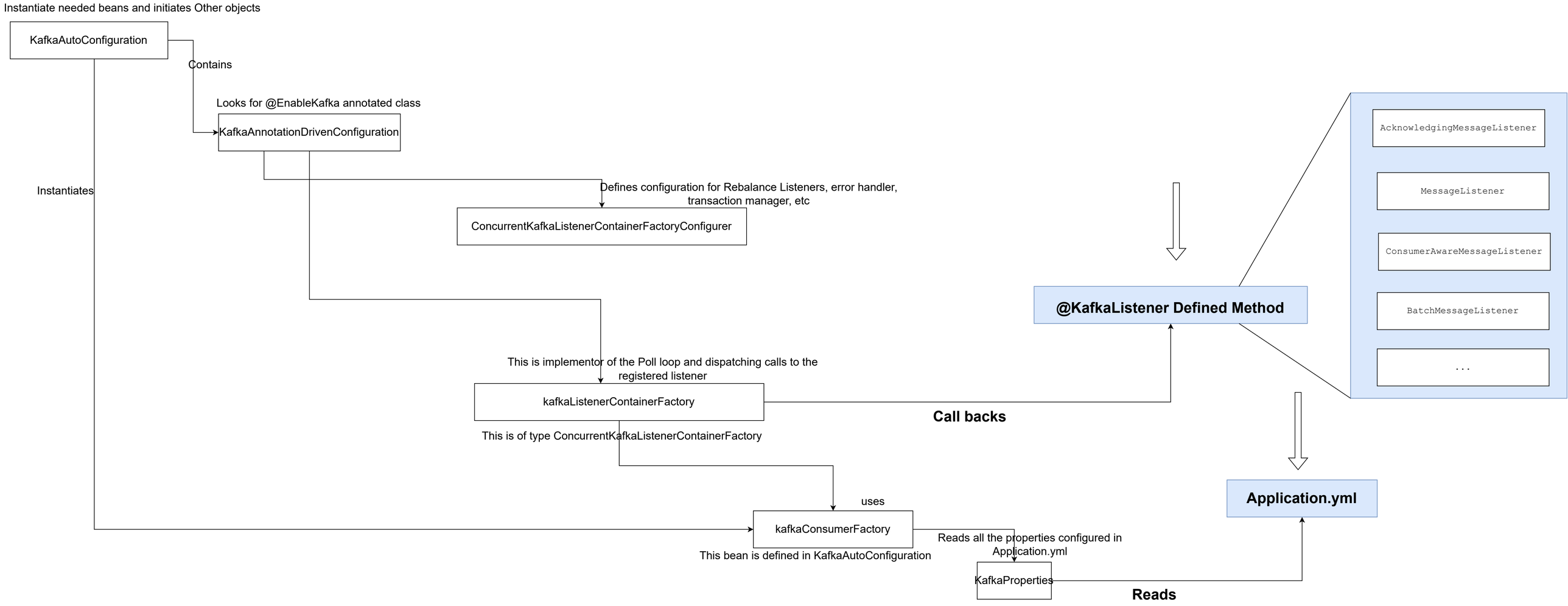
DIY Exercise - day 7

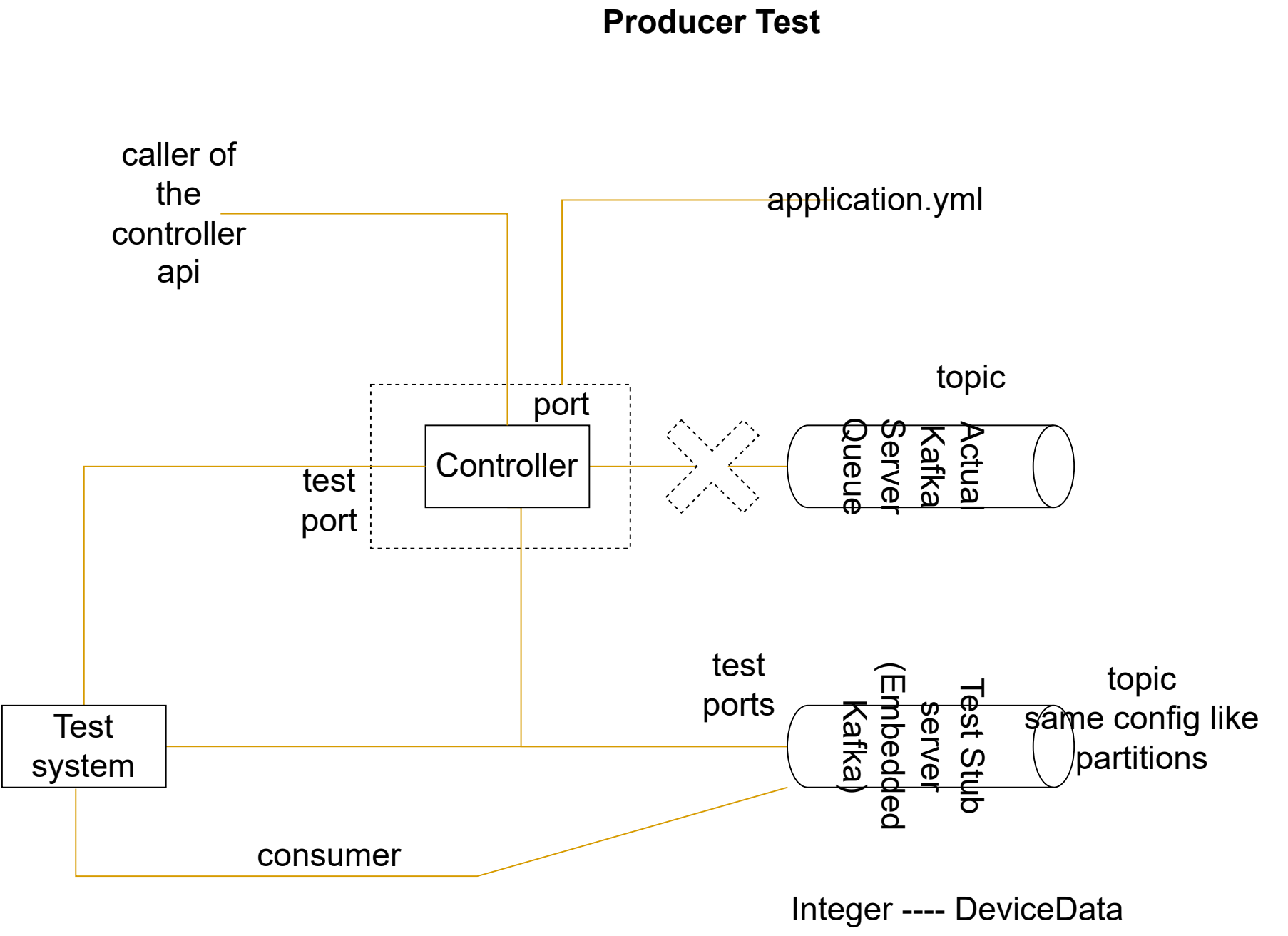


Producer Retry Mechanism



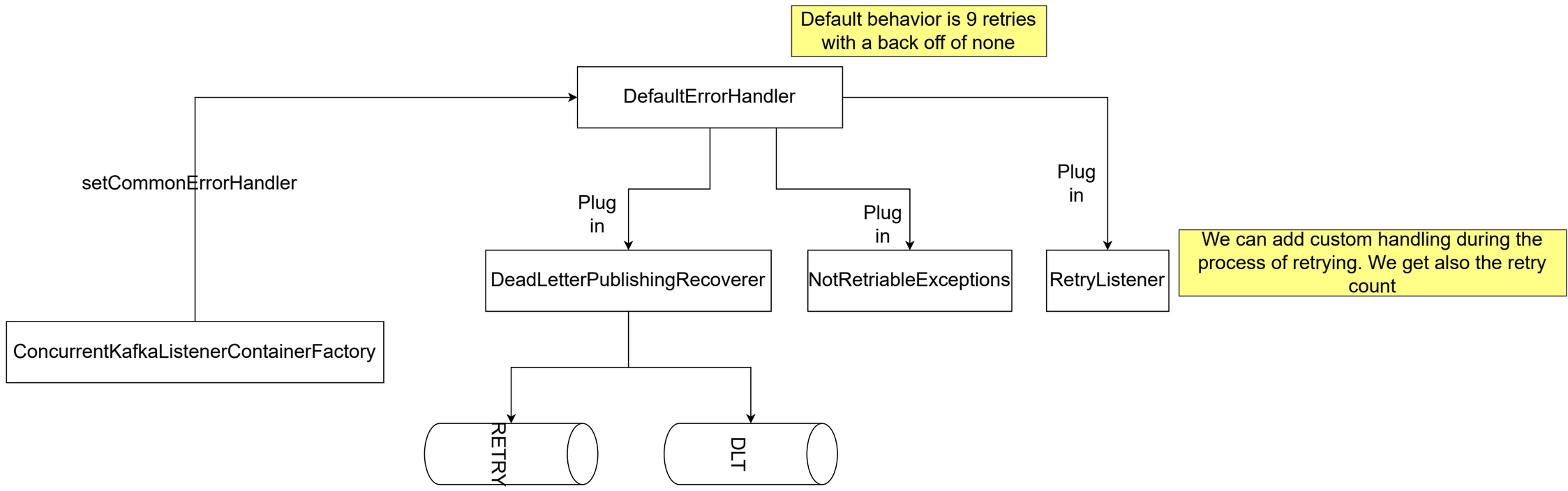
