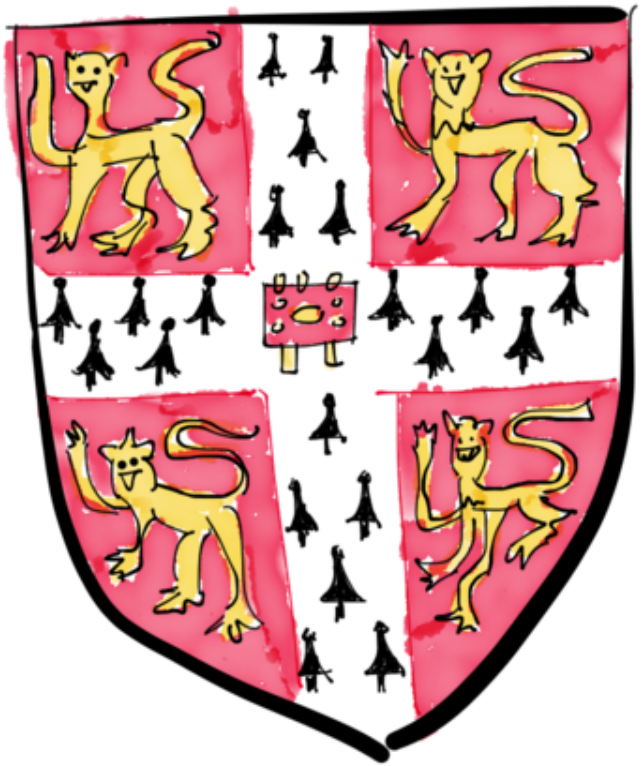
STAYING IN SYNC.

