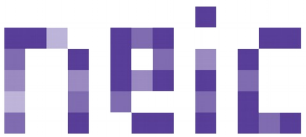
dCache.org

Kafka with dCache

Marina Sahakyan

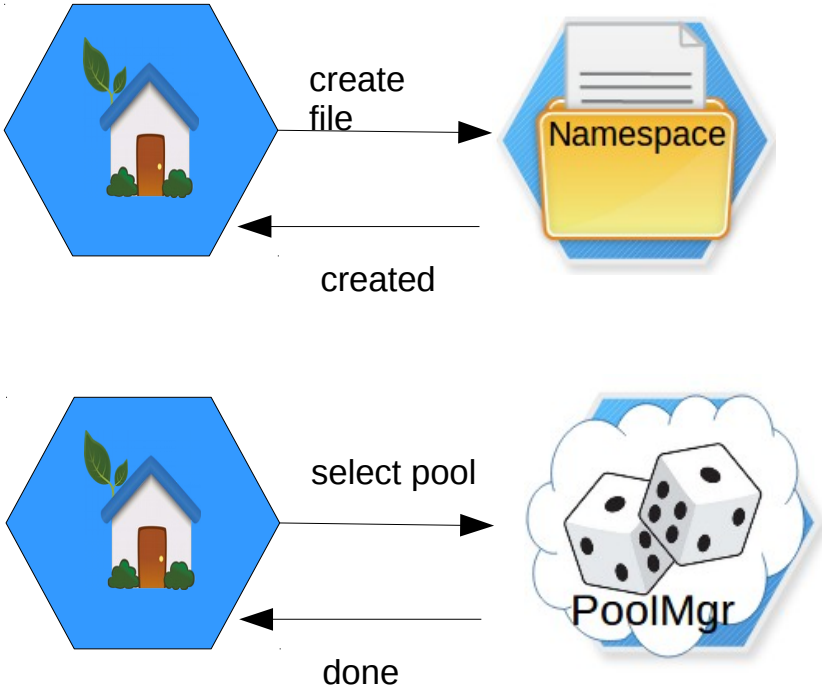
Madrid, 21 May



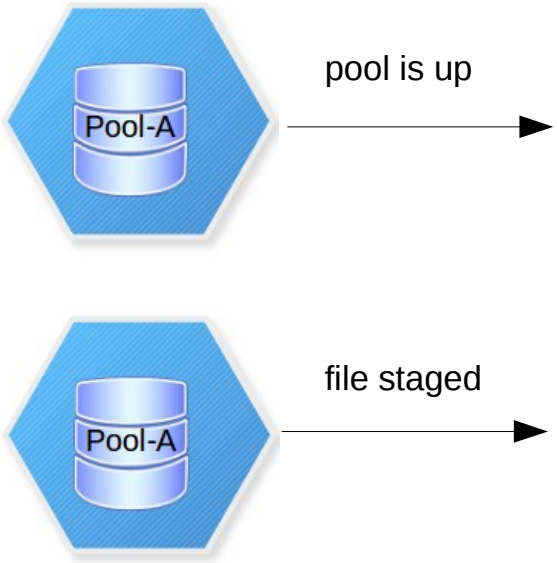
Nordic e-Infrastructure Collaboration

HELMHOLTZ RESEARCH FOR GRAND CHALLENGES

Events inside dCache

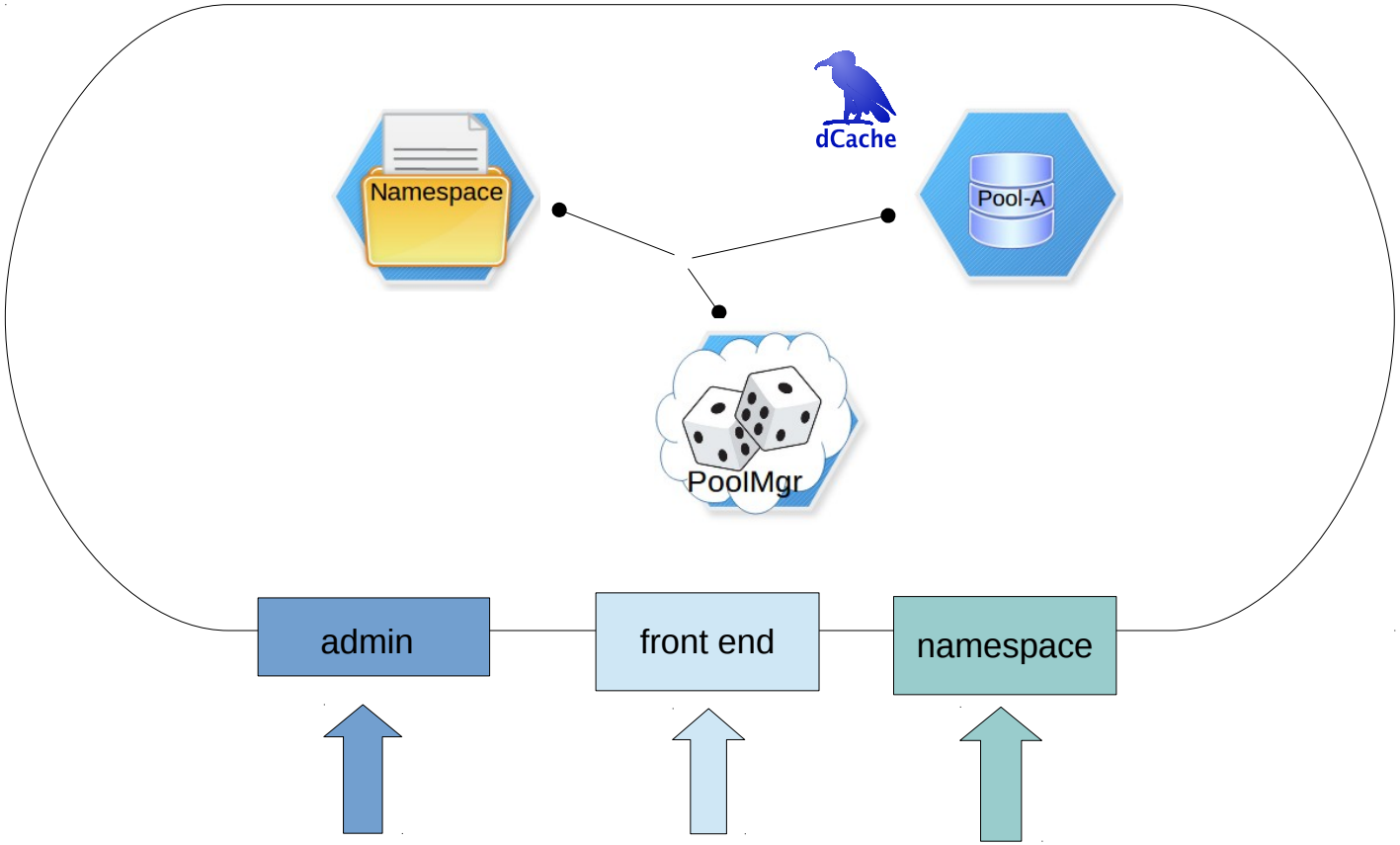


request/response
(rpc like)

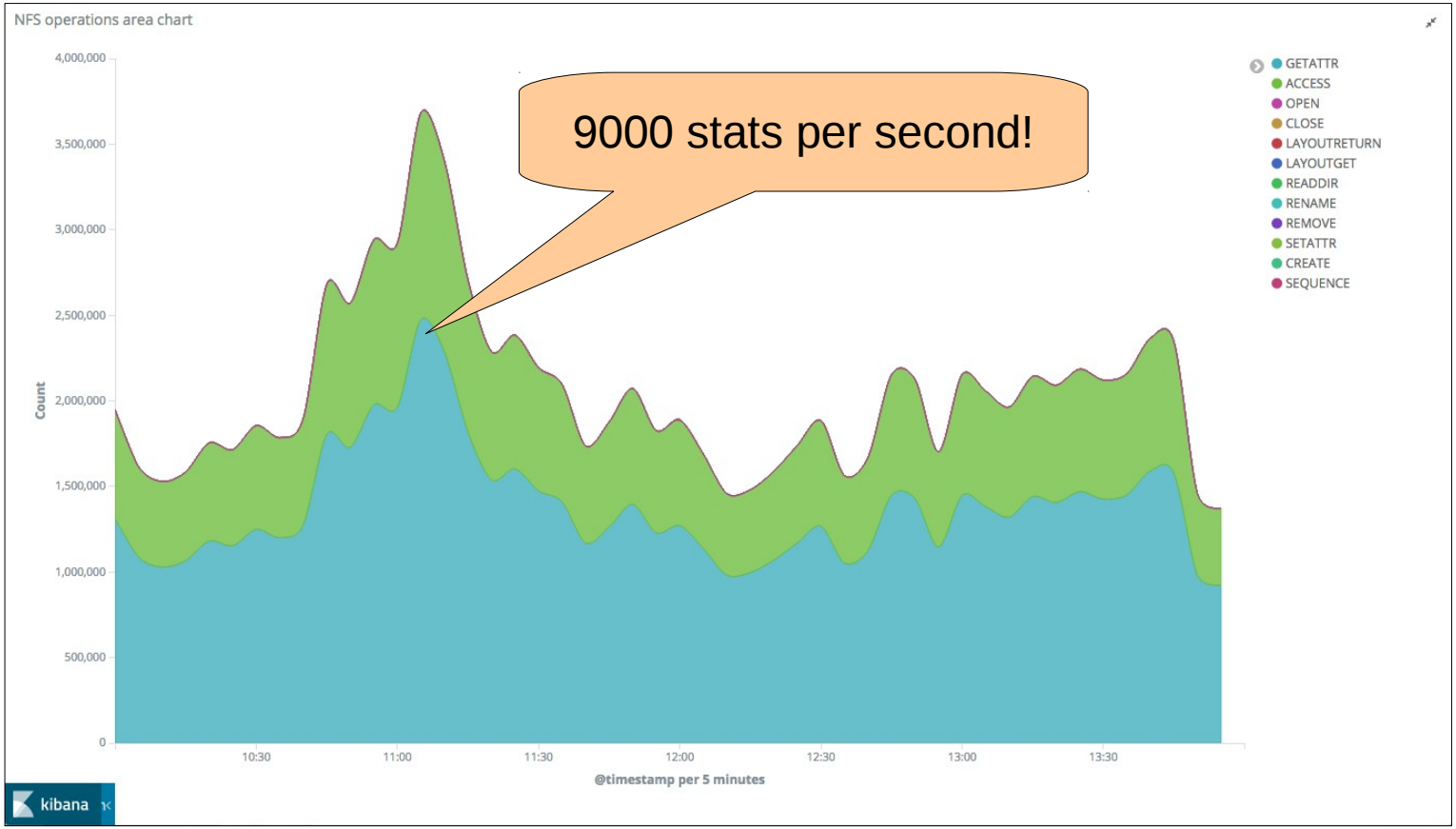


Event notifications

Inside dCache



Getting dCache events



Getting dCache events

9000 stats per second!

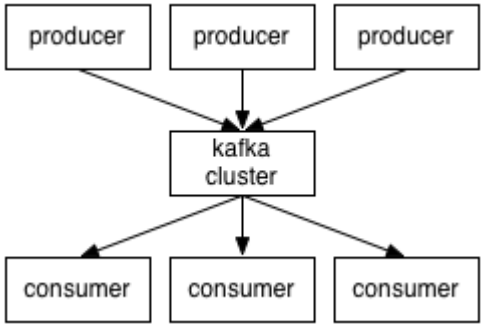
Getting dCache events

Stop polling please!

Storage events in dCache

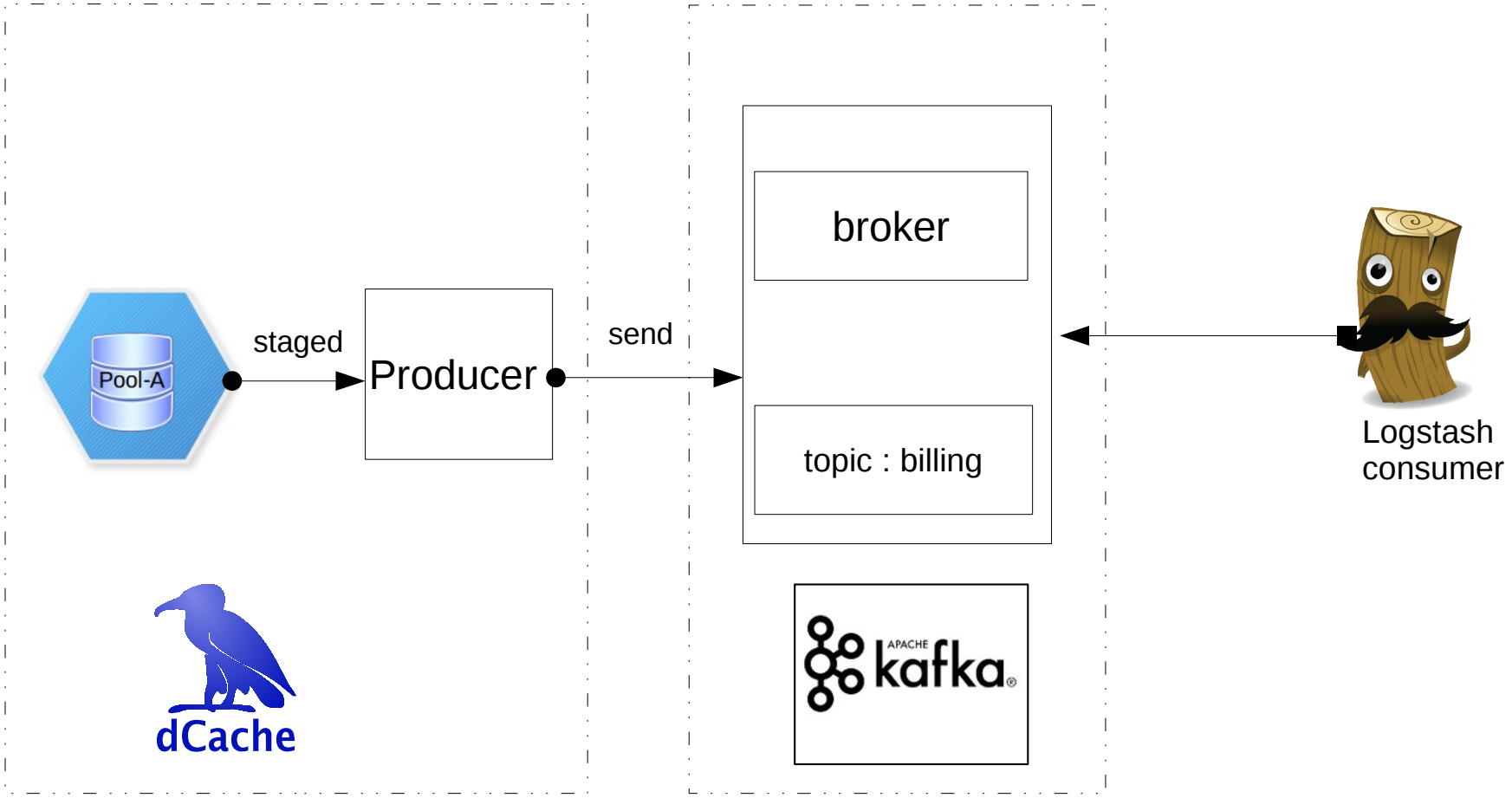
- Kafka stream
  - Producer-consumer model
  - Kafka consumer is required
  - Global events
  - Consumer keeps track of the last seen event
  - Integration with other tools (Spark, ELK, ...)
- Server-Sent Events (SSE)
  - Producer-consumer model
  - HTTP connection "for receiving push notifications from a server"
  - User specific event stream
  - Client keeps track of the "Last-Event-ID"

