

# dCache storage events

Paul Millar  
on behalf of the dCache team

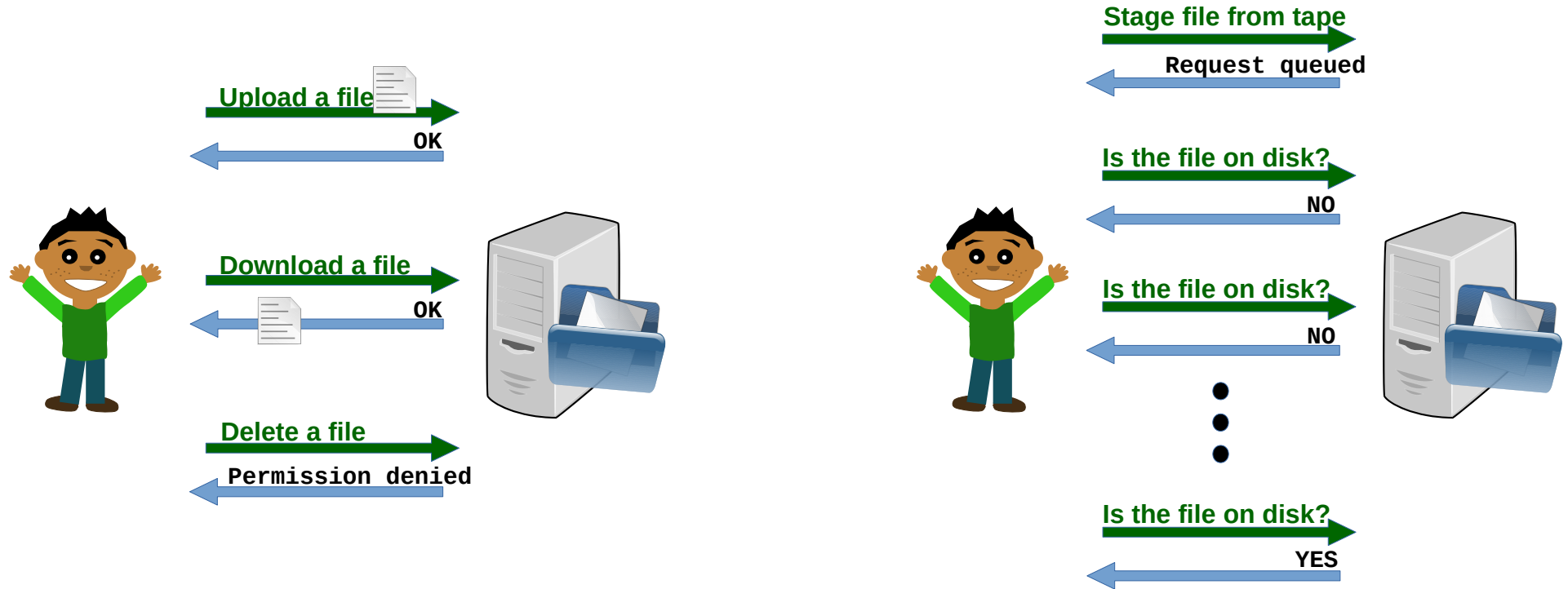
**ESCAPE dCache events mini-workshop**

2020-01-27

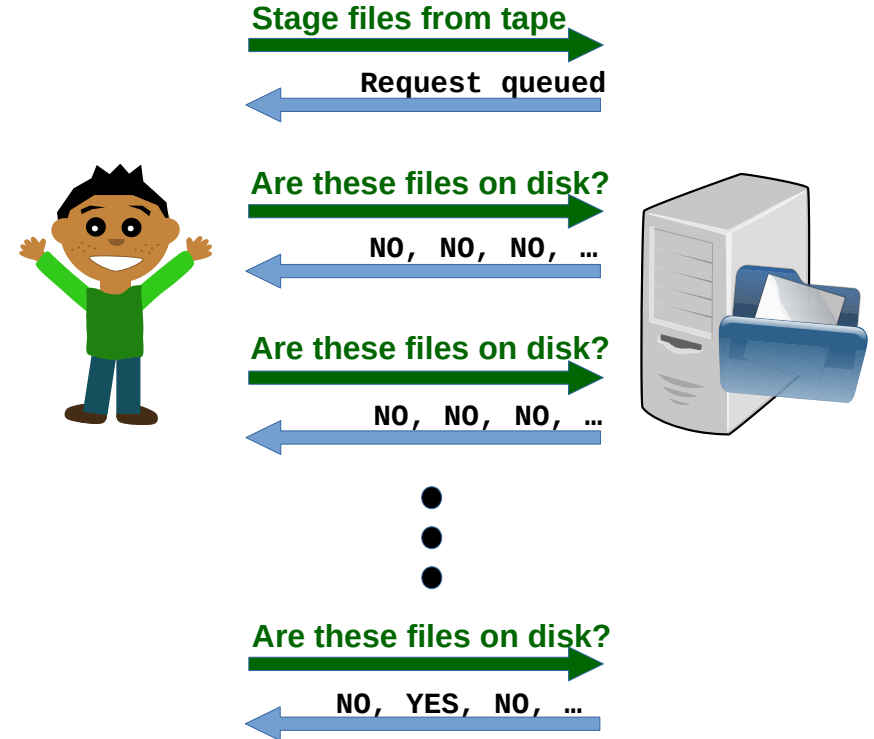
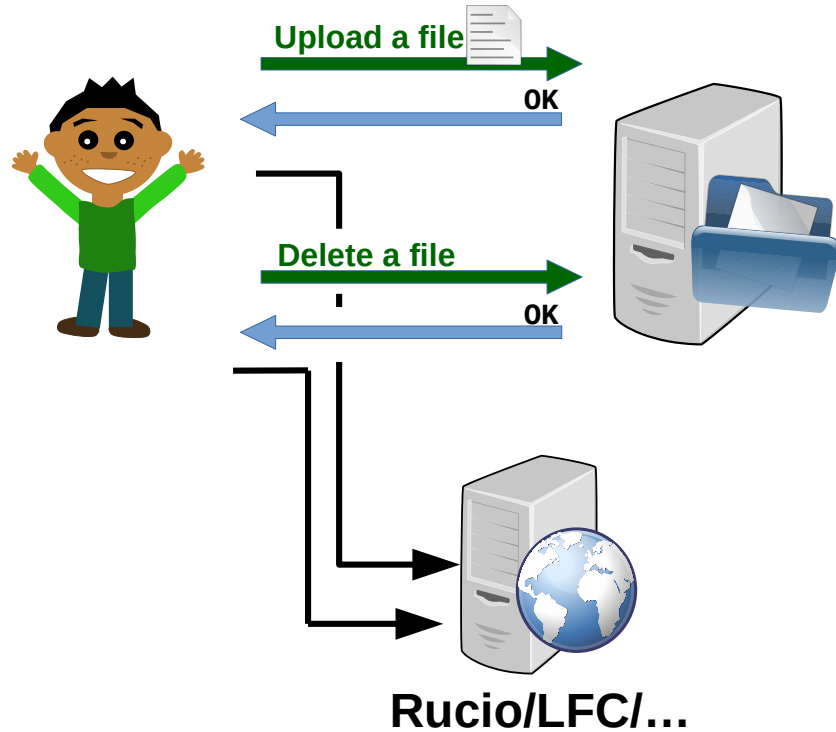
<https://indico.in2p3.fr/event/20525/>