from TRANSACTIONS to STREAMS

@martinkl

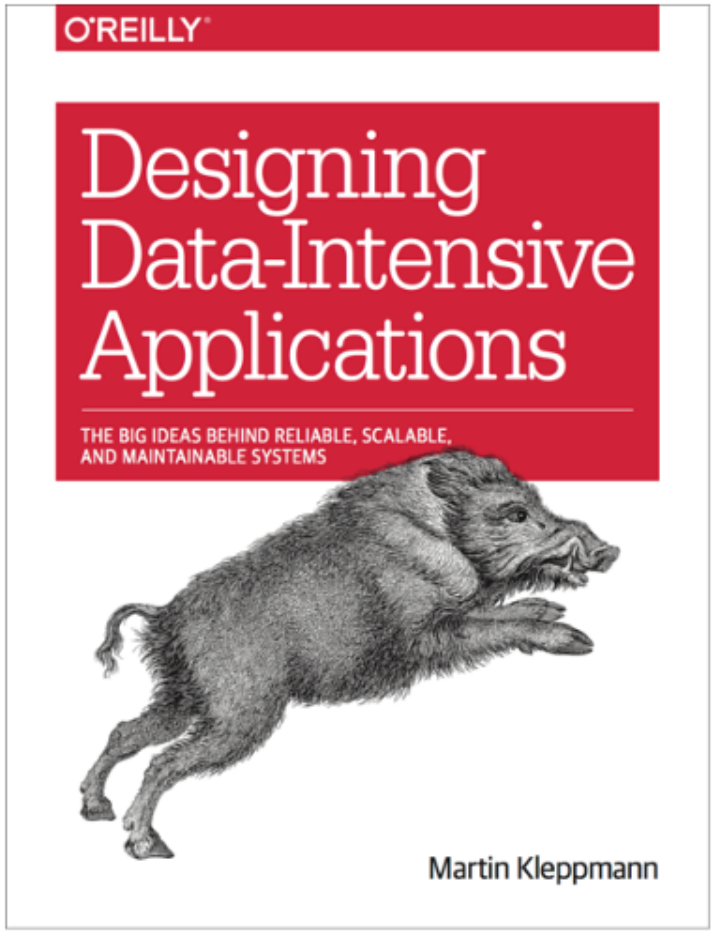
martinkl.com

/qcon

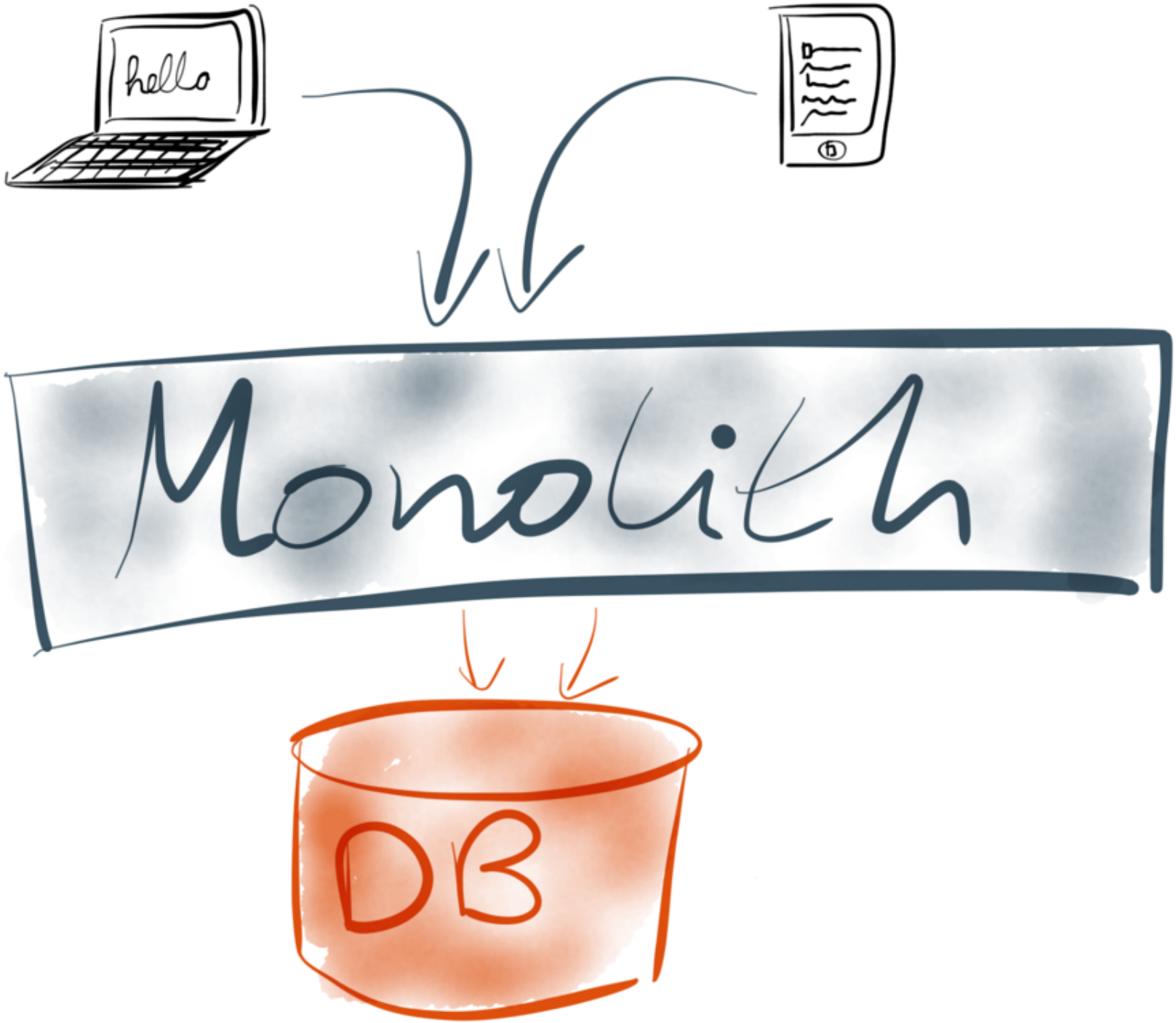


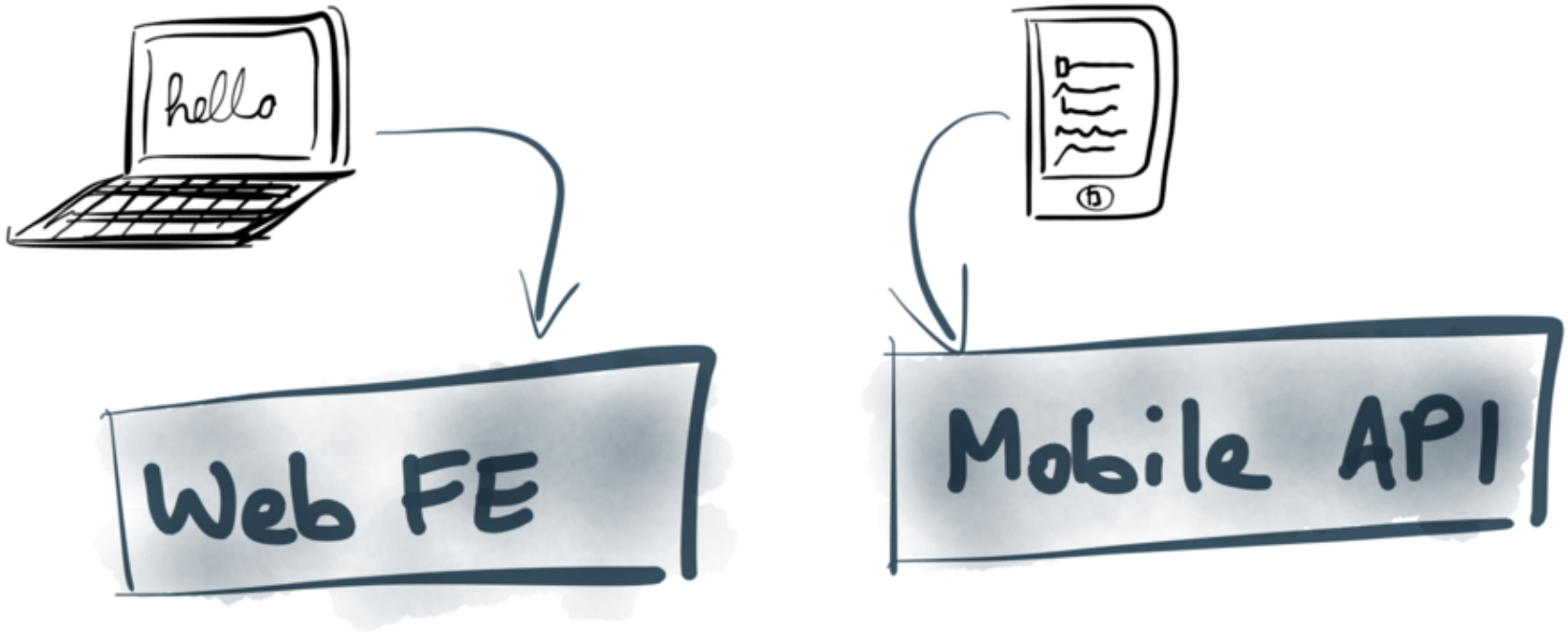
UNIVERSITY OF CAMBRIDGE

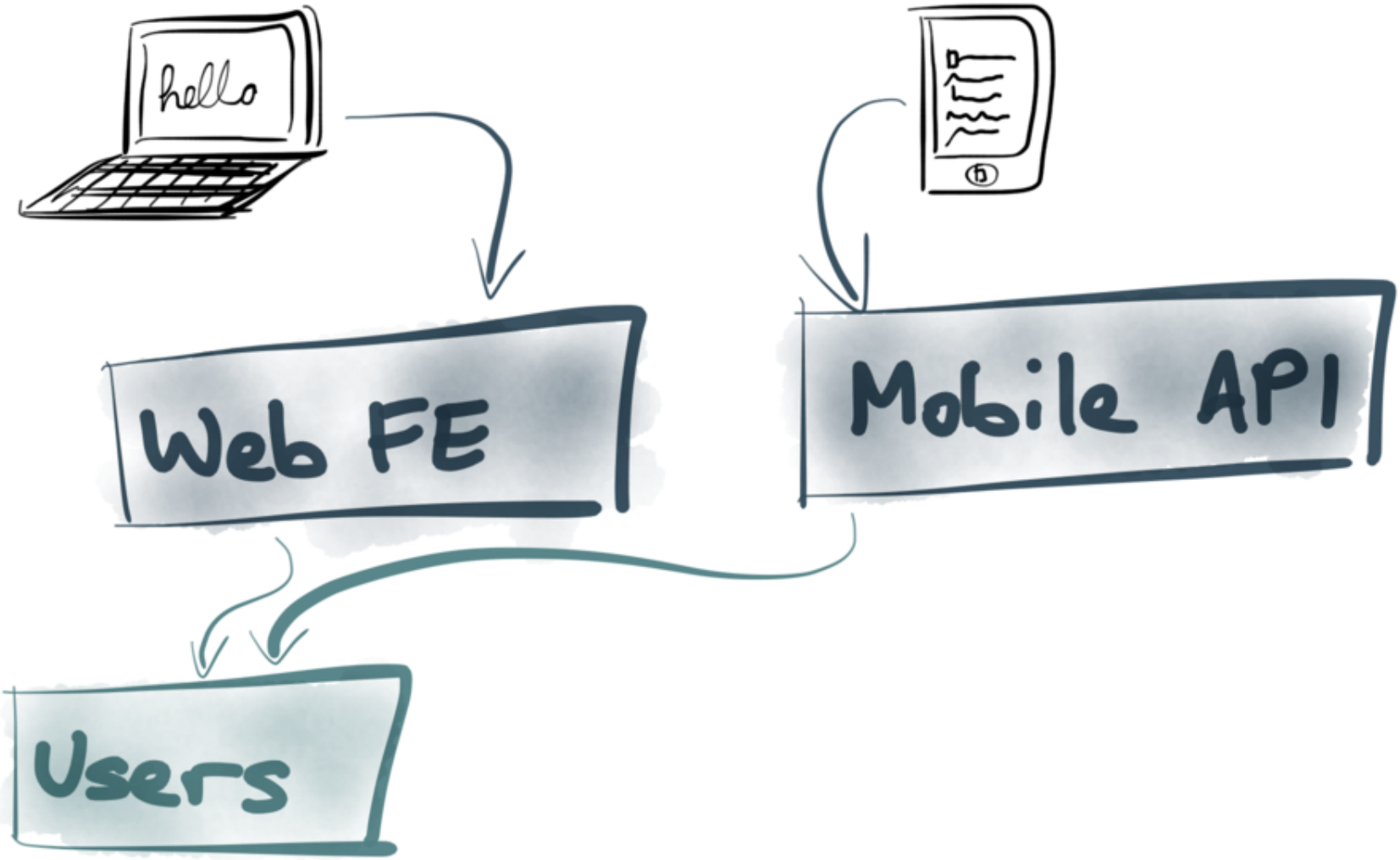
dataintensive.net

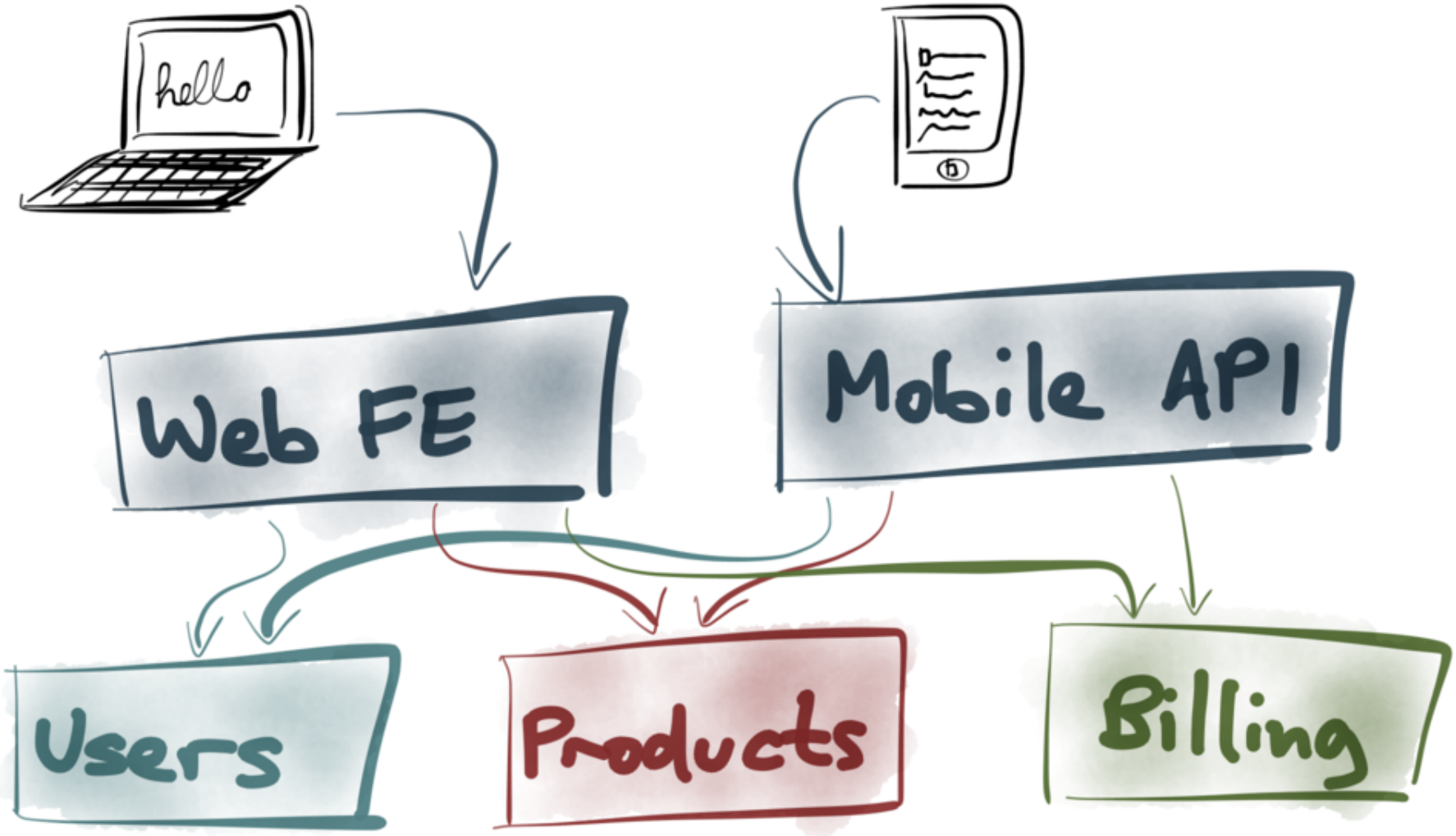


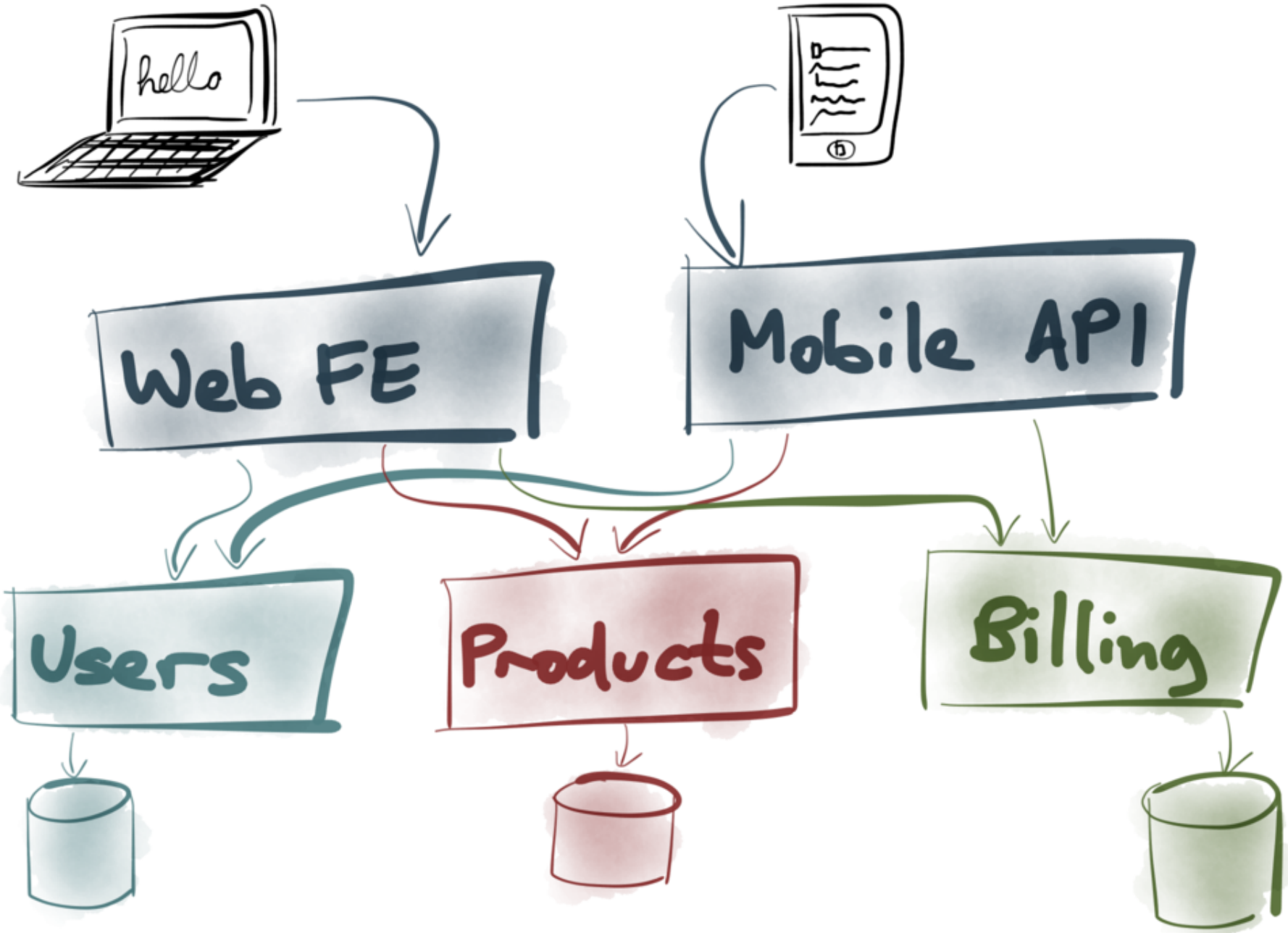
@martinkl

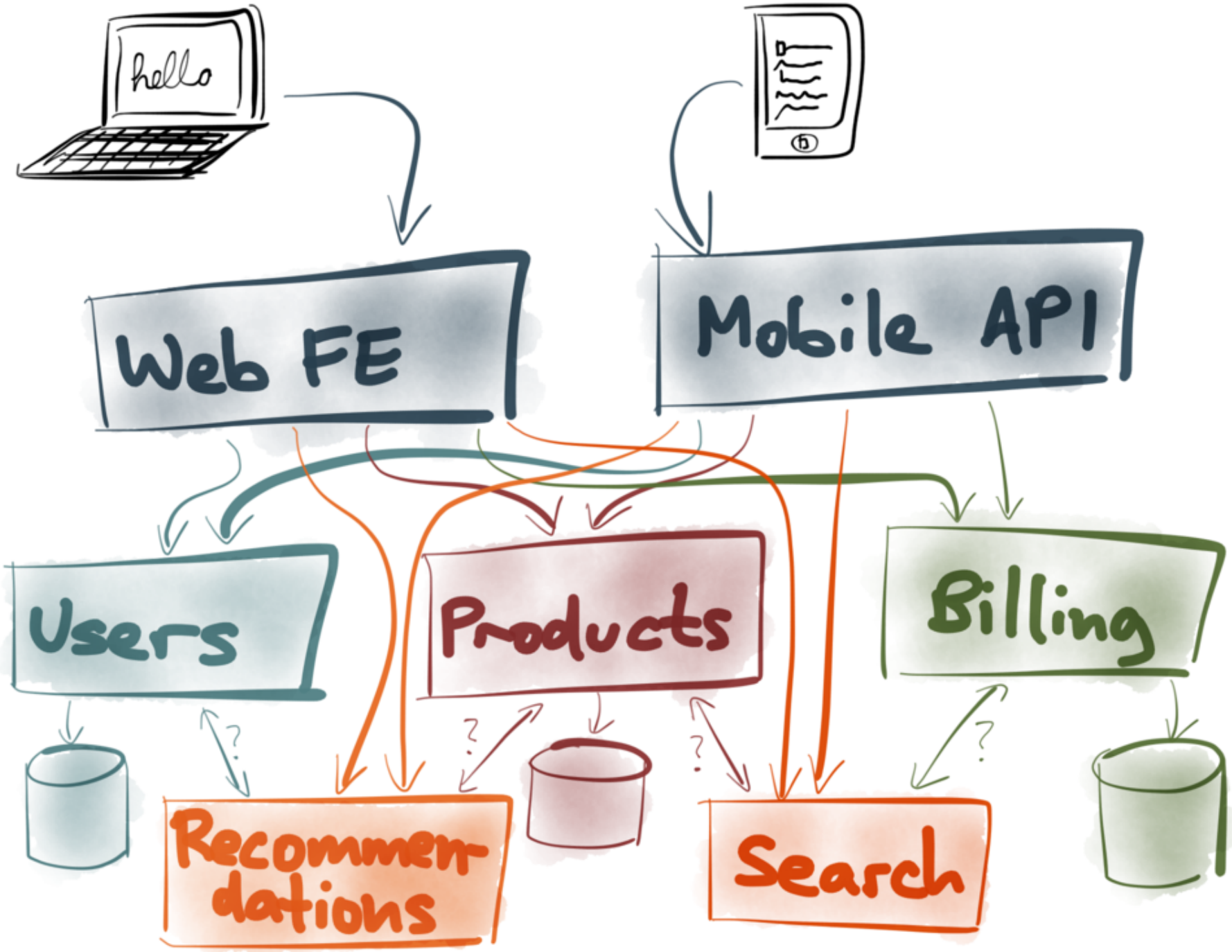

hello
Monolith
DB

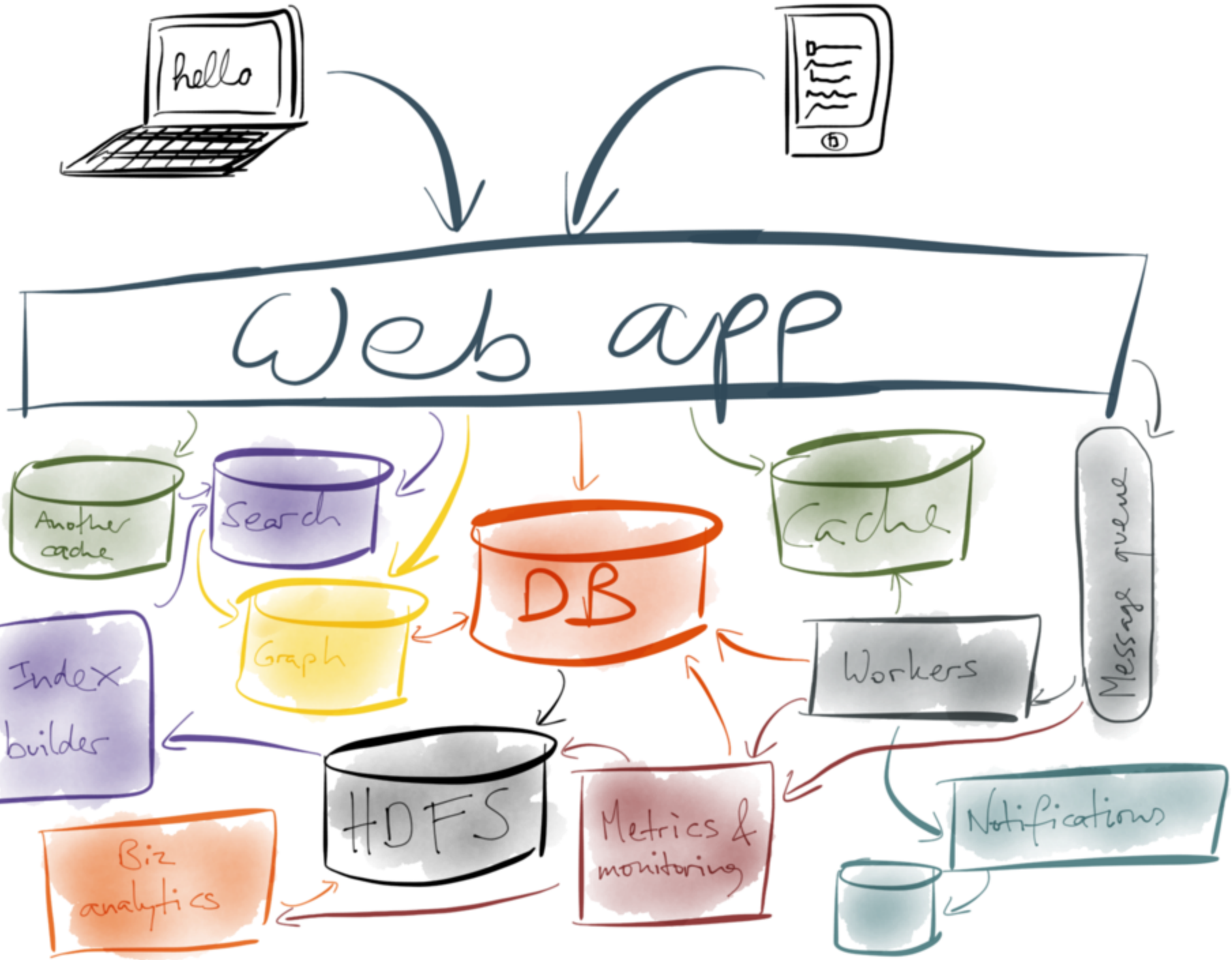

hello
Web FE
Mobile API


hello
Web FE
Mobile API
Users


hello
Web FE
Mobile API
Users
Products
Billing

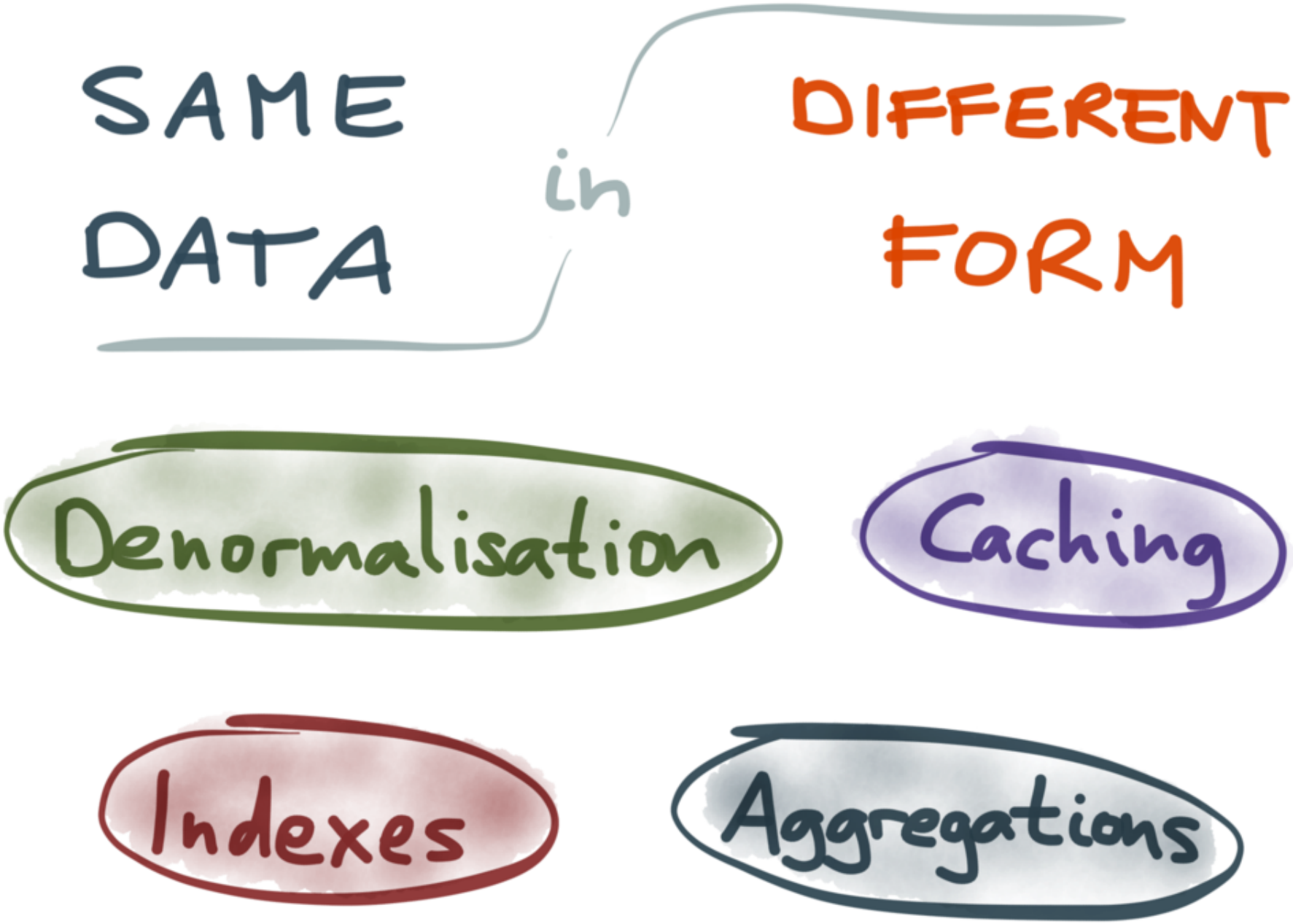

hello
Web FE
Mobile API
Users
Products
Billing

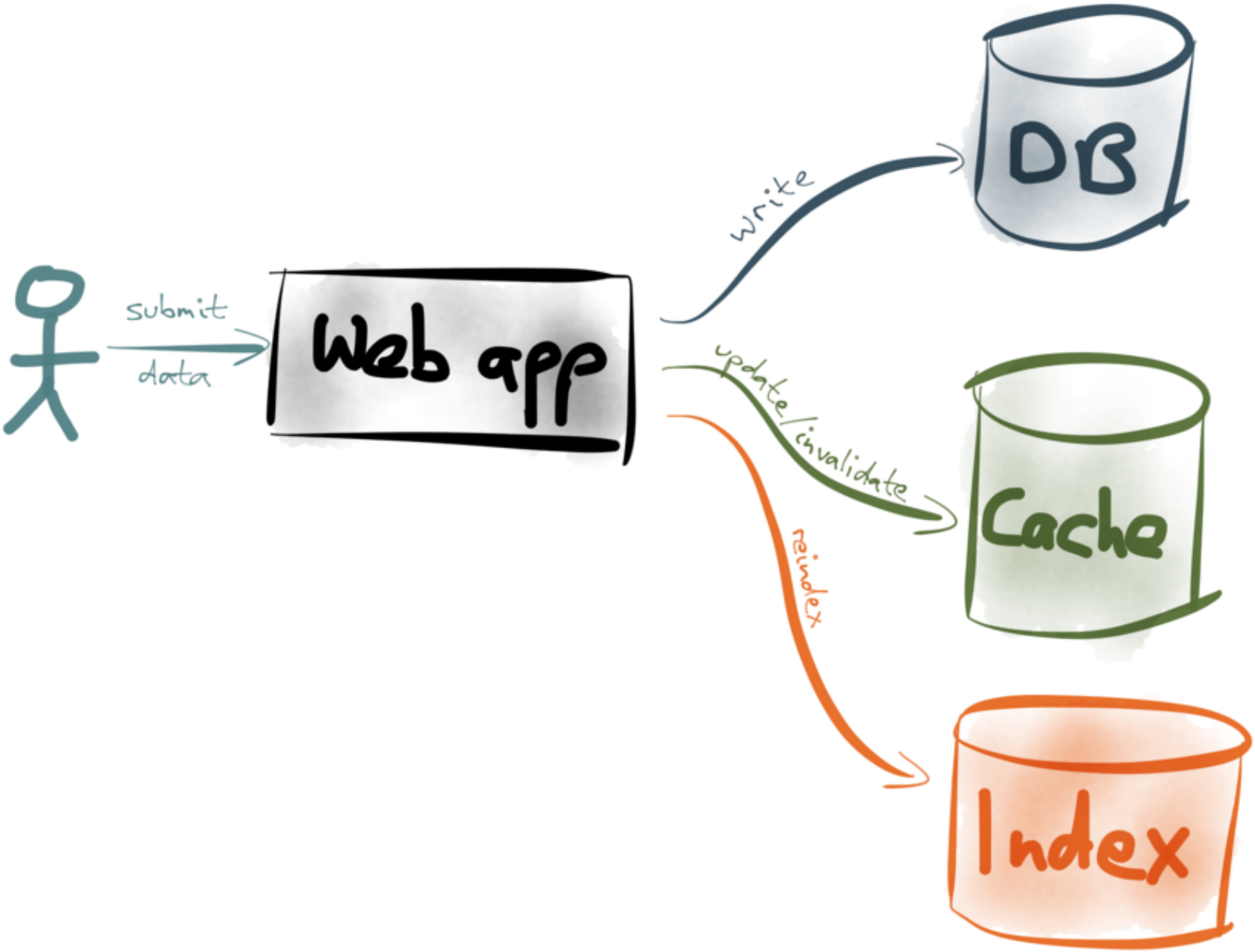

hello
Web FE
Mobile API
Users
Products
Billing
Recommendations
Search

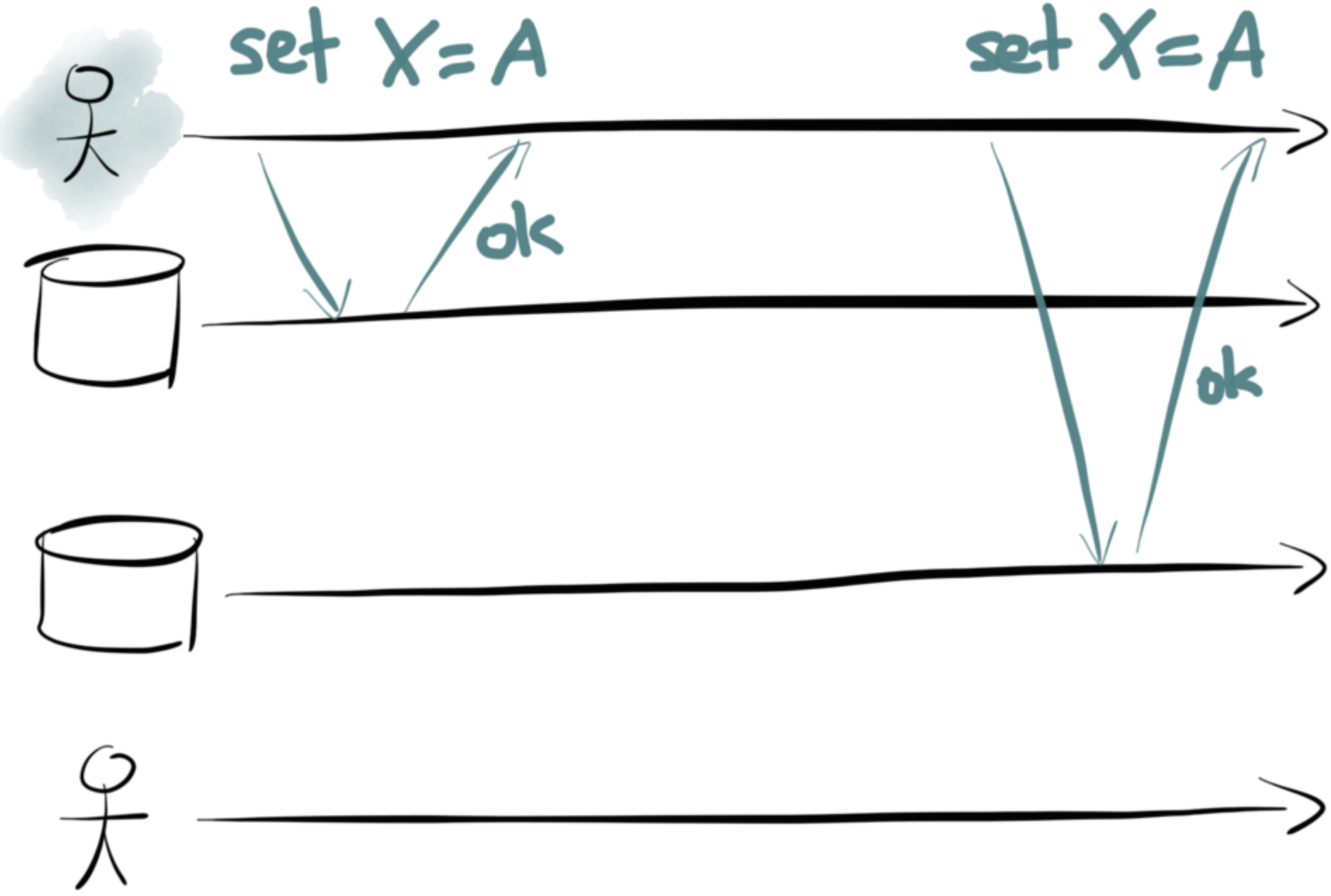

hello
Web app
Another cache
Search
DB
Cache
Message queue
Graph
Index builder
Workers
HDFS
Metrics & monitoring
Notifications
Biz analytics

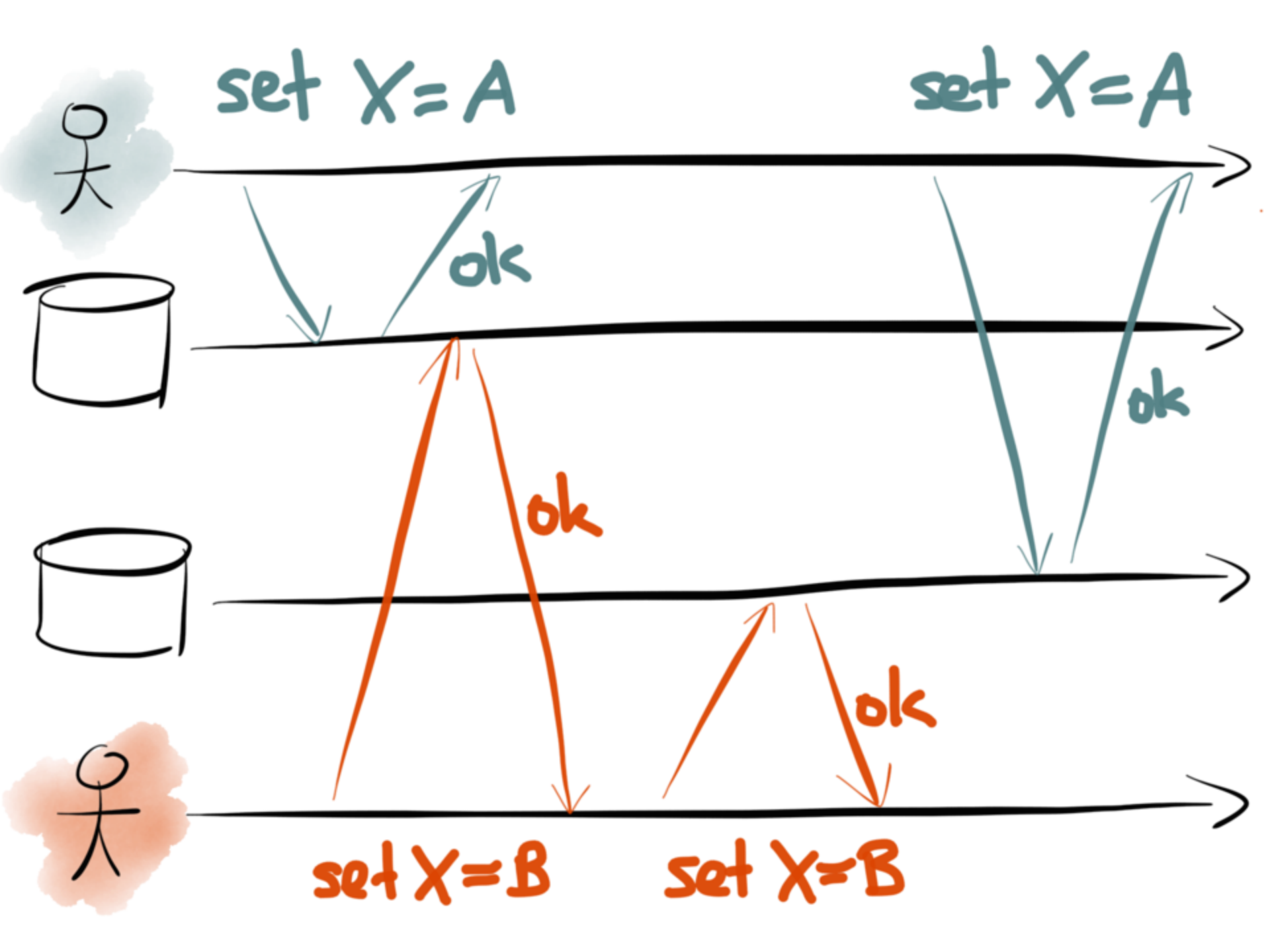

hello
Web app
Another cache
Search
Graph
Index builder
HDFS
Biz analytics
Metrics & monitoring
Workers
Message queue
Notifications
sanity

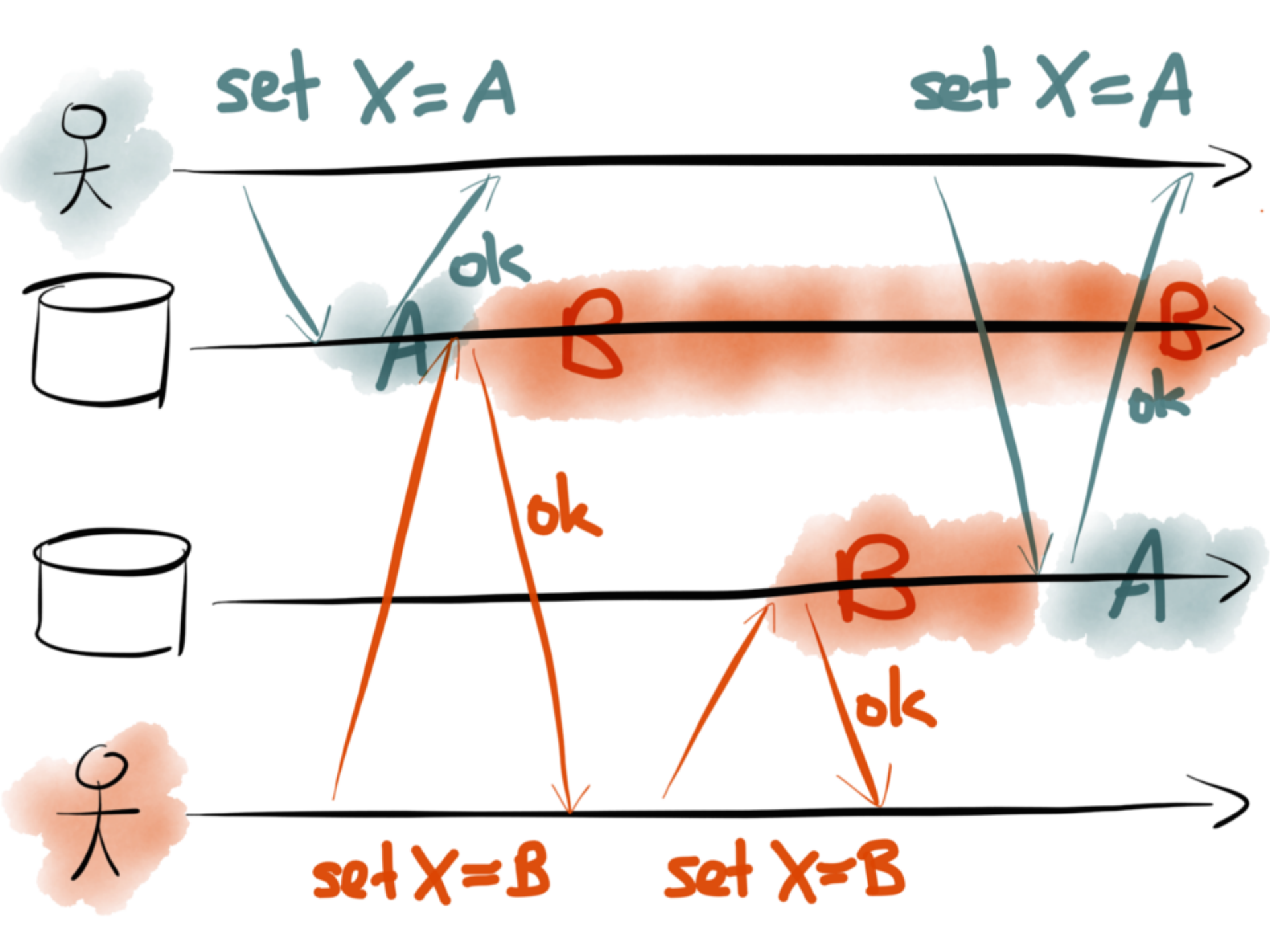
SAME DATA in DIFFERENT FORM