Spring Kafka Consumer Internals



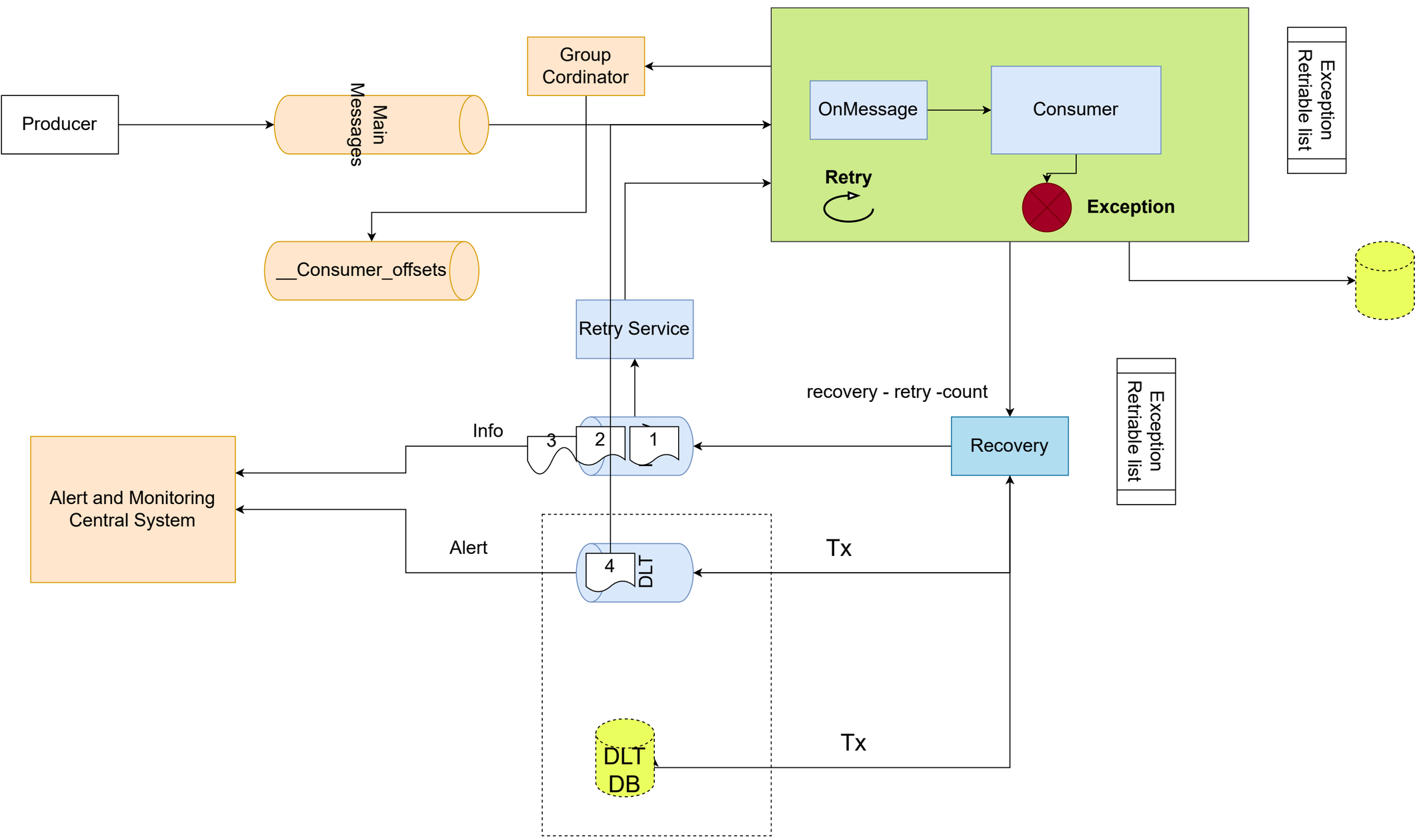

Producer Test
caller of the controller api
application.yml
port
Controller
test port
topic
Actual Kafka Server Queue
test ports
Test Stub server (Embedded Kafka)
topic same config like partitions
Test system
consumer
Integer ---- DeviceData