# How storage is used currently



# The problems...



## Polling: watching for side-effects

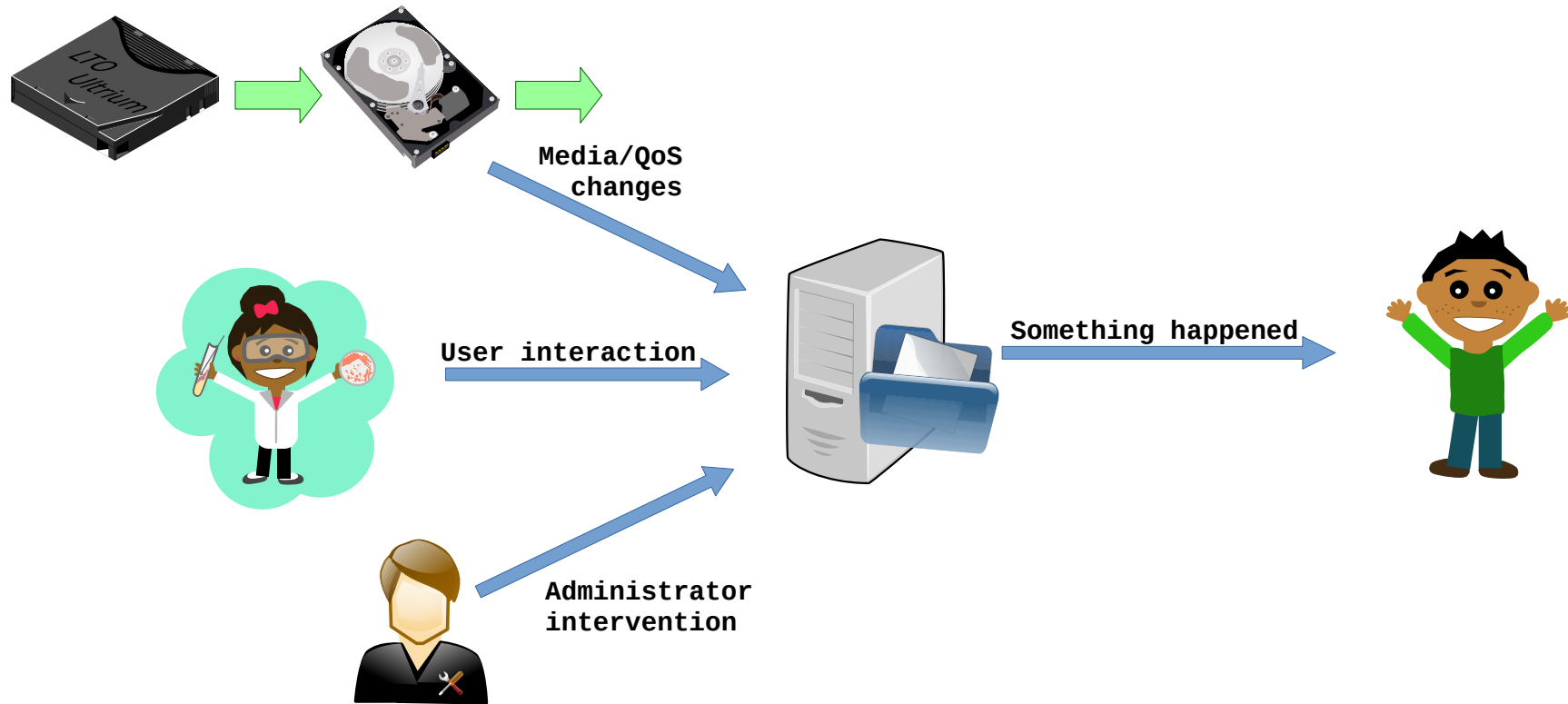
- Internally, dCache generates events, but these events do not propagate outside dCache.
- The only way of detecting changes is to look for **side-effects**:
  - Uploaded file → directory's mtime (or directory listing) changes,
  - A staged file → file locality changes to ONLINE.
- So, you can query the current status:
  - ... and again 5 seconds later,
  - ... and again 5 seconds later,
  - ... and again 5 seconds later,
  - ...

**This is polling**

## Polling is bad

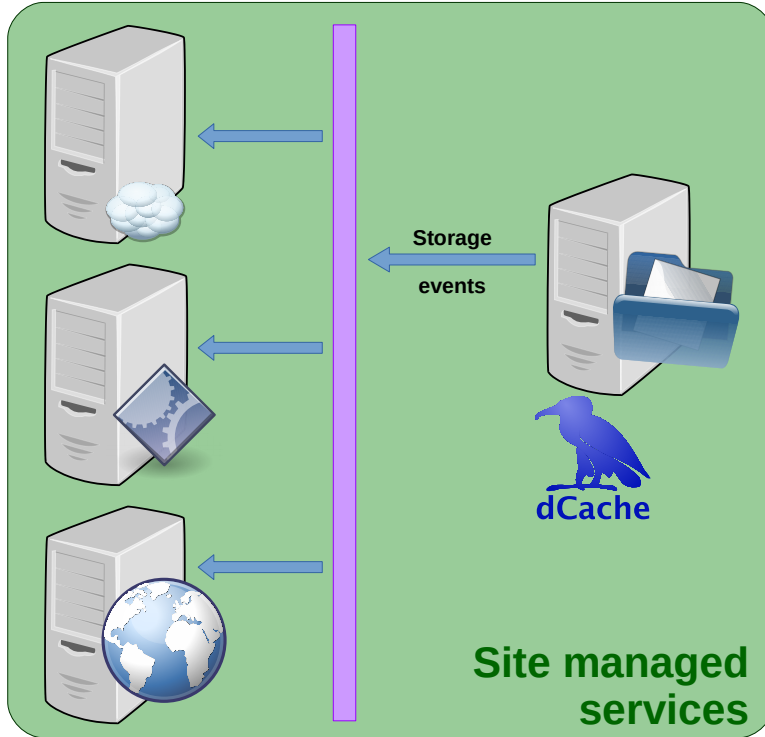


# New way of interacting: storage events

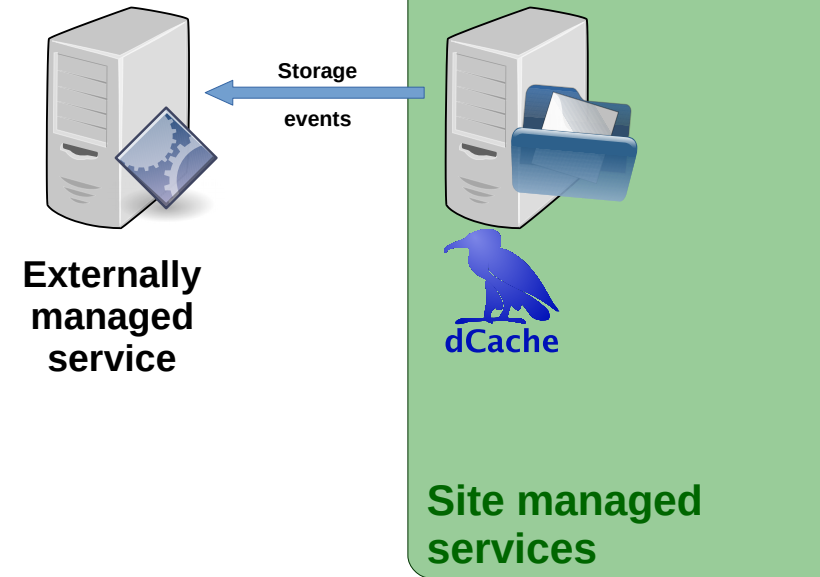


# dCache implementation

# dCache storage events: Kafka and SSE





Complementing, not competing solutions:  
different target audiences



W3C<sup>®</sup> SSE



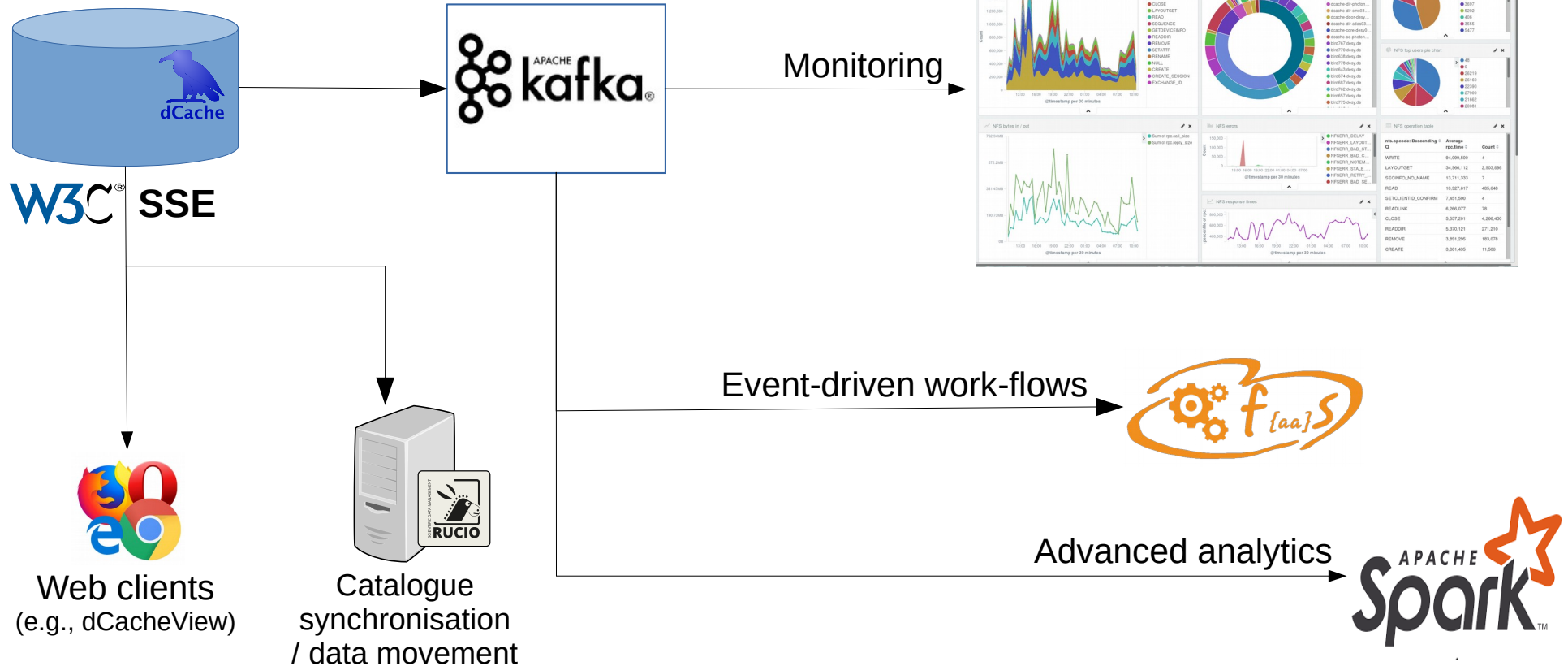
## Comparison: Kafka vs SSE

	 kafka	 SSE
<b>A standard ...</b>	software package	protocol
<b>What events does it see?</b>	dCache billing events	inotify
<b>Main benefit</b>	Easy integration	Built-in security
<b>“Catch-up” storage</b>	Memory & disk	Memory-only (currently)
<b>Target audience</b>	Site-level integration	Events for users