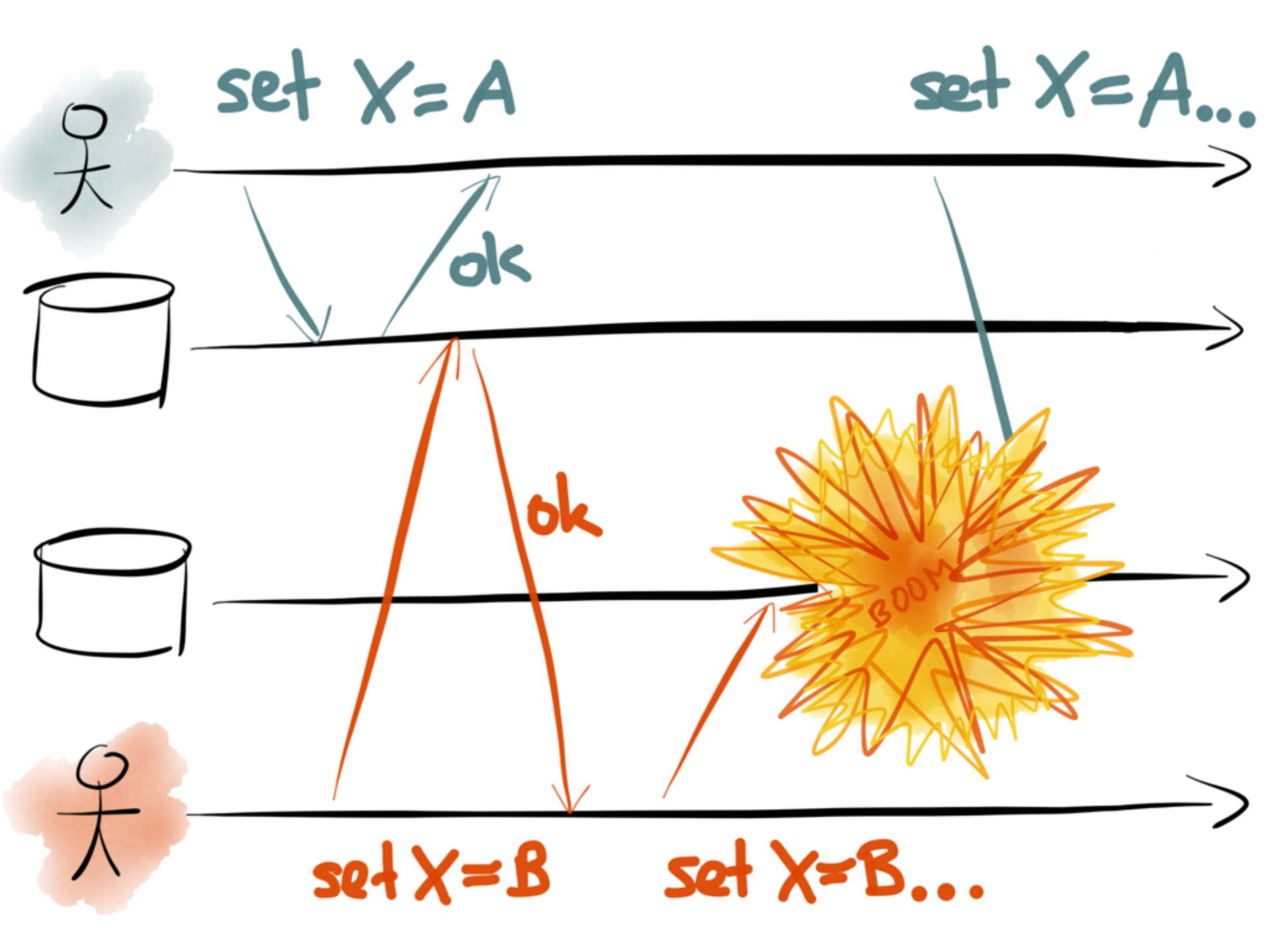

SAME
DATA
in
DIFFERENT
FORM
Denormalisation
Caching
Indexes
Aggregations

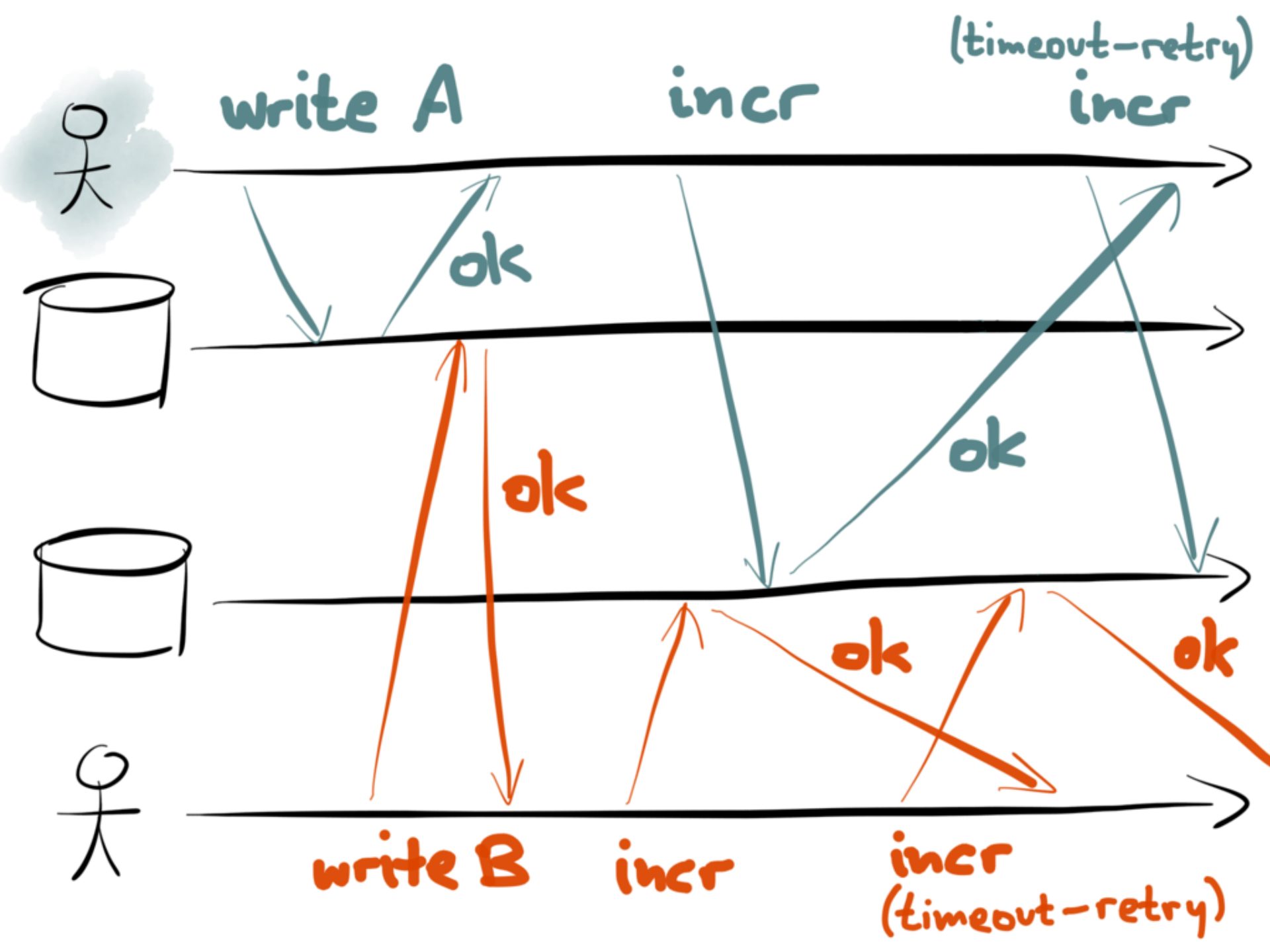

submit
data
Web app
write
DB
update/invalidate
Cache
reindex
Index


set X=A
set X=A
ok
ok


set X=A
set X=A
ok
ok
ok
ok
set X=B
set X=B


set X=A
set X=A
ok
A
B
B
ok
ok
B
A
ok
set X=B
set X=B


set X=A
set X=A...
ok
ok
BOOM
set X=B
set X=B...


(timeout-retry)
write A
incr
incr
ok
ok
ok
ok
ok
write B
incr
incr
(timeout-retry)

TWEETS 6,219 FOLLOWING -20 FOLLOWERS 24.1K

Follow

Лепра

@leprasorium

Хуепра. Добро пожаловать отсюда

Default City

Лепра @leprasorium · 2h

Викторианские советы

Часть 2 pic.twitter.com/21PraRYBaO

Details

Лепра @leprasorium · 2h

Викторианские советы

Часть 1 pic.twitter.com/BVE6ao8711

Details

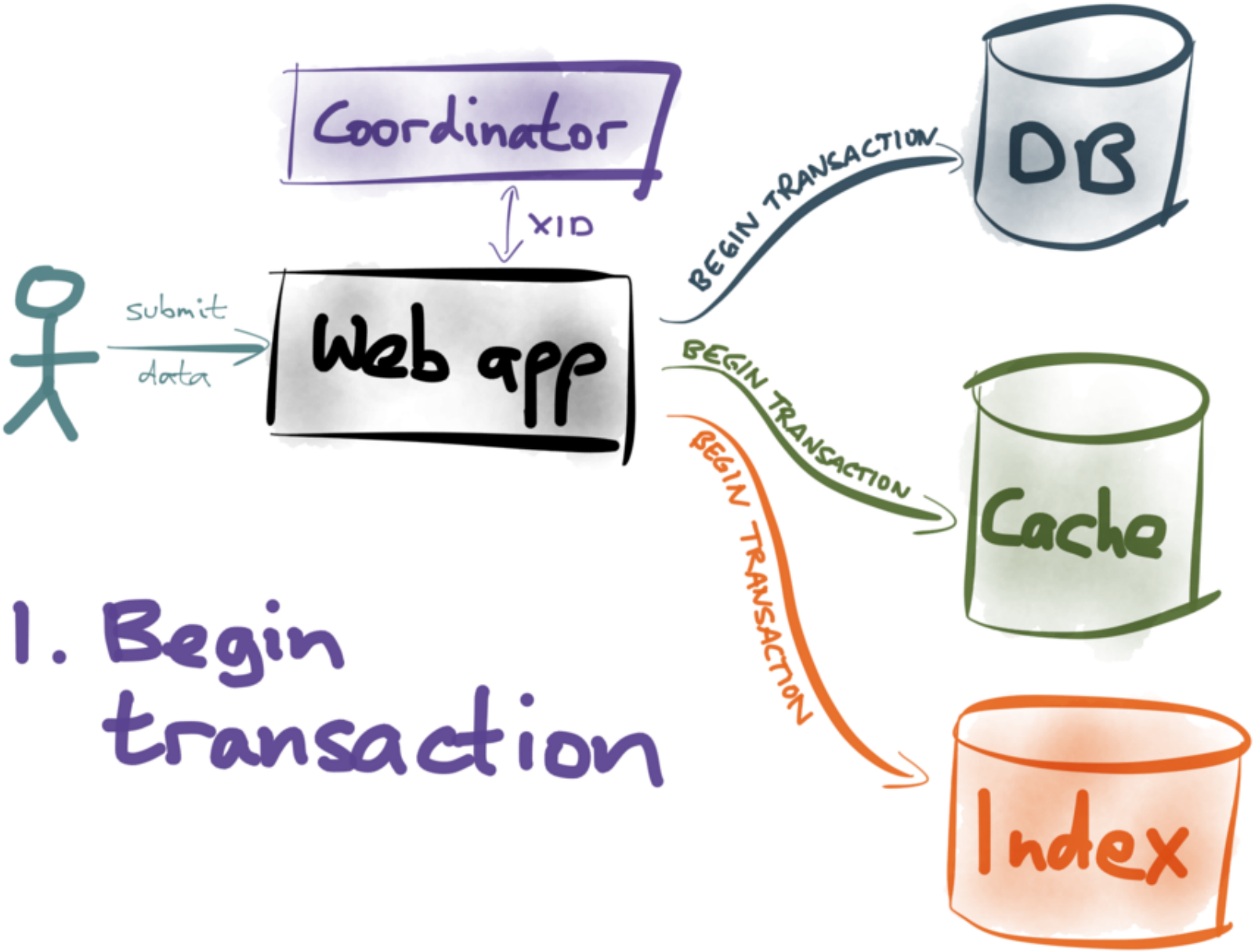
Go to full profile

```
insert into emails
  (mailbox_id, unread, body)
  values(42,   true,  'Hello!');
update mailboxes
  set unread_count += 1
  where id=42;
```

```
begin transaction;

insert into emails
  (mailbox_id, unread, body)
  values(42,   true,   'Hello!');

update mailboxes
  set unread_count += 1
  where id=42;

commit;
```

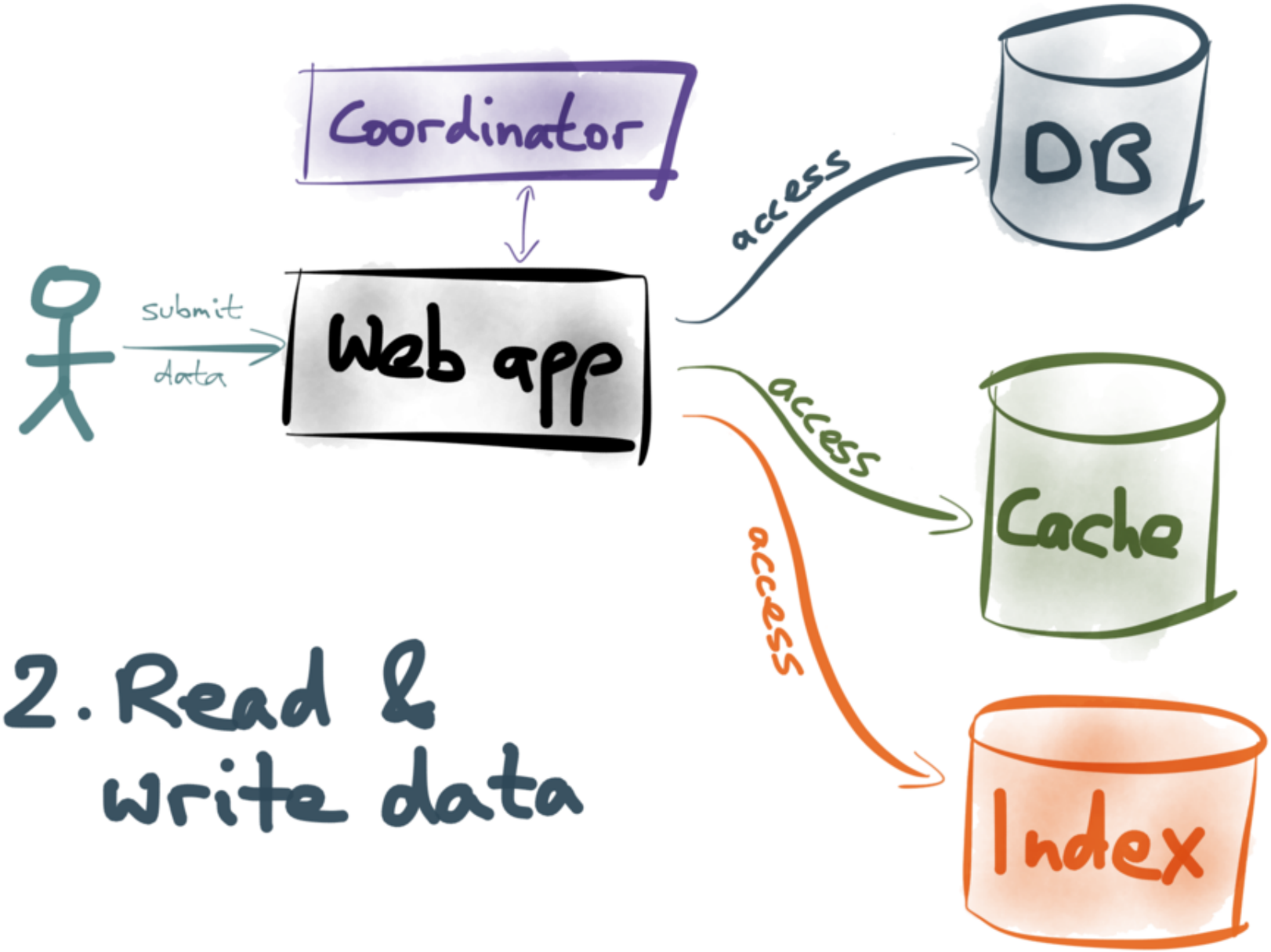
2-Phase commit?