Consumer Error Handling & Recovery Mechanism

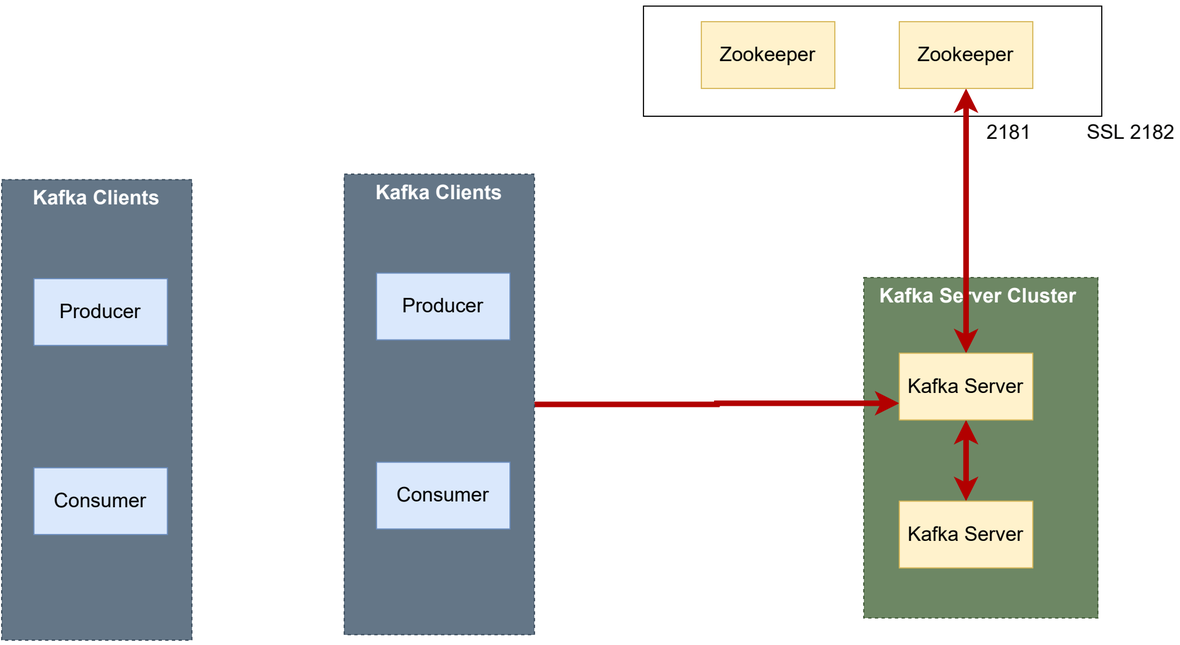
Spring Kafka Consumer Classes



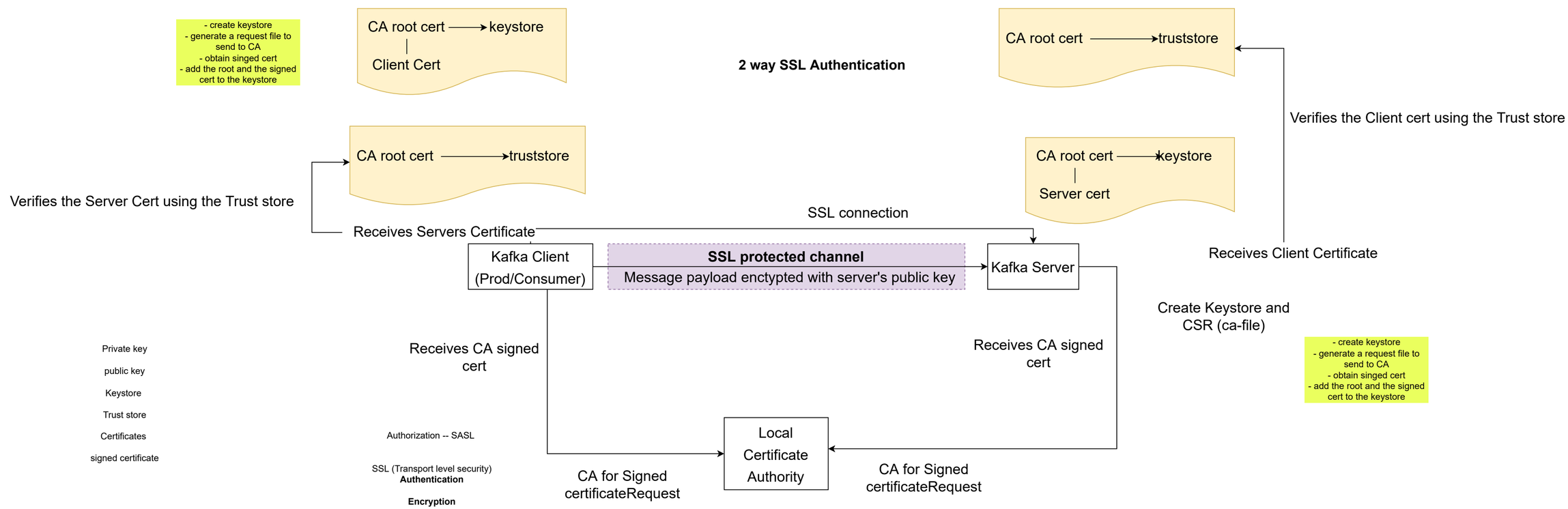
Recovery



Communications in Kafka Infra To Be Secured



Securing Kafka Client and Server using SSL



SSL Setup for securing Kafka Server and Clients - 1 Way

Client

Server

Cert Authority

1. Create keystore

keytool -keystore server.keystore.jks -alias localhost -validity 365 -genkey -keyalg RSA

creates a file named **server.keystore.jks**

2. Create local CA

openssl req -new -x509 -keyout ca-key -out ca-cert -days 365 -subj "/CN=local-security-CA"

creates files **ca-cert and ca-key**

3. create request file for geneating signed server cert

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

creates file **cert-file**

4. send the info file created in step 3 to local CA to generate signed cert

openssl x509 -req -CA ca-cert -CAkey ca-key -in ..\server-keys\cert-file -out ..\server-keys\cert-signed -days 365 -CAcreateserial -passin pass:kafkassl

creates file **cert-signed**

5. Add the local CA cert to root of server keystore

keytool -keystore server.keystore.jks -alias CARoot -import -file ..\local-ca\ca-cert

Adds the local ca root certificate to the root of the server keystore file named s**erver.keystore.jks**

6. Add the server signed certificate to the server keystore under this root ca

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