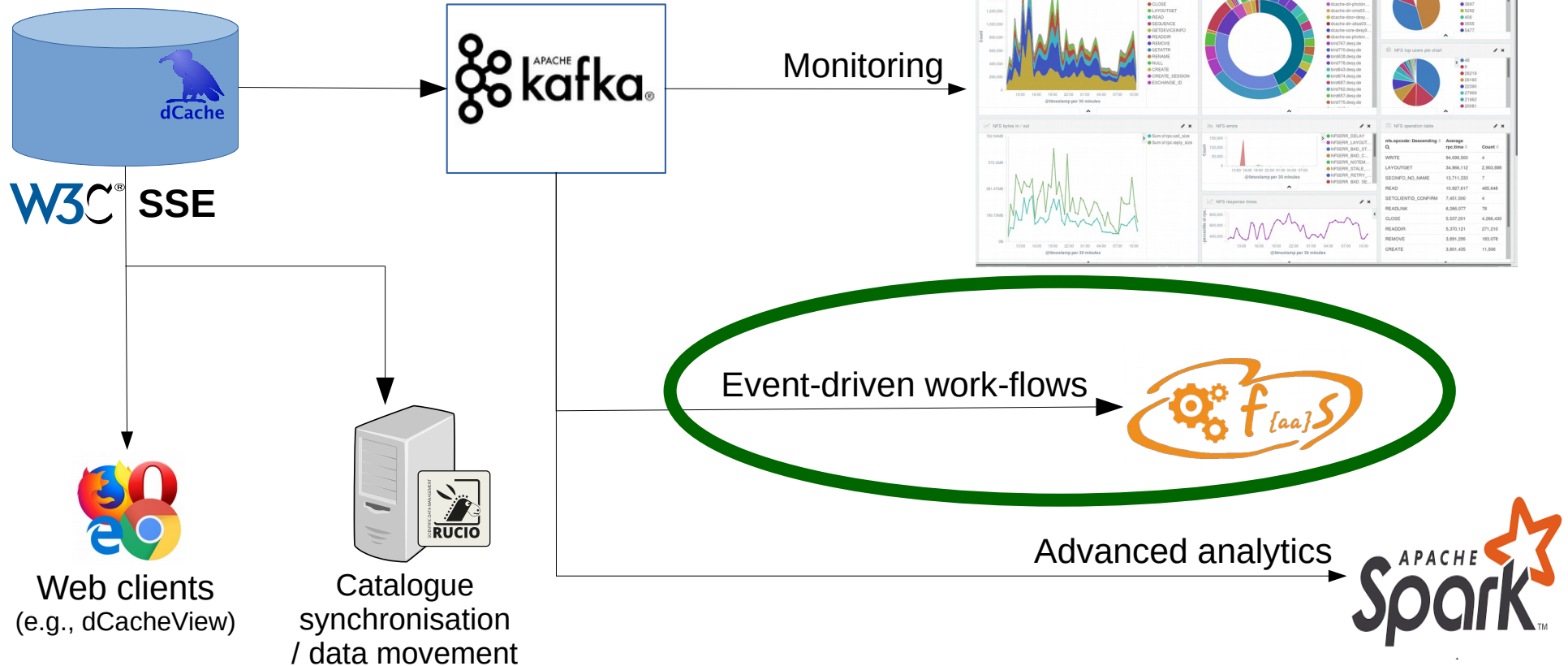
## Comparison: billing events vs inotify

Event	Billing records (kafka)	Inotify (SSE)
<b>File uploaded</b>	Door upload, Pool replica created	CREATE, OPEN, MODIFY... , ATTRIB, CLOSE_WRITE
<b>File read</b>	Door read, Pool replica delivered	OPEN, ACCESS..., CLOSE_NOWRITE
<b>File deleted</b>	Door delete, Pool replica removed	DELETE
<b>Directory created</b>	(nothing)	CREATE
<b>Directory deleted</b>	(nothing)	DELETE
<b>File/directory renamed</b>	(nothing)	MOVE_FROM and MOVE_TO
<b>File/directory moved</b>	(nothing)	MOVE_FROM and/or MOVE_TO
<b>File written to tape</b>	Pool flushed	ATTRIB
<b>File read back from tape</b>	Pool staged	ATTRIB
<b>Internal replica created</b>	Pool replica created	ATTRIB
<b>Internal replica deleted</b>	Pool replica removed	ATTRIB

# Using storage events



# Using storage events



# Thanks for listening

# Backup slides

# Pool replica created record

```
{
  "date": "2020-01-23T11:20:44.05+01:00",
  "writeActive": "PT0.005386747S",
  "msgType": "transfer",
  "transferTime": 17,
  "cellName": "dcache-atlas148-10",
  "session": "pool:dcac[...]11",
  "subject": [
    "FQANPrincipal[/atlas/lcg1]",
    "UidPrincipal[40001]",
    "GidPrincipal[4000,primary]",
    "Origin[...]",
    "/DC=ch/DC=cern/OU=Organic Units/OU=Users/...",
    "FQANPrincipal[/atlas/Role=production,primary]",
    "LoAPPrincipal[IGTF-AP:Classic]",
    "GroupNamePrincipal[atlasusr001]",
    "UserNamePrincipal[atlasusr001]",
    "GroupNamePrincipal[atlasusr001,primary]",
    "EmailAddressPrincipal[...]",
    "FQANPrincipal[/atlas]"
  ],
  "initiator": "door:GFT[...]0",
  "transferPath": "/upload/15/3[...]d",
  "version": "1.0",
  "storageInfo": "atlas:atlasdatadisk@osm",
  "transferSize": 5272605,
  "meanWriteBandwidth": 1061642810.2714789,
  "protocolInfo": {
    "protocol": "GFtp",
    "port": 52406,
    "host": "[...]",
    "versionMajor": 2,
    "versionMinor": 0
  },
  "cellType": "pool",
  "fileSize": 5272605,
  "queuingTime": 0,
  "cellDomain": "dcache-atlas148-10Domain",
  "isP2p": false,
  "pnfsid": "0000FAEAF22BA17C48FC91D4E6C0AB66BDA6",
  "writeIdle": "PT0.017285714S",
  "billingPath": "/pnfs/d[...]d",
  "isWrite": "write",
  "status": {
    "msg": "",
    "code": 0
  }
}
```

## Door read record

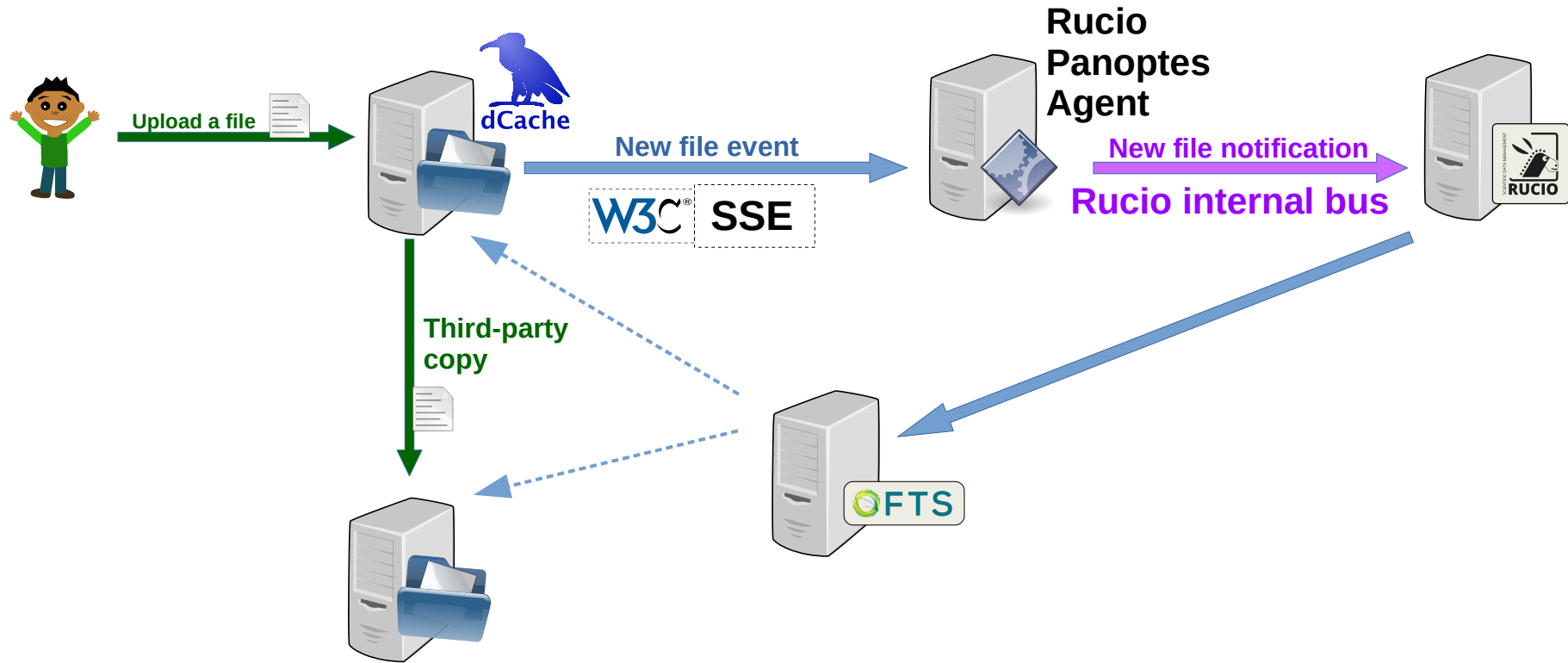
```
{
  "date": "2020-01-23T11:20:44.056+01:00",
  "owner": "/DC=ch/DC=cern/OU=Organic Units/...",
  "msgType": "request",
  "clientChain": "[IP address]",
  "mappedGID": 4000,
  "cellName": "GFTP-dcache-door-atlas13-AAWcy_5lvcA",
  "session": "door:GFTP-dcache-door[...]0",
  "subject": [
    "FQANPrincipal[/atlas/lcg1]",
    "UidPrincipal[40001]",
    "GidPrincipal[4000,primary]",
    "Origin[...]",
    "/DC=ch/DC=cern/OU=Organic Units/...",
    "FQANPrincipal[/atlas/Role=production,primary]",
    "LoAPrincipal[IGTF-AP:Classic]",
    "GroupNamePrincipal[atlasusr001]",
    "UserNamePrincipal[atlasusr001]",
    "GroupNamePrincipal[atlasusr001,primary]",
    "EmailAddressPrincipal[...]",
    "FQANPrincipal[/atlas]"
  ],
  "transferPath": "/upload/15/390ea6d9-6d10-488d-aa...",
  "sessionDuration": 62,
  "storageInfo": "atlas:atlasdatadisk@osm",
  "cellType": "door",
  "fileSize": 5272605,
  "mappedUID": 40001,
  "VERSION": "1.0",
  "queuingTime": 0,
  "cellDomain": "dcache-door-atlas13_gridftpDomain",
  "client": "[IP address]",
  "pnfsid": "0000FAEAF22BA17C48FC91D4E6C0AB66BDA6",
  "billingPath": "/pnfs/desy.de/atlas/dq2/...",
  "status": {
    "msg": "",
    "code": 0
  }
}
```