2-PHASE COMMIT



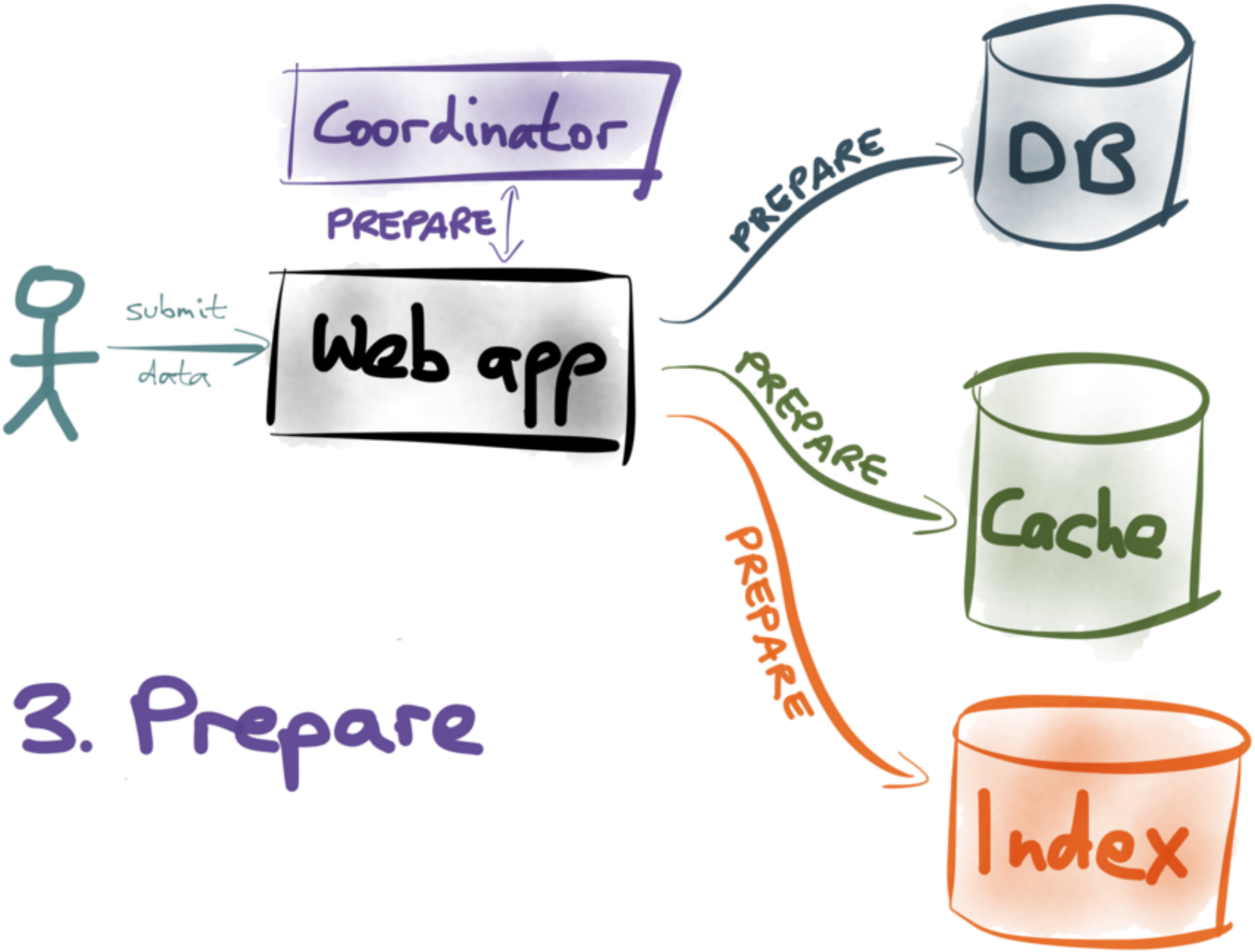
1. Begin transaction

2-PHASE COMMIT



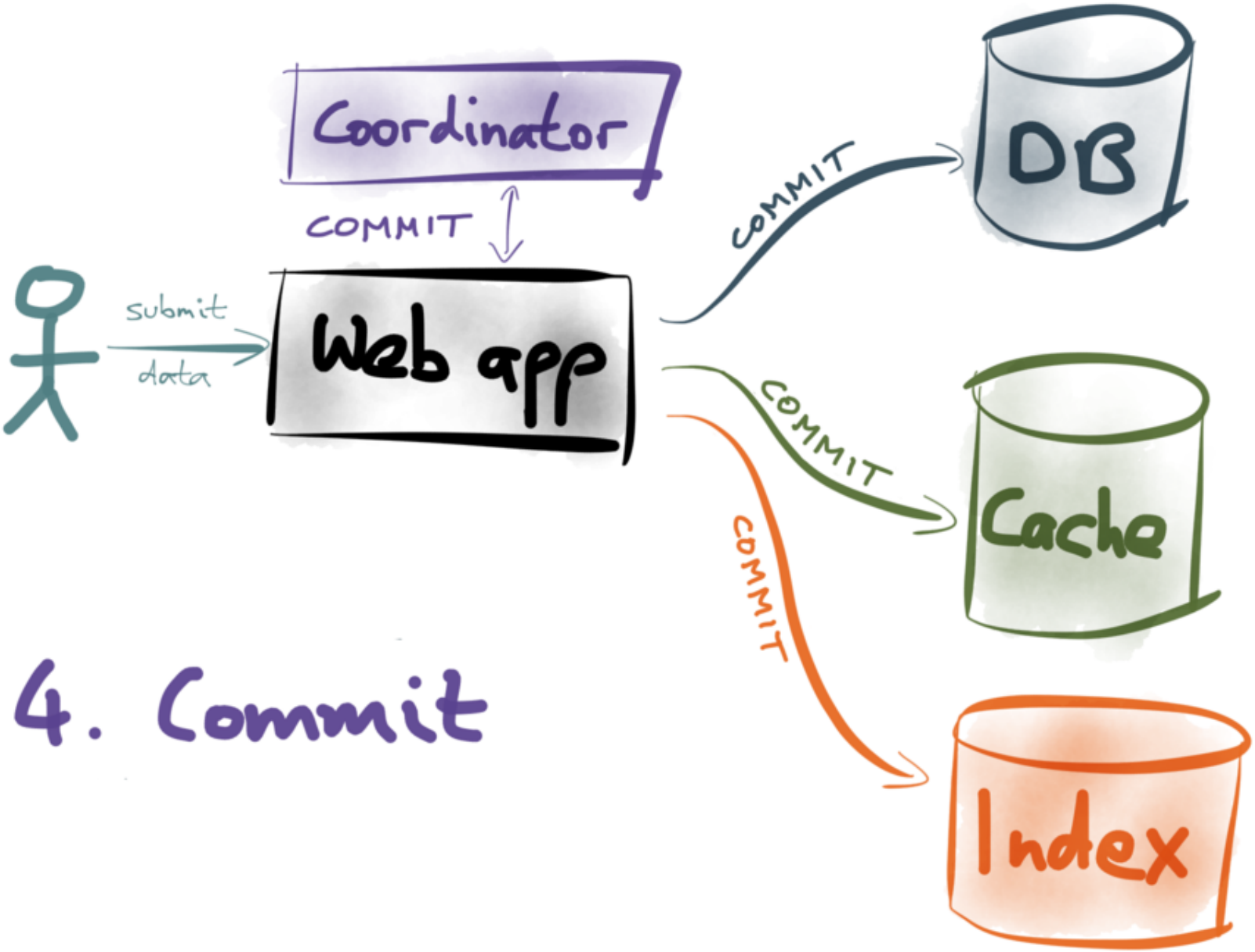
2. Read & write data

2-PHASE COMMIT

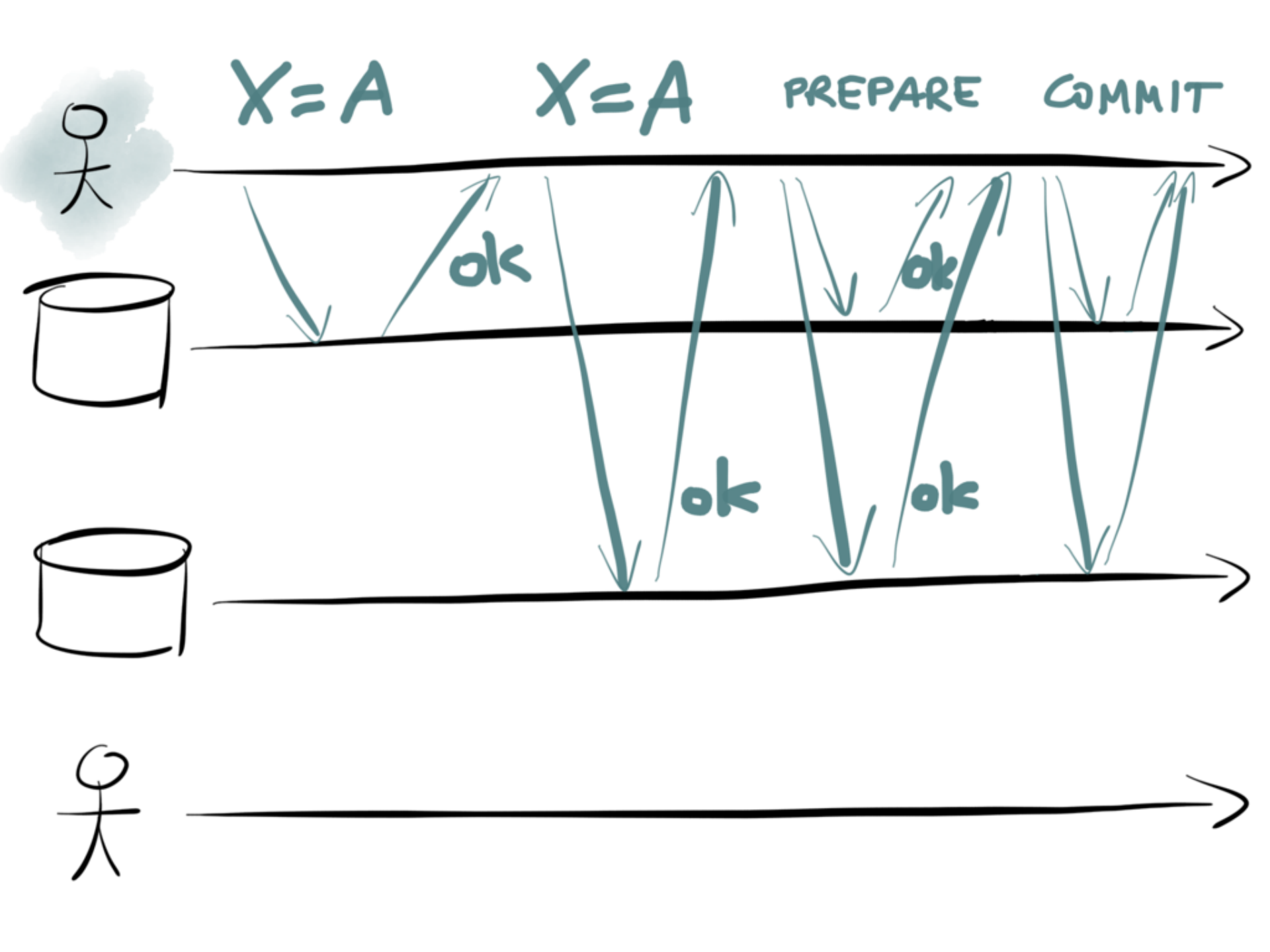


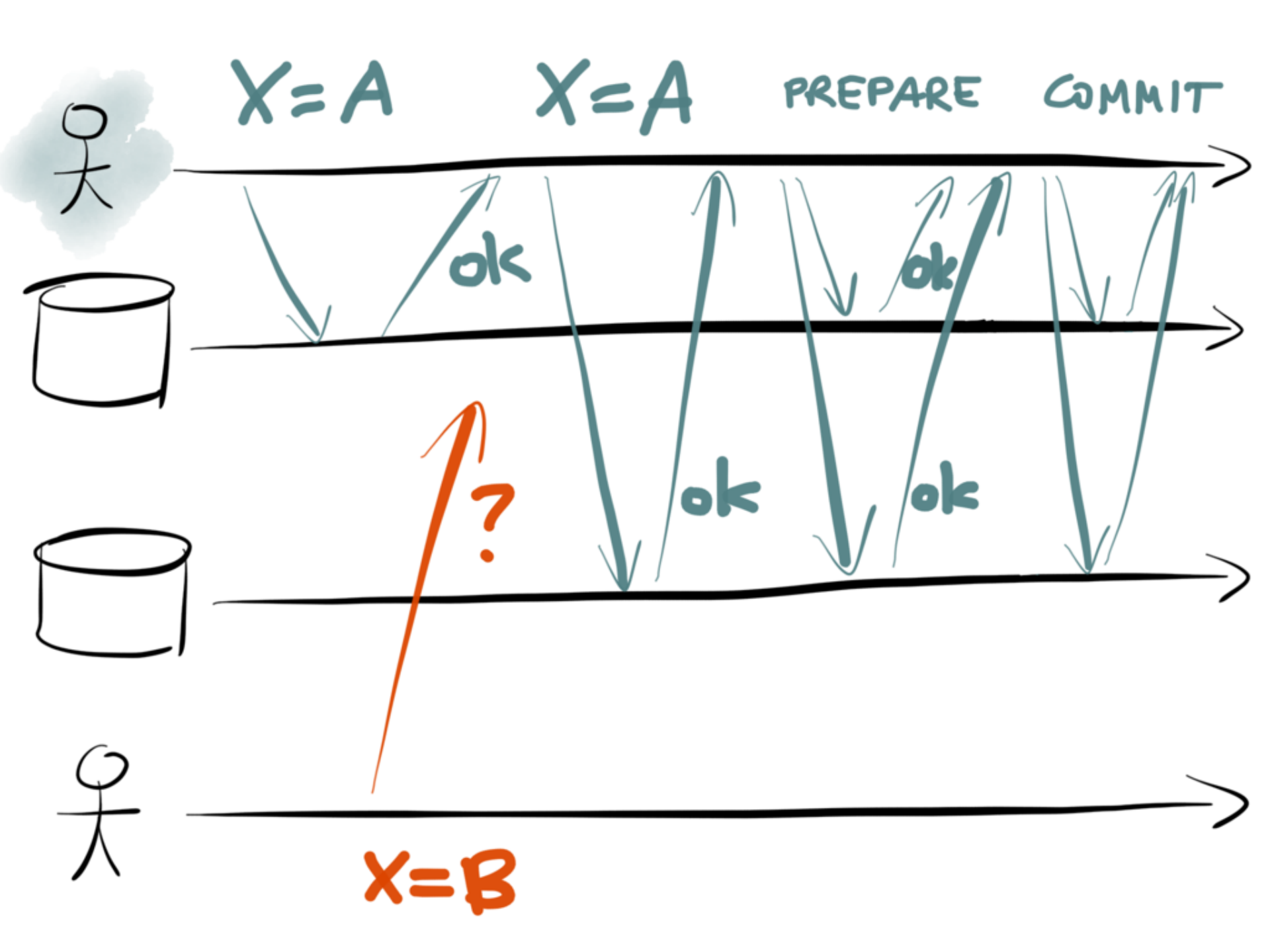
3. Prepare

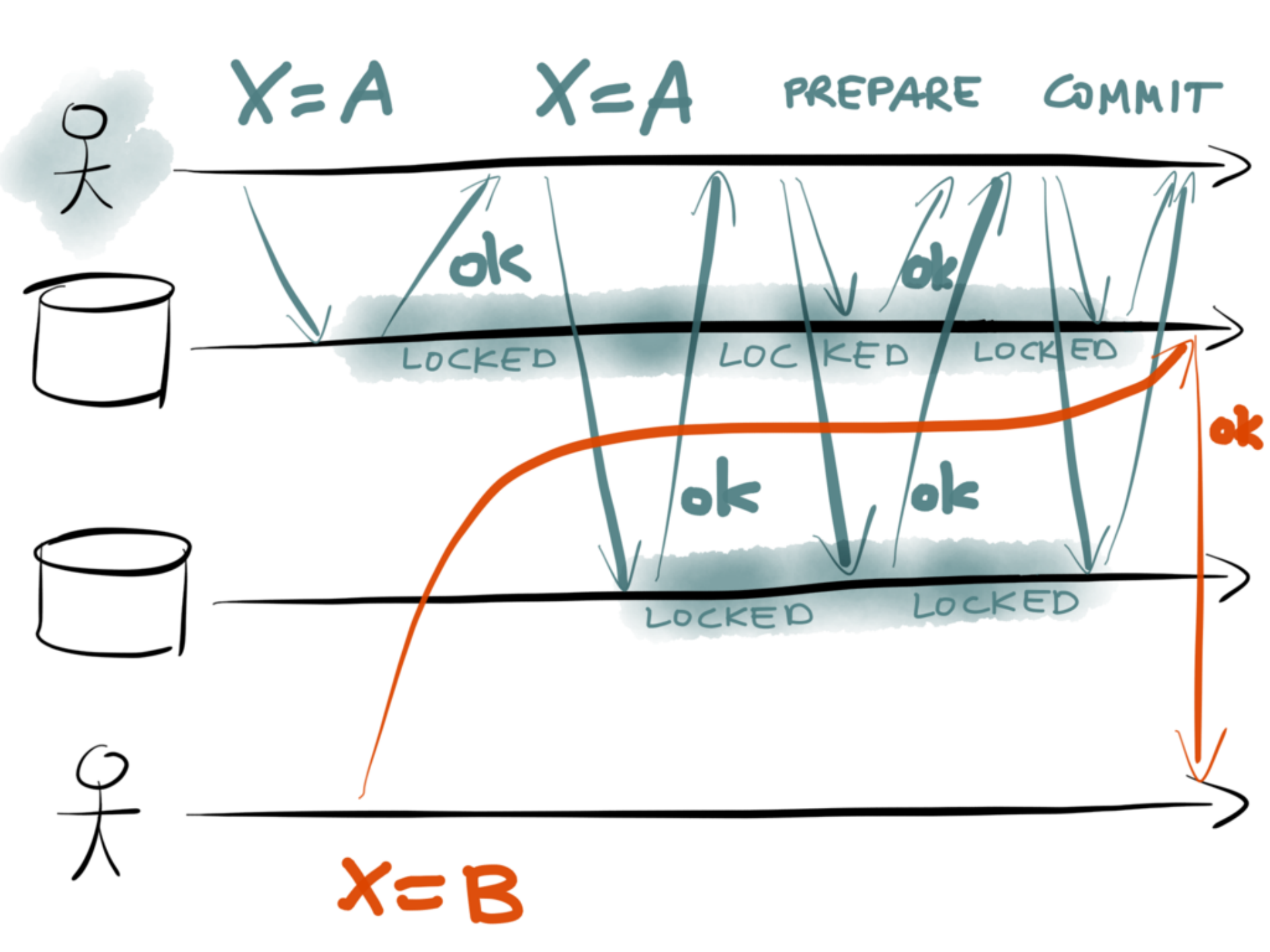
2-PHASE COMMIT

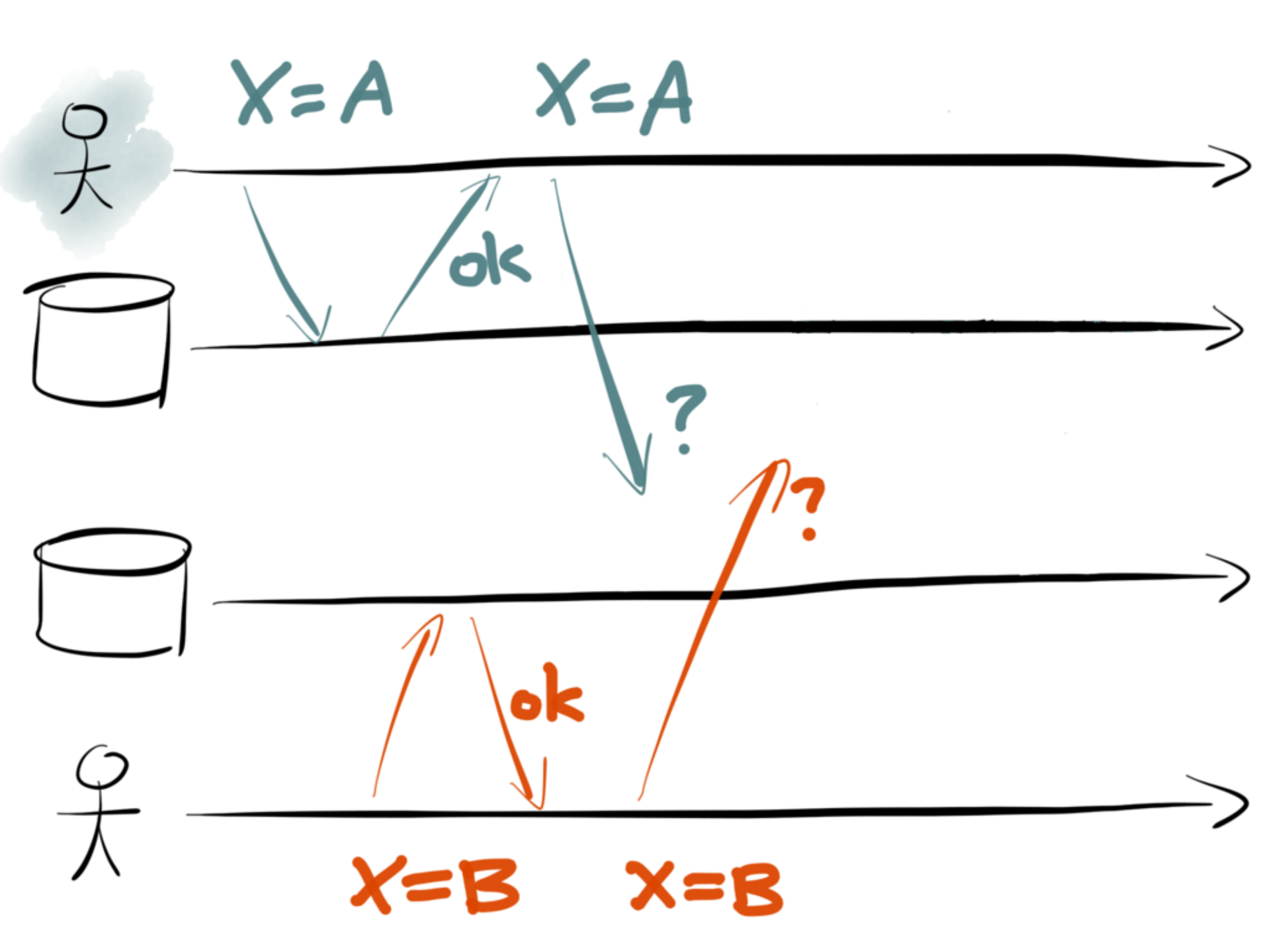


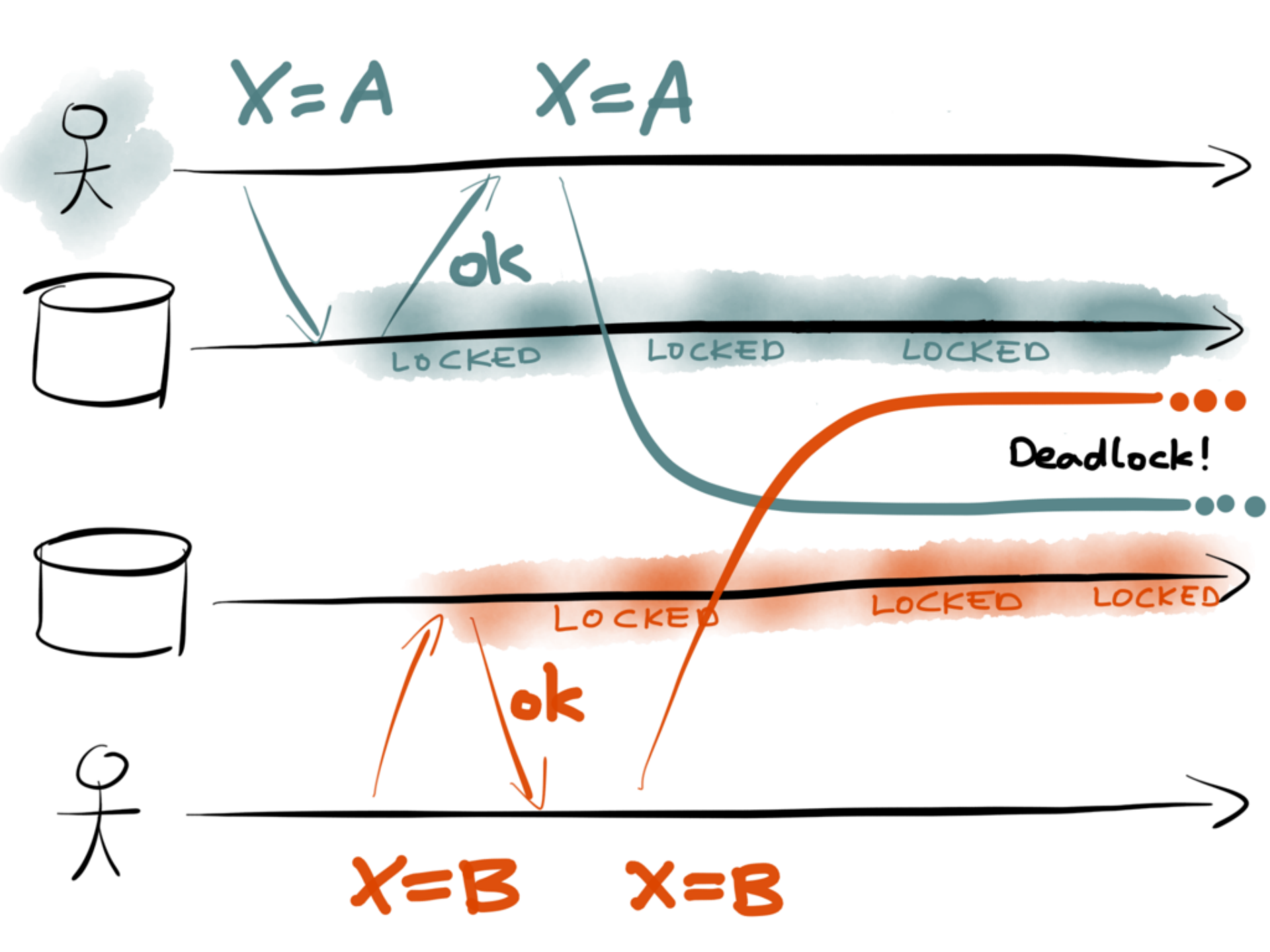
4. Commit

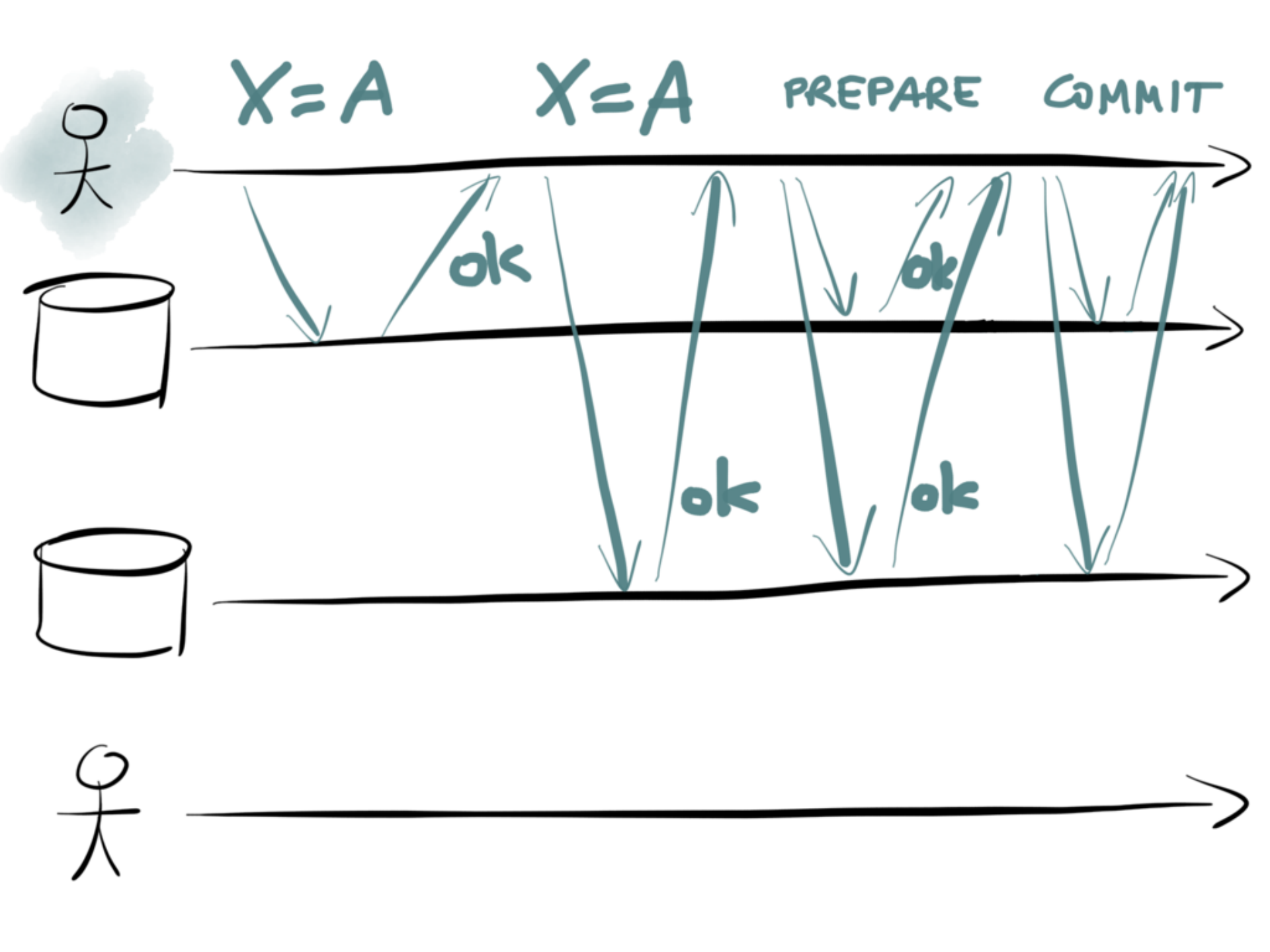

X=A
X=A
PREPARE
COMMIT
ok
ok
ok
ok

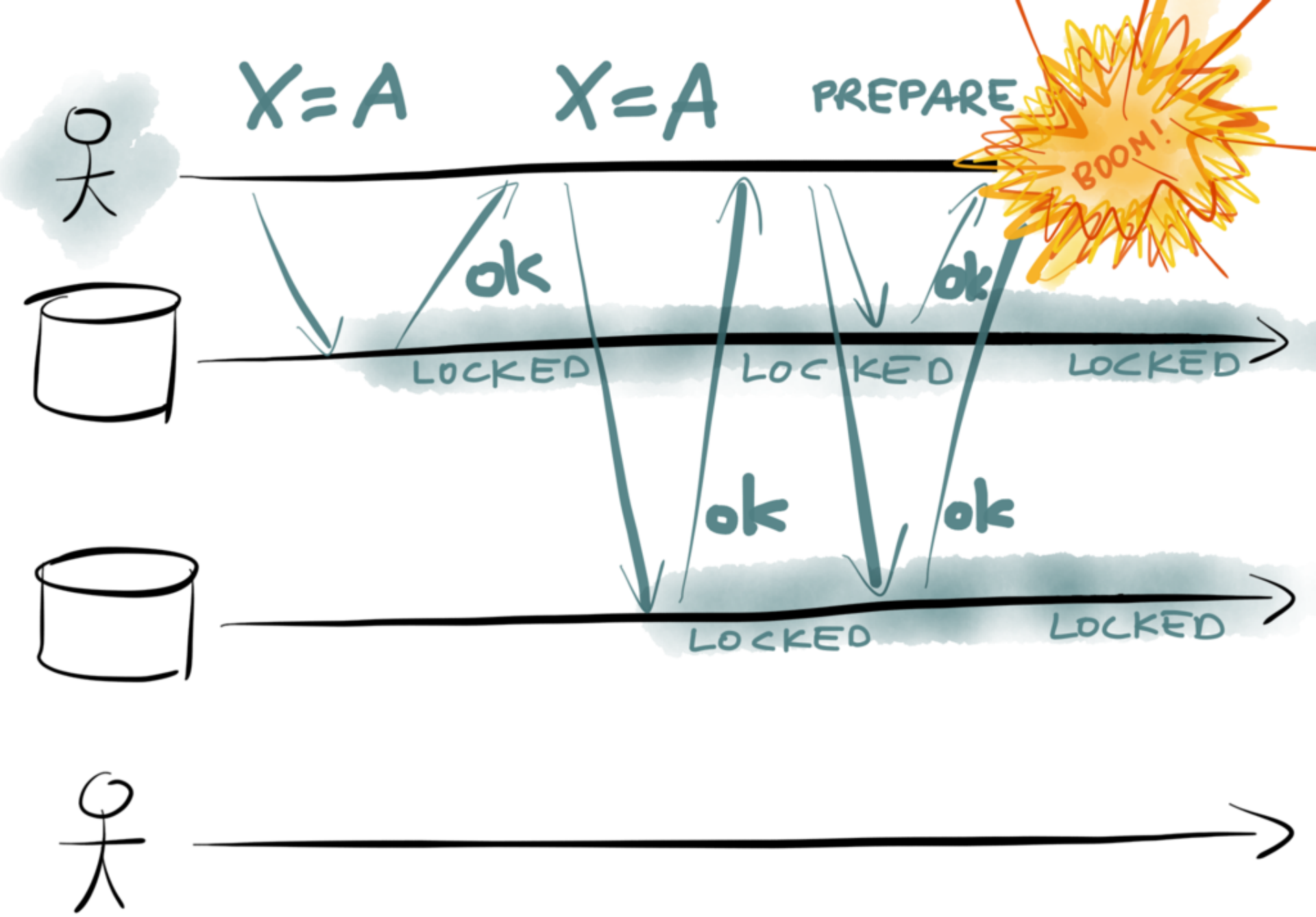

X=A
X=A
PREPARE
COMMIT
ok
ok
?
ok
ok
X=B

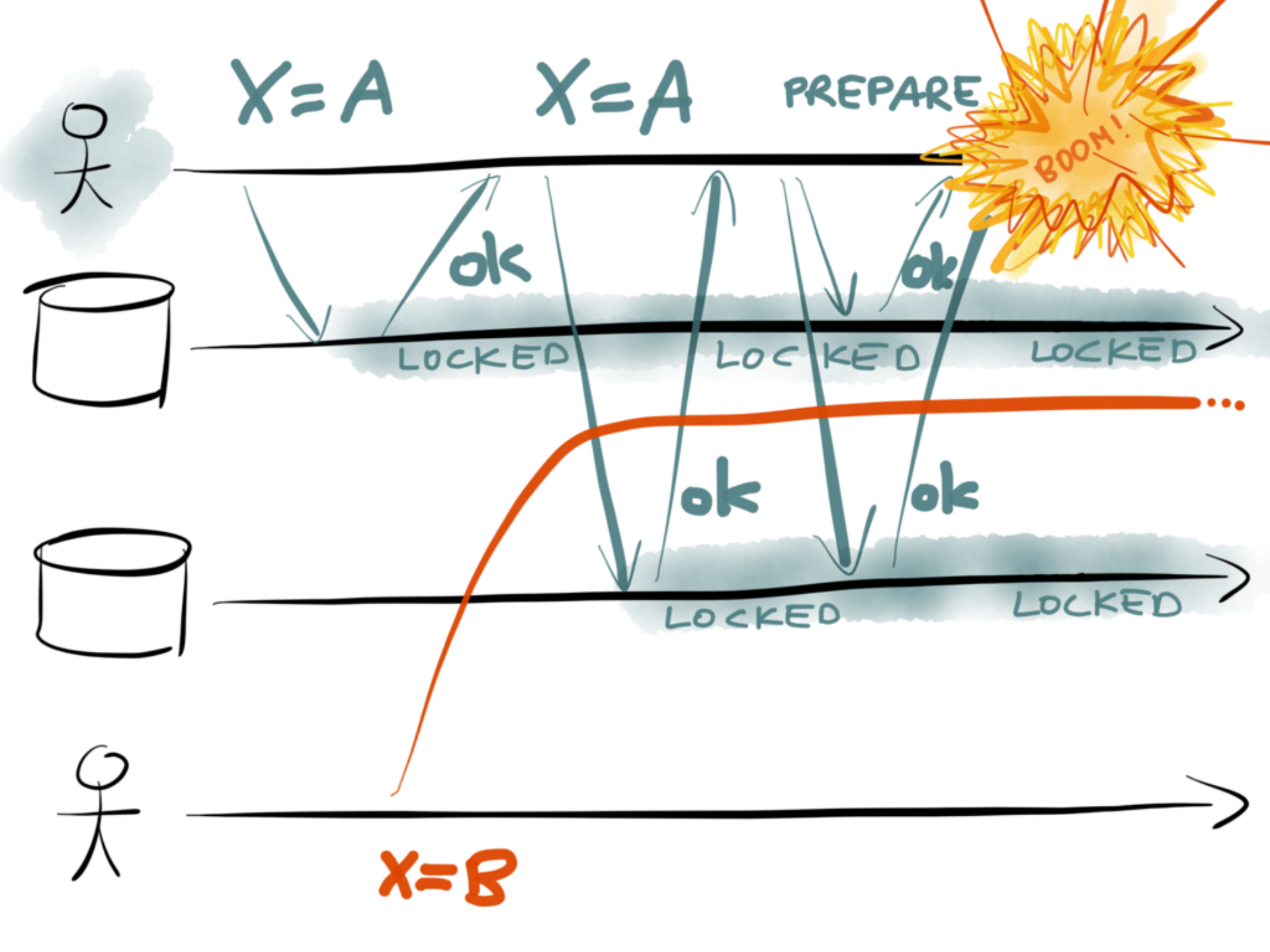

X=A
X=A
PREPARE
COMMIT
ok
ok
LOCKED
LOCKED
LOCKED
ok
ok
ok
LOCKED
LOCKED
X=B

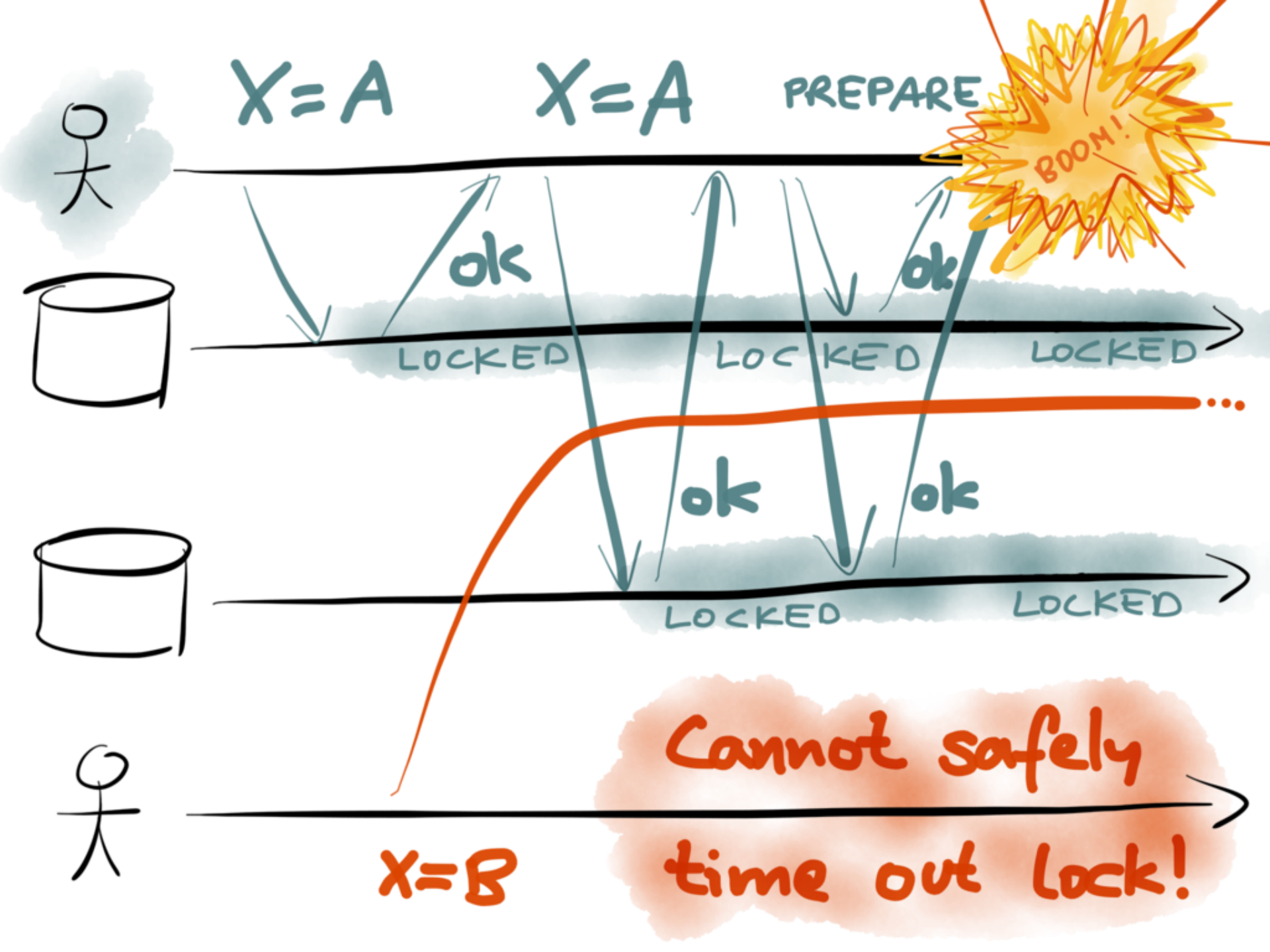

X=A
X=A
ok
?
?
ok
X=B
X=B


X=A
X=A
ok
LOCKED
LOCKED
LOCKED
Deadlock!
LOCKED
LOCKED
LOCKED
ok
X=B
X=B


X=A
X=A
PREPARE
COMMIT
ok
ok
ok
ok


X=A
X=A
PREPARE
BOOM!
ok
ok
ok
ok
LOCKED
LOCKED
LOCKED
LOCKED
LOCKED


X=A
X=A
PREPARE
BOOM!
ok
ok
LOCKED
LOCKED
LOCKED
ok
ok
LOCKED
LOCKED
X=B


X=A
X=A
PREPARE
BOOM!
ok
ok
LOCKED
LOCKED
LOCKED
ok
ok
LOCKED
LOCKED
X=B
Cannot safely
time out lock!

BETWEEN THE DEVIL AND THE DEEP BLUE SEA

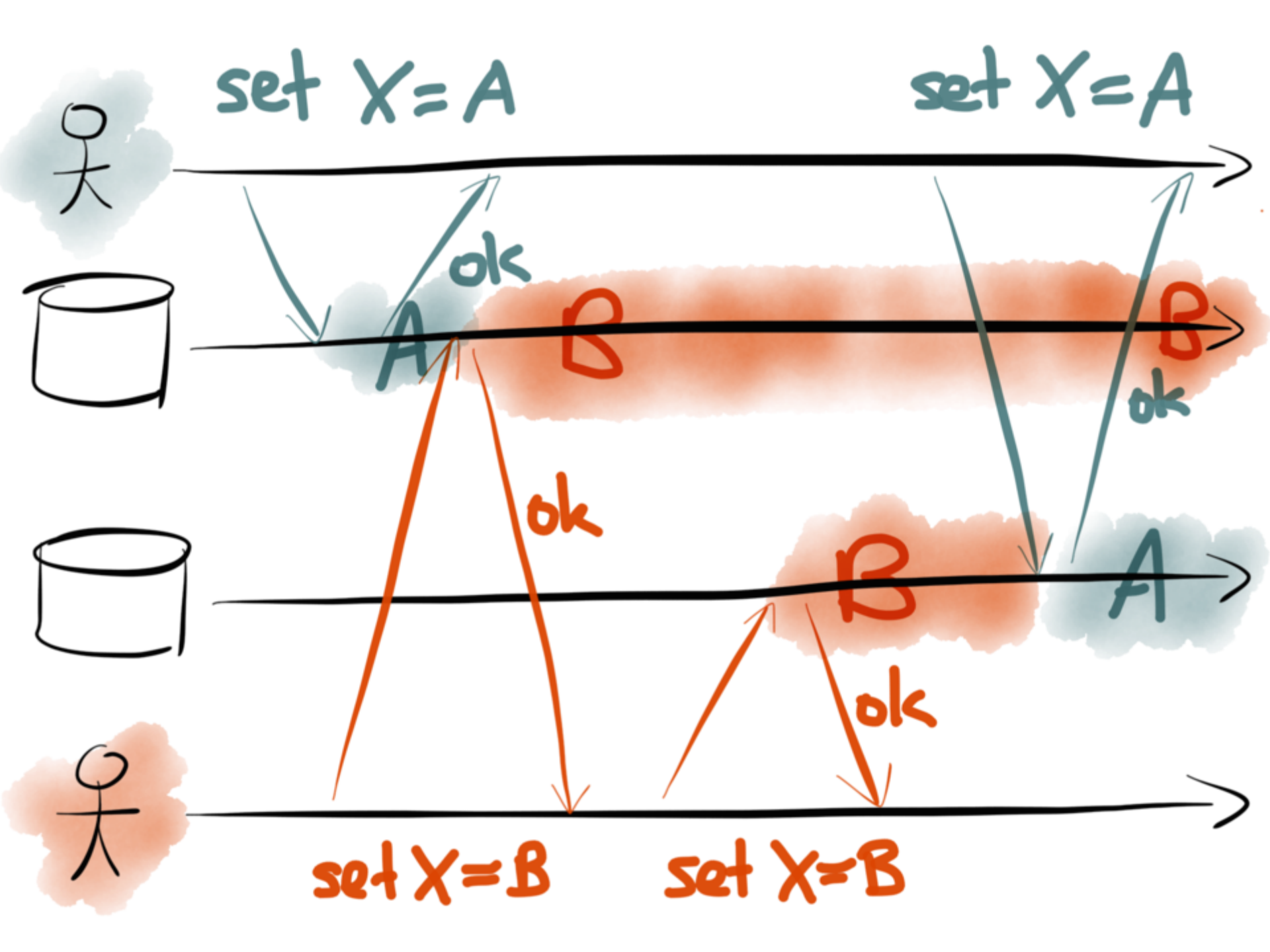
Distributed transactions

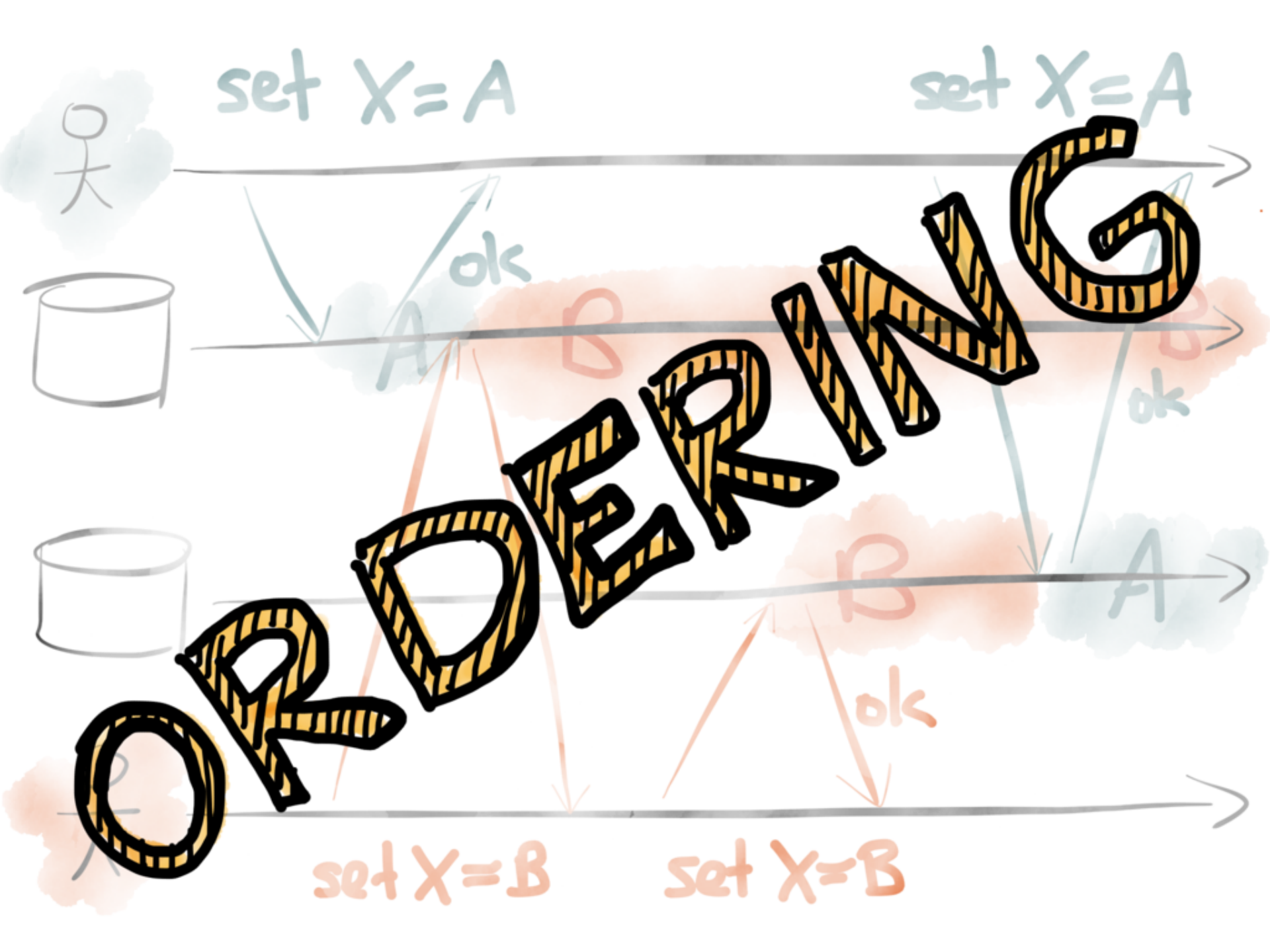
poor performance, operational problems, ...

OR

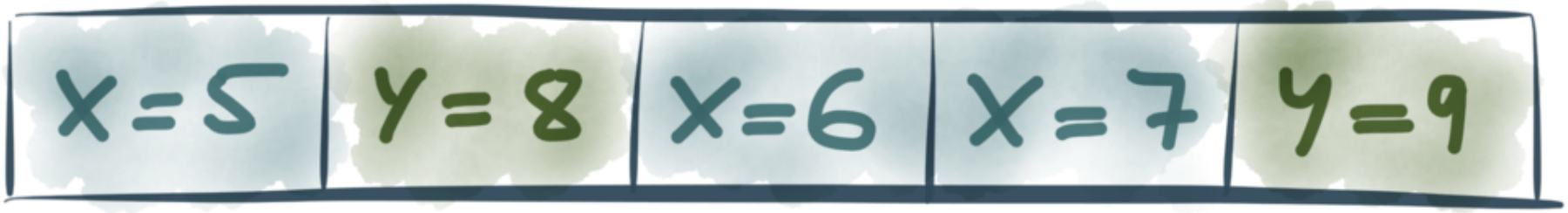
Eventual consistency

more like perpetual inconsistency, amirite?

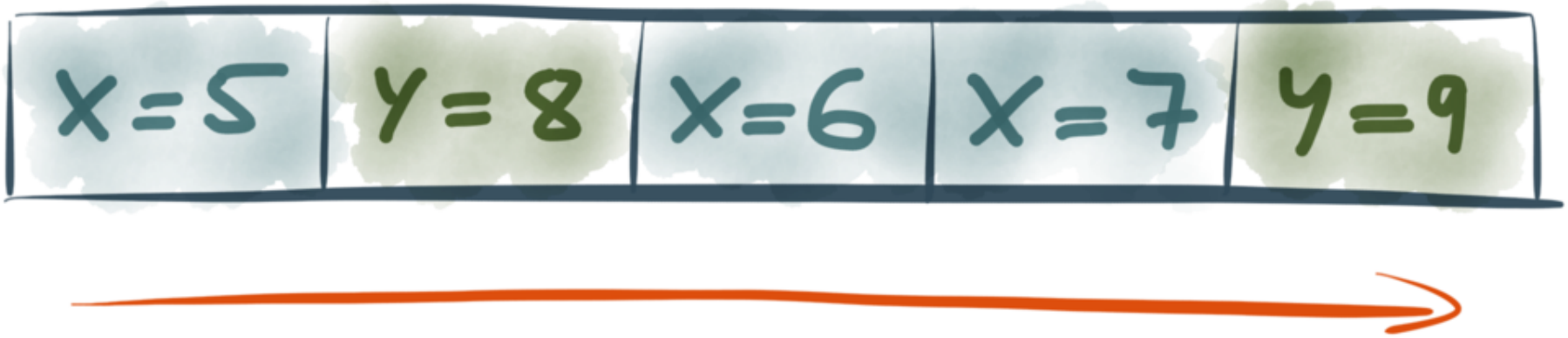

set X=A
set X=A
ok
A
B
B
ok
ok
B
A
ok
set X=B
set X=B


set X=A
set X=A
ok
A
B
B
ok
B
A
ok
set X=B
set X=B
ORDERING

STUPIDLY SIMPLE SOLUTIONS ARE THE BEST

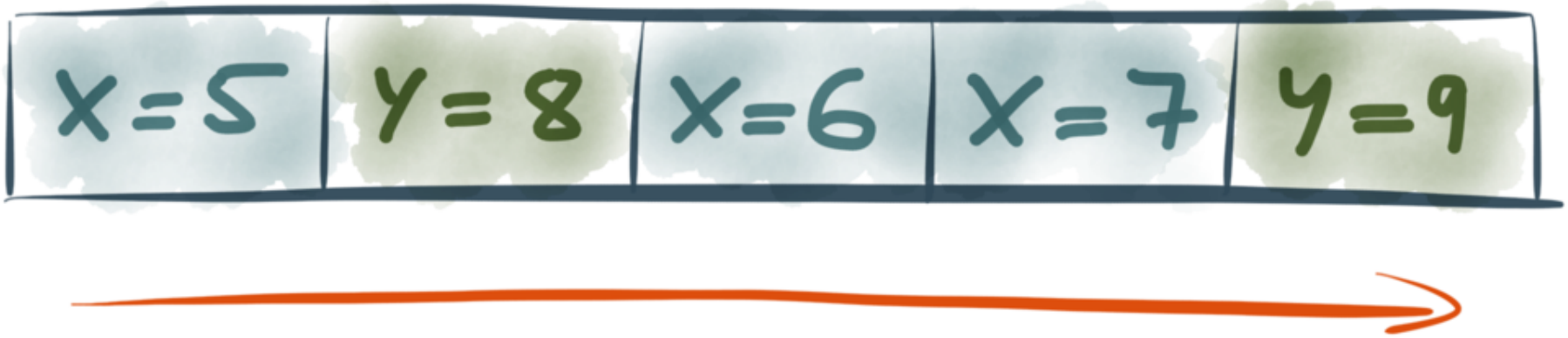


STUPIDLY SIMPLE SOLUTIONS ARE THE BEST



totally ordered sequence

STUPIDLY SIMPLE SOLUTIONS ARE THE BEST



totally ordered sequence

append-only, persistent

State machine replication

(Lamport 1978 ; Schneider 1990)

State machine replication

(Lamport 1978 ; Schneider 1990)

State machine replication

(Lamport 1978 ; Schneider 1990)

Single writer principle

(Thompson 2011)

State machine replication

(Lamport 1978 ; Schneider 1990)

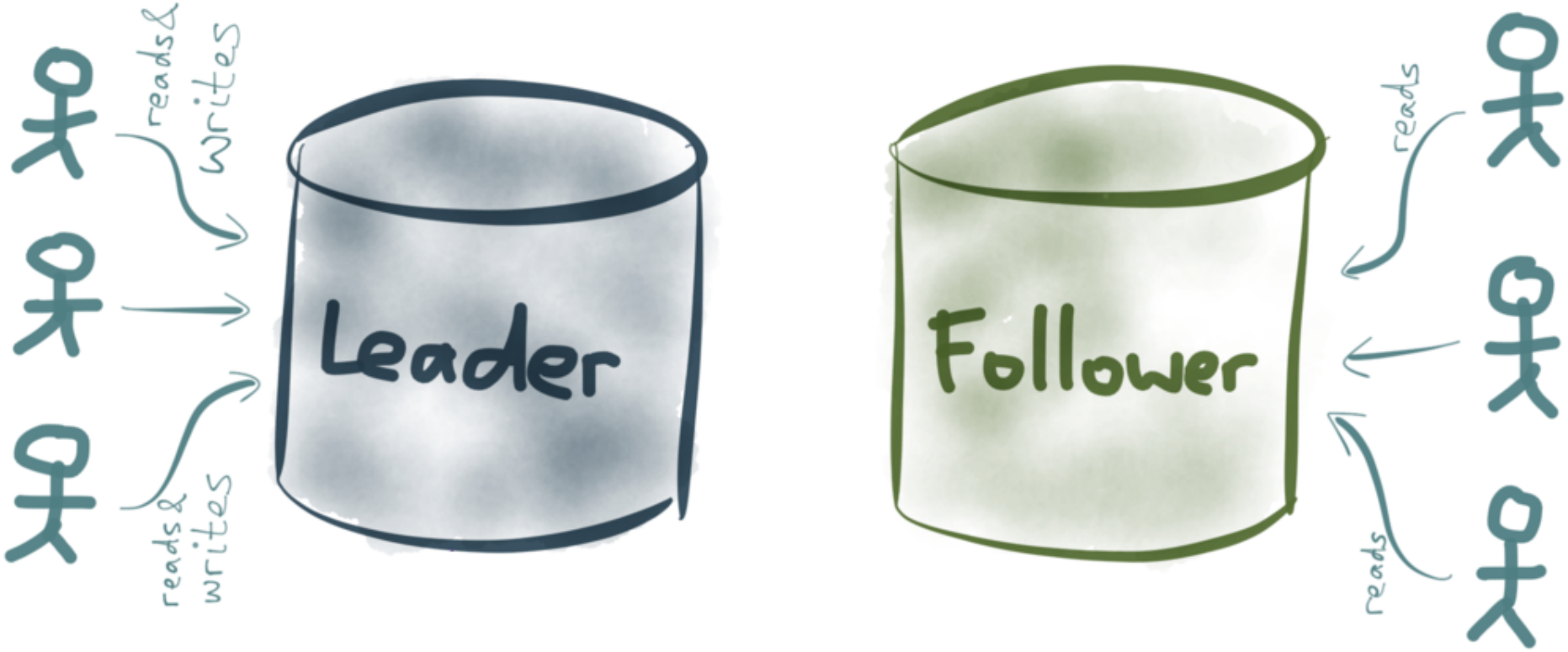
Single writer principle

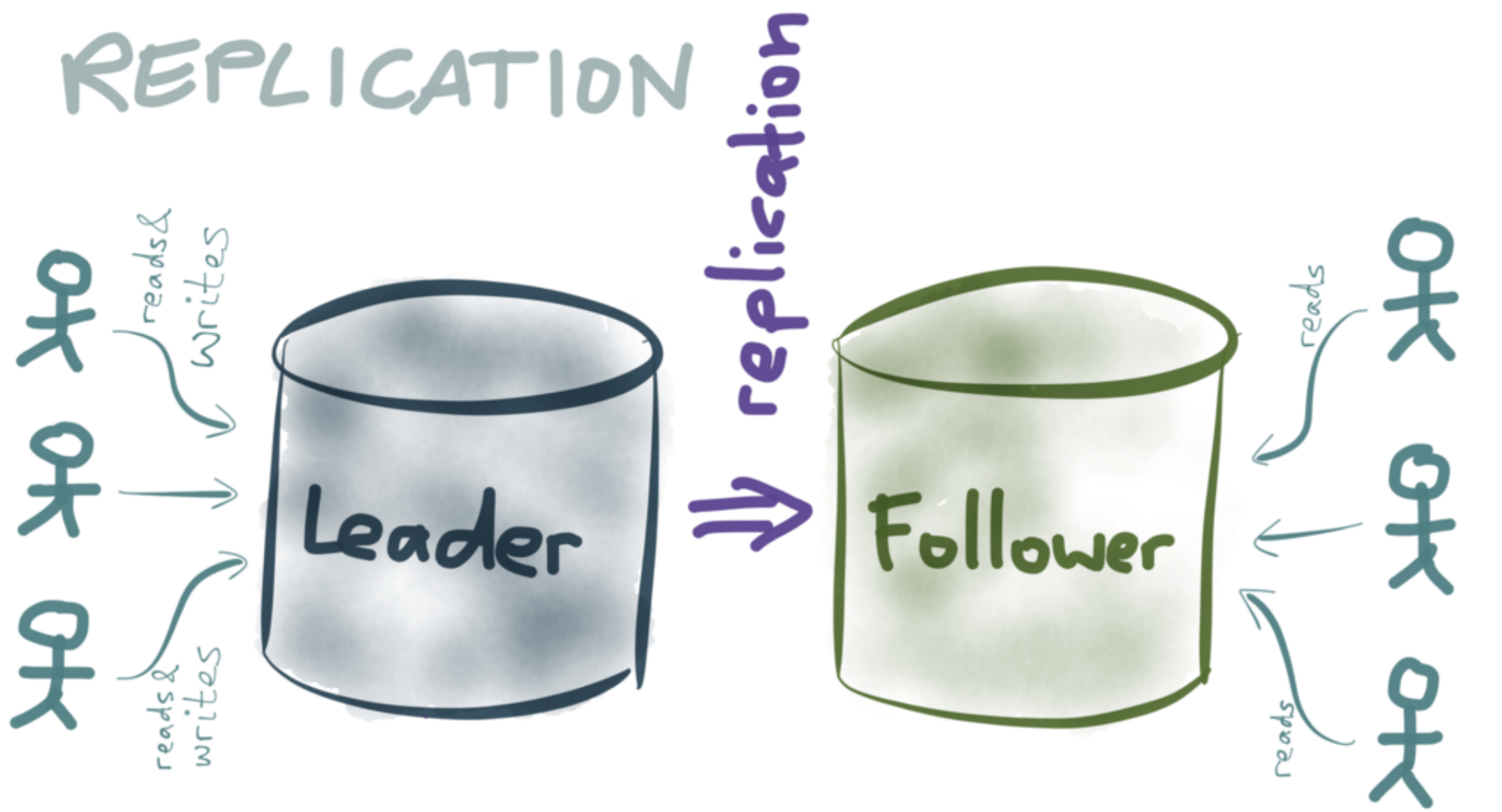
(Thompson 2011)