Storage events in dCache

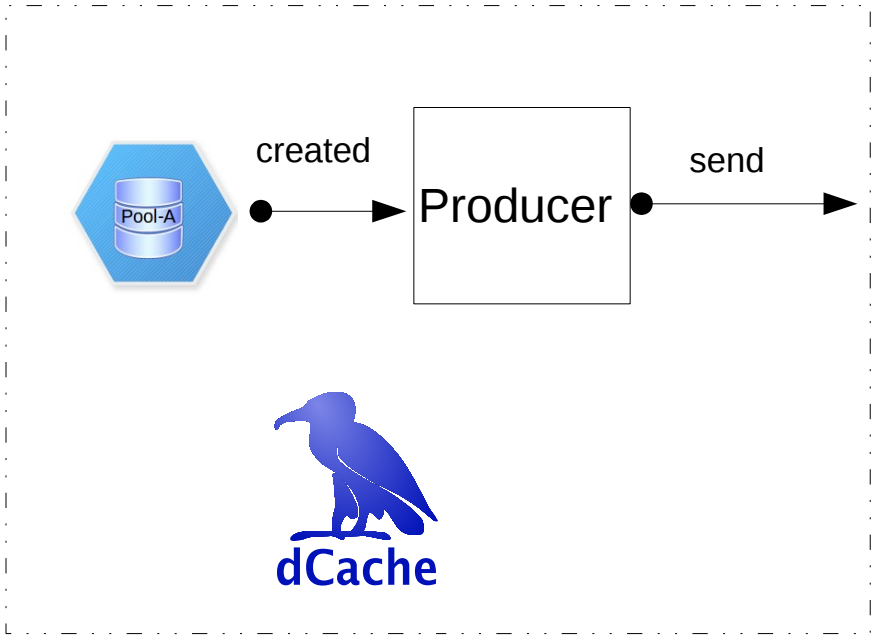


- Kafka stream
  - Producer-consumer model
  - Kafka consumer is required
  - Global events
  - Consumer keeps track of the last seen event
  - Integration with other tools (Spark, ELK, ...)
- Server-Sent Events (SSE)
  - Producer-consumer model
  - HTTP connection "for receiving push notifications from a server"
  - User specific event stream
  - Client keeps track of the "Last-Event-ID"

dCache as Kafka producer

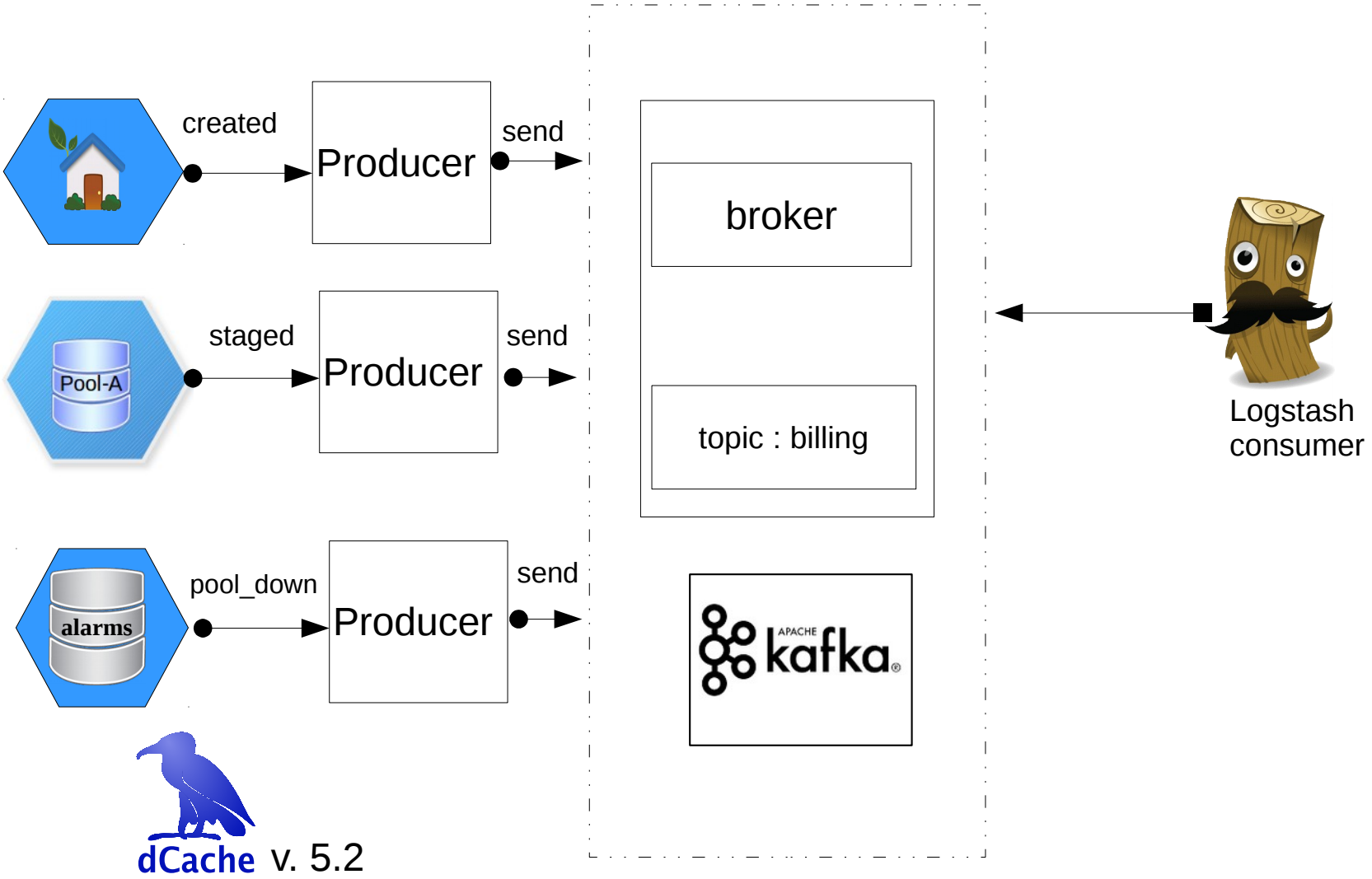


dCache as Kafka producer

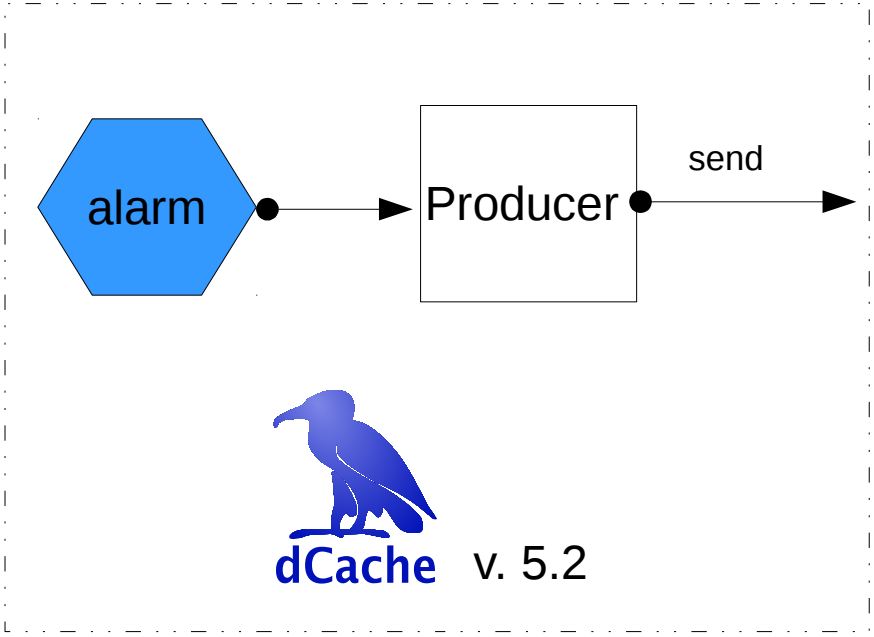



```
{
"msgType": "transfer",
"cellType": "pool",
"meanWriteBandwidth ": 9.751456E8 ,
"isP2p": false ,

"isWrite": "write",

"protocolInfo": {
"protocol": "NFS4",
"port": 746,
"host": "192.168.163.49",
"versionMajor": 4,
"versionMinorv: 1,
}
```