## Door remove record

```
{
  "date": "2020-01-23T11:38:12.956+01:00",
  "owner": "/DC=ch/DC=cern/OU=Organic Units/[...]",
  "msgType": "remove",
  "clientChain": "unknown",
  "mappedGID": 4000,
  "cellName": "xrootd-wan-dcache-door-atlas15",
  "session": "door:xrootd-wa[...]",
  "subject": [
    "FQANPrincipal[/atlas/lcg1]",
    "EmailAddressPrincipal[...]",
    "UidPrincipal[40001]",
    "GidPrincipal[4000,primary]",
    "Origin[...]",
    "FQANPrincipal[/atlas/Role=production,primary]",
    "/DC=ch/DC=cern/OU=Organic Units/OU=Users/...",
    "LoAPPrincipal[IGTF-AP:Classic]",
    "GroupNamePrincipal[atlasusr001]",
    "UserNamePrincipal[atlasusr001]",
    "GroupNamePrincipal[atlasusr001,primary]",
    "FQANPrincipal[/atlas]",
    "FQANPrincipal[/atlas/usatlas]"
  ],
  "transferPath": "/pnfs/desy.de/a[...]",
  "sessionDuration": 0,
  "cellType": "door",
  "fileSize": 0,
  "mappedUID": 40001,
  "VERSION": "1.0",
  "queuingTime": 0,
  "cellDomain": "dcache-door-atlas15_xrootd-wan-Domain",
  "client": "[...]",
  "pnfsid": "0000F9DA5227B44445FBBFBC4EDD09526C1A",
  "billingPath": "/pnfs/desy.d[...].txt",
  "status": {
    "msg": "",
    "code": 0
  }
}
```

# Rucio automatic registration (SSE)

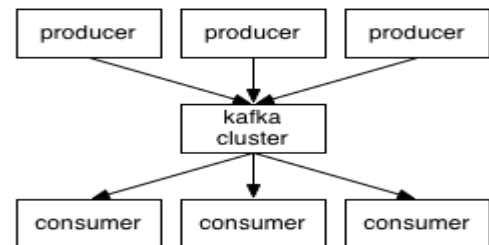


# Storage events in dCache

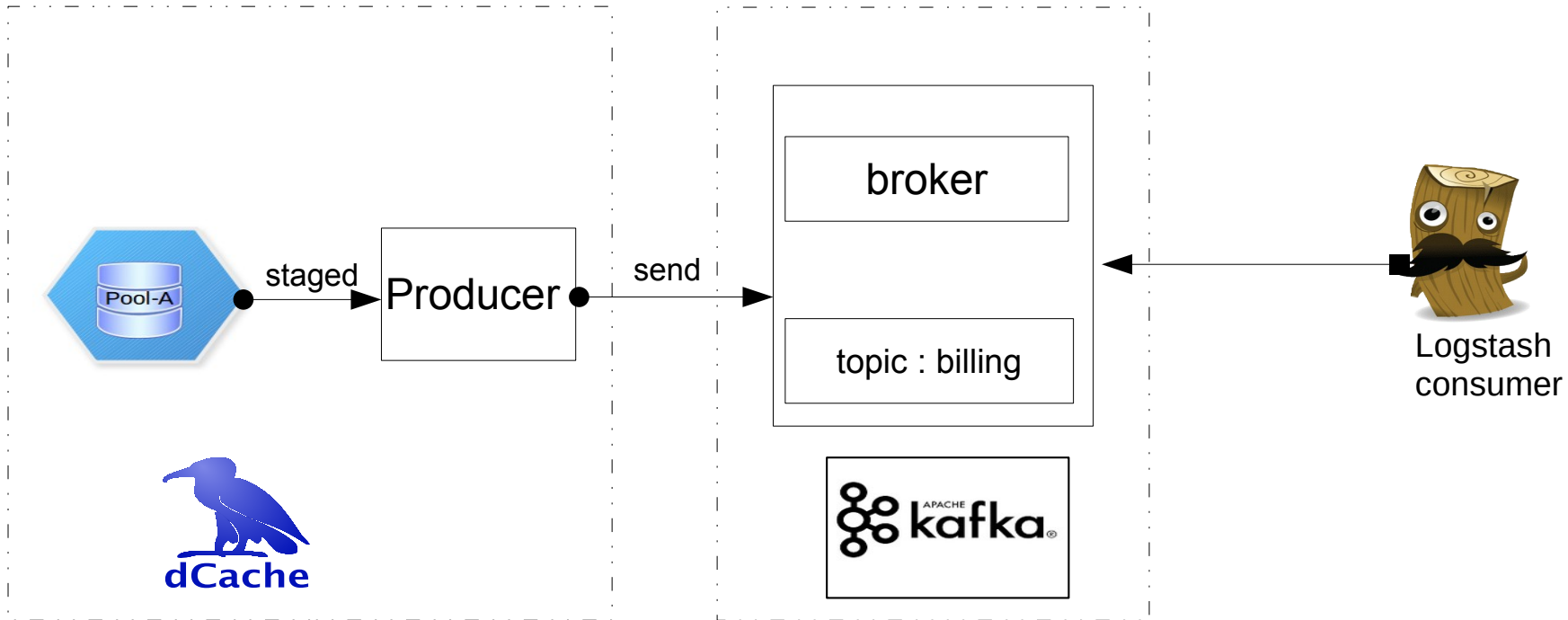
- Kafka stream
  - Producer-consumer model
  - Kafka consumer is required
  - Global events
  - Consumer keeps track of the last seen event
  - Integration with other tools (Spark, ELK, ...)
- Server-Sent Events (SSE)
  - Producer-consumer model
  - HTTP connection “for receiving push notifications from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

# Storage events in dCache

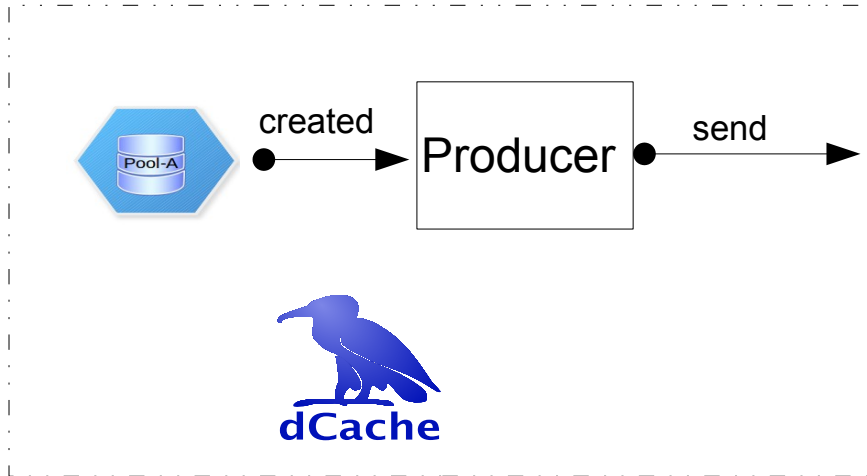
- Kafka stream
  - Producer-consumer model
  - Kafka consumer is required
  - Global events
  - Consumer keeps track of the last seen event
  - Integration with other tools (Spark, ELK, ...)
- Server-Sent Events (SSE)
  - Producer-consumer model
  - HTTP connection “for receiving push notifications from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