Adds the server signed cert to ca root certificate of the server keystore file named s**erver.keystore.jks**

7. Configure the Kafka Servers to use the certificates from the server keystore and restart servers

Make following entries in the server.properties

listeners=PLAINTEXT://localhost:9094, **SSL://localhost:10094**

 # security settings
ssl.keystore.location=E:/MyWork/trainings/Kafka/Code/KafkaTutorials/security-ssl/server-keys/server.keystore.jks
ssl.keystore.password=kafkassl
ssl.key.password=kafkassl
ssl.endpoint.identification.algorithm=

8. Add the Local CA root Cert to the tursted store of client

keytool -keystore client.truststore.jks -alias CARoot -import -file ..\local-ca\ca-cert

it will update the **client.truststore.jks** with ca root cert details

9. Create a config file eg **client-secure-ssl.properties in CLI tools folder with SSL configuration**

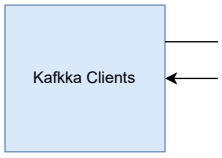
Create file with following content

security.protocol=SSL
ssl.truststore.location=E:/MyWork/trainings/Kafka/Code/KafkaTutorials/security-ssl/client-keys/client.truststore.jks
ssl.truststore.password=kafkassl
ssl.truststore.type=JKS
ssl.endpoint.identification.algorithm=

10. Perform Client Operations such as Producing, consuming messages

.\kafka-console-producer.bat --topic devices-topic --bootstrap-server localhost:10092 --producer.config .\client-secure-ssl.properties

