What is Apache Kafka?

How it’s similar to the databases you know and love, and how it’s not.

Kenny Gorman
Founder and CEO

www.eventador.io
www.kennygorman.com
@kennygorman

eventador.io

I am a database nerd

I have done database foo for my whole career, going on 25 years.

Sybase, Oracle DBA, PostgreSQL DBA, MySQL aficionado, MongoDB early adopter, founded two companies based on data technologies

Broke lots of stuff, lost data before, recovered said data, stayed up many nights, on-call shift horror stories

Apache Kafka is really cool, as fellow database nerds you will appreciate it.

‘02 had hair ^

Now… lol

eventador.io

Kafka

Comparison with the databases you are familiar with

Kafka

Apache Kafka is an open-source ~~stream processing platform~~ pub/sub message platform developed by the Apache Software Foundation written in Scala and Java. The project aims *blah blah blah* pub/sub message queue architected as a distributed transaction log,"[3] Blah blah blah to process streaming data. *Blah blah blah*.

The design is heavily influenced by transaction logs.[4]

Why Kafka?

High Performance Streaming Data

Persistent

Distributed

Fault Tolerant

K.I.S.S.

Many Modern Use Cases

Pub/Sub Messaging Attributes

- It’s a stream of data. A boundless stream of data.

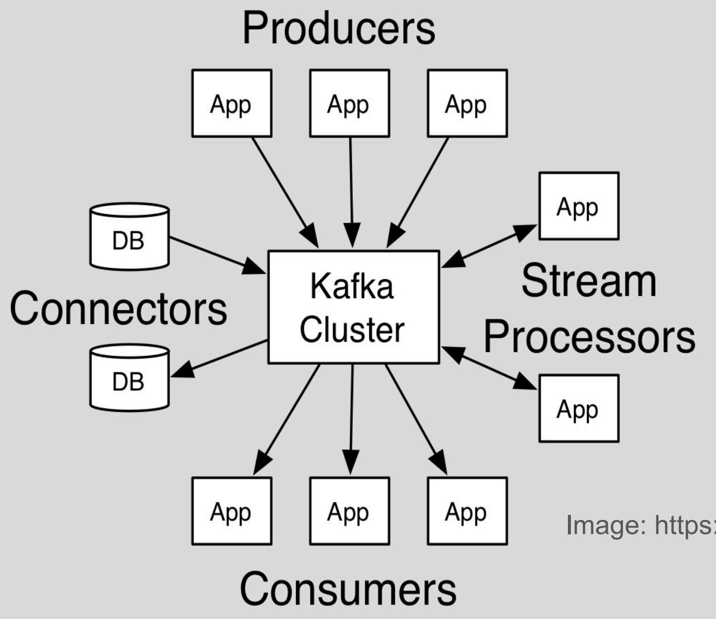
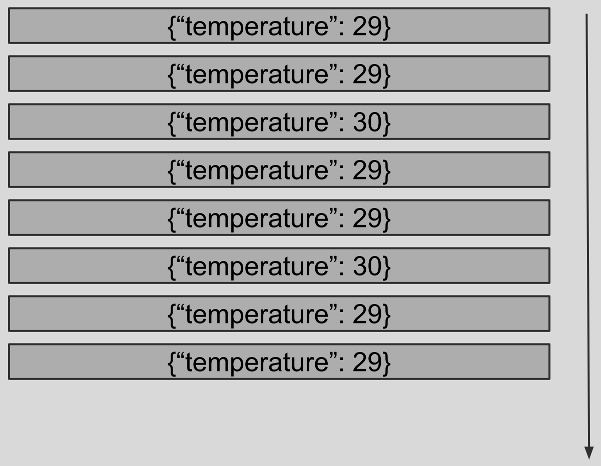


Image: https://kafka.apache.org



eventador.io

Logical Data Organization

PostgreSQL	**MongoDB**	**Kafka**
Database	Database	Topic Files
Fixed Schema	Non Fixed Schema	Key/Value Message
Table	Collection	Topic
Row	Document	Message
Column	Name/Value Pairs	
	Shard	Partition

eventador.io

Storage Architecture

PostgreSQL	**MongoDB**	**Kafka**
Stores data in files on disk	Stores data in files on disk	Stores data in files on disk
Has journal for recovery (WAL)	Has journal for recovery (Oplog)	**Is a commit log**
FS + Buffer Cache	FS for caching *	FS for caching
Random Access, Indexing	Random Access, Indexing	Sequential access

eventador.io

Topics

- Core to design of Kafka
- Partitioning
- Consumers and Consumer Groups
- Offsets ~= High Water Mark