# dCache as Kafka producer



# dCache as Kafka producer

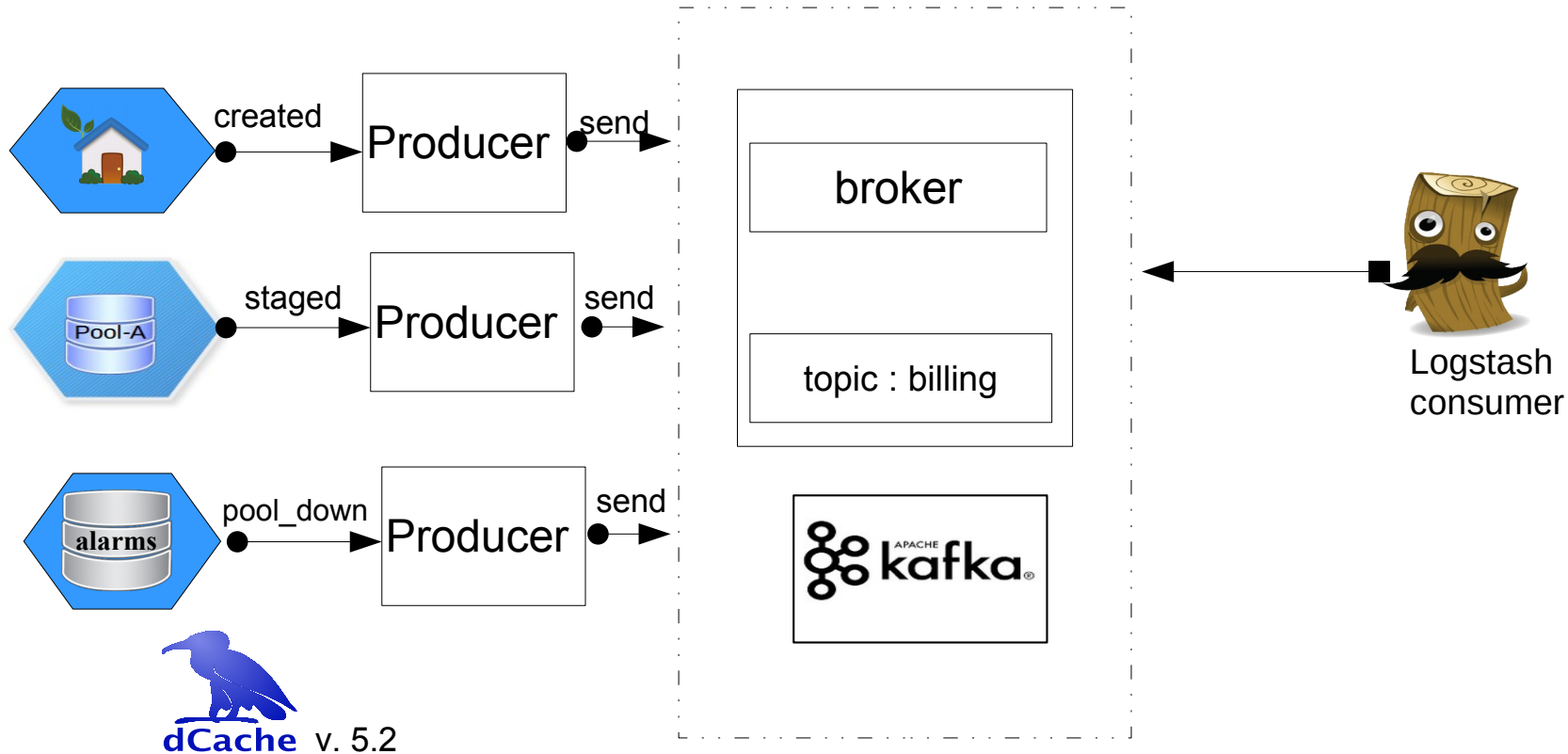


```
{
  "msgType": "transfer",
  "cellType": "pool",
  "meanWriteBandwidth": 9.751456E8,
  "isP2p": false,

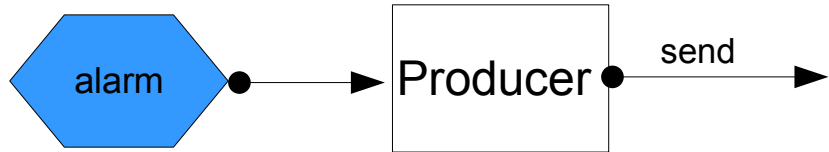
  "isWrite": "write",

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

# dCache as Kafka producer



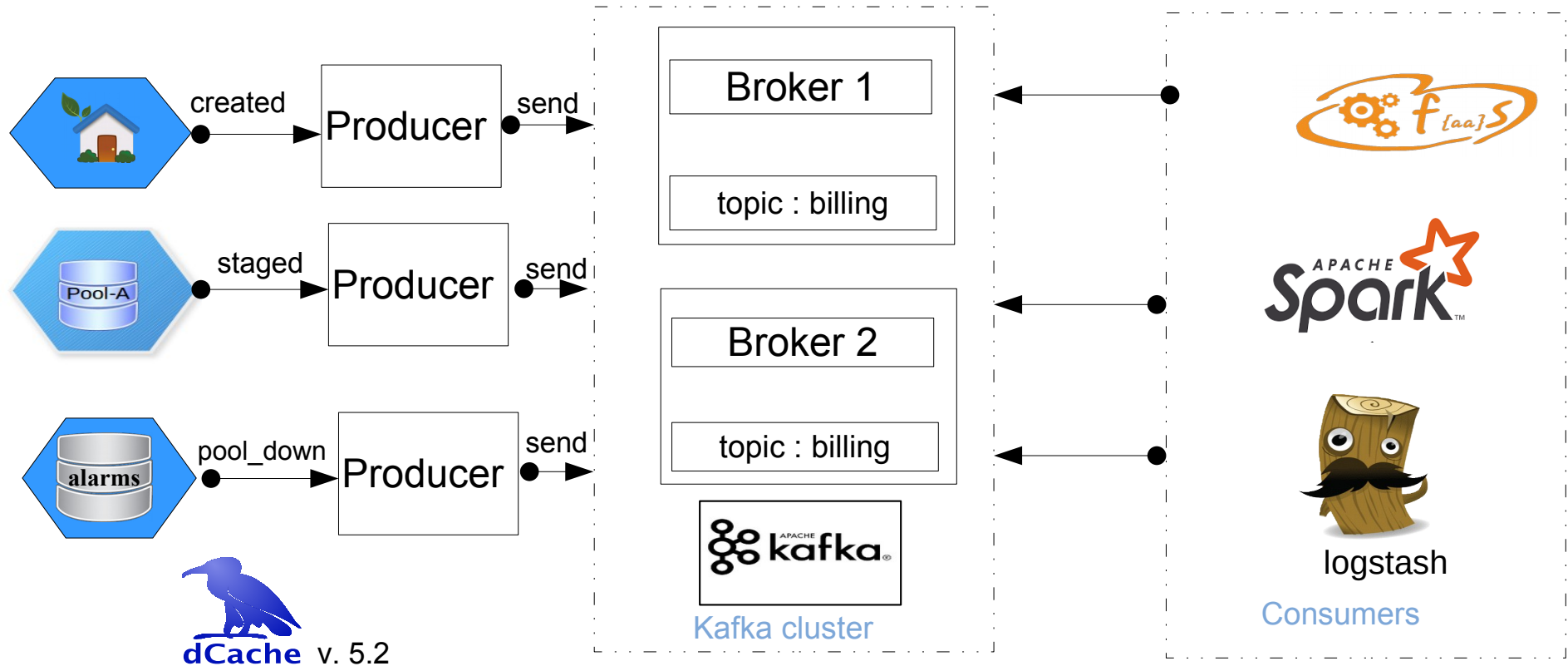
# dCache as Kafka producer



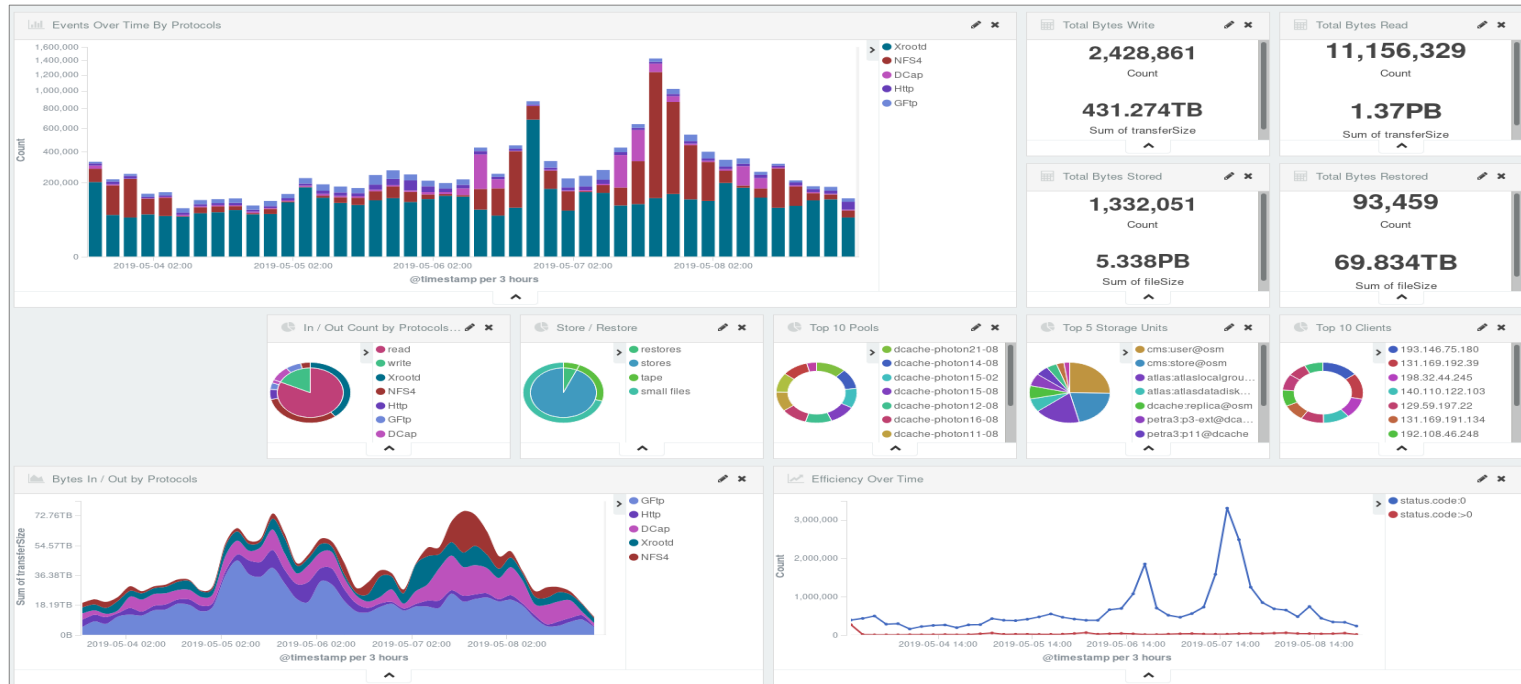
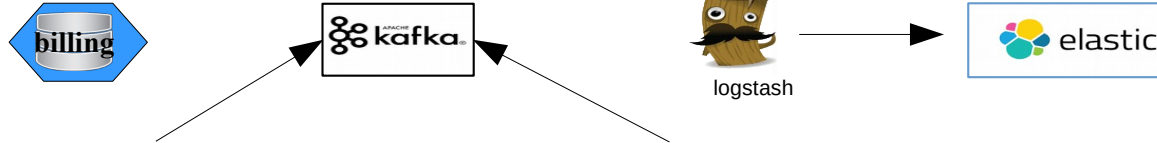
```
{
  "timestamp" : "2019-04-24T13:46:06.449Z",
  "level" : "ERROR",
  "thread" : "Thread-125",
  "marker" : {
    "key" : "OUT_OF_FILE_DESCRIPTOR;pool_name",
    "firstArrived" : 1556113566449,
    "lastUpdate" : 1556113566449,
    "type" : "OUT_OF_FILE_DESCRIPTOR"
  }
}
```



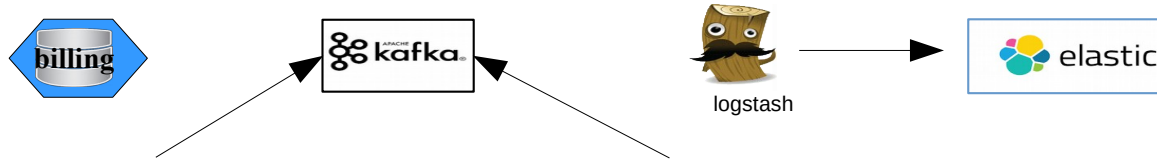
# dCache as Kafka producer



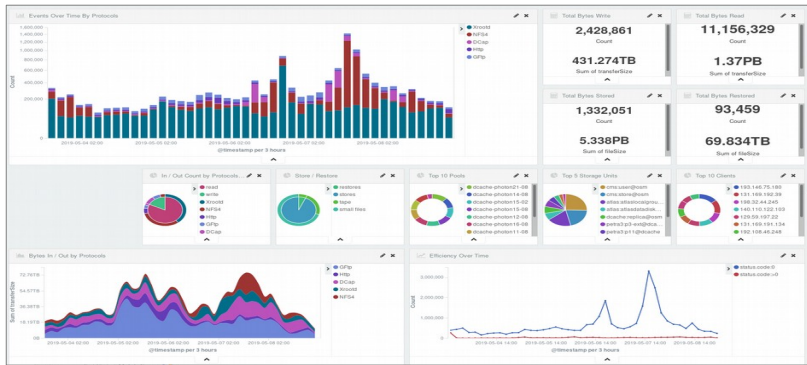
# Example for Monitoring with ELK



# 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")
  }
}
```