dCache as Kafka producer

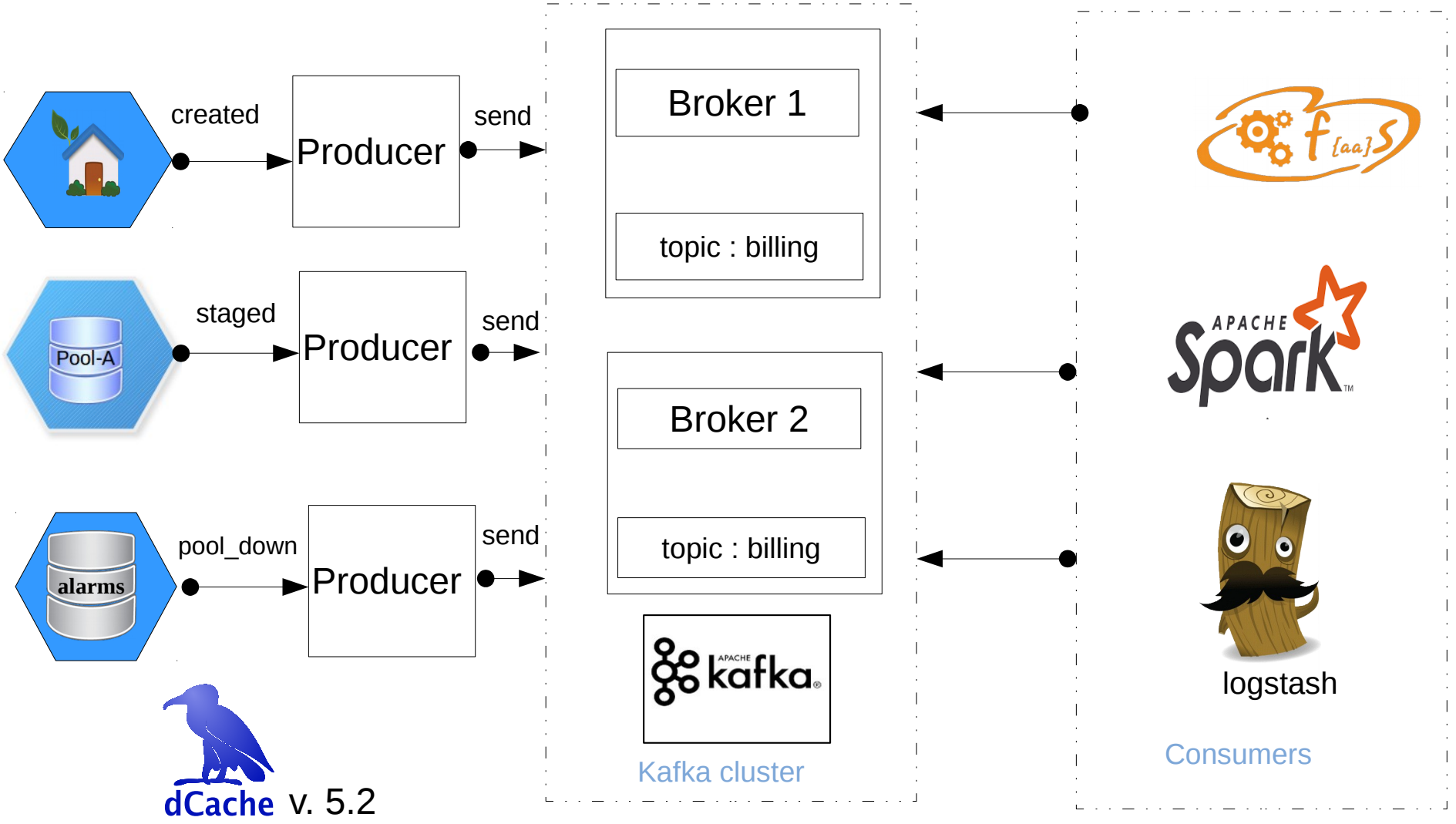


dCache as Kafka producer

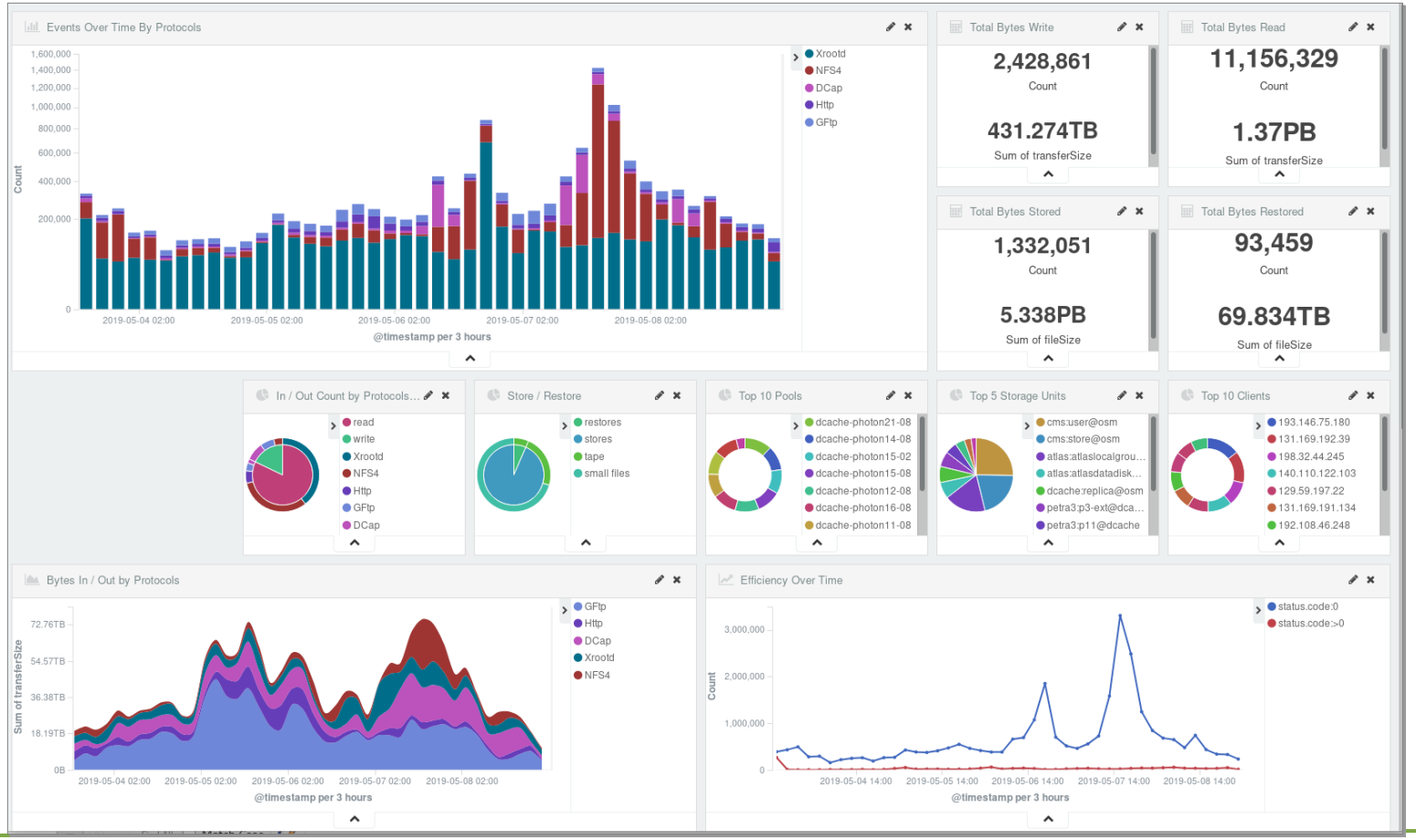



```
{
 "timestamp" : "2019-04-24T13:46:06.449Z",
 "level" : "ERROR",
 "thread" : "Thread-125",
 "marker" : {
   "key" : "OUT_OF_FILE_DESCRIPTORS:pool_name",
   "firstArrived" : 1556113566449,
   "lastUpdate" : 1556113566449,
   "type" : "OUT_OF_FILE_DESCRIPTORS"
    }
}
```

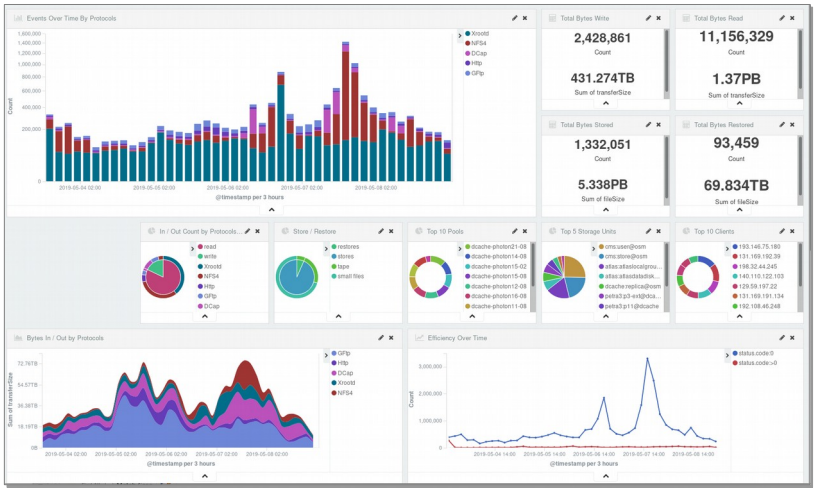
dCache as Kafka producer



Example for Monitoring with ELK



Example for Monitoring with ELK




```
{
"msgType": "transfer",
"cellType": "pool",
"meanWriteBandwidth ": 9.751456E8 ,
"isP2p": false ,

"isWrite": "write",
```

Example of configuring Logstash

```
input {
    kafka {
            bootstrap_servers => "dcache-billing-cloud.desy.de:9092"
            topics => ("billing")
            codec => "json"
            tags => ("cloud","billing")
    }
}

filter {
  date {
    match => ( "date", "ISO8601" )
    timezone => "CET"
  }
}

output {
  elasticsearch {
    hosts => ("itelk01", "itelk02", "itelk04")
  }
}
```

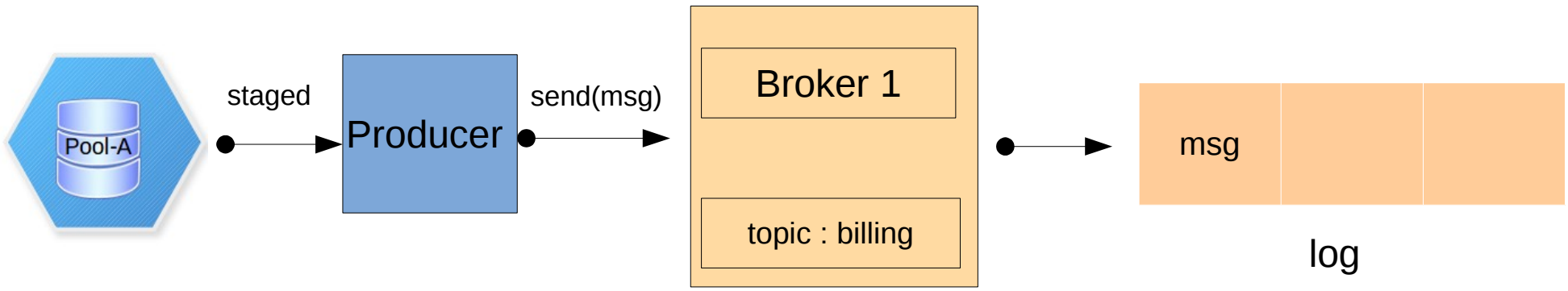
Enabling Kafka

- Enabling Kafka - globally
  - **dcache.enable.kafka = true**
  - **dcache.kafka.bootstrap-servers = localhost:9092**
  - **dcache.kafka.topic = billing**
  - **Alarms (v. 5.2)**
    - **dcache.log.kafka.topic = alarms**
    - **dcache.log.level.kafka = error**
- Enabling Kafka – for a specific service
  - **{ nfs, ftp, dcap, … }.dcache.enable.kafka = true**
    - **pool.dcache.enable.kafka = true**

Message delivery policy

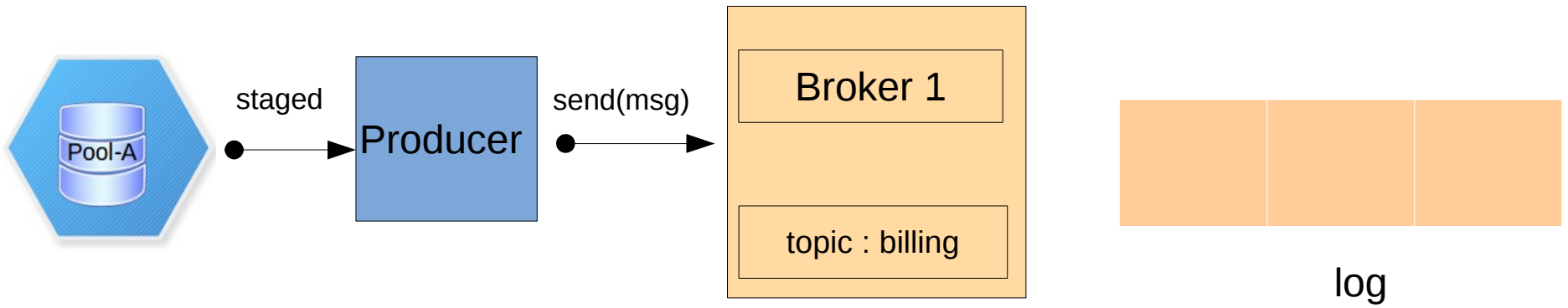
1. At most once
2. At least once
3. Exactly once

1. At most once



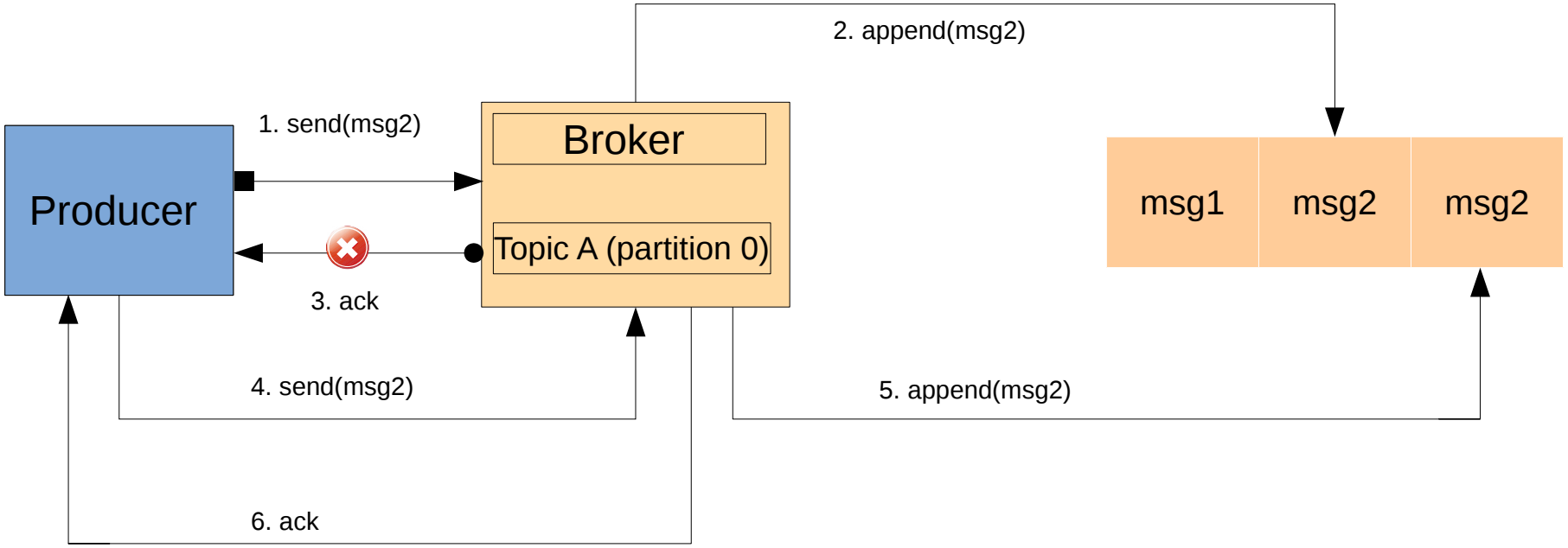
- Producer does not retry when when no ack is received