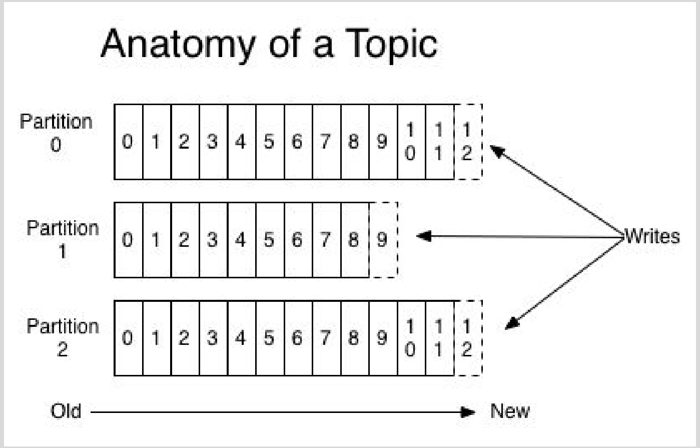


Image: https://kafka.apache.org

eventador.io

Performance

- Kafka topics are glorified distributed write ahead logs
- Append only
- k/v pairs where the key decides the partition it lives in
- Sendfile system call optimization
- Client controlled routing

eventador.io

Availability and Fault Tolerance

- **Topics are replicated among any number of servers (brokers)**
- **Topics can be configured individually**
- **Topic partitions are the unit of replication**

The unit of replication is the topic partition. Under non-failure conditions, each partition in Kafka has a single leader and zero or more followers.

MongoDB	Majority Consensus (Raft-like in 3.2)
Kafka	ISR set vote, stored in ZK

eventador.io

Application Programming Interfaces

	PostgreSQL	MongoDB	Kafka

Insert	`sql = "insert into mytable .."`
	`db.execute(sql)`
	`db.commit()`
		`db.mytable.save({"baz":1})`	`producer.send("mytopic", "{'baz':1}")`

Query	`sql = "select * from …"`
	`cursor = db.execute(sql)`
	`for record in cursor:`
	`    print record`
		`db.mytable.find({"baz":1})`	`consumer = get_from_topic("mytopic")`
			`for message in consumer:`
			`    print message`

Update	`sql = "update mytable set .."`
	`db.execute(sql)`
	`db.commit()`
		`db.mytable.update({"baz":1, "baz":2})`	

Delete	`sql = "delete from mytable .."`
	`db.execute(sql)`
	`db.commit()`
		`db.mytable.remove({"baz":1})`	

eventador.io

Typical RDBMS

```
conn = database_connect()
cur = conn.cursor(cursor_factory=psycopg2.extras.RealDictCursor)
cur.execute(
    """
        SELECT a.lastname, a.firstname, a.email,
        a.userid, a.password, a.username, b.orgname
        FROM users a, orgs b
        WHERE a.orgid = b.orgid
        AND a.orgid = %(orgid)s
    """, {"orgid": orgid}
)
results = cur.fetchall()

for result in results:
    print result
```

eventador.io

Publishing

```
from kafka import KafkaProducer

producer = KafkaProducer(bootstrap_servers='localhost:1234')

for _ in range(100):
    producer.send('foobar', b'some_message_bytes')
```

- Flush frequency/batch
- Partition keys

eventador.io

Subscribing (Consume)

```
from kafka import KafkaConsumer

consumer = KafkaConsumer(bootstrap_servers='localhost:9092')
consumer.subscribe('my-topic')

for msg in consumer:
    print (msg)
```

eventador.io

Subscribing (Consume)

```
try:
    msg_count = 0
    while running:
        msg = consumer.poll(timeout=1.0)
        if msg is None: continue

        msg_process(msg) # application-specific processing
        msg_count += 1
        if msg_count % MIN_COMMIT_COUNT == 0:
            consumer.commit(async=False)
finally:
    # Shut down consumer
    consumer.close()
```

- Continuous ‘cursor’
- Offset management
- Partition assignment

eventador.io

Tooling

- No simple command console like psql or mongo shell
- BOFJCiS
- Kafkacat, jq
- Shell scripts, mirrormaker, etc.
- PrestoDB

Settings and Tunables

PostgreSQL:

- Shared Buffers
- WAL/recovery

MongoDB (mmapv2)

- directoryPerDB
- FStuning

Kafka:

- Xmx ~ 90% memory
- log.retention.hours

eventador.io

Contact

https://kafka.apache.org/documentation

We are hiring!

www.eventador.io
@kennygorman

eventador.io