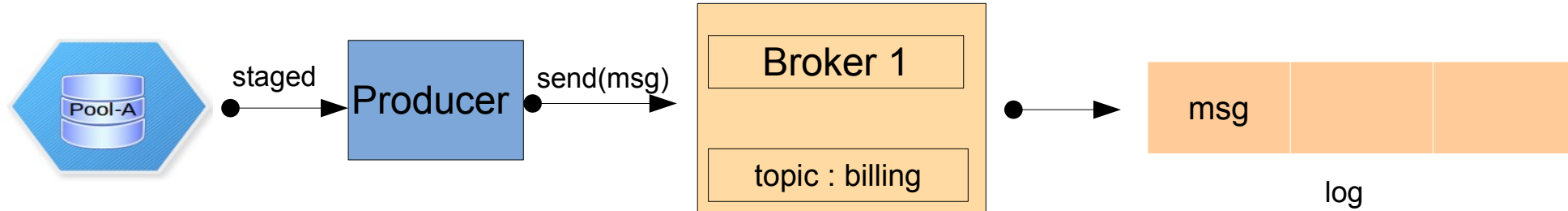
# Enabling Kafka

- 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

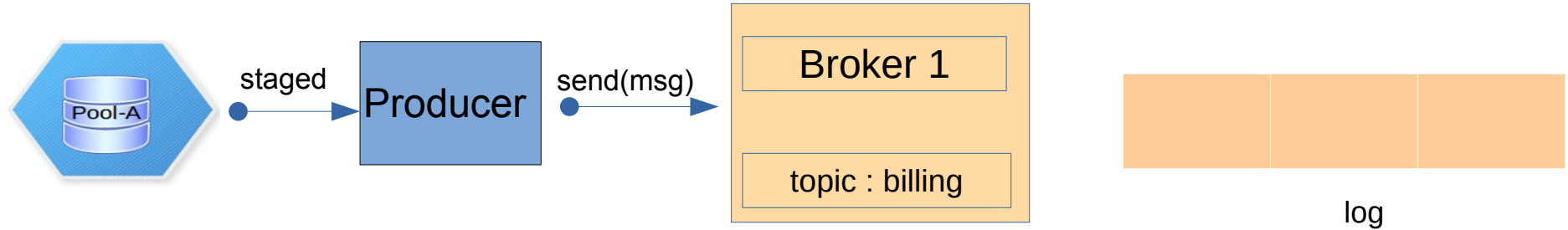
1. At most once
2. At least once
3. Exactly once