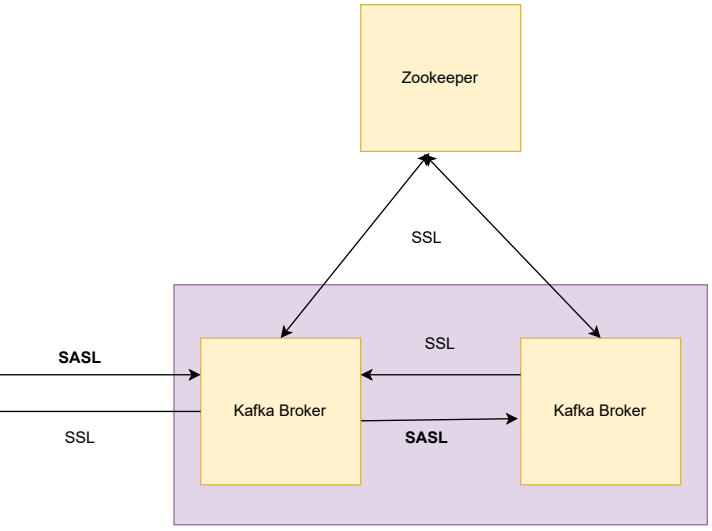
.\kafka-console-consumer.bat --topic devices-topic --bootstrap-server localhost:10092 --from-beginning --consumer.config .\client-secure-ssl.properties



User creation

ACL creation

SASL Topology



Important commands

Admin	./bin/kafka-configs.sh --zookeeper localhost:2182 --zk-tls-config-file ./bin/zookeeper-client.properties --alter --add-conf
Producer	./bin/kafka-configs.sh --zookeeper localhost:2182 --zk-tls-config-file ./bin/zookeeper-client.properties --alter --add-conf
Consumer	./bin/kafka-configs.sh --zookeeper localhost:2182 --zk-tls-config-file ./bin/zookeeper-client.properties --alter --add-con
Producer	./bin/kafka-acls.sh --authorizer-properties zookeeper.connect=localhost:2182 --zk-tls-config-file ./bin/zoo
Consumer	./bin/kafka-acls.sh --authorizer-properties zookeeper.connect=localhost:2182 --zk-tls-config-file ./bin/zooke
Consumer to group	./bin/kafka-acls.sh --authorizer-properties zookeeper.connect=localhost:2182 --zk-tls-config-file ./bin/zook

fig 'SCRAM-SHA-512=[password=kafkassl]' --entity-type users --entity-name broker-admin

fig 'SCRAM-SHA-512=[password=kafkassl]' --entity-type users --entity-name kafkaprod

fig 'SCRAM-SHA-512=[password=kafkassl]' --entity-type users --entity-name kafkacon

okeeper-client.properties --add --allow-principal User:kafkaprod --topic secure-topic --operation WRITE --operation DESCRIBE --c

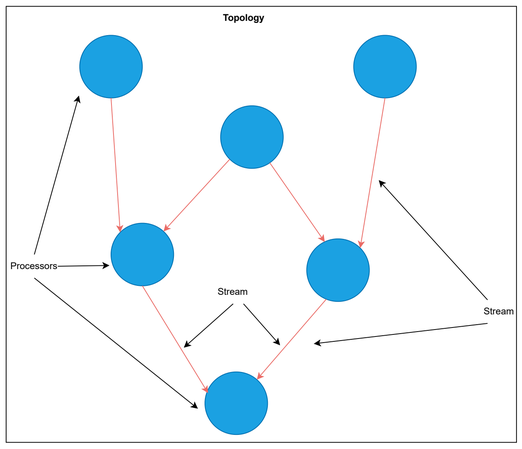
eeper-client.properties --add --allow-principal User:kafkacon --topic secure-topic --operation READ --operation DESCRIBE --opera

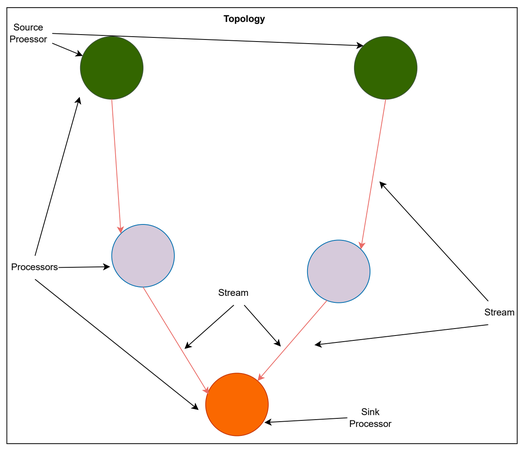
keeper-client.properties --add --allow-principal User:kafkacon --group secure-group --operation READ