1. At most once



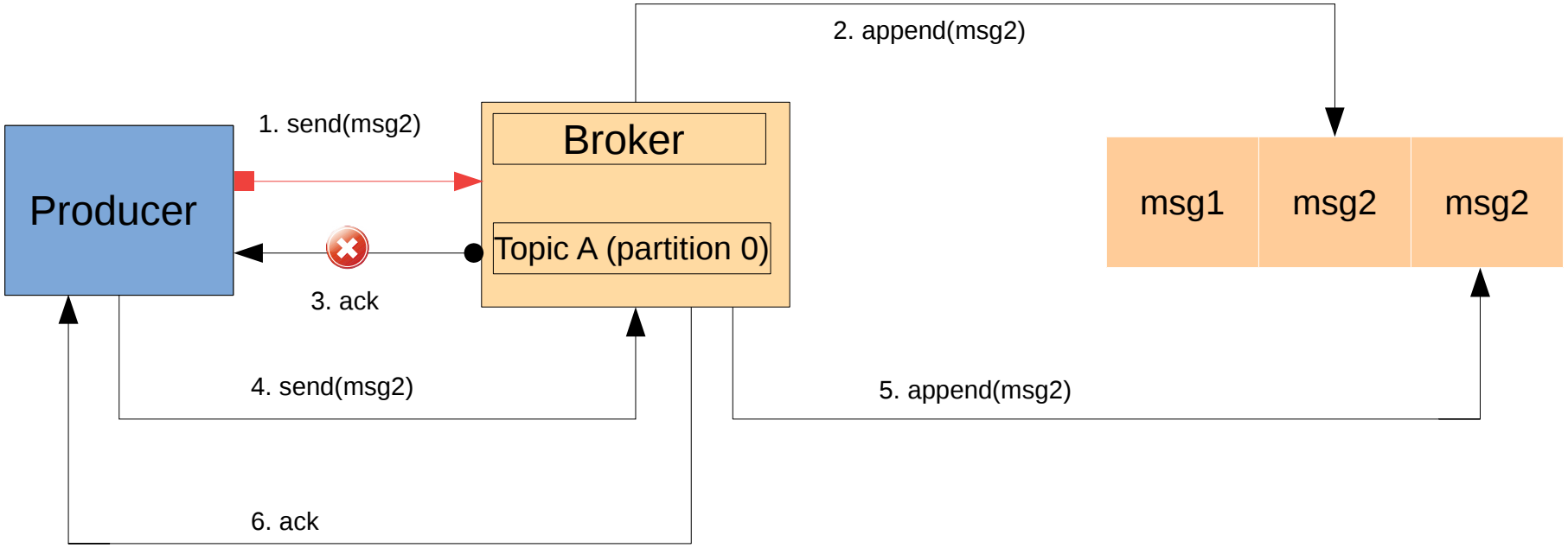
- Producer does not retry when when no ack is received
- The message might end up not being written to the Kafka topic

2. At least once



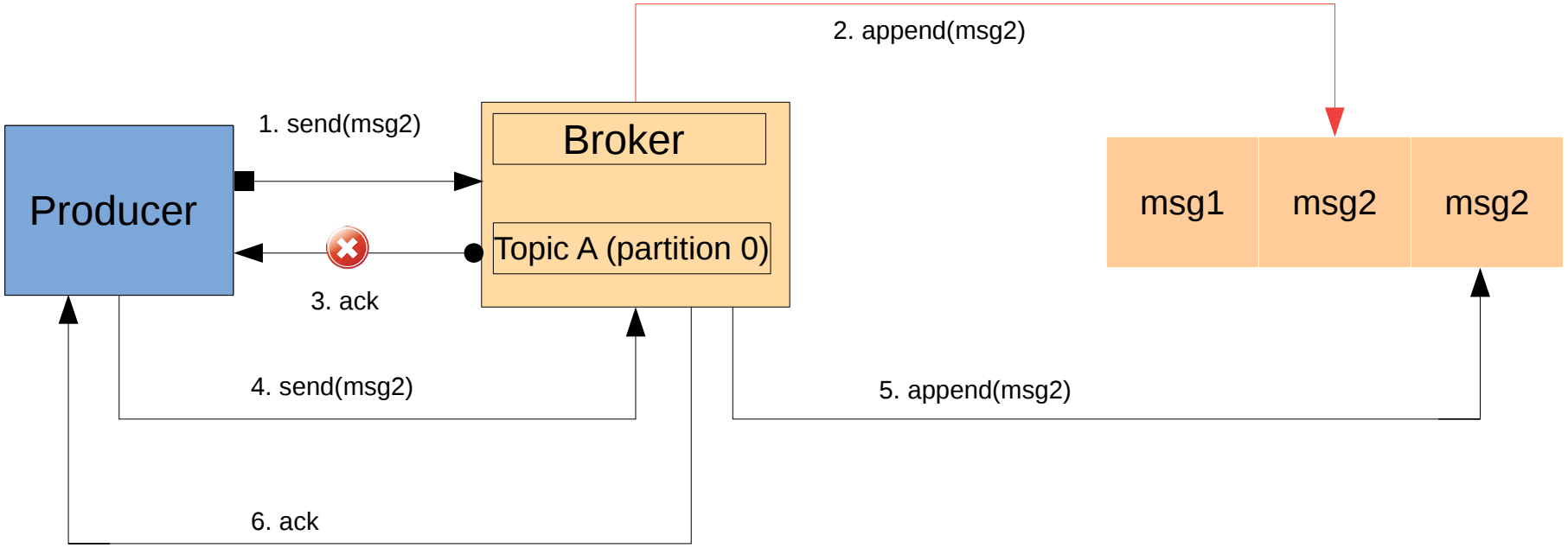
- Producer retries as long as it doesn't get ack
- Implications – duplicated messages

2. At least once



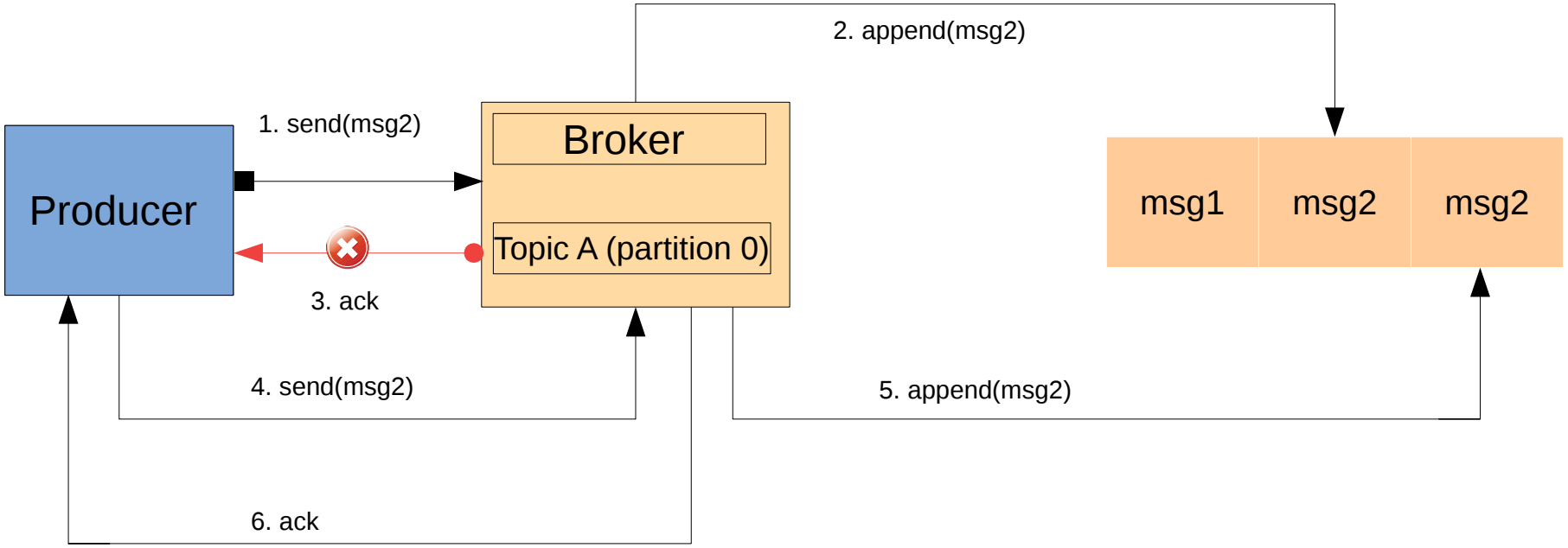
- Producer retries as long as it doesn't get ack
- Implications – duplicated messages

2. At least once



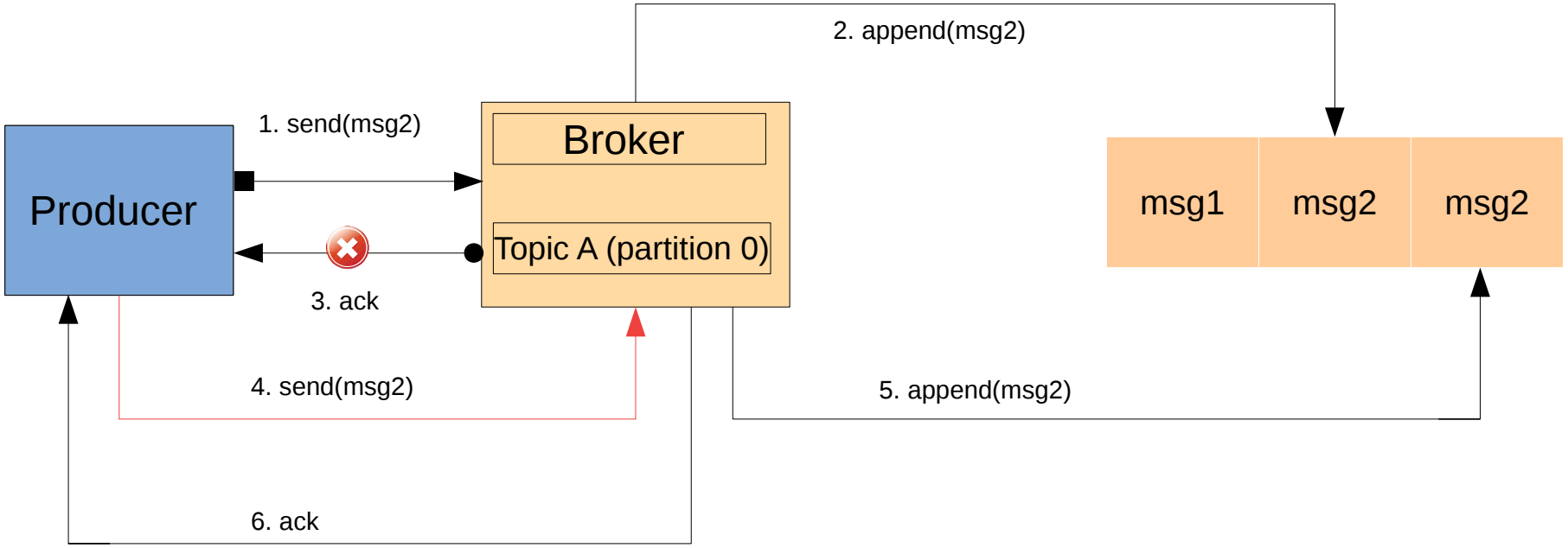
- Producer retries as long as it doesn't get ack
- Implications – duplicated messages

2. At least once



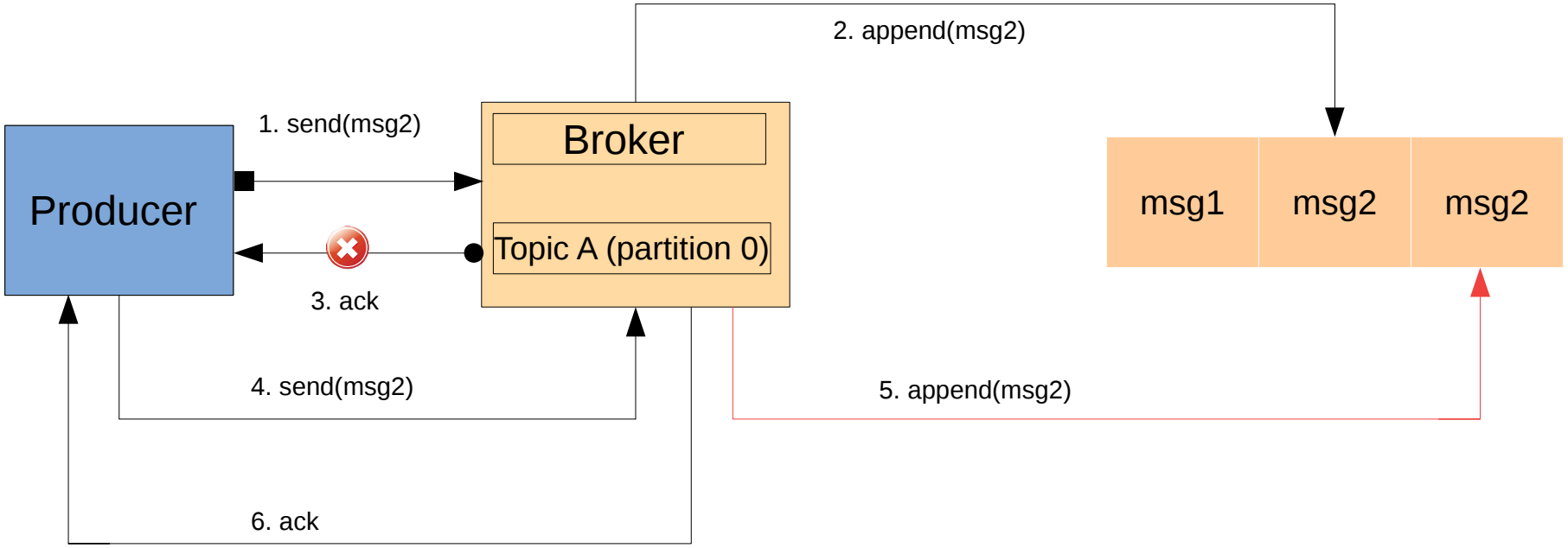
- Producer retries as long as it doesn't get ack
- Implications – duplicated messages