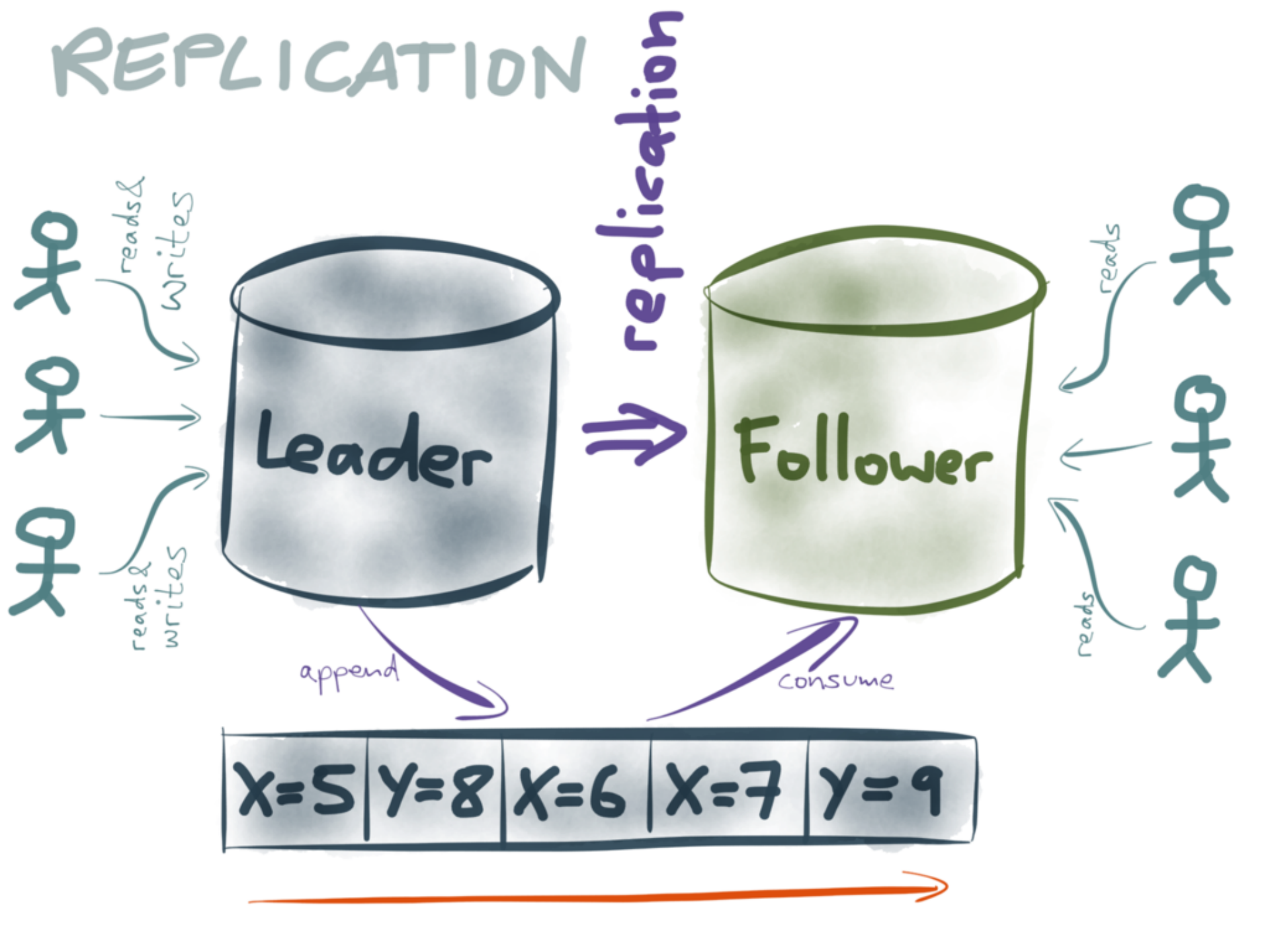
Event sourcing

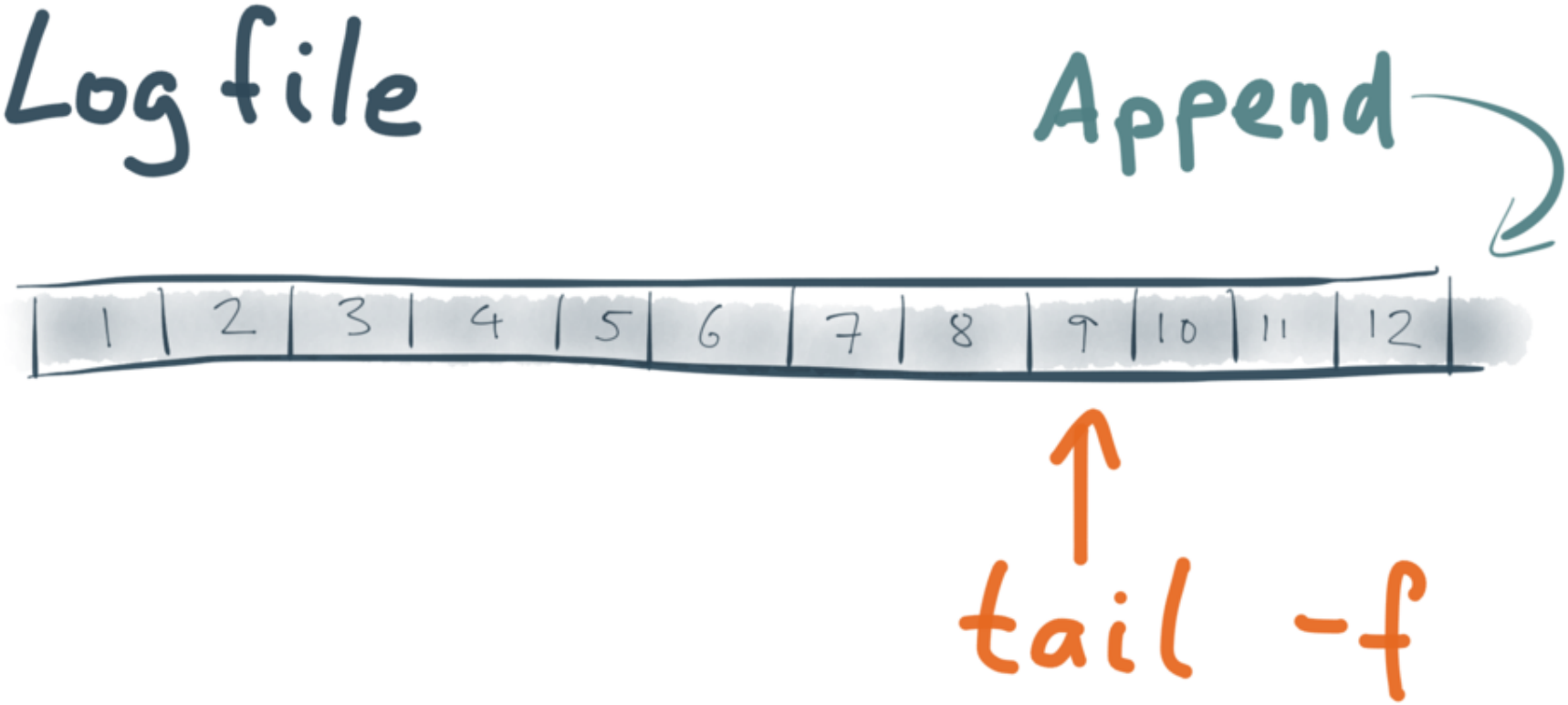
(Vernon 2013)

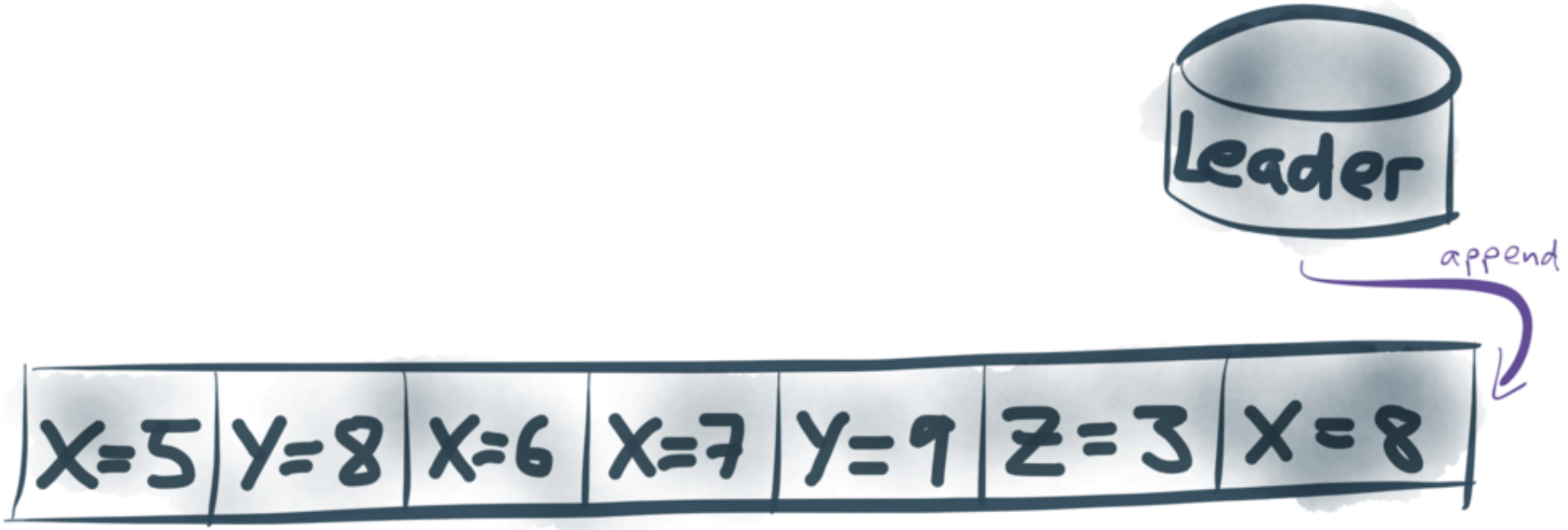
REPLICATION

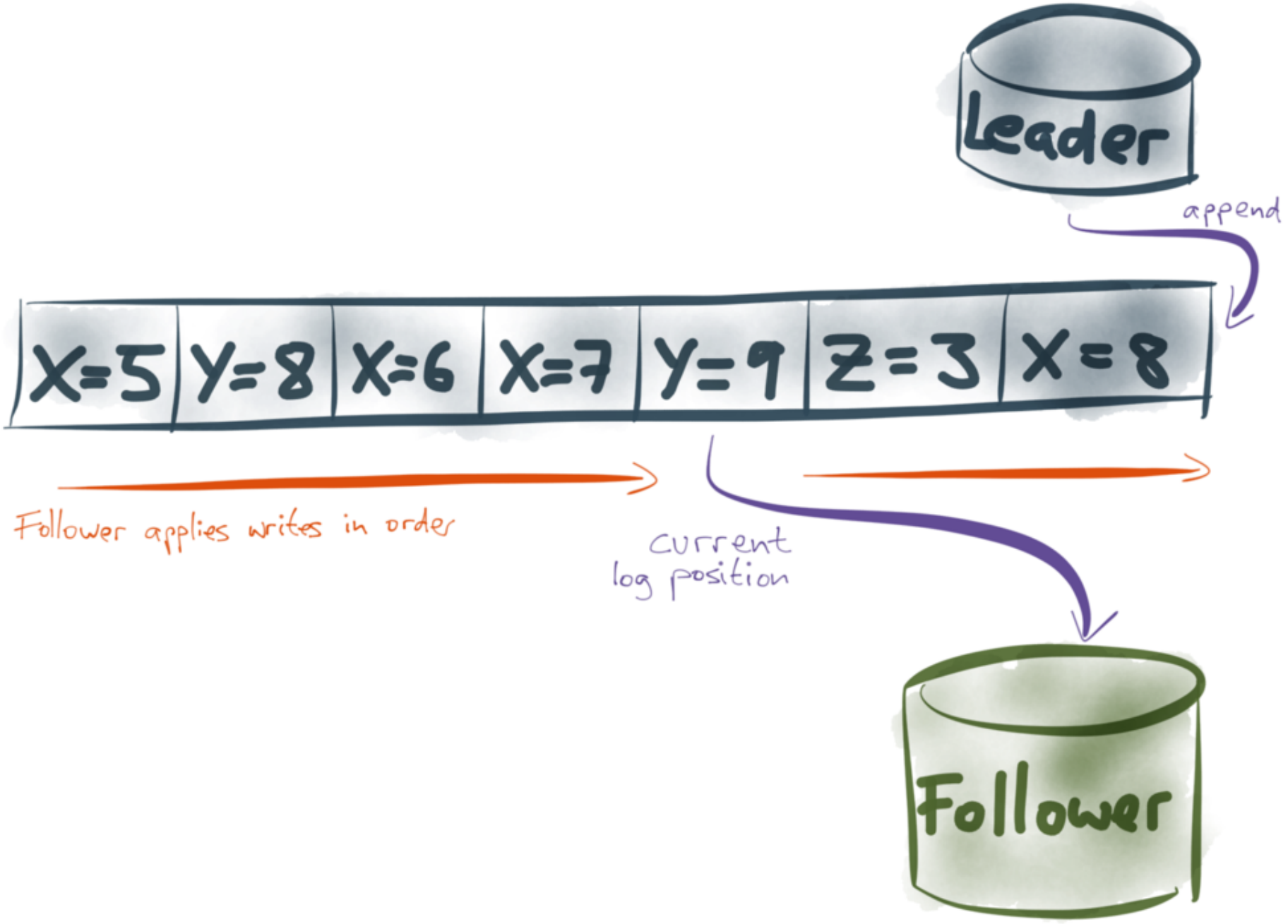


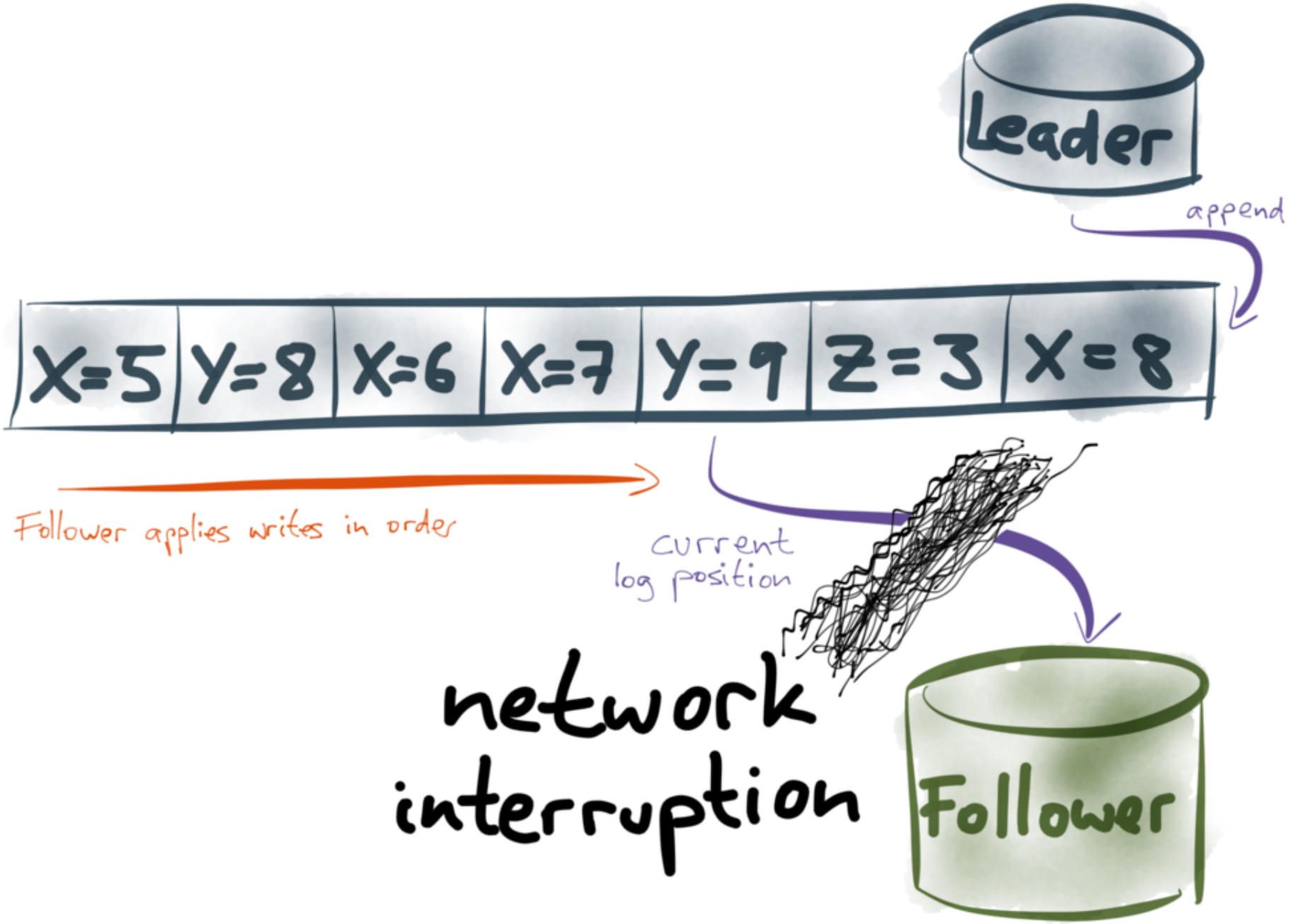

REPLICATION
reads & writes
reads & writes
Leader
replication
Follower
reads
reads

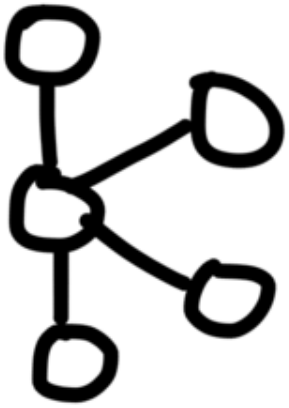

REPLICATION
replication
reads & writes
reads & writes
Leader
Follower
reads
reads
append
consume
X=5
Y=8
X=6
X=7
Y=9


Log file
Append
1
2
3
4
5
6
7
8
9
10
11
12
tail -f

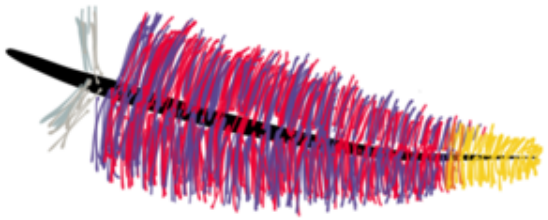

Leader
append
X=5
Y=8
X=6
X=7
Y=9
Z=3
X=8


Leader
append
X=5
Y=8
X=6
X=7
Y=9
Z=3
X=8
Follower applies writes in order
current
log position
Follower

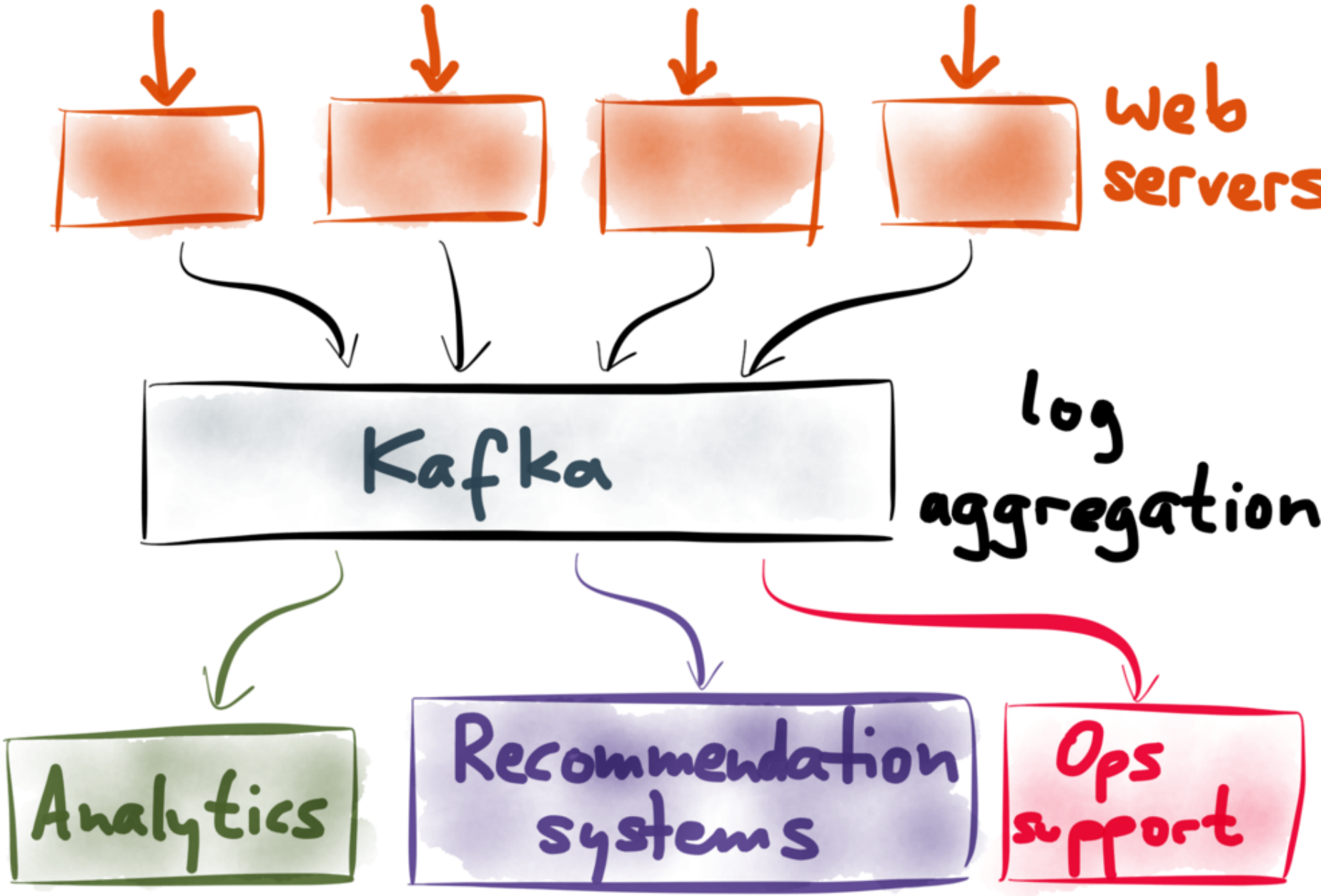

Leader
append
X=5
Y=8
X=6
X=7
Y=9
Z=3
X=8
Follower applies writes in order
current
log position
network
interruption
Follower