operation DESCRIBECONFIGS

ation DESCRIBECONFIGS

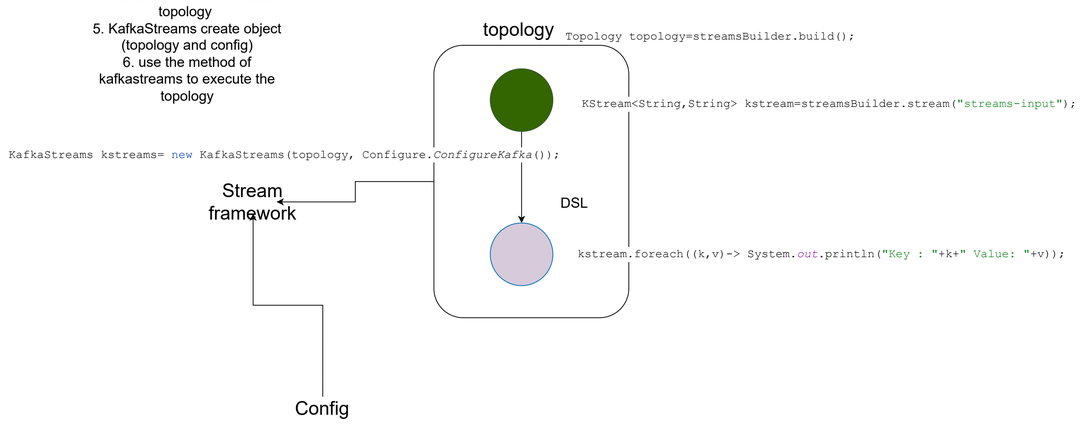
Kafka Streams



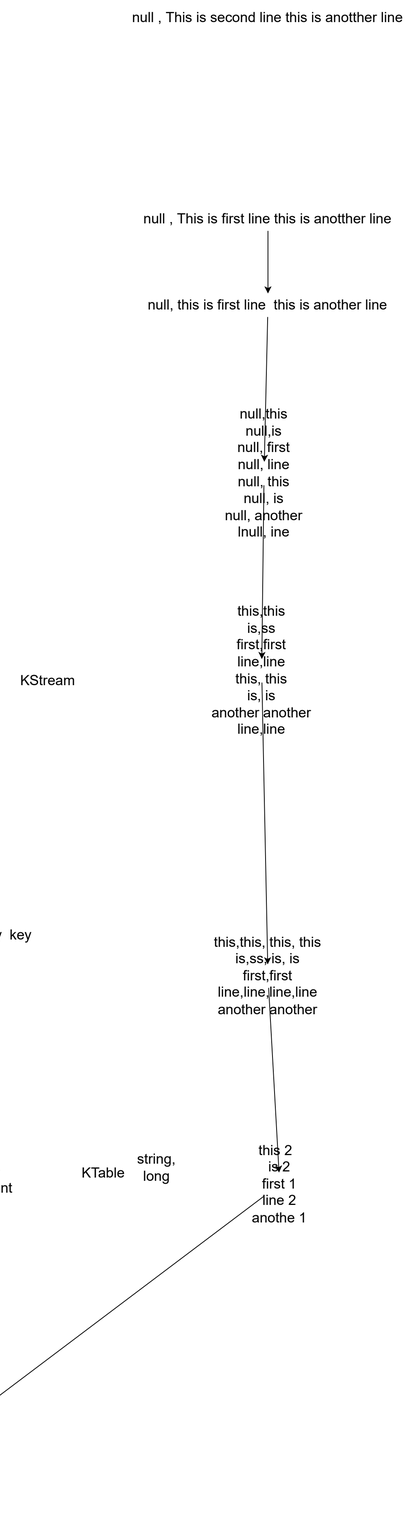


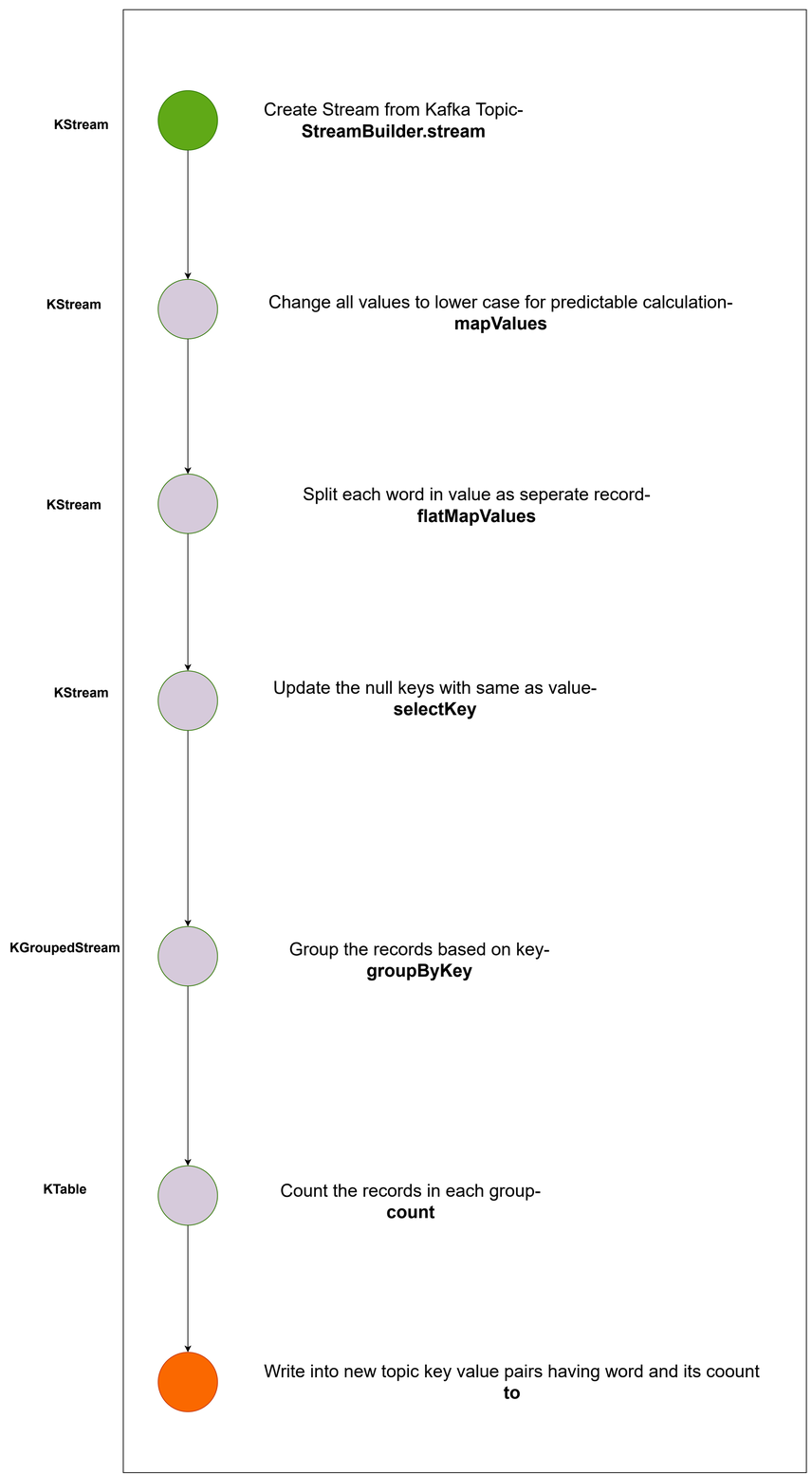
1. create StreamBuilder
2. Source Stream processor
3. Processing processors