2. At least once



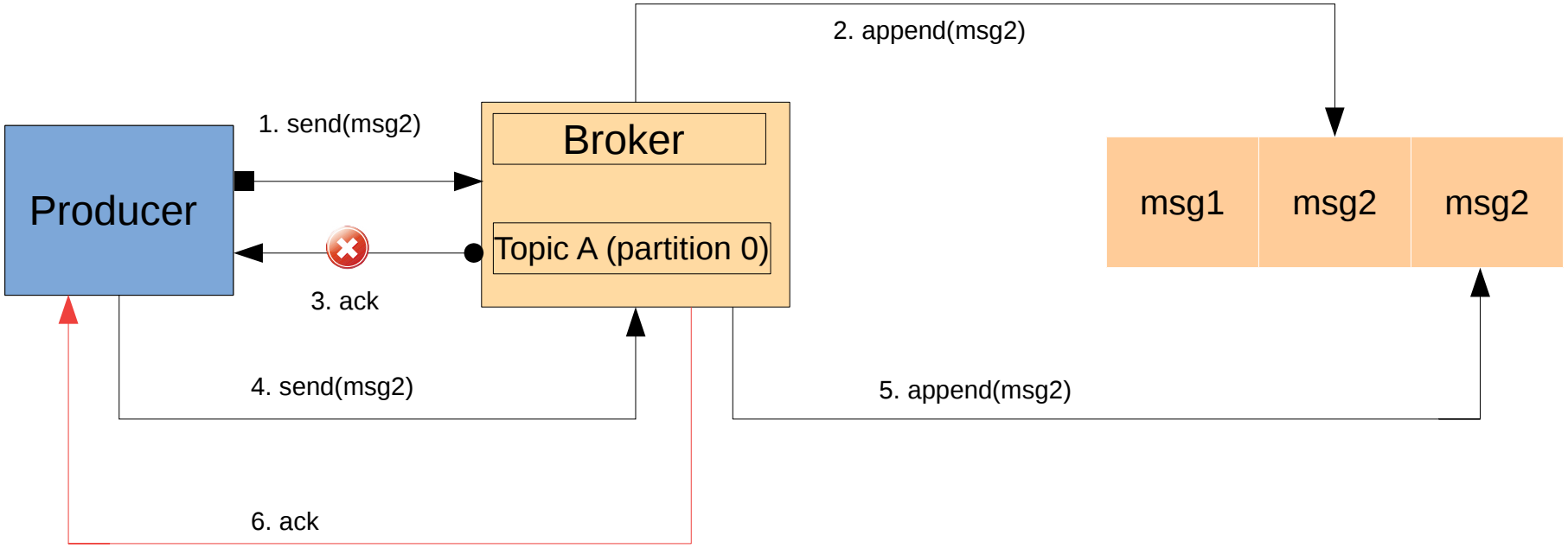
- Producer retries as long as it doesn't get ack
- Implications – duplicated messages

2. At least once



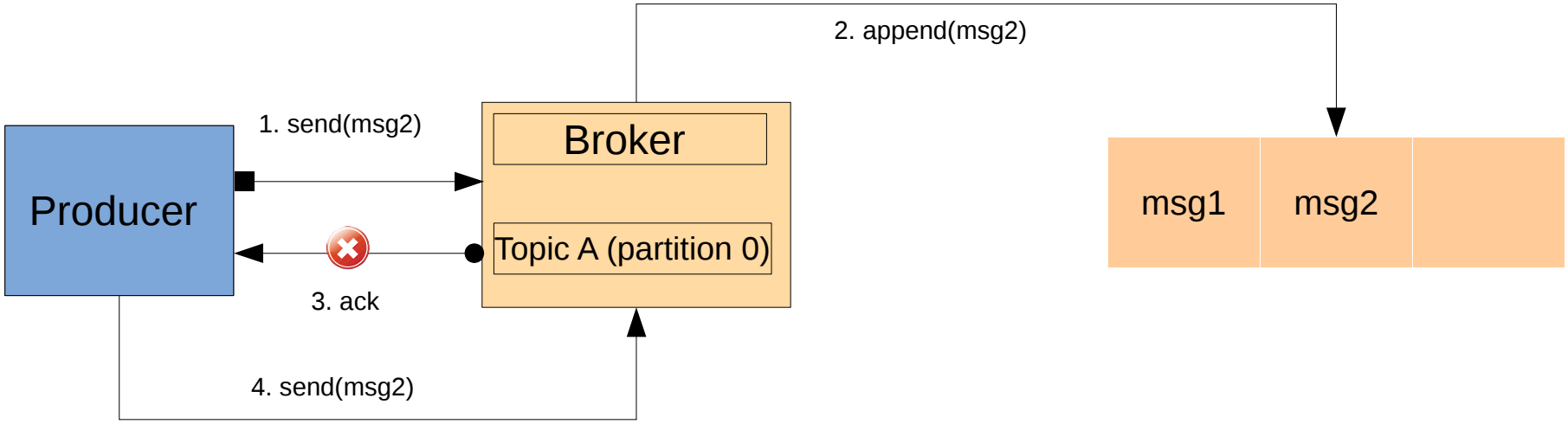
- Producer retries as long as it doesn't get ack
- Implications – duplicated messages

2. At least once



- Producer retries as long as it doesn't get ack
- Implications – duplicated messages

3. Exactly once



- Similar to at least one delivery, but broker detects duplicates.

At least/At most/Exactly once

	Properties in dCache	Delivery guaranty	Duplicates	Message throughput	Impact on dCache
At most once	acks = 0	No	No	High	No
At least once	acks = all|1 retries > 0	Yes	Yes	Lower	Yes
Exactly once	acks = all|1 Retries > 0 enable.idempotence = true max.in.flight.requests.per.connection = 1	Yes	No	Lower	Yes

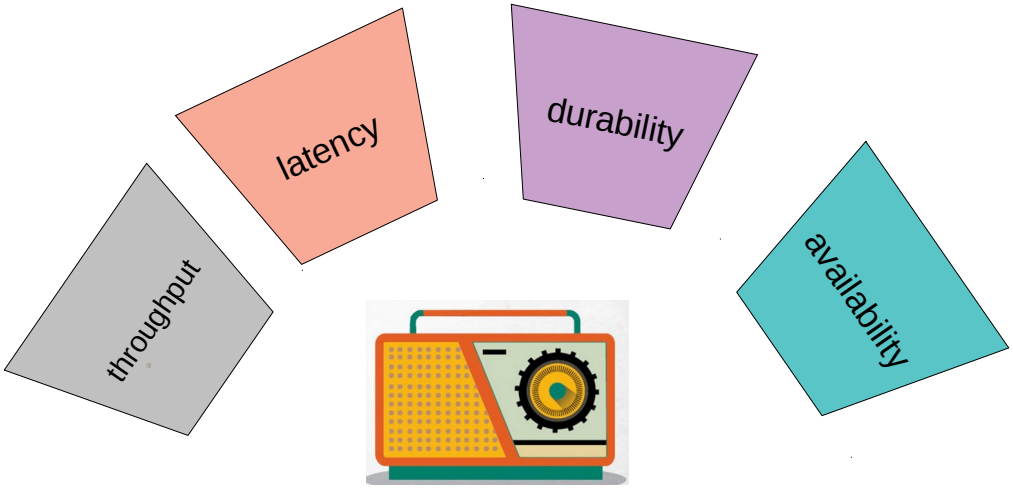
At least/At most/Exactly once

	Properties in dCache	Delivery guaranty	Duplicates	Message throughput	Impact on dCache
At most once	acks = 0	No	No	High	No
At least once	acks = all|1 retries > 0	Yes	Yes	Lower	Yes
Exactly once	acks = all|1 Retries > 0 enable.idempotence = true max.in.flight.requests.per.connection = 1	Yes	No	Lower	Yes

{pool,...}.kafka.producer.configs!ack = 1

Tweaking Kafka Producer

- Like any Kafka producer…



- More info
  - https://www.confluent.io/wp-content/uploads/Optimizing-Your-Apache-Kafka-Deployment-1.pdf
  - https://kafka.apache.org/documentation

Summary

- dCache provide 2 type of events
- Analysis
- Monitoring
- Workflow engine (like FaaS)
- Tweak it according to you needs (v. 5.1)

Thank you/Questions

Tweaking kafka producer

- Throughput
- Latency
- Durability
- **Availability**
  - Broker cluster

Tweaking kafka producer

- **batch.size**
  - when multiple records are sent to the same partition, the producer will batch them together
- **linger.ms**
  - controls the amount of time to wait for additional messages before sending the current batch

Tweaking kafka producer

- **batch.size - size base**
  - when multiple records are sent to the same partition, the producer will batch them together
- **linger.ms - time based**
  - controls the amount of time to wait for additional messages before sending the current batch

Tweaking kafka producer

- **Throughput**
  - batch.size - default 16384, increase to 100000 - 200000
  - linger.ms – default is 0, increase to 10 - 100
  - acks = 1
  - retries = 0
  - compression.type = lz4
  - buffer.memory - increase if there are a lot of partitions
- Latency
- Durability
- Availability

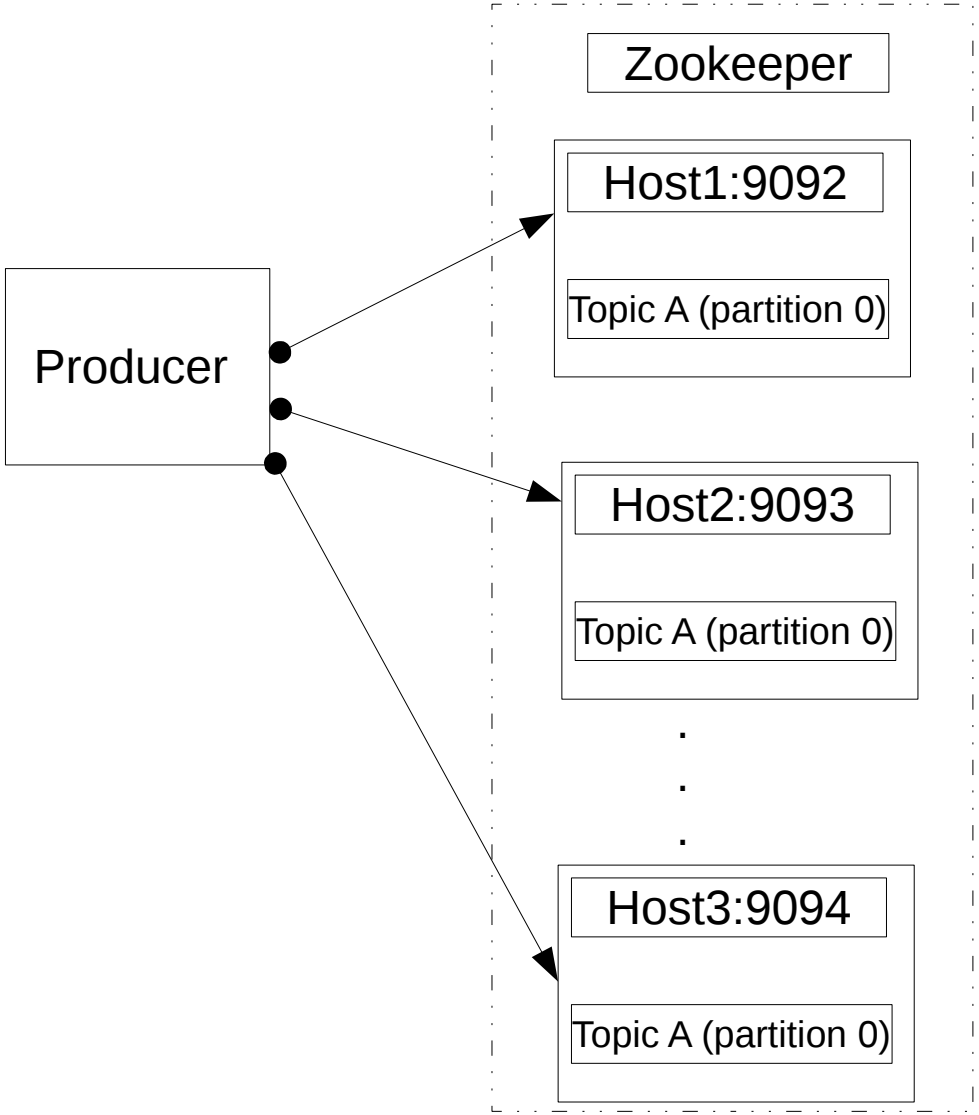
Tweaking kafka producer

- Throughput
- **Latency**
    - linger.ms = 0
    - compression.type = none
    - acks = 1
- Durability
- Availability

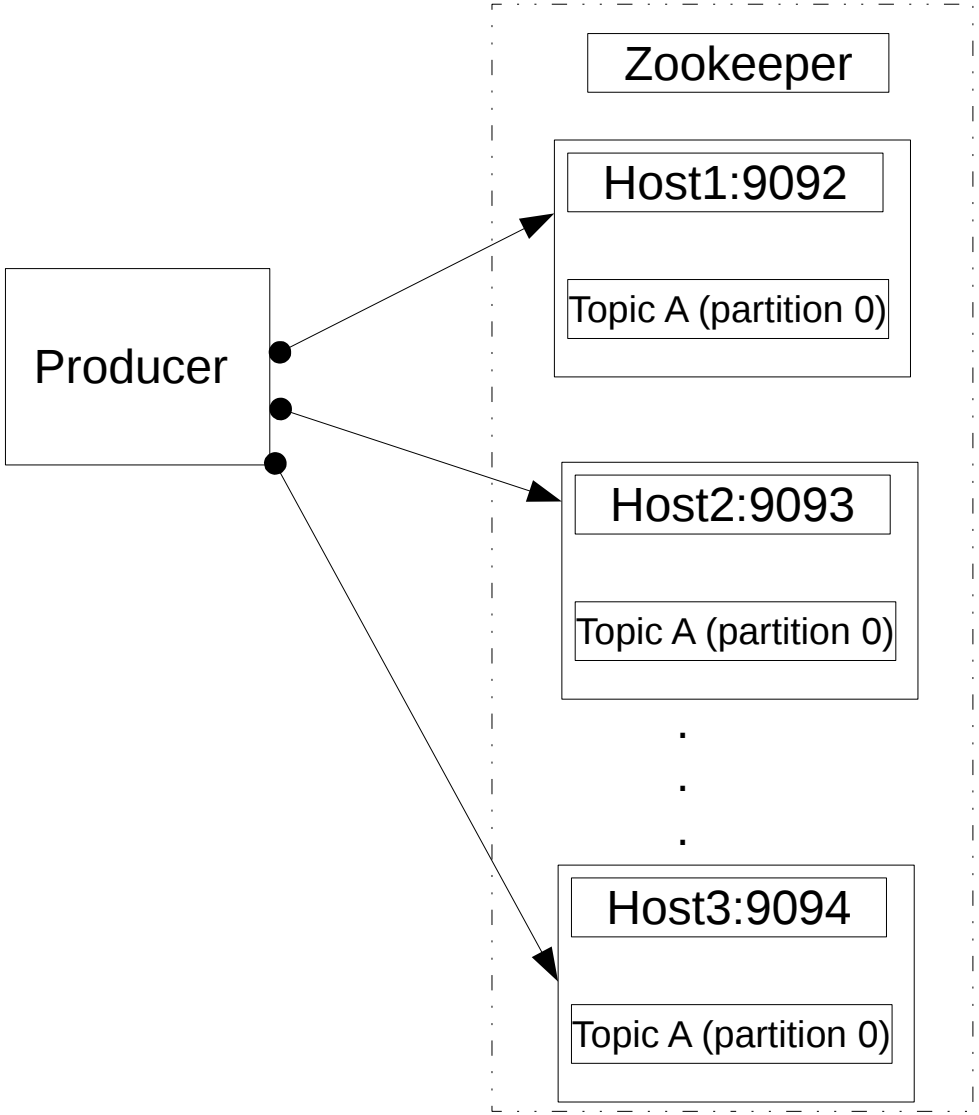
Tweaking kafka producer

- Throughput
- Latency
- **Durability**
    - retries - 1 or more
    - acks = all
    - replication.factor – 3, configure per topic
    - max.in.flight.requests.per.connection = 1
- Availability