# 1. At most once



- Producer does not retry when when no ack is received

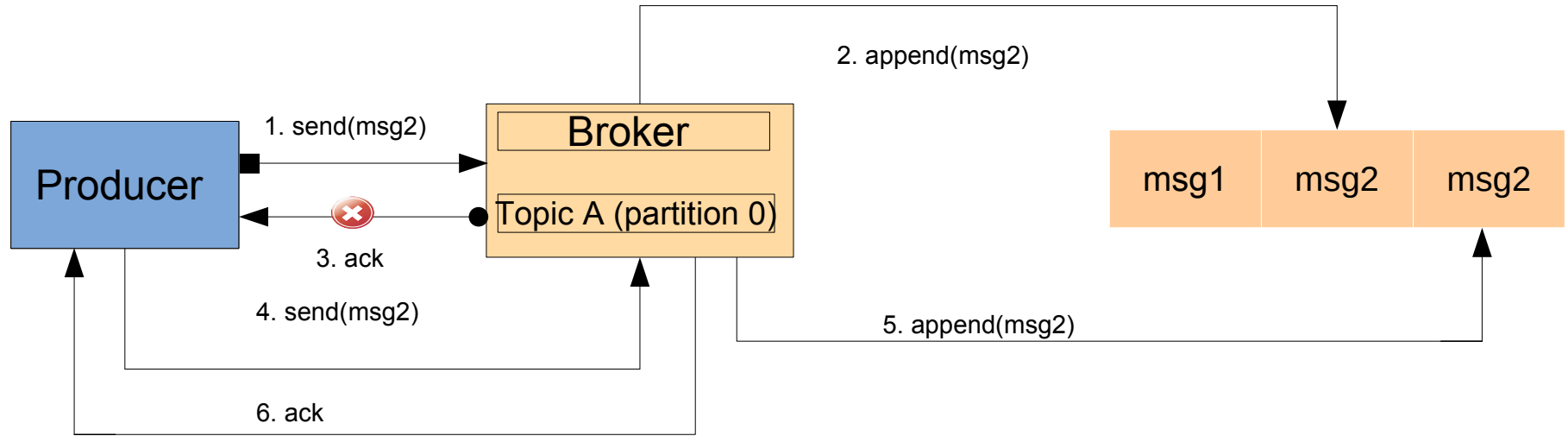
# 1. At most once



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

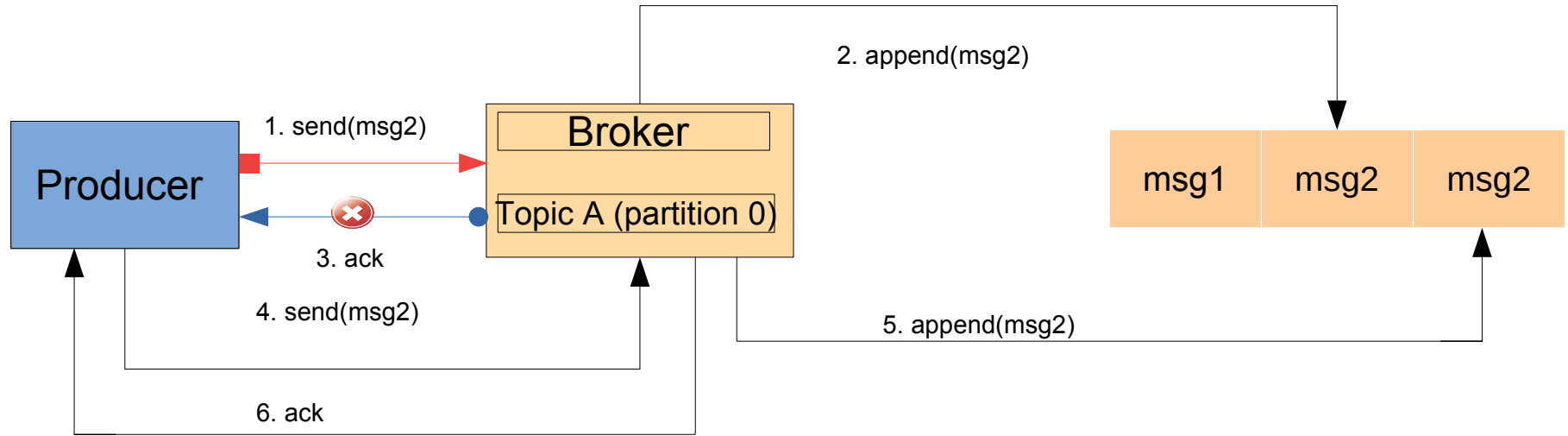


## 2. At least once



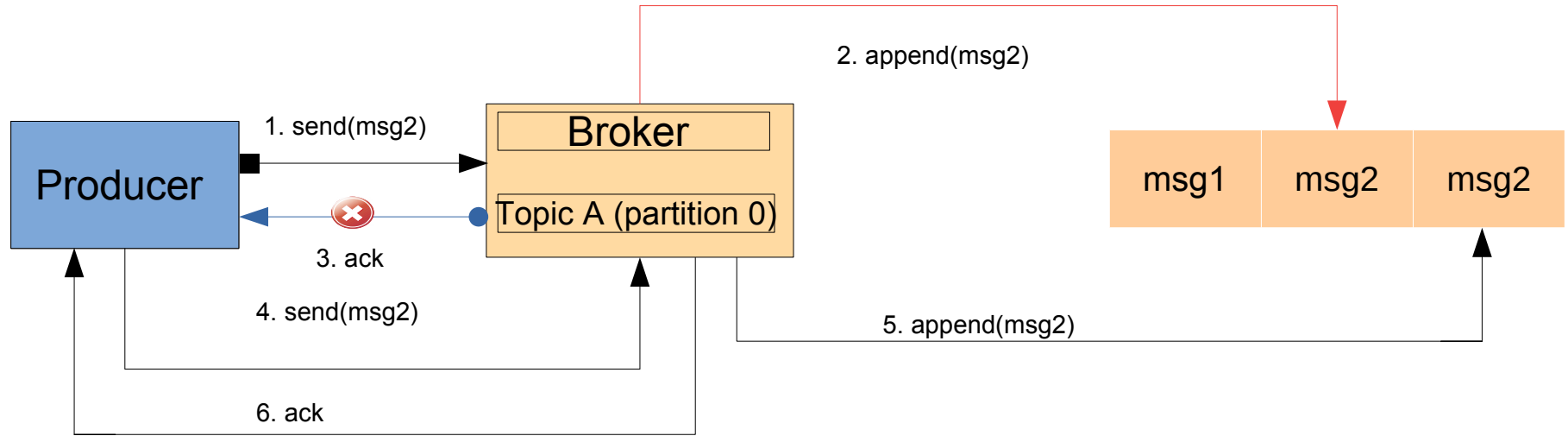
- Producer retries as long as it doesn't get ack
- Implications – duplicated messages

## 2. At least once



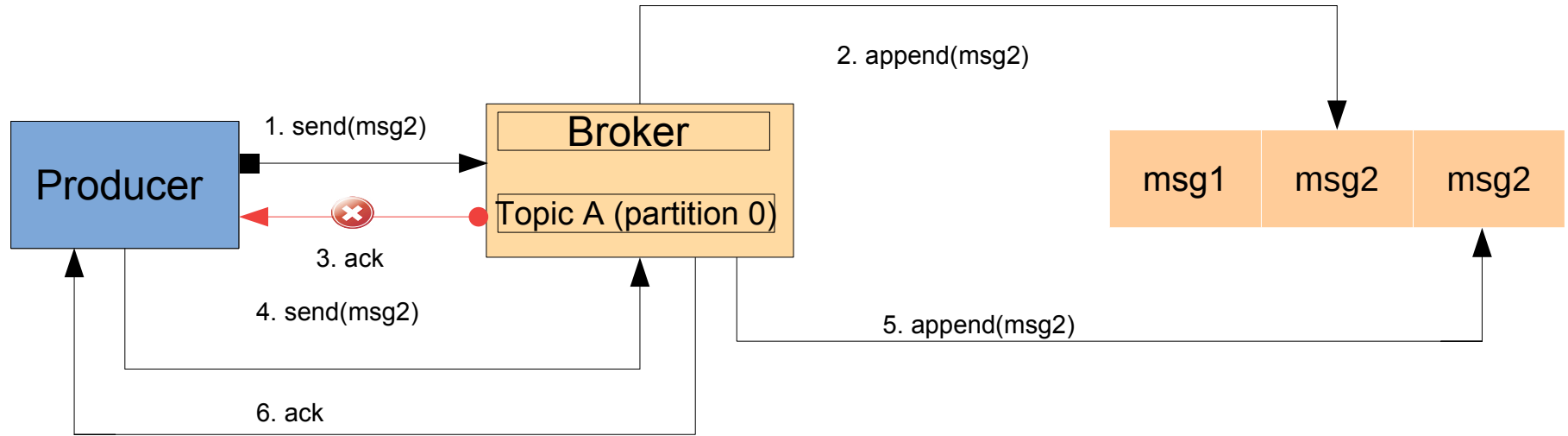
- Producer retries as long as it doesn't get ack
- Implications – duplicated messages

## 2. At least once



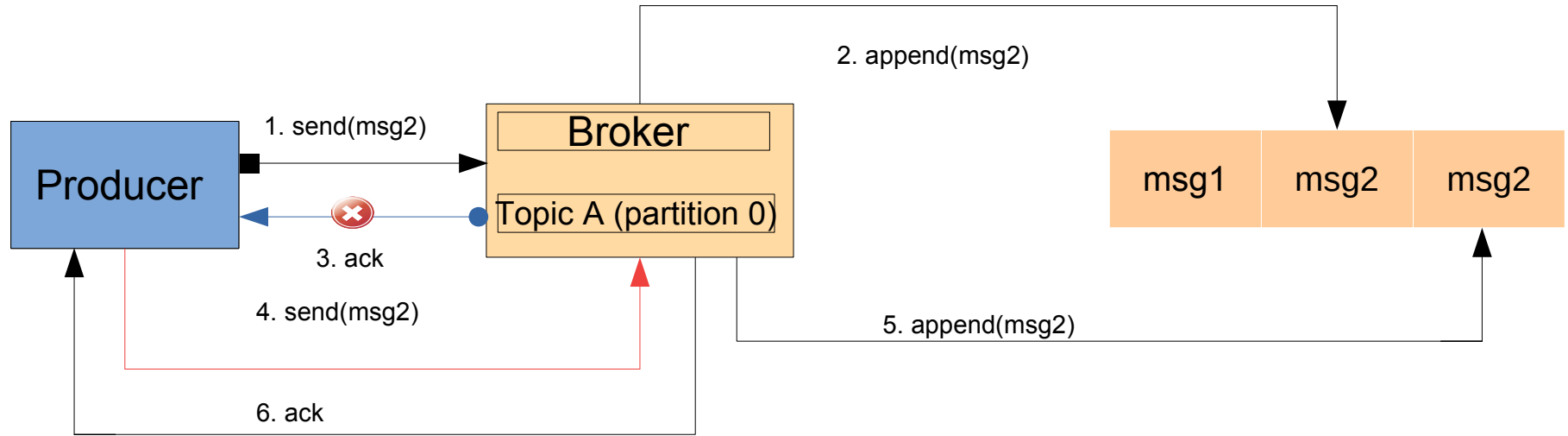
- Producer retries as long as it doesn't get ack
- Implications – duplicated messages