4. build the builder to create topology
5. KafkaStreams create object (topology and config)
6. use the method of kafkastreams to execute the topology

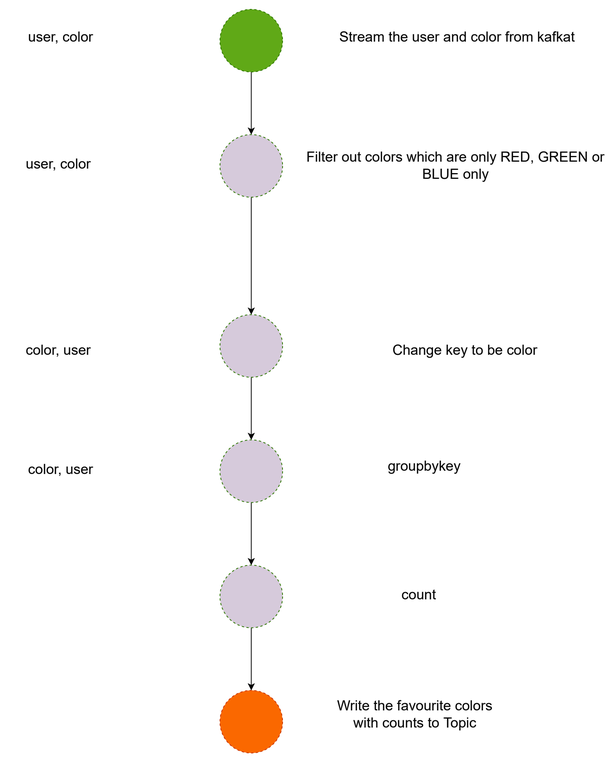
Basic Topology



WordCount Topology







Kafka Streaming Architecture