Kafka broker cluster

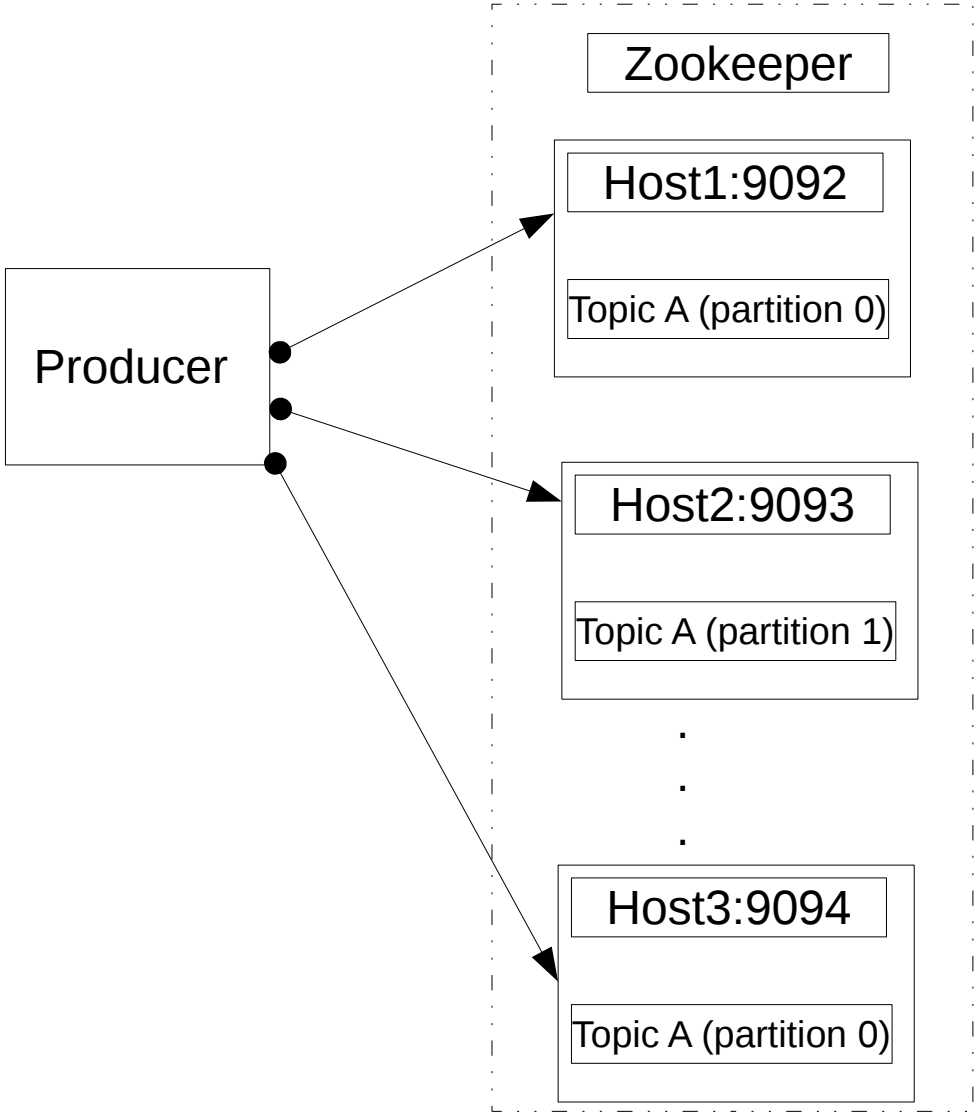


Kafka broker cluster




```
bootstrap.server=host1:9092:host2:9093
```

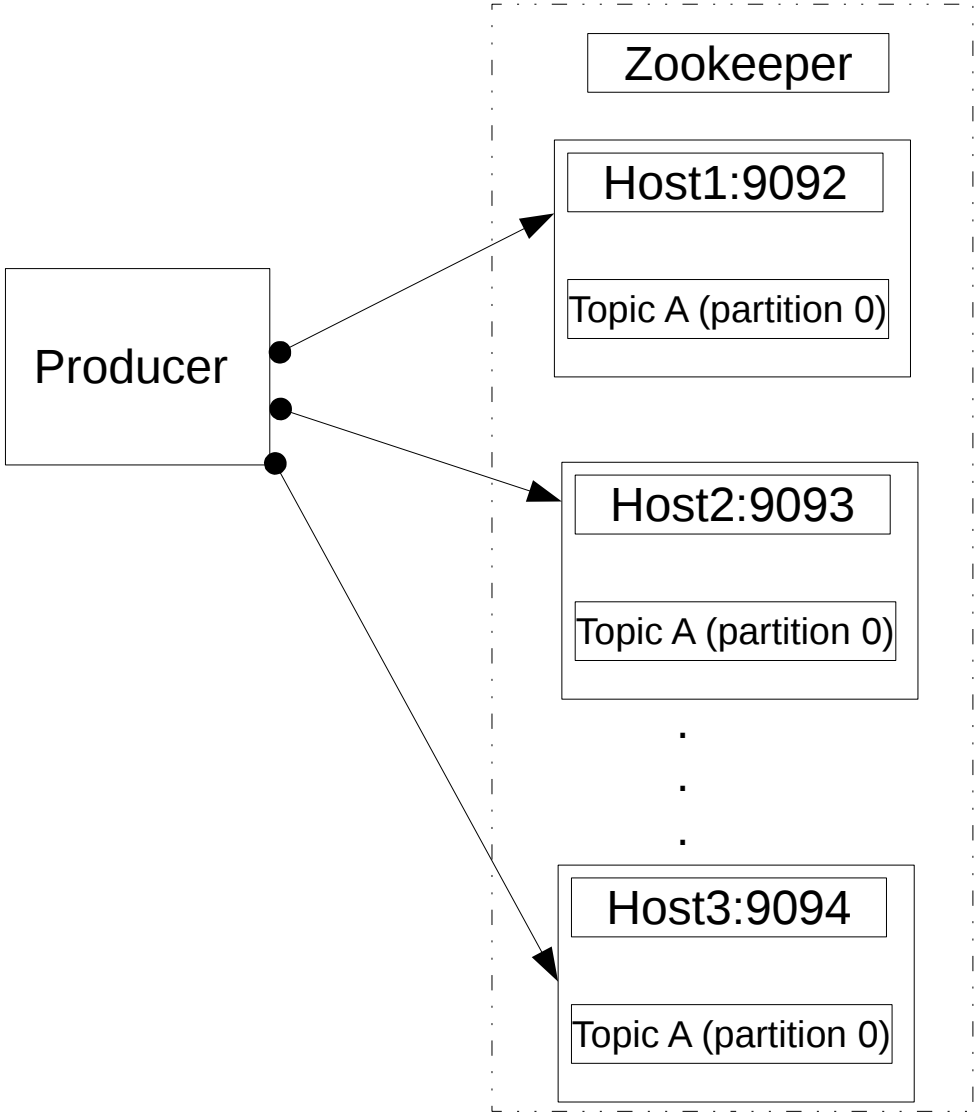
Kafka broker cluster



bootstrap.server=host1:9092:host2:9093

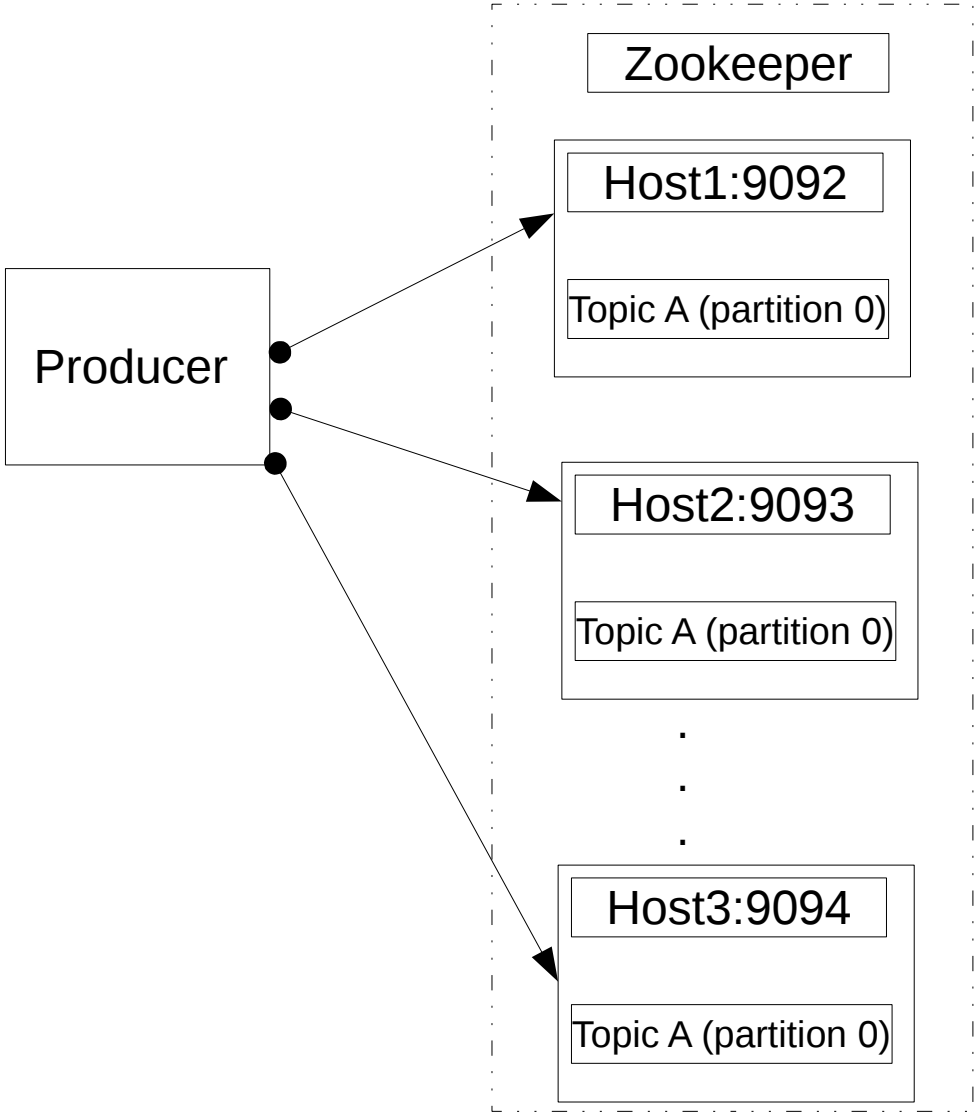
doesn't need to include all brokers

Kafka broker cluster



default.replication.factor

Kafka broker cluster

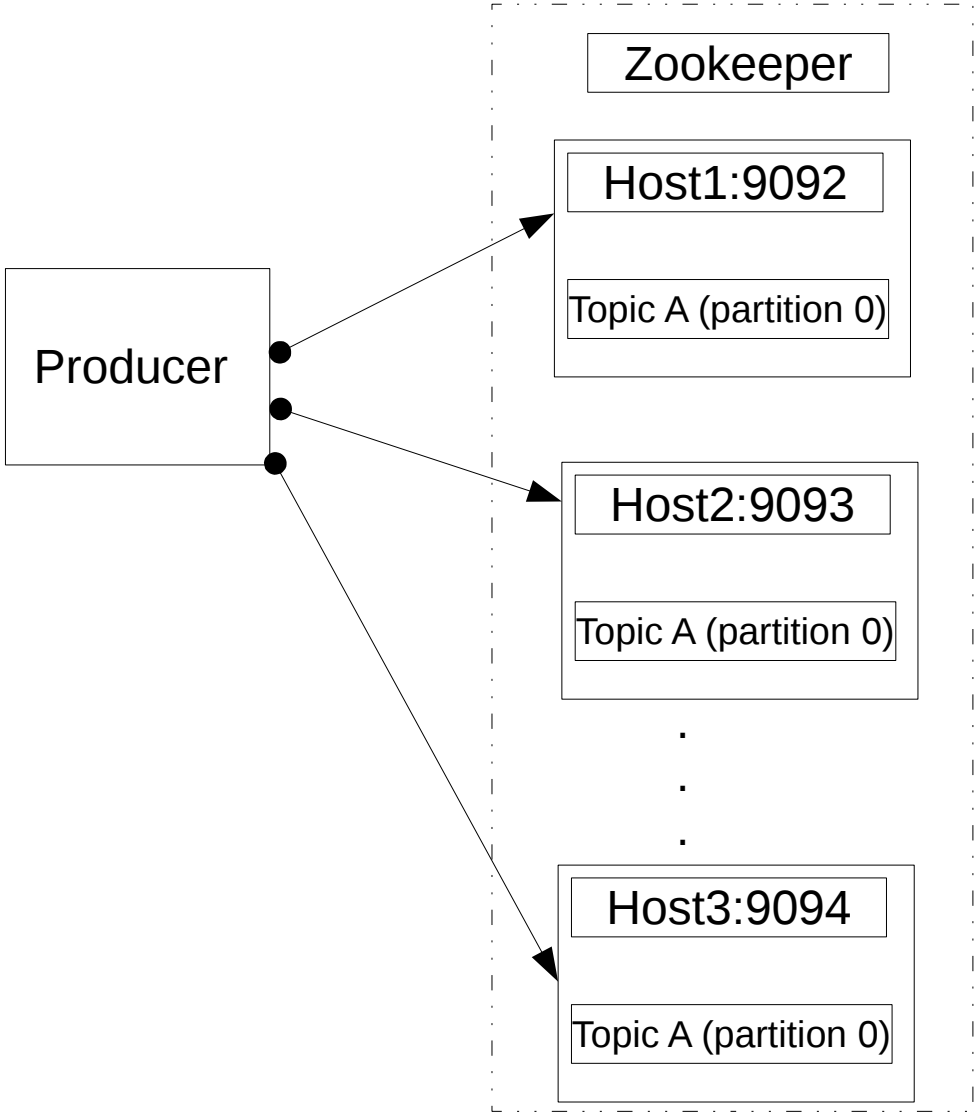


default.replication.factor = 3

Default replication factor

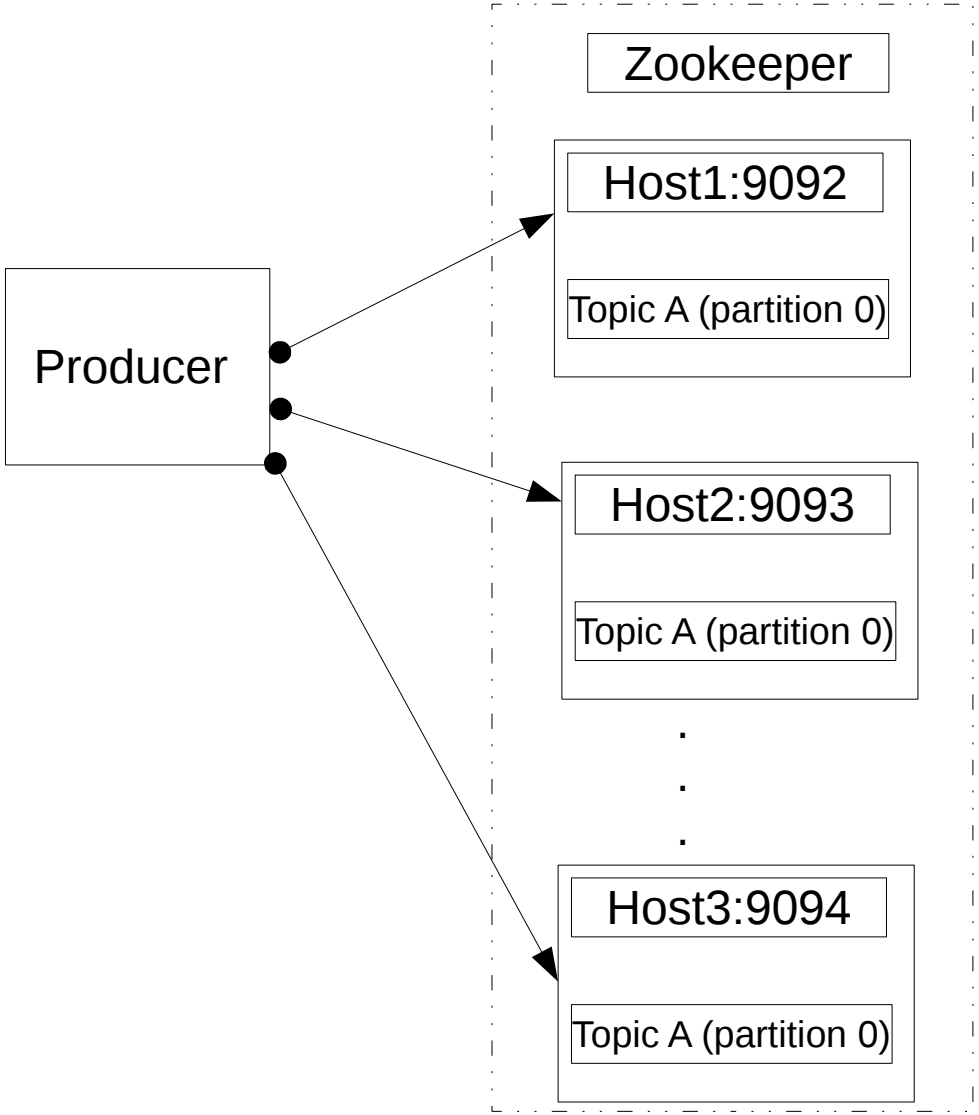
- Ex. = 3 means that each partition is replicated thre times on three different brokers.
- Even after a topic exists, you can choose to add or remove replicas and thereby modify the replication factor.