## 2. At least once



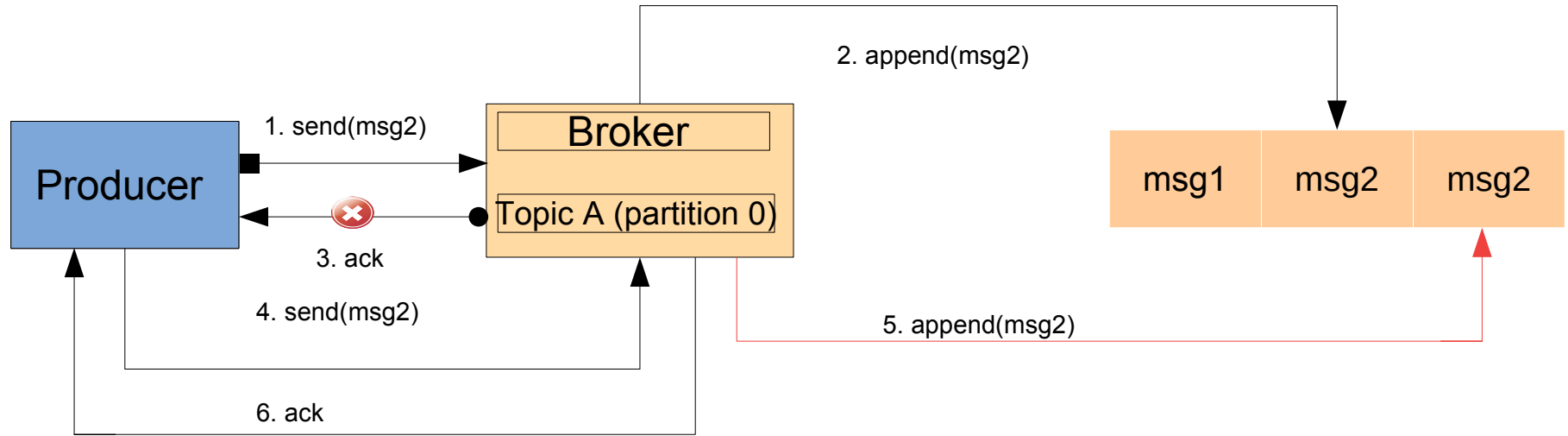
- Producer retries as long as it doesn't get ack
- Implications – duplicated messages

## 2. At least once



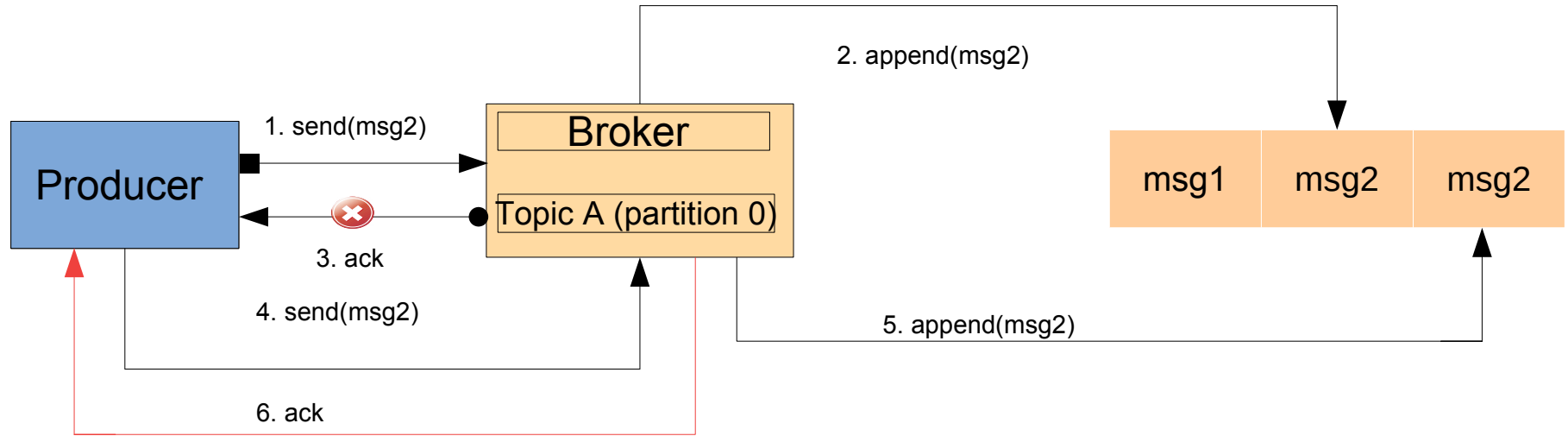
- Producer retries as long as it doesn't get ack
- Implications – duplicated messages

## 2. At least once



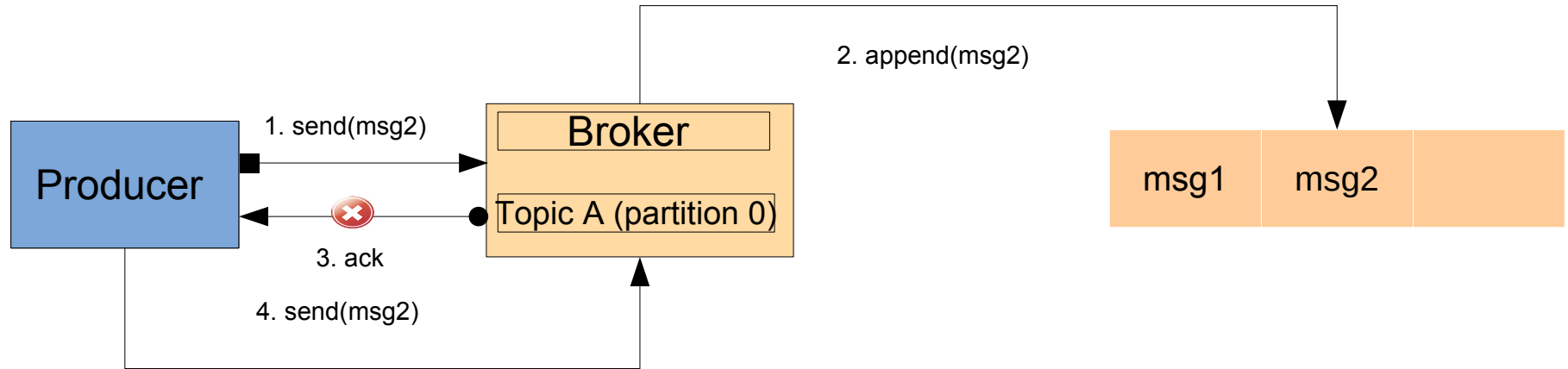
- Producer retries as long as it doesn't get ack
- Implications – duplicated messages

## 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
<i>At most once</i>	acks = 0	No	No	High	No
<i>At least once</i>	acks = all 1 retries > 0	Yes	Yes	Lower	Yes
<i>Exactly once</i>	acks = all 1 Retries > 0 enable.idempotence = true max.in.flight.requests. per.connection = 1	Yes	No	Lower	Yes

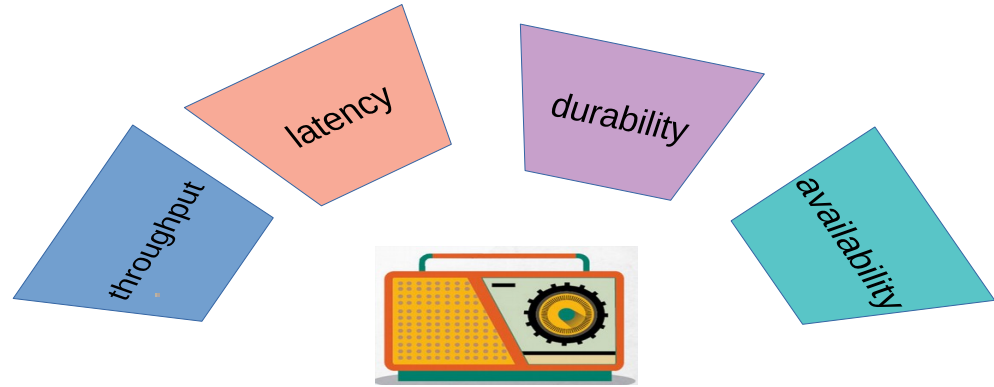
## At least/At most/Exactly once

	Properties in dCache	Delivery guaranty	Duplicates	Message throughput	Impact on dCache
<i>At most once</i>	acks = 0	No	No	High	No
<i>At least once</i>	acks = all 1 retries > 0	Yes	Yes	Lower	Yes
<i>Exactly once</i>	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>