Kafka



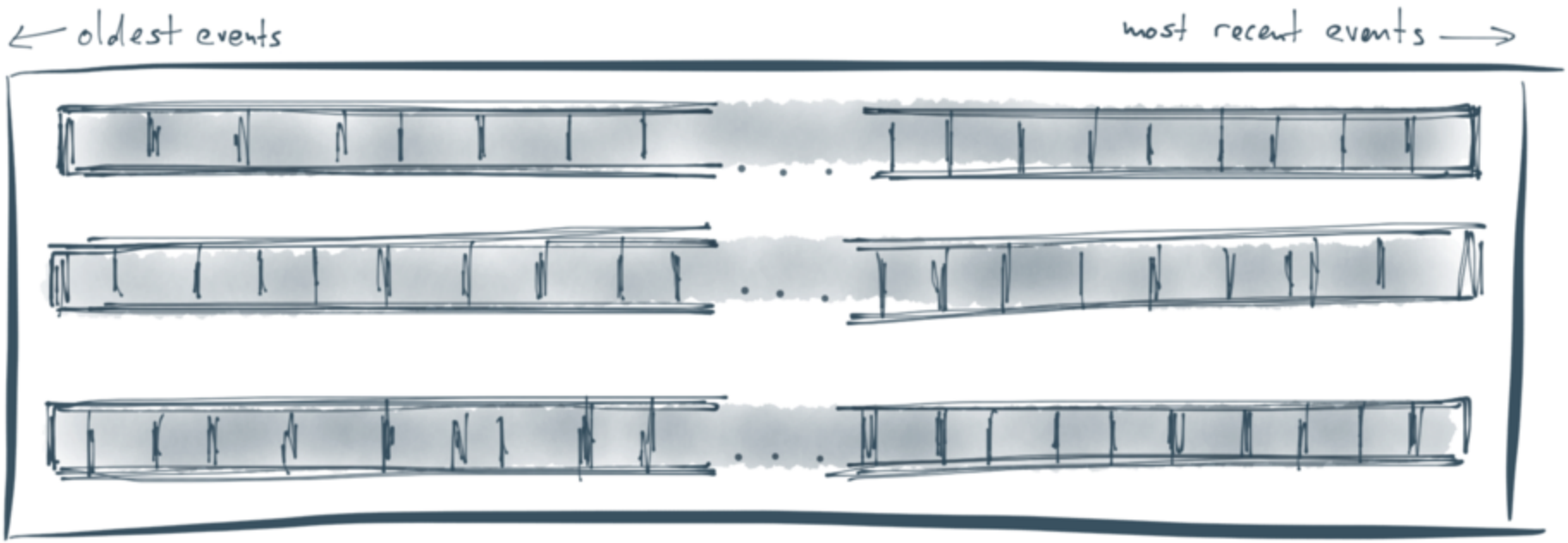
APACHE
SOFTWARE
FOUNDATION

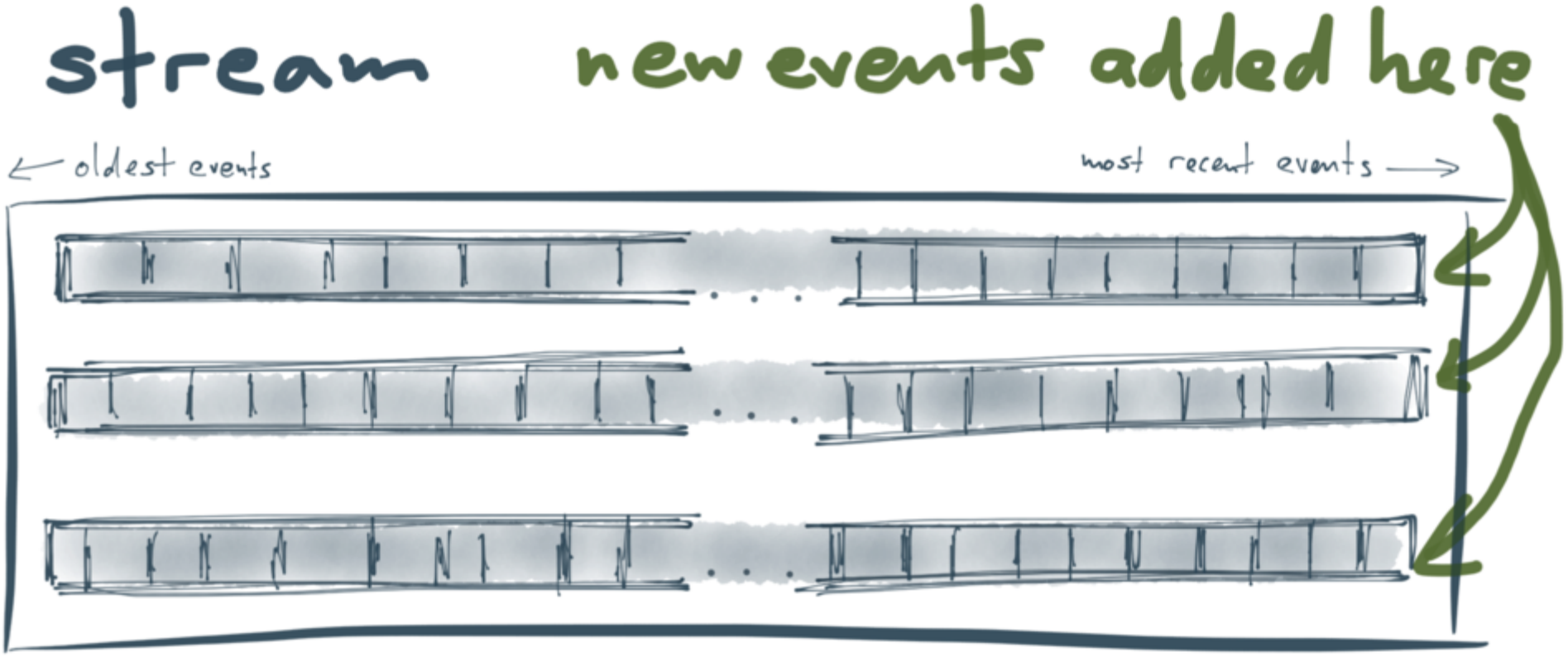

web servers
Kafka
log aggregation
Analytics
Recommendation systems
Ops support

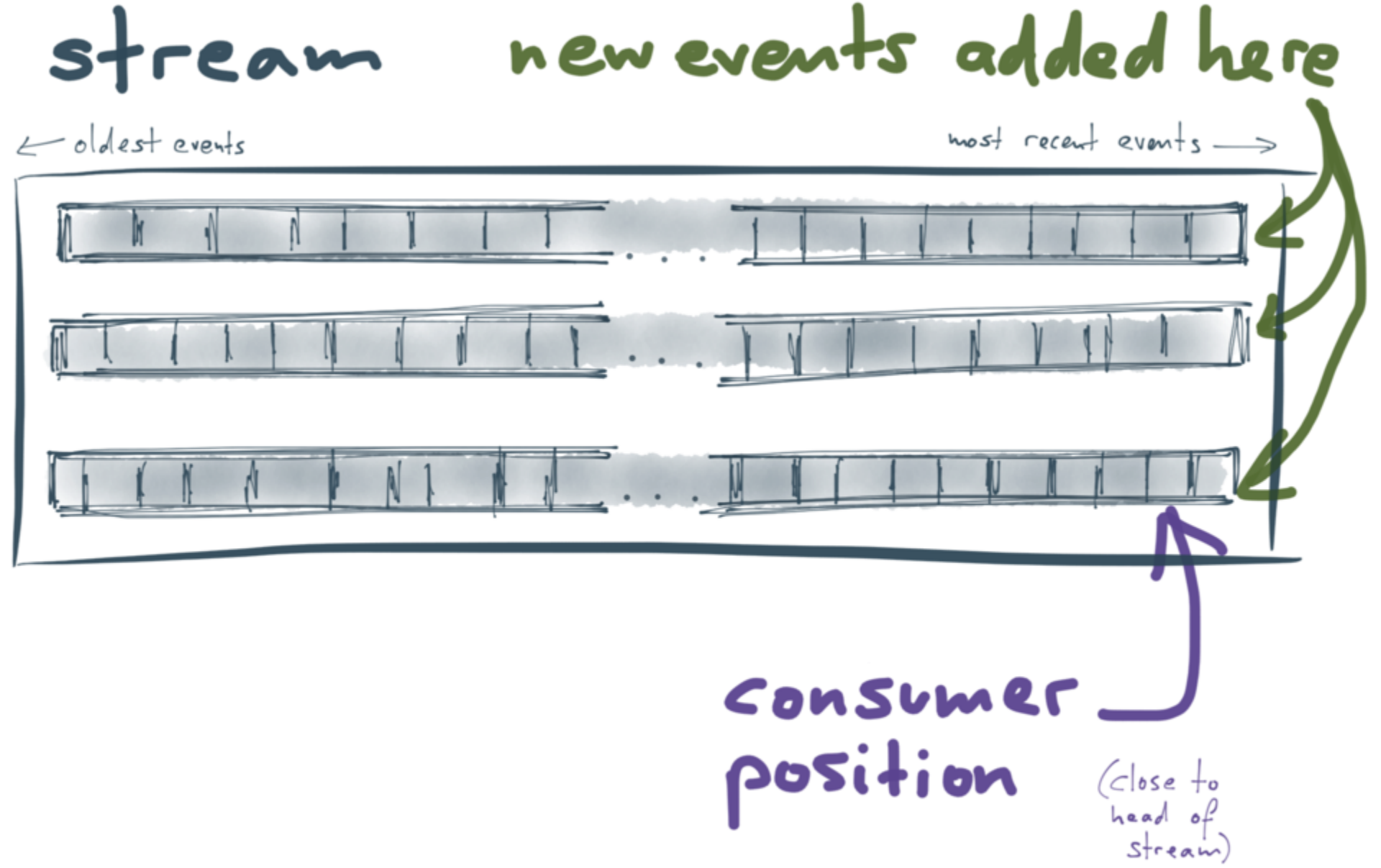
Log file/
stream

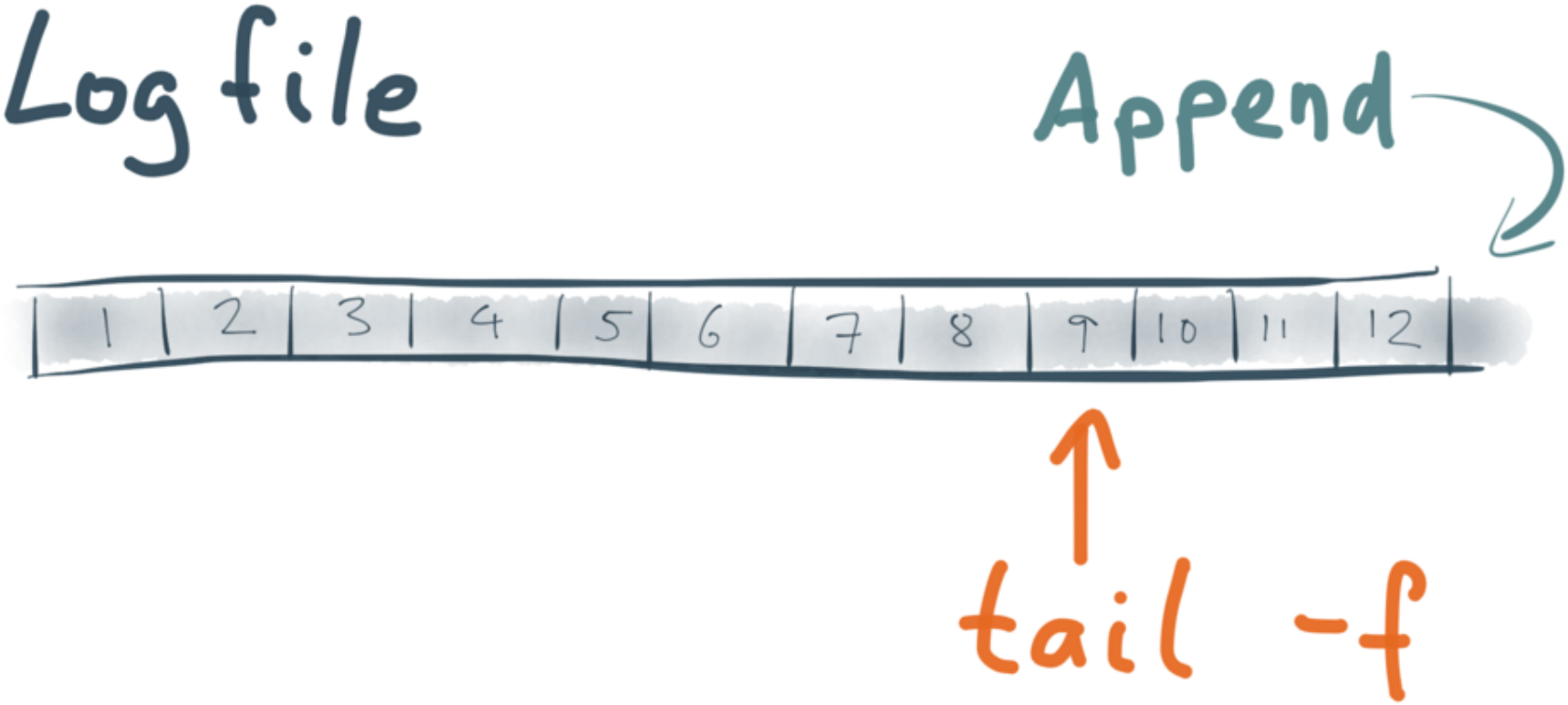
request: GET
url: /hello.html
user: 123456
client_ip: 12.34.5.6
browser: Chrome 90
referrer: google.com
timestamp: 14255006

stream

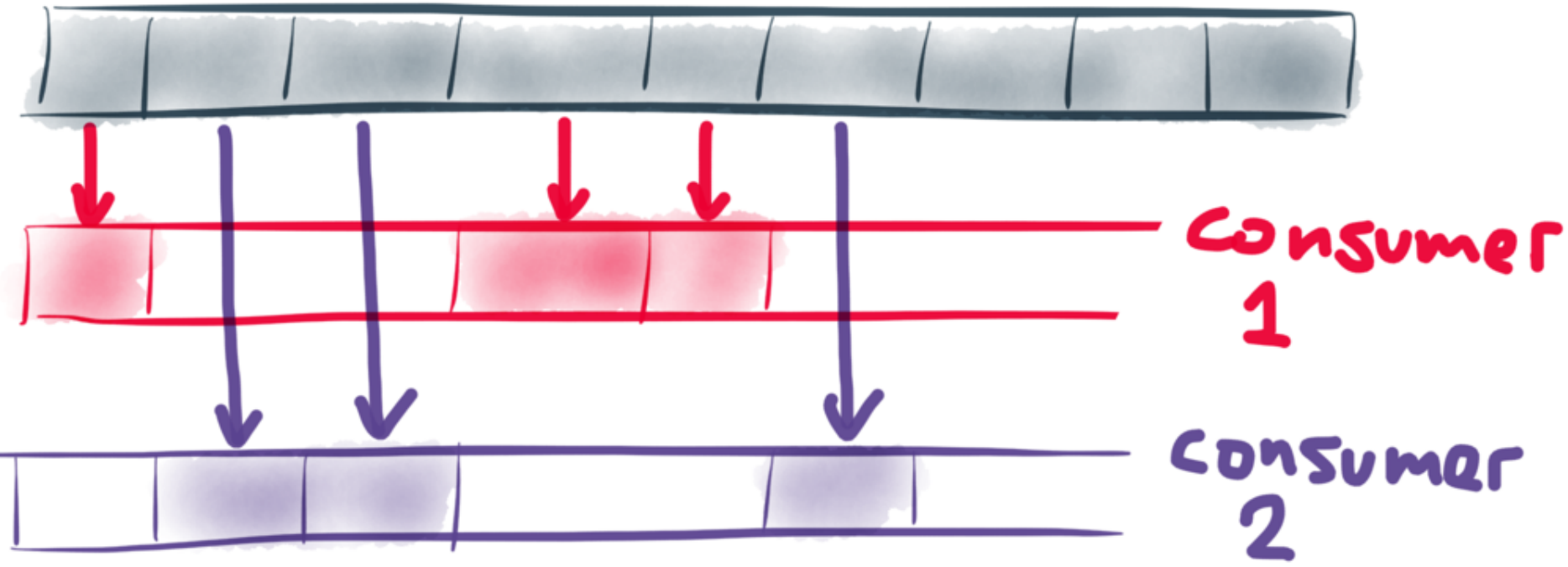



stream
new events added here
← oldest events
most recent events →

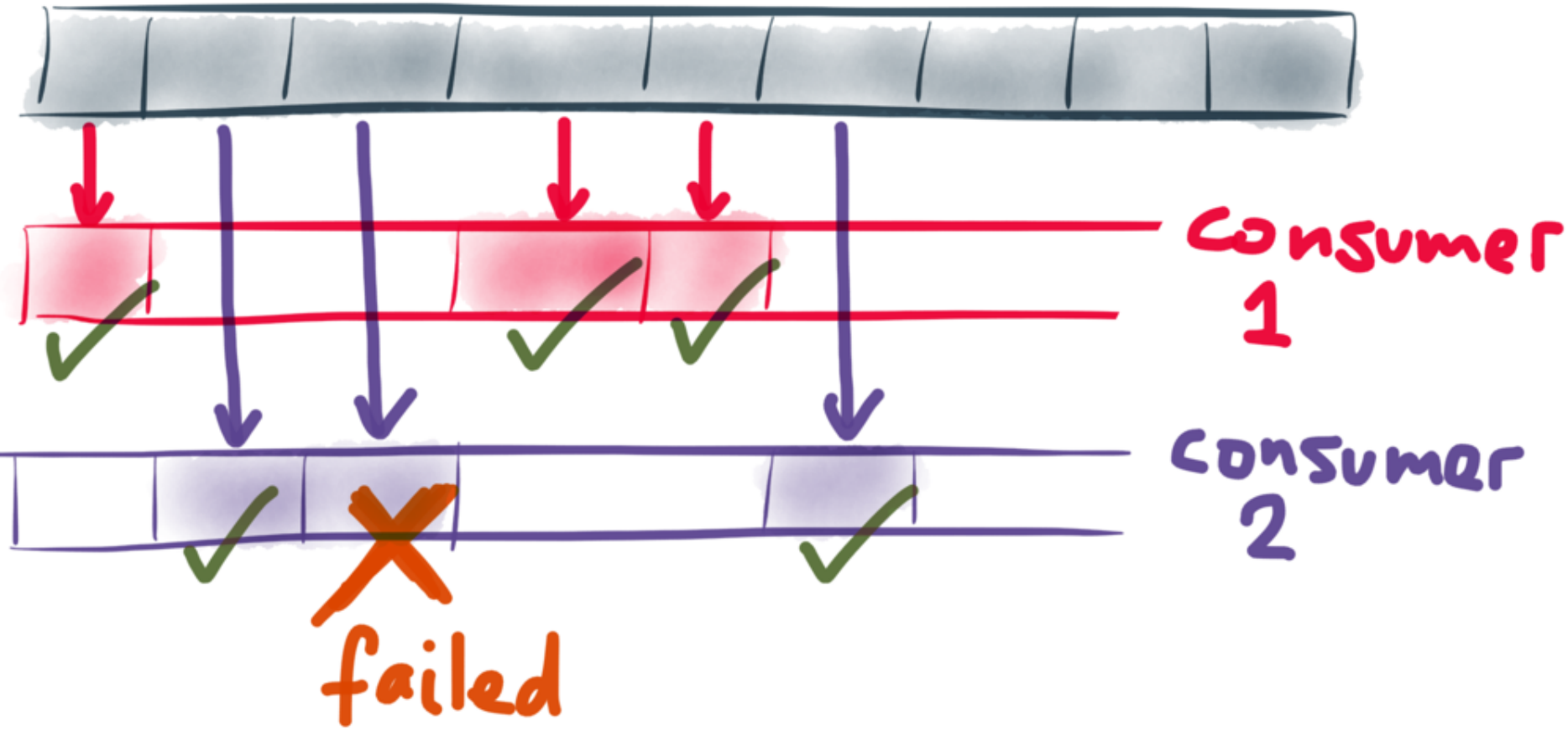

stream
new events added here
← oldest events
most recent events →
consumer position
(close to head of stream)


Log file
Append
1
2
3
4
5
6
7
8
9
10
11
12
tail -f

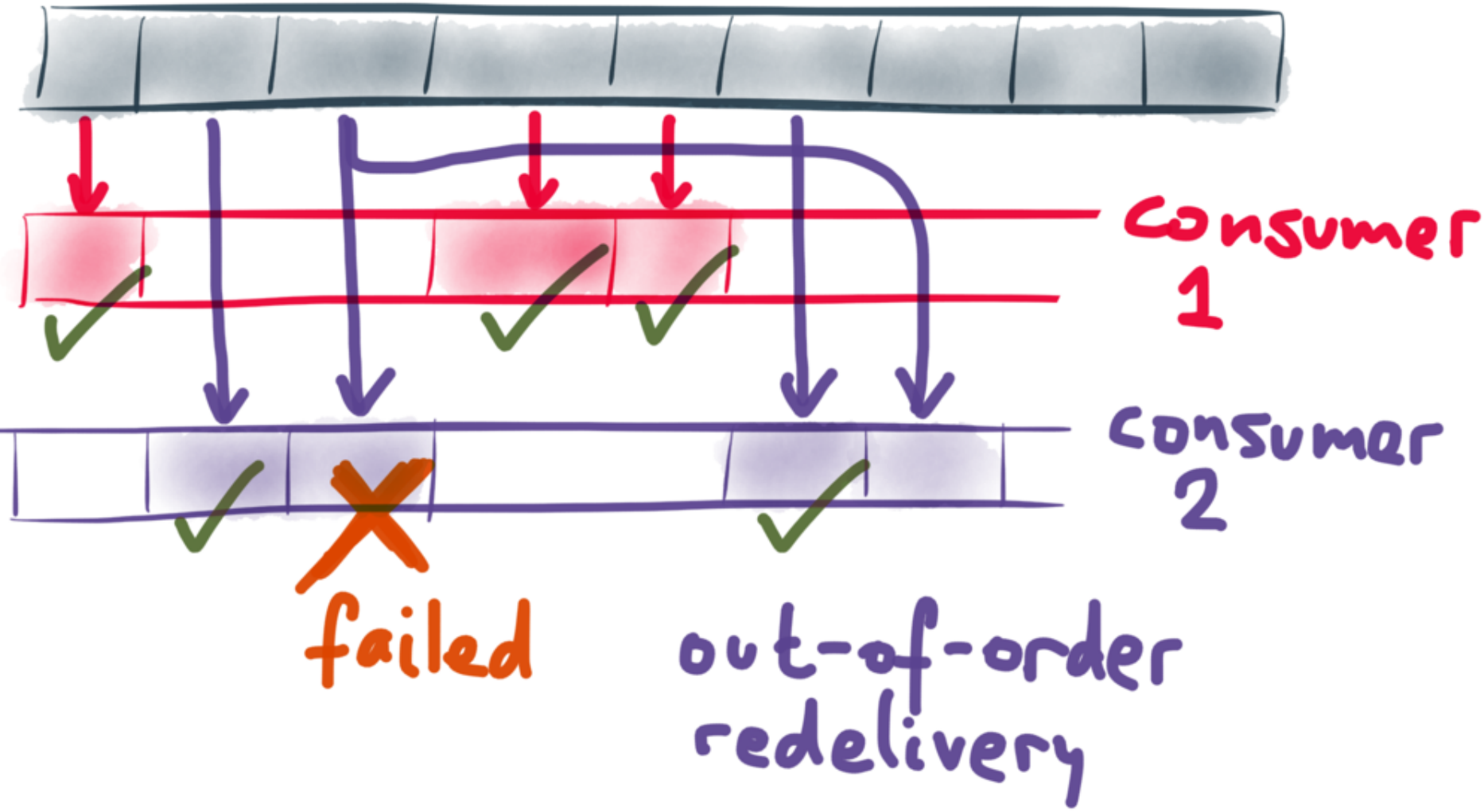
Comparison: AMQP



Comparison: AMQP



Comparison: AMQP



Job queue

"please send this email for me"

"please charge this credit card"

vs.

Event log

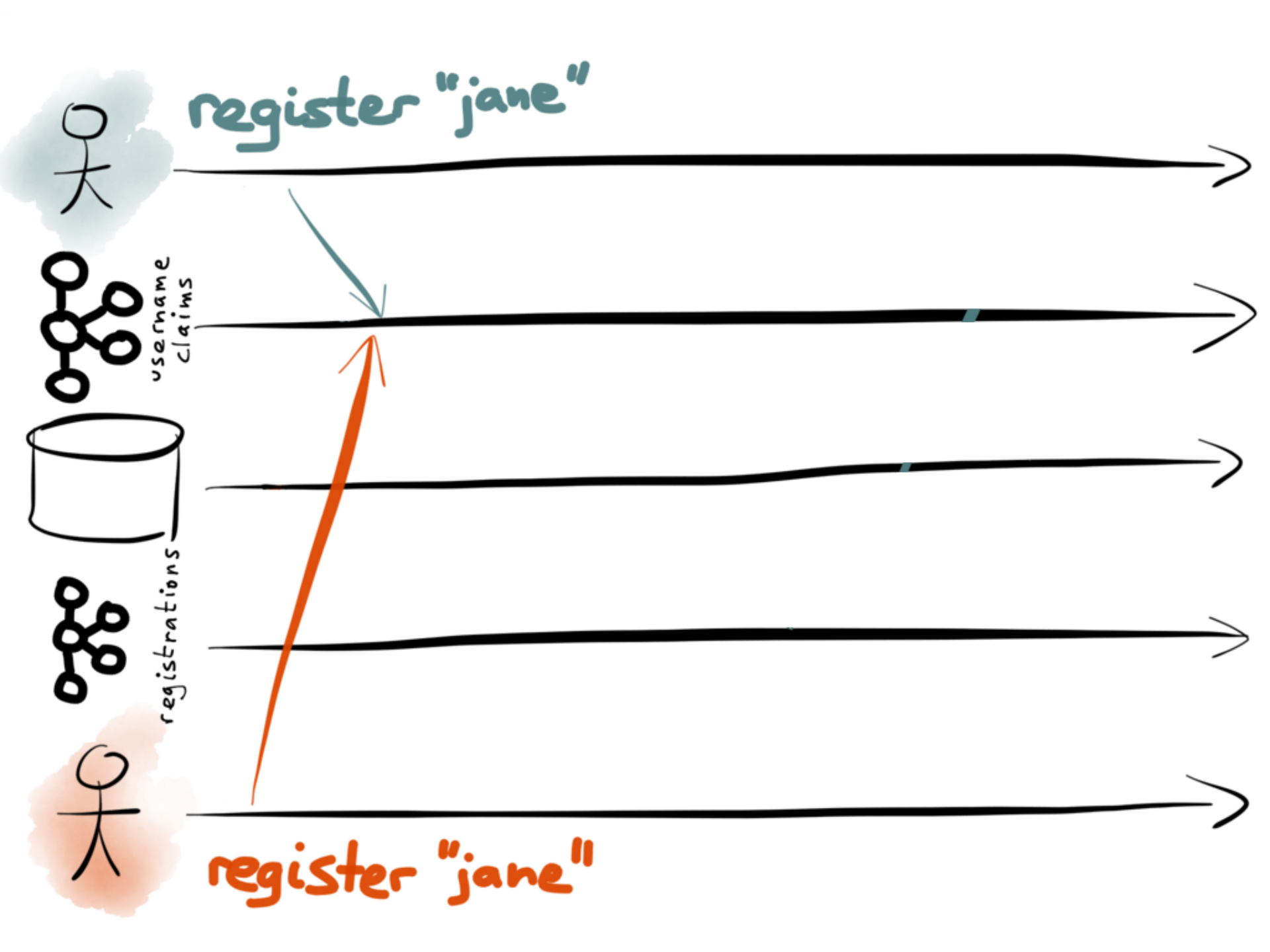
"user viewed a web page"

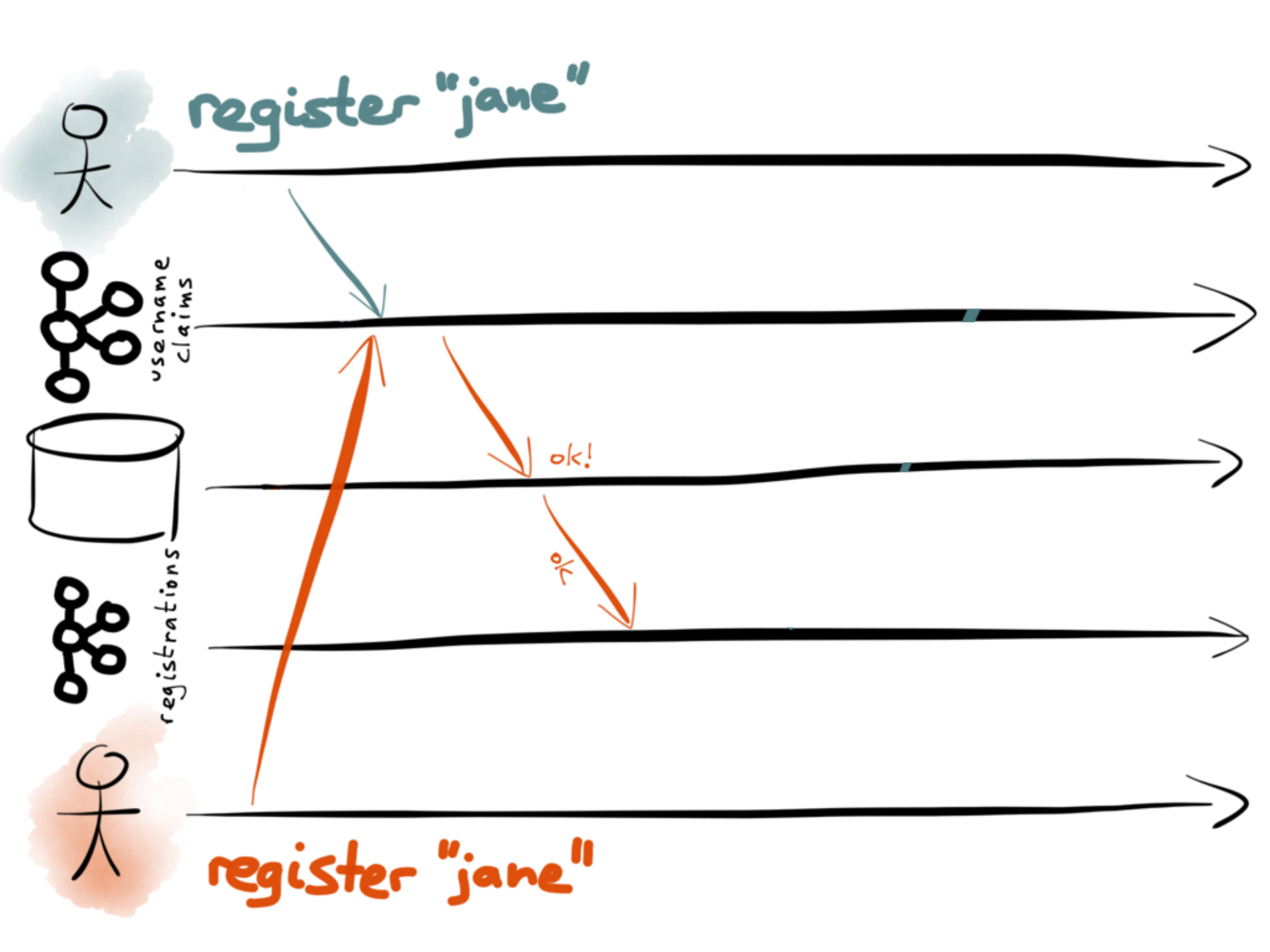
"customer X purchased product Y"

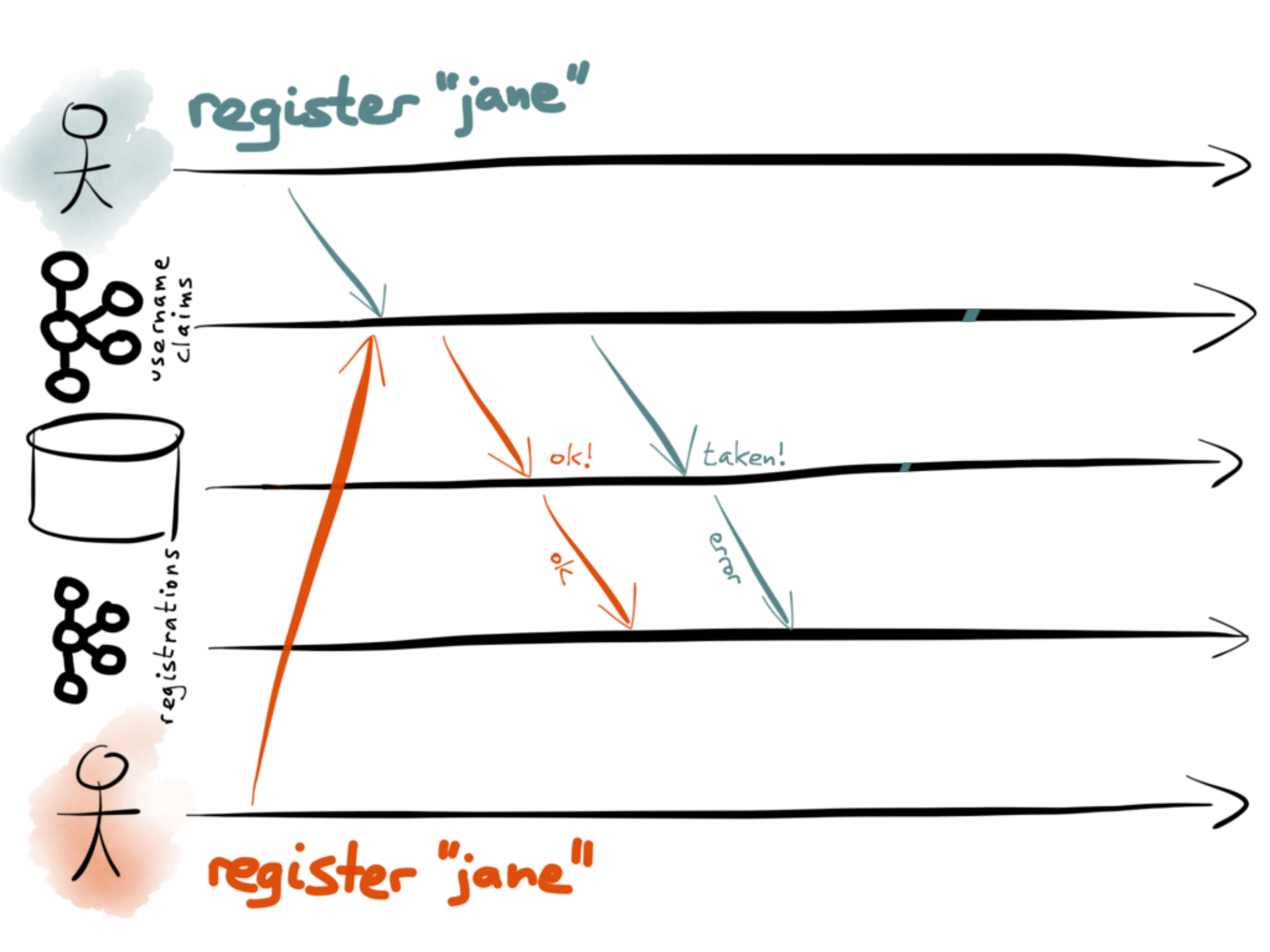
Enforcing invariants
Integrity constraints