- Replication factor of N allows you to lose N-1 brokers
  - A single producer send results in exactly one copy of the message

Kafka broker cluster



unclean.leader.election.enable

Kafka broker cluster



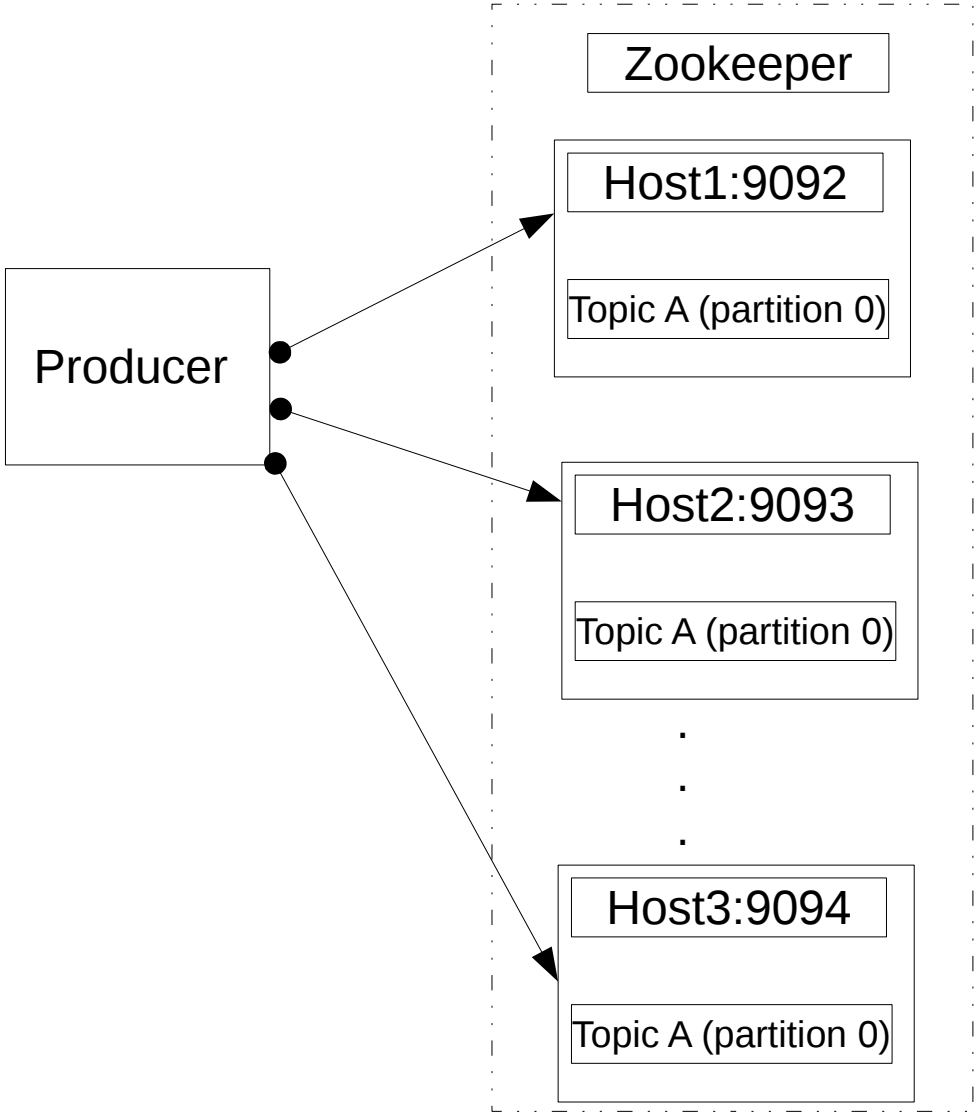
unclean.leader.election.enable

default true

Default replication factor

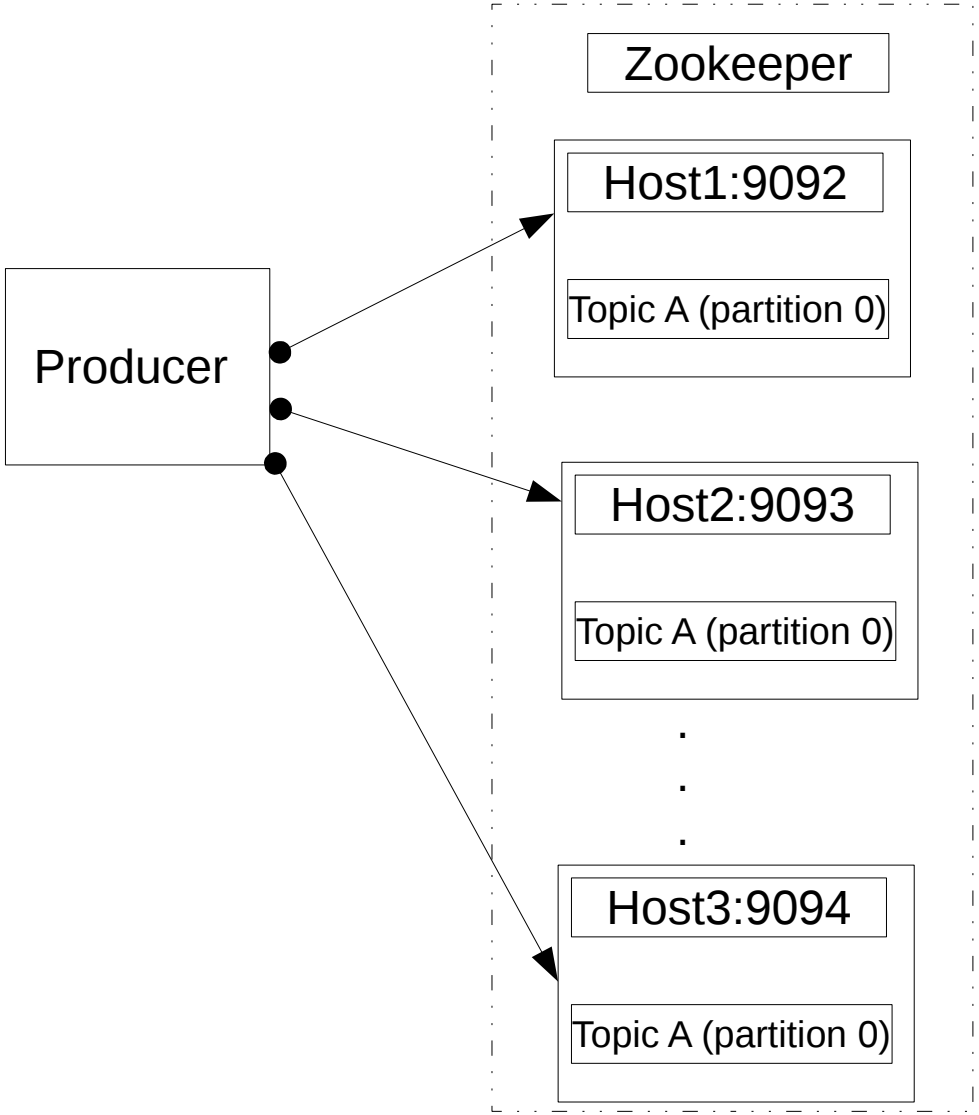
- unclean.leader.election.enable = true
  - out-of-sync replicas are allow to become leaders
  - messages loss , lower consistency
- unclean.leader.election.enable = false
  - waiting for the original leader to come back online
  - resulting in lower availability.