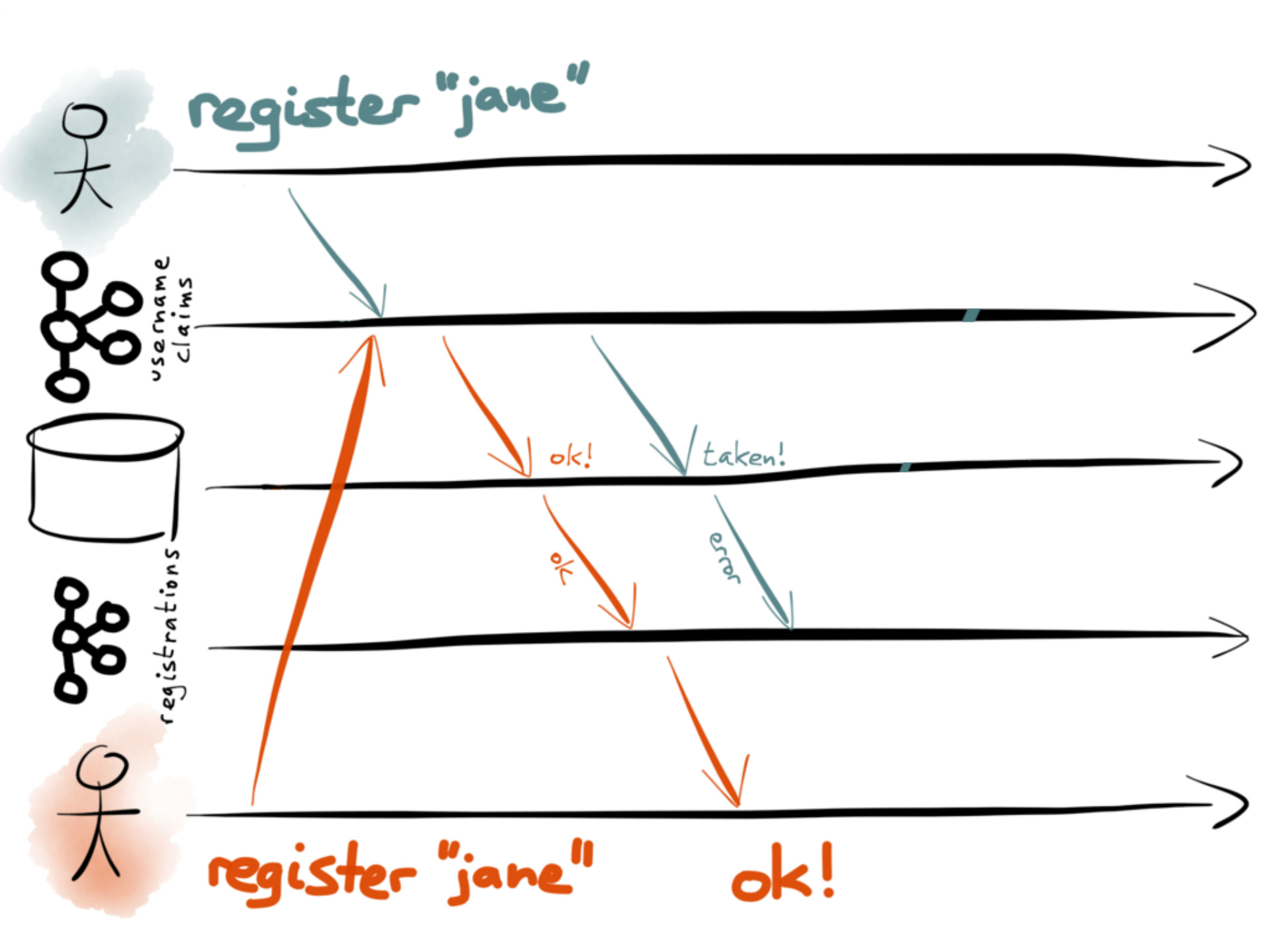
unique username

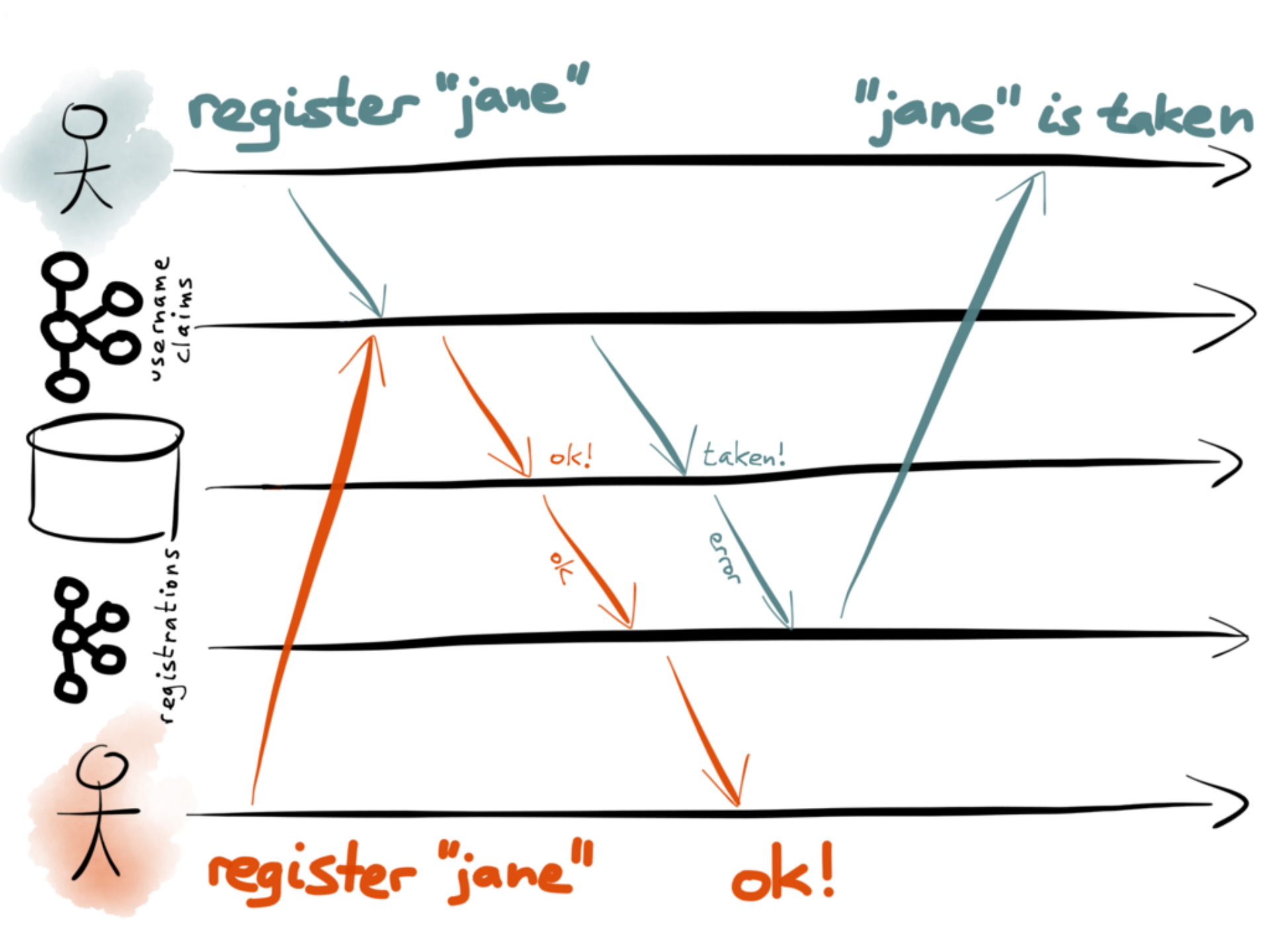
account balance positive

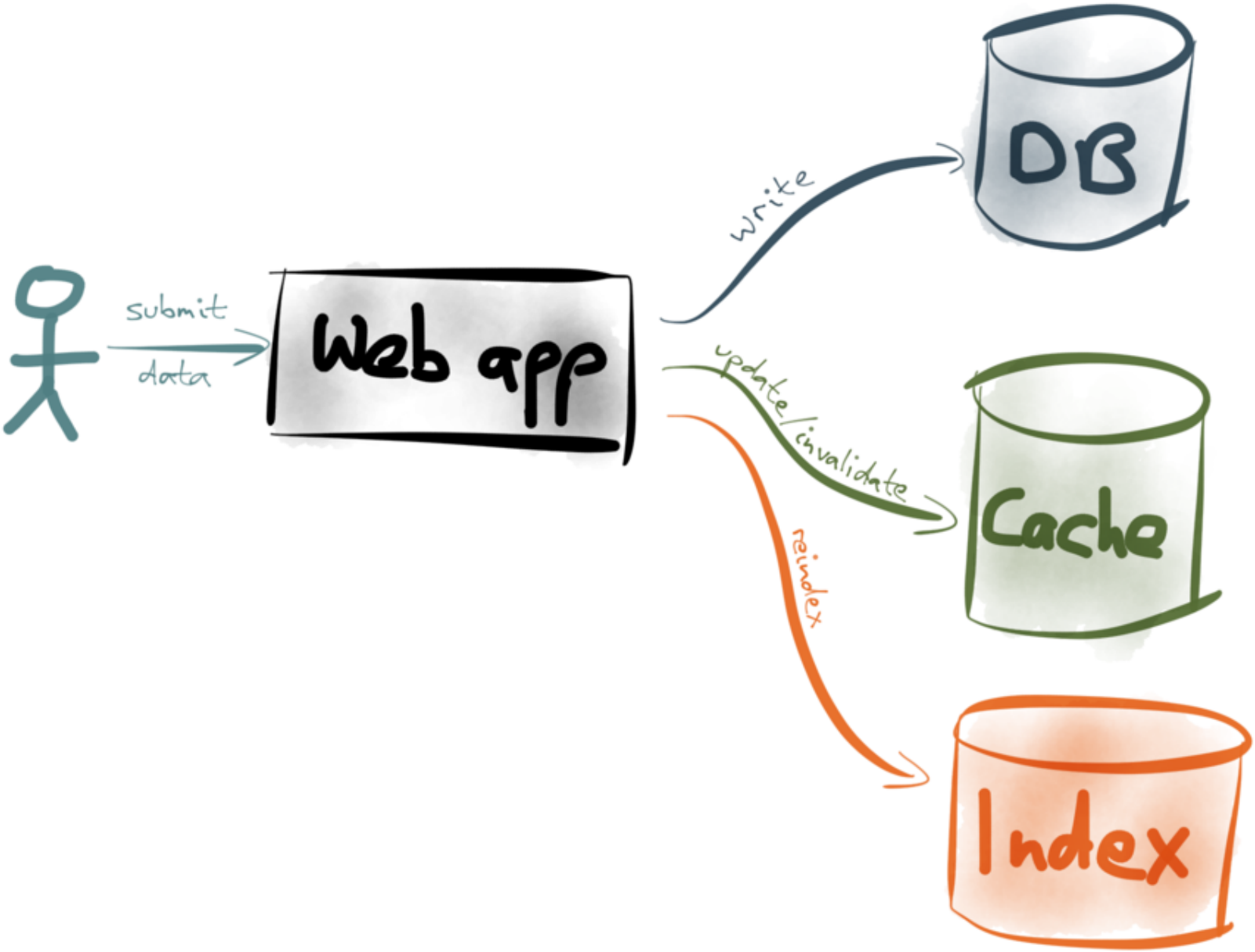

register "jane"
username claims
registrations
register "jane"


register "jane"
username claims
registrations
ok!
ok
register "jane"

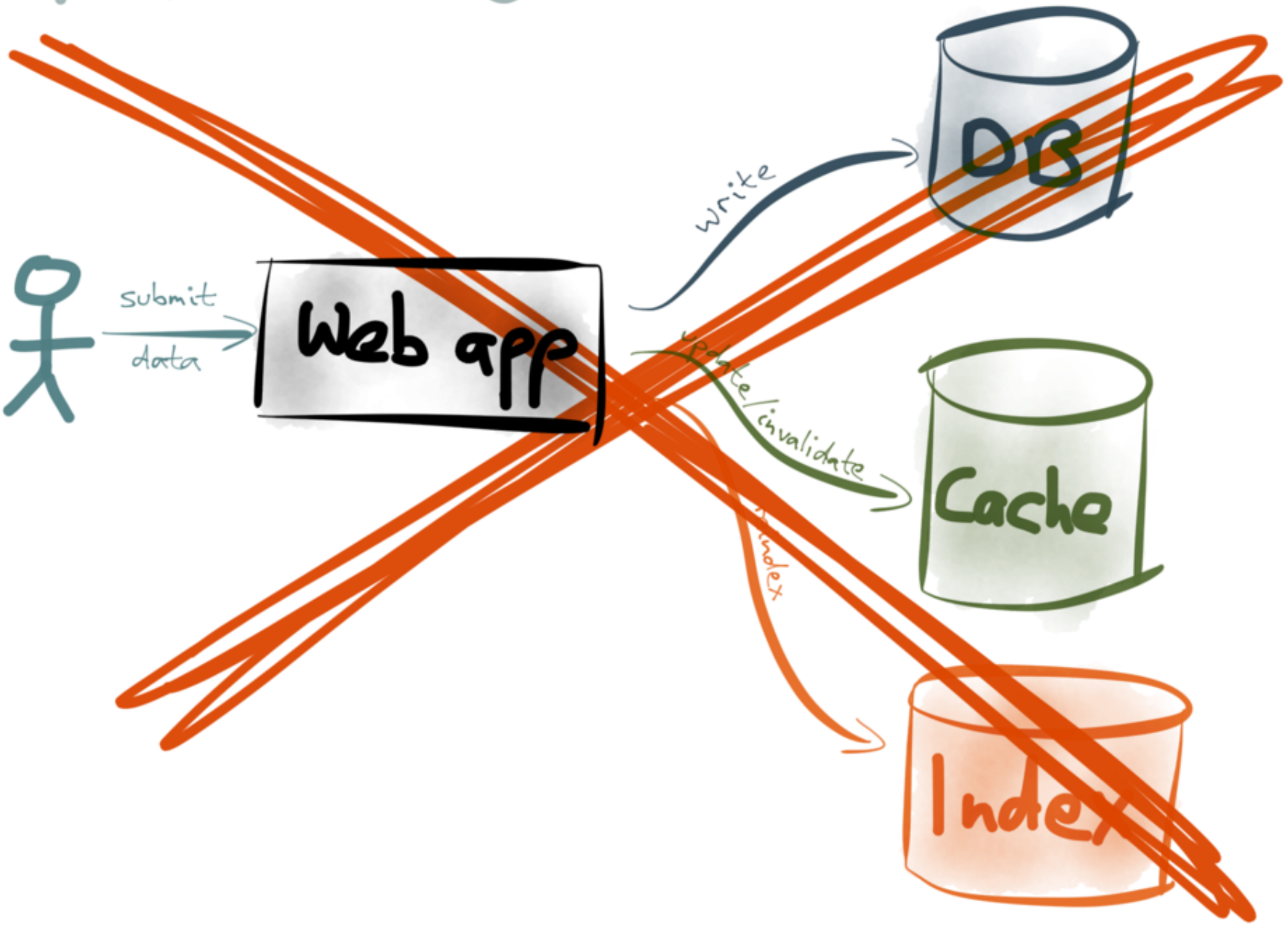

register "jane"
username claims
registrations
ok!
taken!
ok
error
register "jane"


register "jane"
username claims
registrations
ok!
taken!
ok
error
register "jane"
ok!

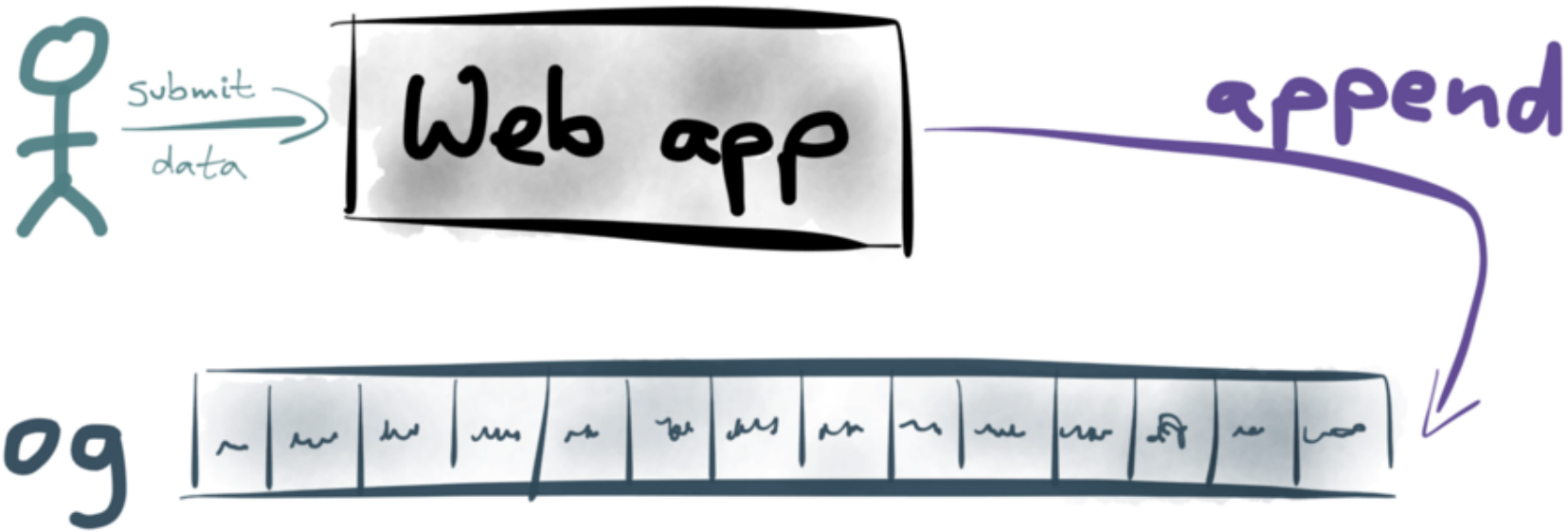

register "jane"
"jane" is taken
username claims
registrations
ok!
taken!
ok
error
register "jane"
ok!


submit
data
Web app
write
DB
update/invalidate
Cache
reindex
Index

STOP DOING THIS.



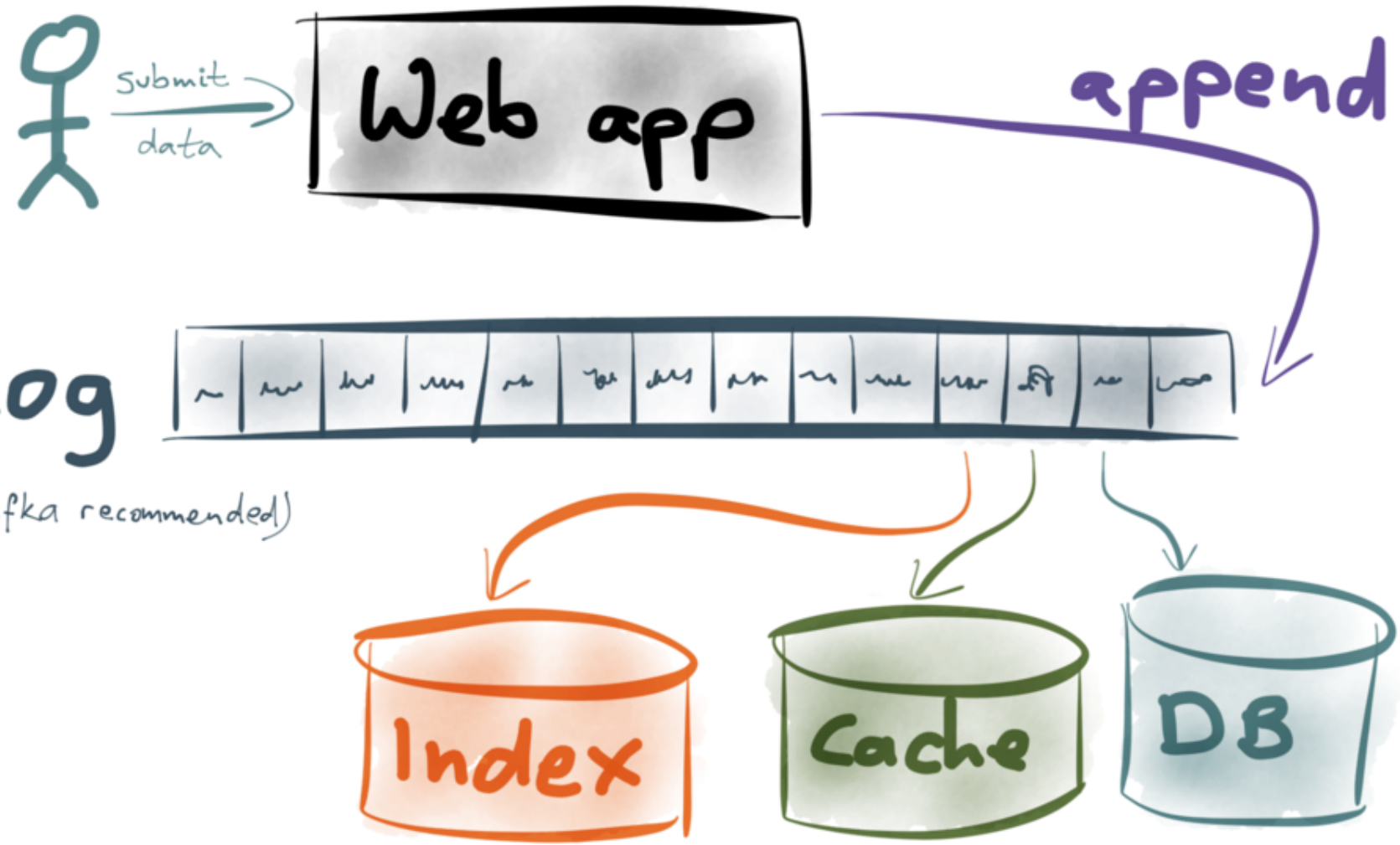
INSTEAD, EMBRACE THE LOG



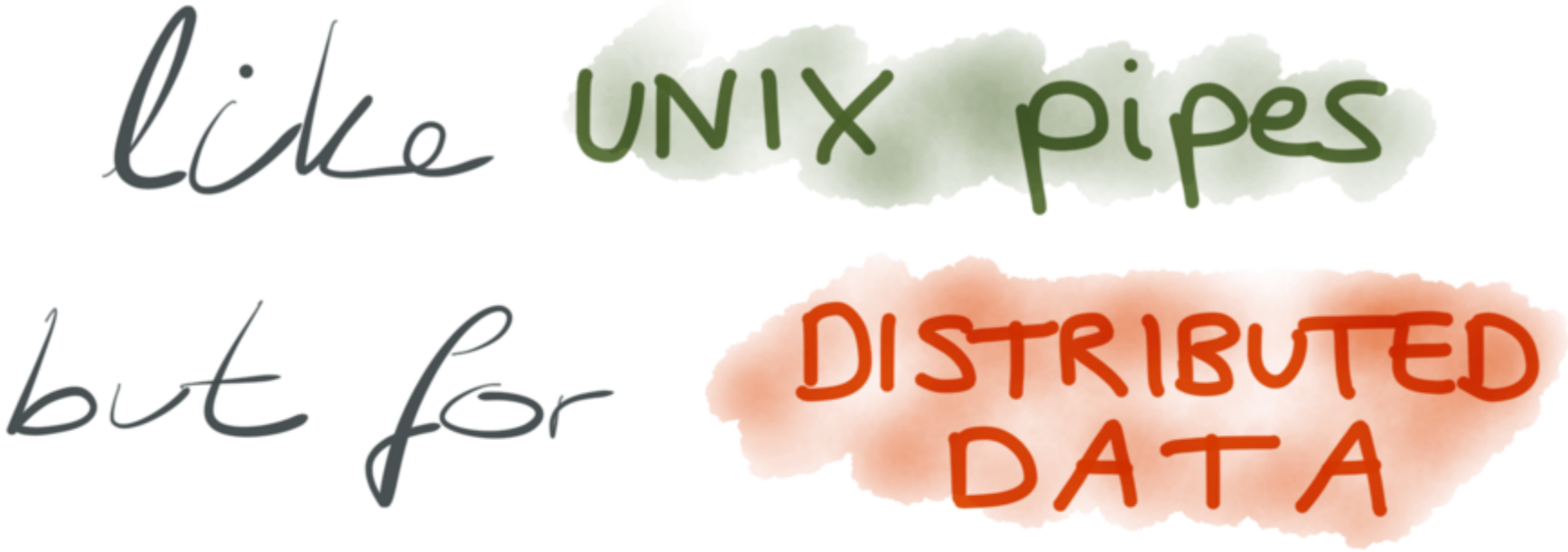
Log

(Kafka recommended)

INSTEAD, EMBRACE THE LOG

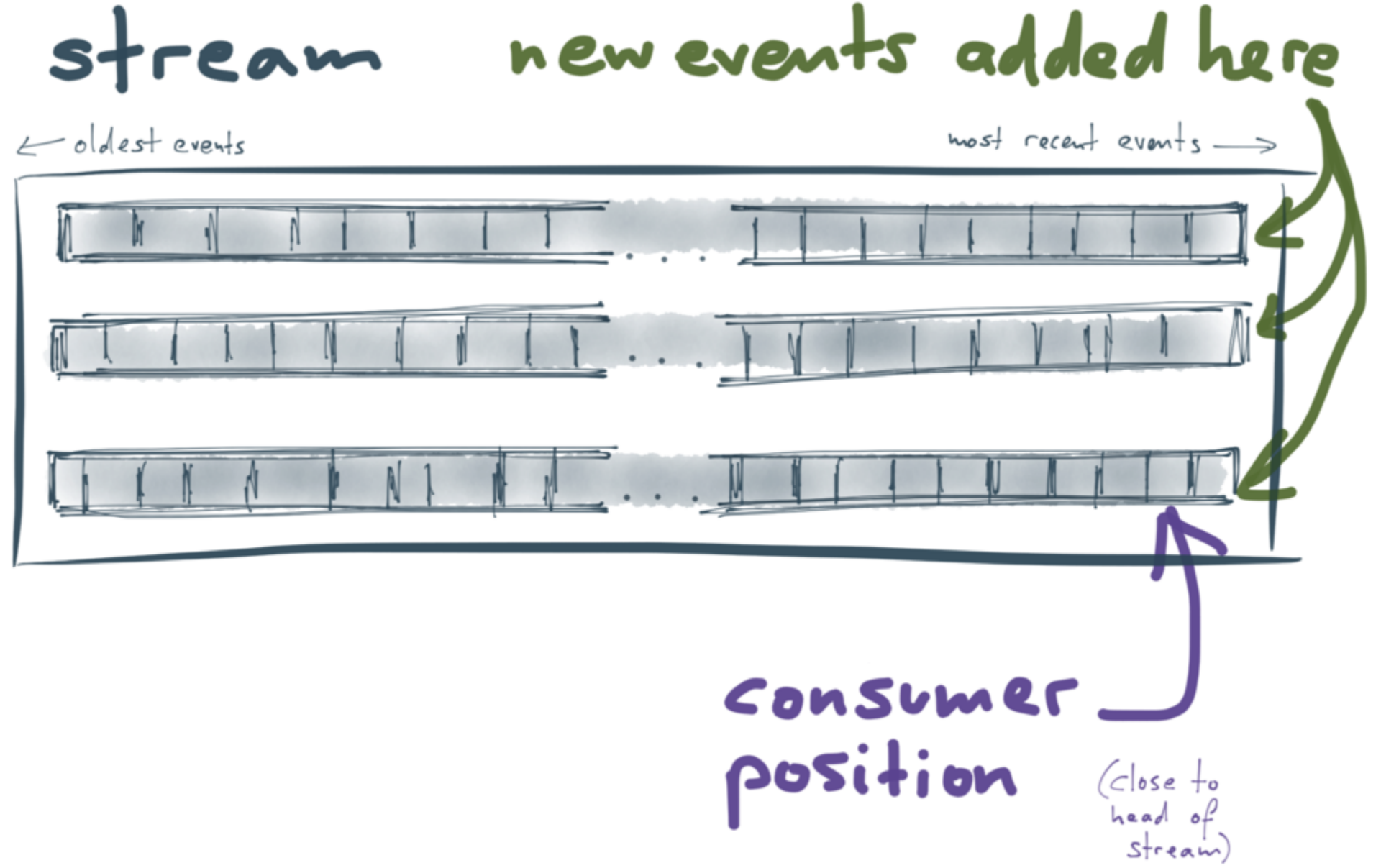


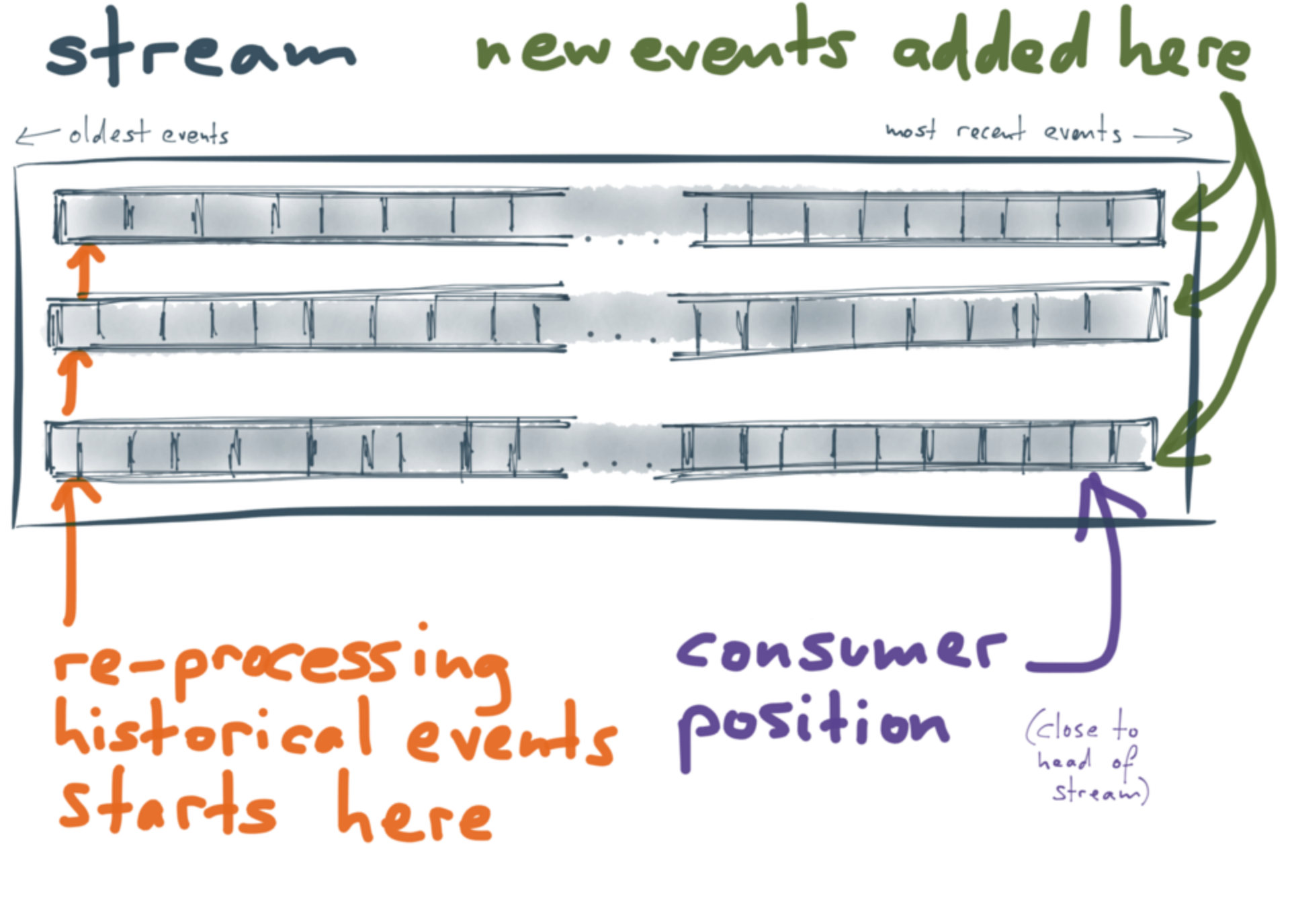
Consumers independently apply writes in LOG ORDER