Kafka broker cluster



min.insync.replicas

Kafka broker cluster



min.insync.replicas =2

default.replication.factor = 3

Tuning kafka producer

- **max.inflight.requests.per.connection**
  - controls how many messages the producer will send to the server without receiving responses
  - affects ordering → setting to 1 quarantines be written to the broker in the order
- **retries**
- **acks(affects durability)**

Tuning kafka producer

- **max.inflight.requests.per.connection**
  - controls how many messages the producer will send to the server without receiving responses
  - affects ordering → setting to 1 quarantines be written to the broker in the order
- **Retries**
  - If the request fails, the producer can automatically retry
- **acks**
  - controls how many partition replicas must receive the record before the producer can consider the write successfully

At least once

- `[service].kafka.producer.configs!acks = all|1`
- `[service].kafka.producer.configs!retries = > 0`

At most once

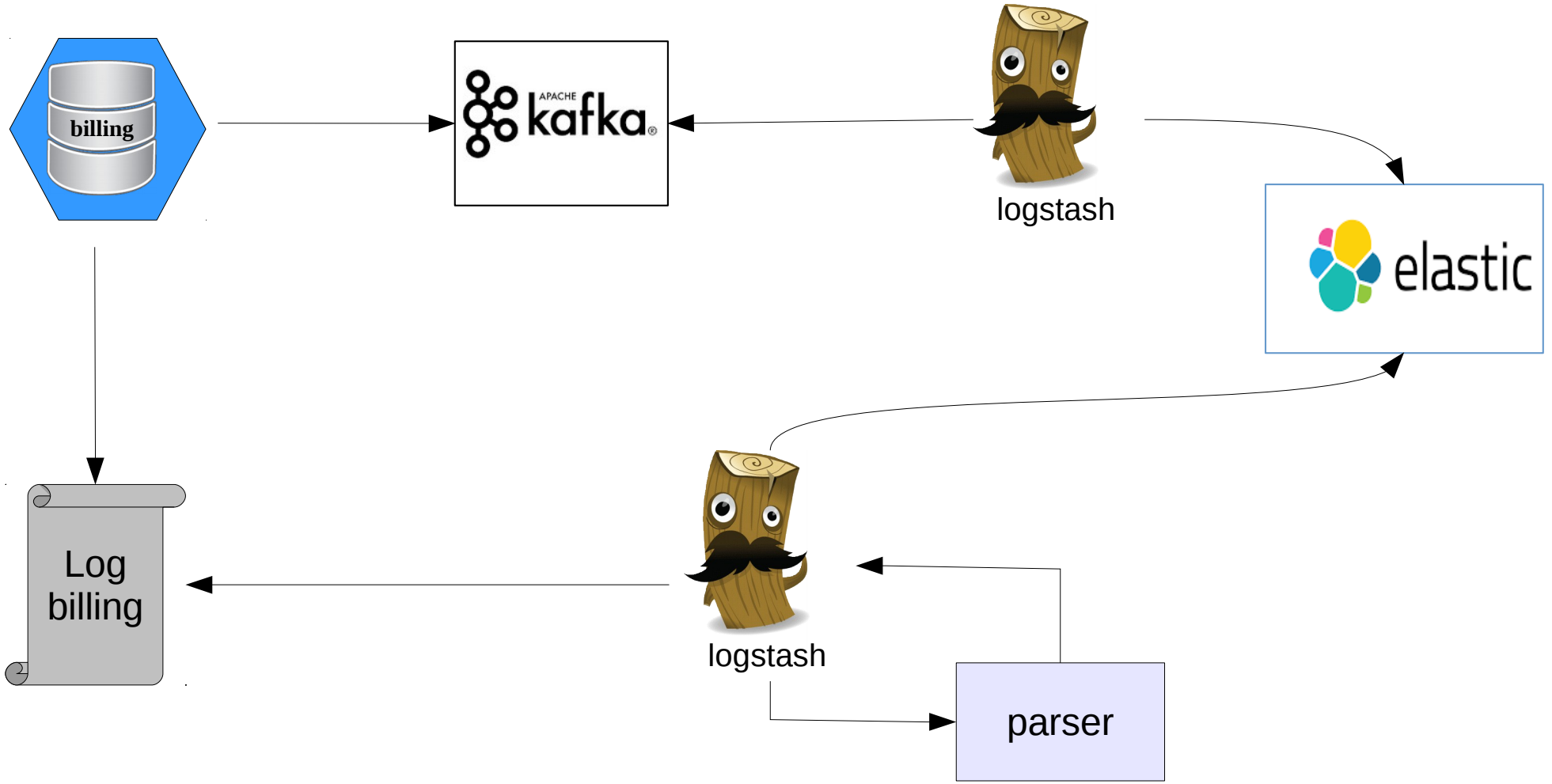
- The number of acknowledgments the producer requires the leader to have received before considering a request complete. Default value is 1.

Setting	Description	Risk of Data loss	Performance
ACKS = 0	No acknowledgment from the server at all	Highest	Highest
ACKS = 1	Leader completes write data	Medium	Medium
ACKS = all	All leaders and followers have written the data	Lowest	Lowest

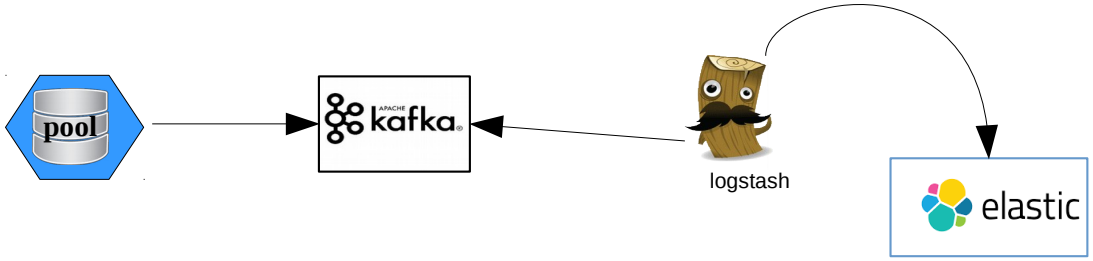
Tuning kafka producer

- **Tuning properties for a specific service**
  - [service].kafka.producer.configs = Configuration for Kafka Producer
  - [service].kafka.producer.configs!max.block.ms = 1000
  - [service].kafka.producer.configs!retries = 1
  - [service] = nfs, ftp, webdav, pool, dcap , xrootd

Kafka with dCache



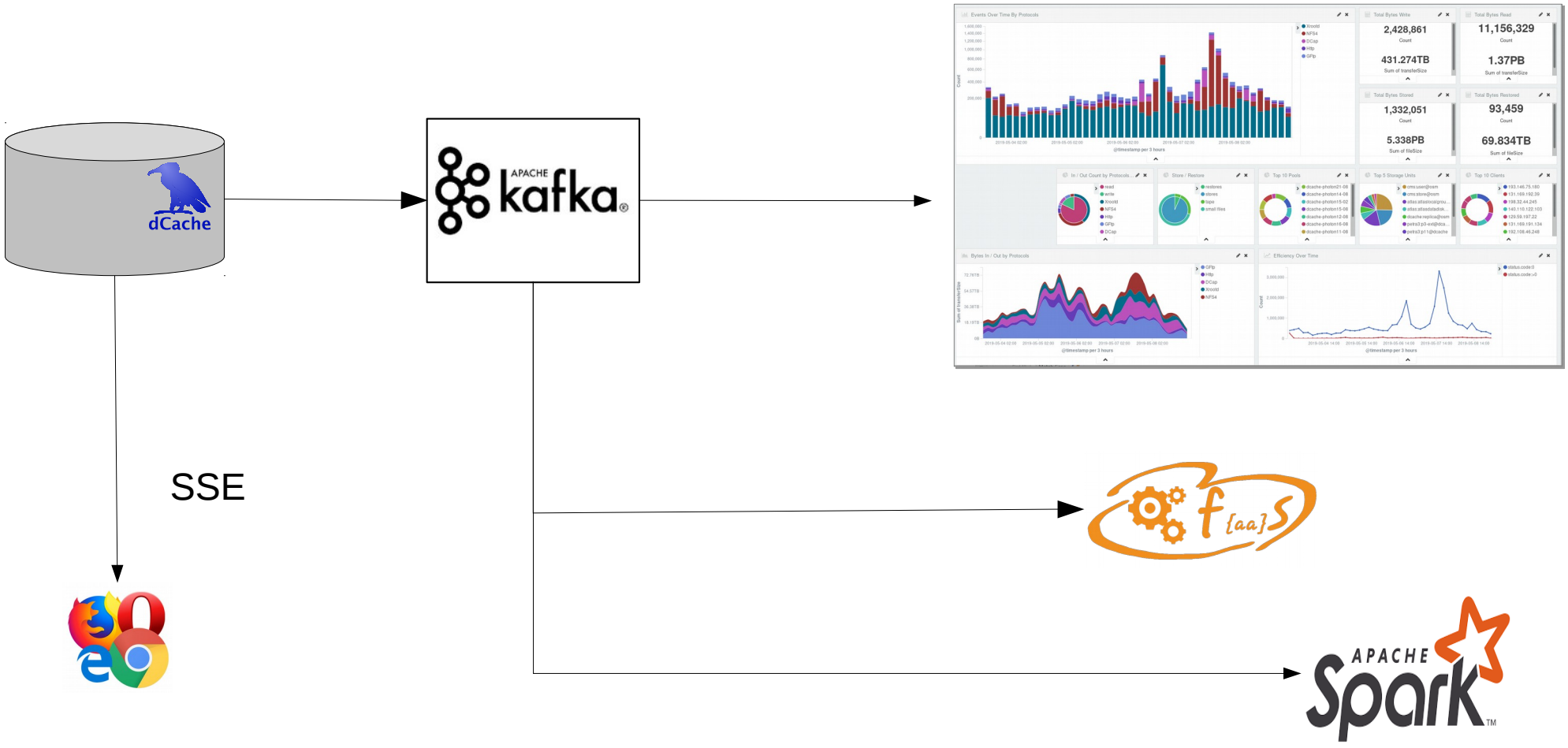
Kafka with dCache



Example/write to storag


```
{
"date": "Fri May 11 12:14:45 CEST 2018”,
"msgType": "store",
"transferTime": 1062,
"cellName": "pool_write",
"session": "pool:pool_write@dCacheDomain:1526033685223-22",
"subject": ("UidPrincipal(0)",
"GidPrincipal(0,primary)"),
"version": "1.0",
"storageInfo": "test:tape@osm",
"cellType": "pool",
"fileSize": 378,
"queuingTime": 1148,
"cellDomain": "dCacheDomain",
"pnfsid": "000003EBFAC026BB4521B8B68E7FE7734D9A",
"transaction": "pool:pool_write@dCacheDomain:1526033685223-22",
"billingPath": "/",
"status" :{
"msg": "",
"code": 0}
}
```

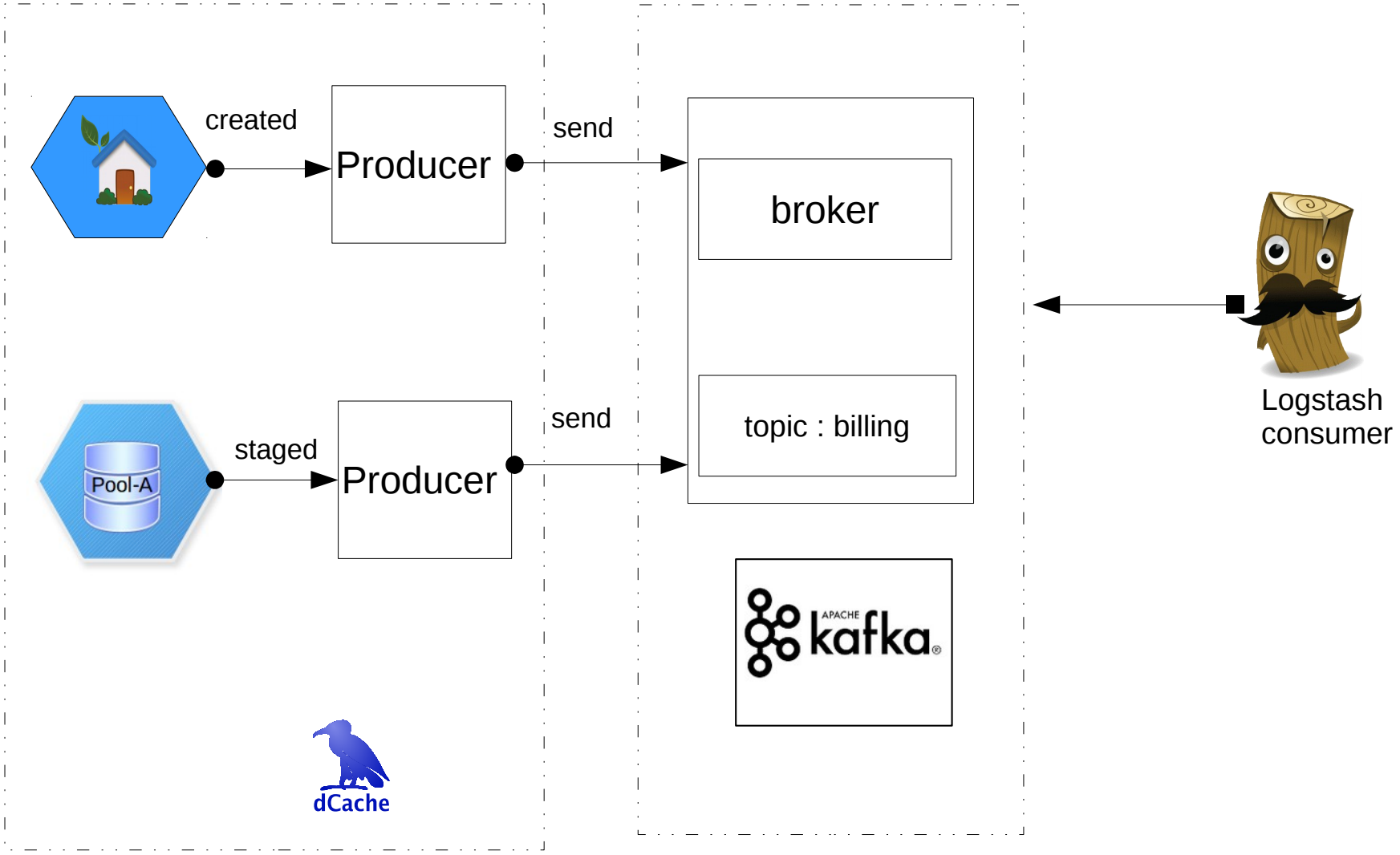
Workflow control



Example/alarm event


```
{
 "timestamp" : "2019-04-24T13:46:06.449Z",
 "level" : "ERROR",
 "thread" : "Thread-125",
 "logger" : "diskCacheV111.admin.UserAdminShell",
 "message" : "test error ",
 "context" : "default",
 "marker" : {
   "key" : "OUT_OF_FILE_DESCRIPTORS:pool_name",
   "firstArrived" : 1556113566449,
   "lastUpdate" : 1556113566449,
   "type" : "OUT_OF_FILE_DESCRIPTORS",
   "host" : "eduroam-1436.desy.de",
   "domain" : null,
   "service" : null,
   "info" : "test error ",
   "notes" : null,
   "closed" : false,
   "alarm" : true,
   "received" : null,
   "severity" : null,
   "formattedDateOfFirstArrival" : "Wed Apr 24 15:46:06 CEST 2019",
   "formattedDateOfLastUpdate" : "Wed Apr 24 15:46:06 CEST 2019"
 }
}
```

dCache as Kafka producer



dCache as Kafka producer