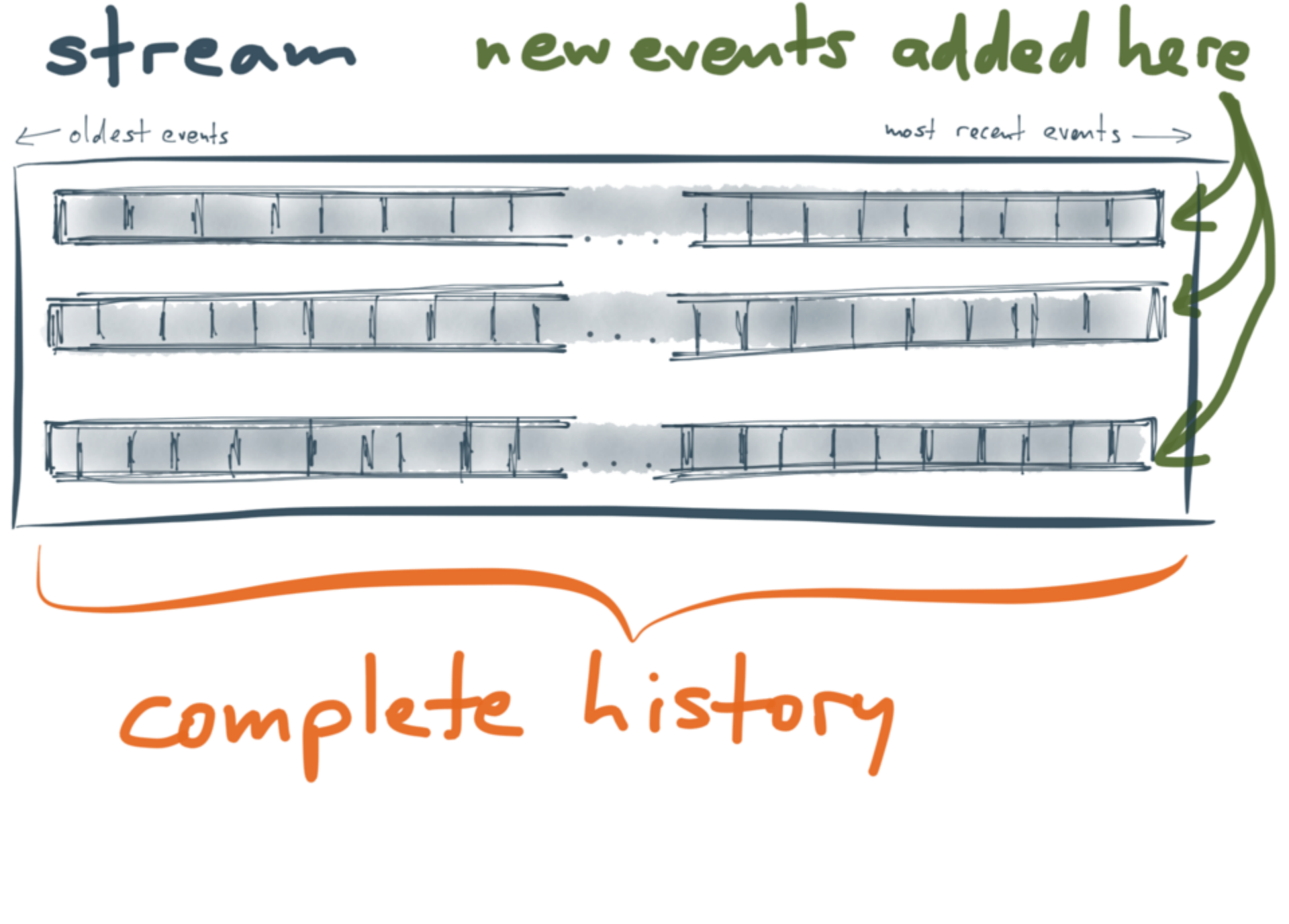


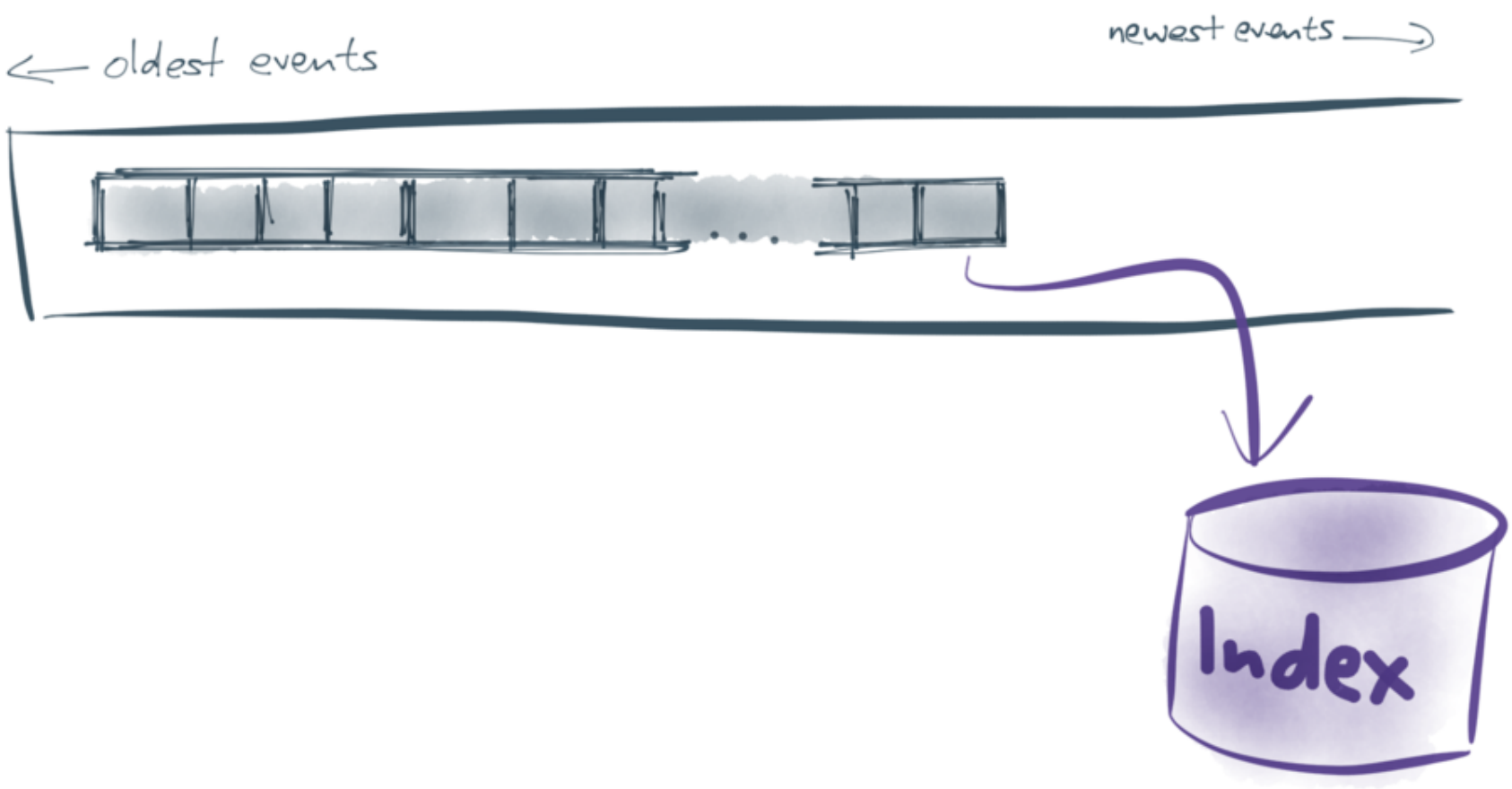
mysql | elasticsearch

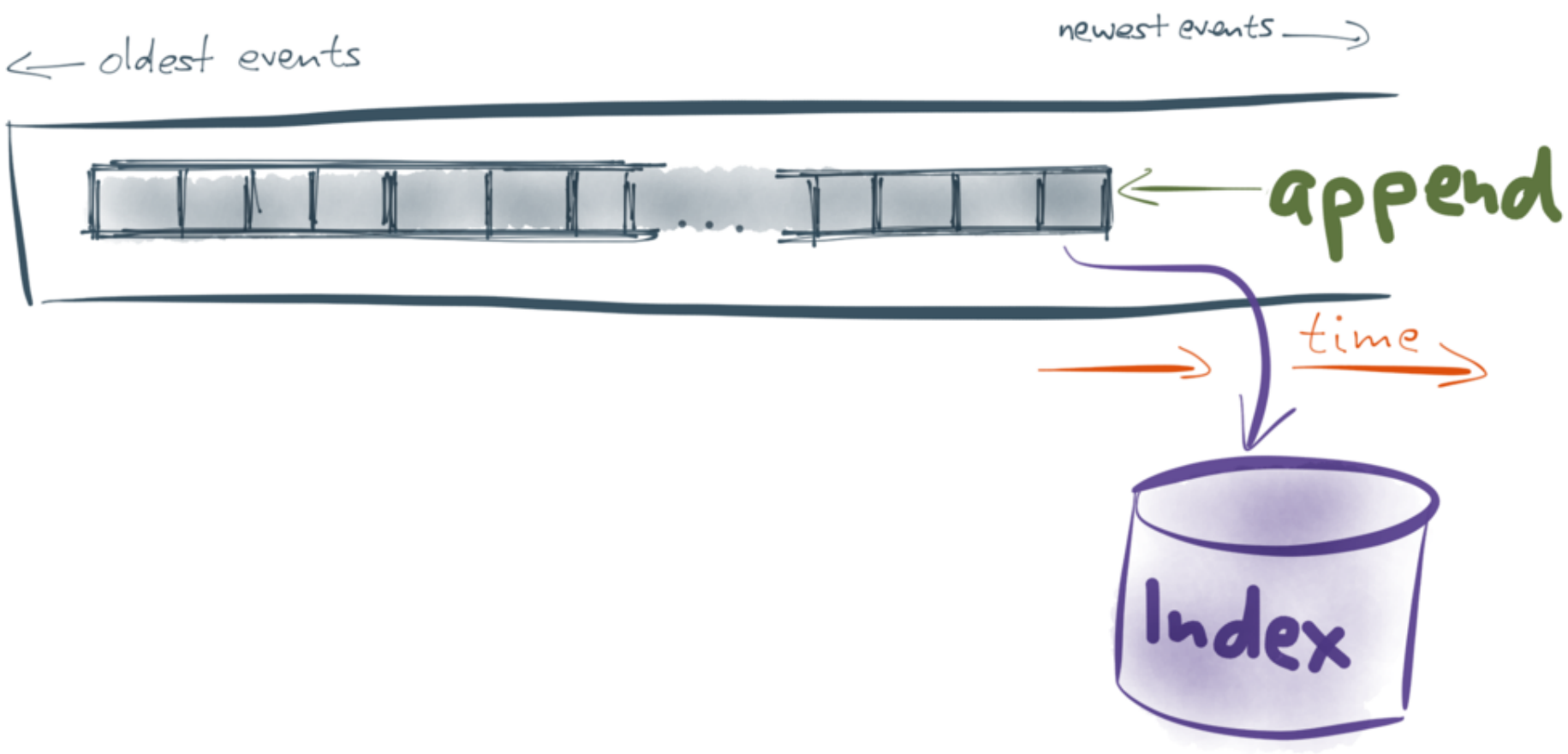
kafka | sed | awk | memcached

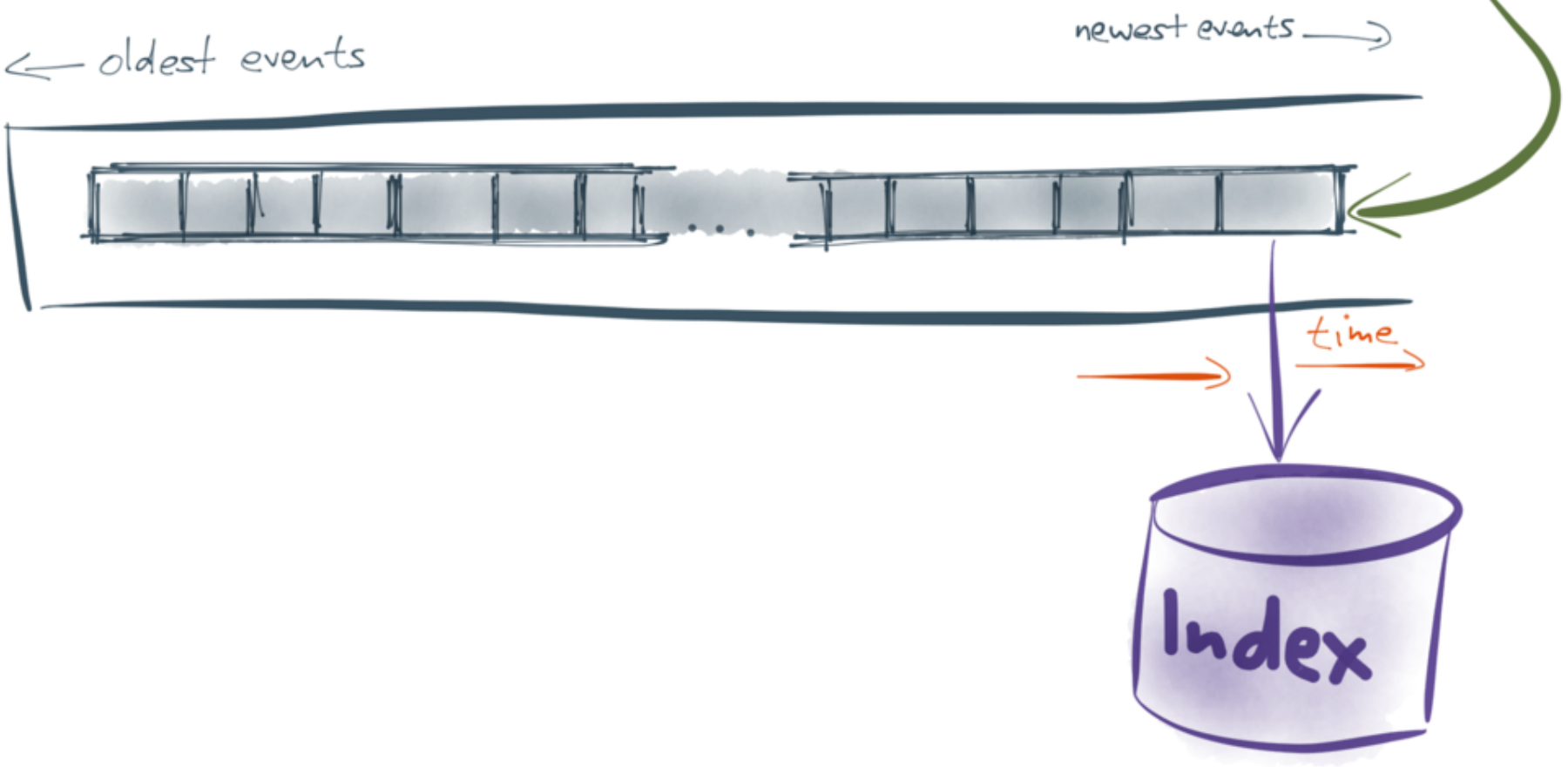

stream
new events added here
← oldest events
most recent events →
consumer position
(close to head of stream)

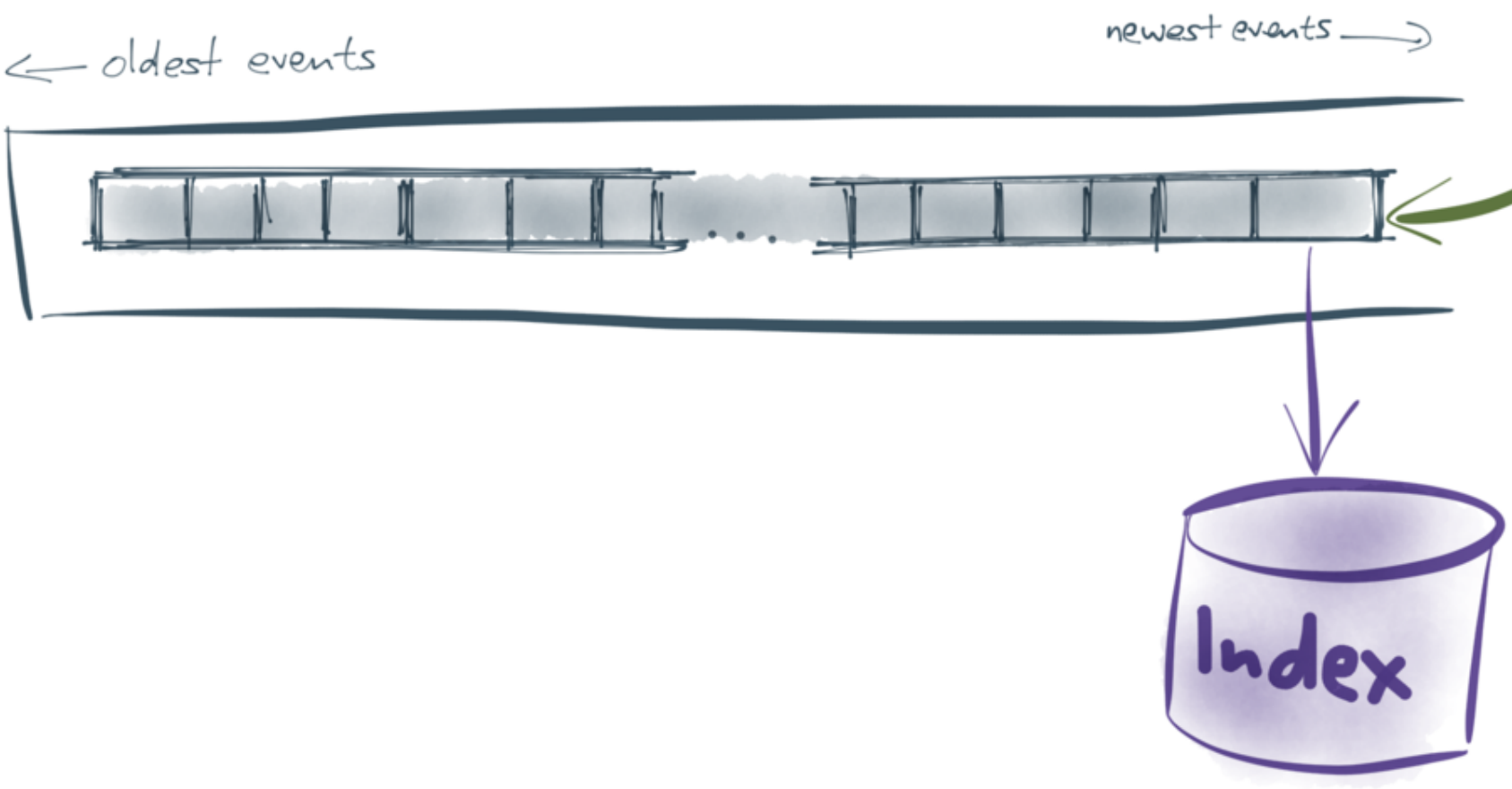

stream
new events added here
← oldest events
most recent events →
re-processing historical events starts here
consumer position
(close to head of stream)


stream
new events added here
← oldest events
most recent events →
complete history

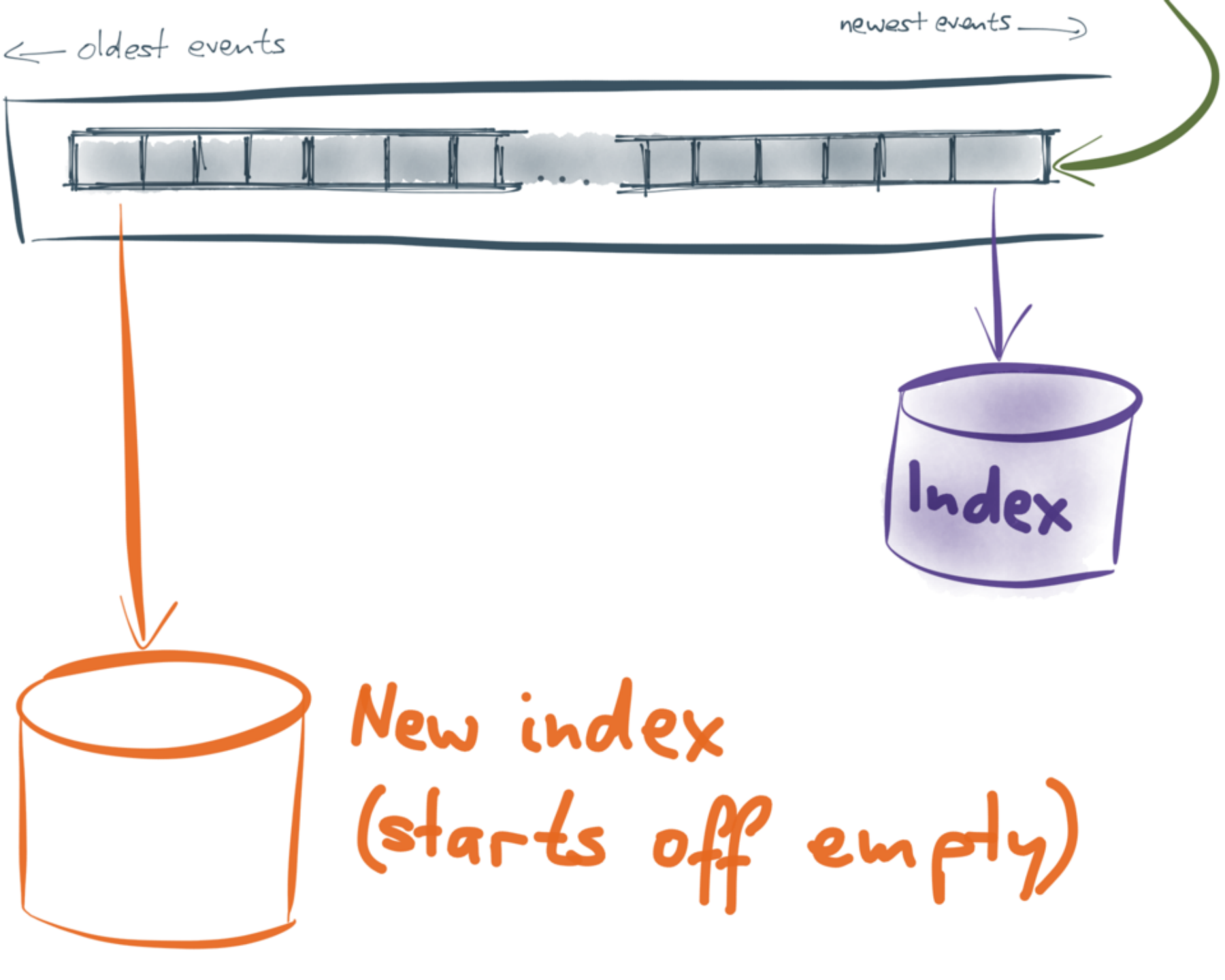

oldest events
newest events
Index

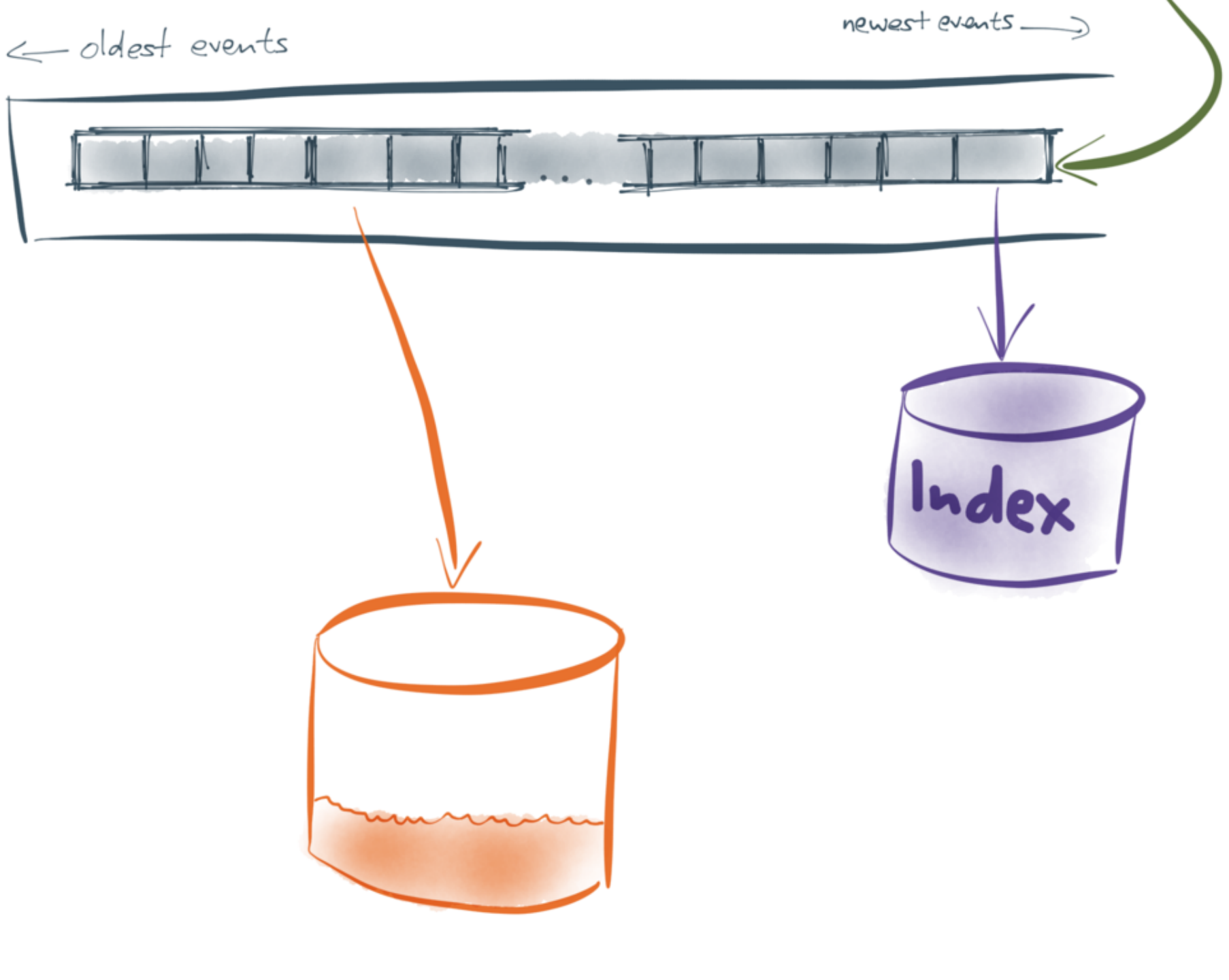

oldest events
newest events
append
time
Index

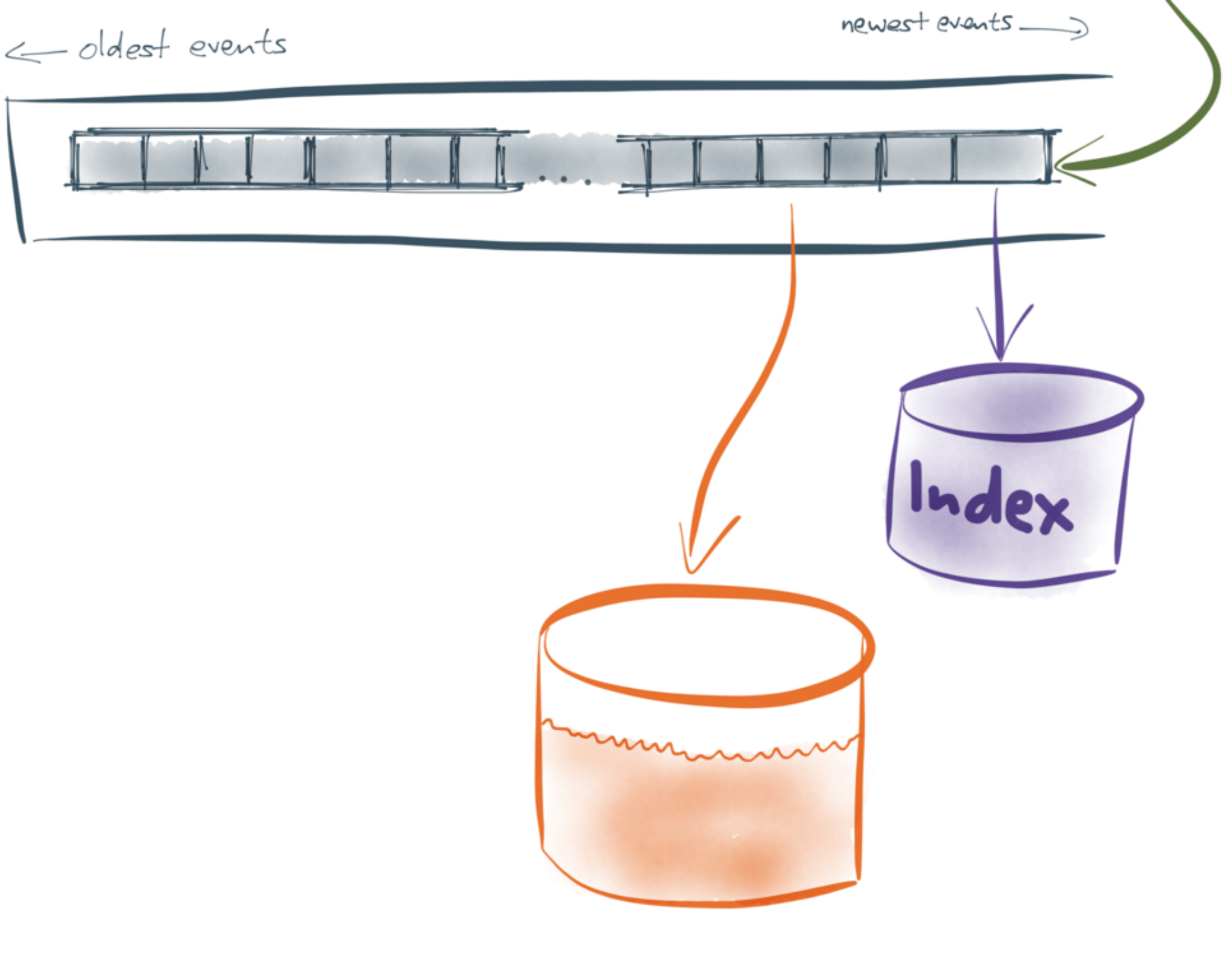

oldest events
newest events
time
Index

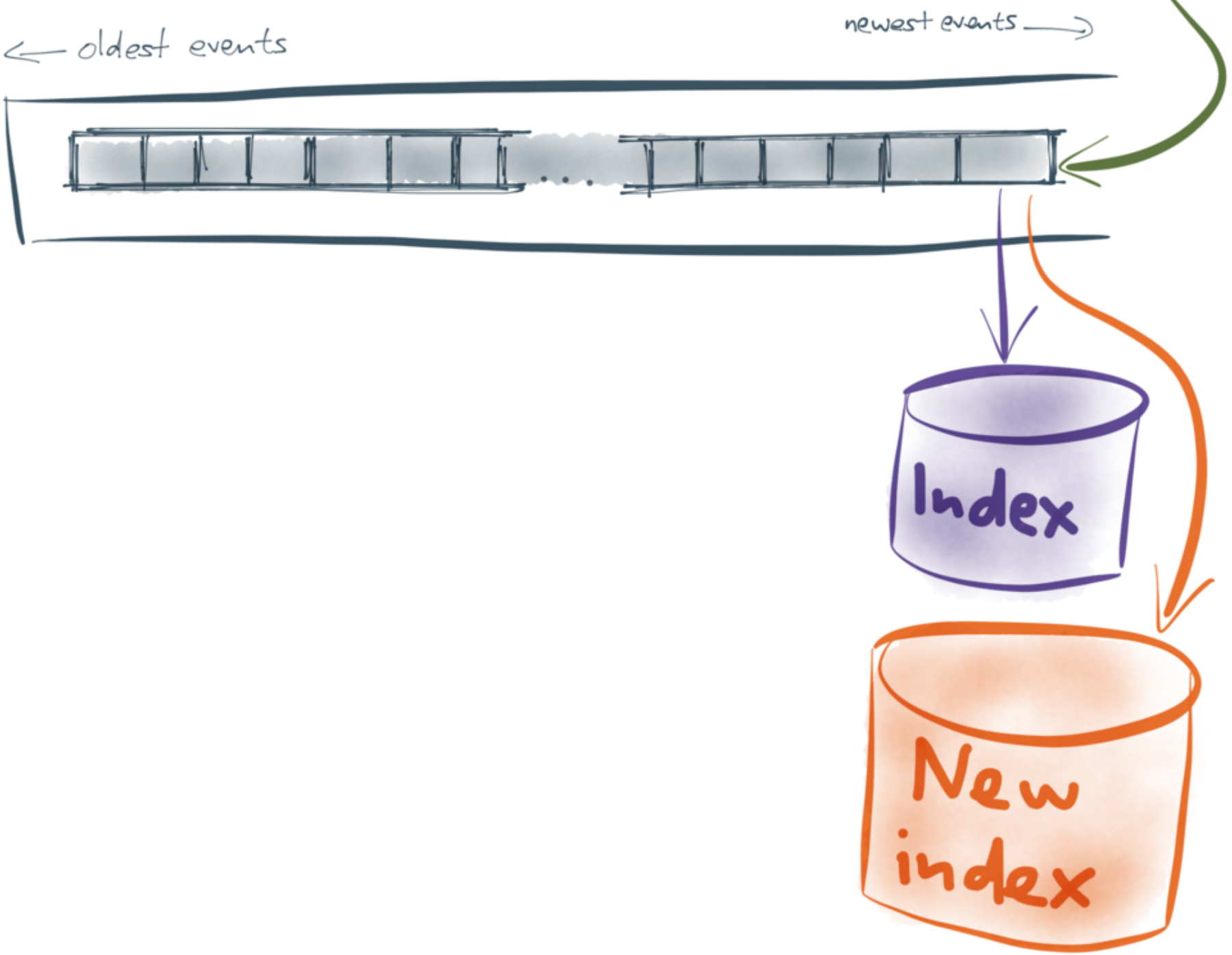


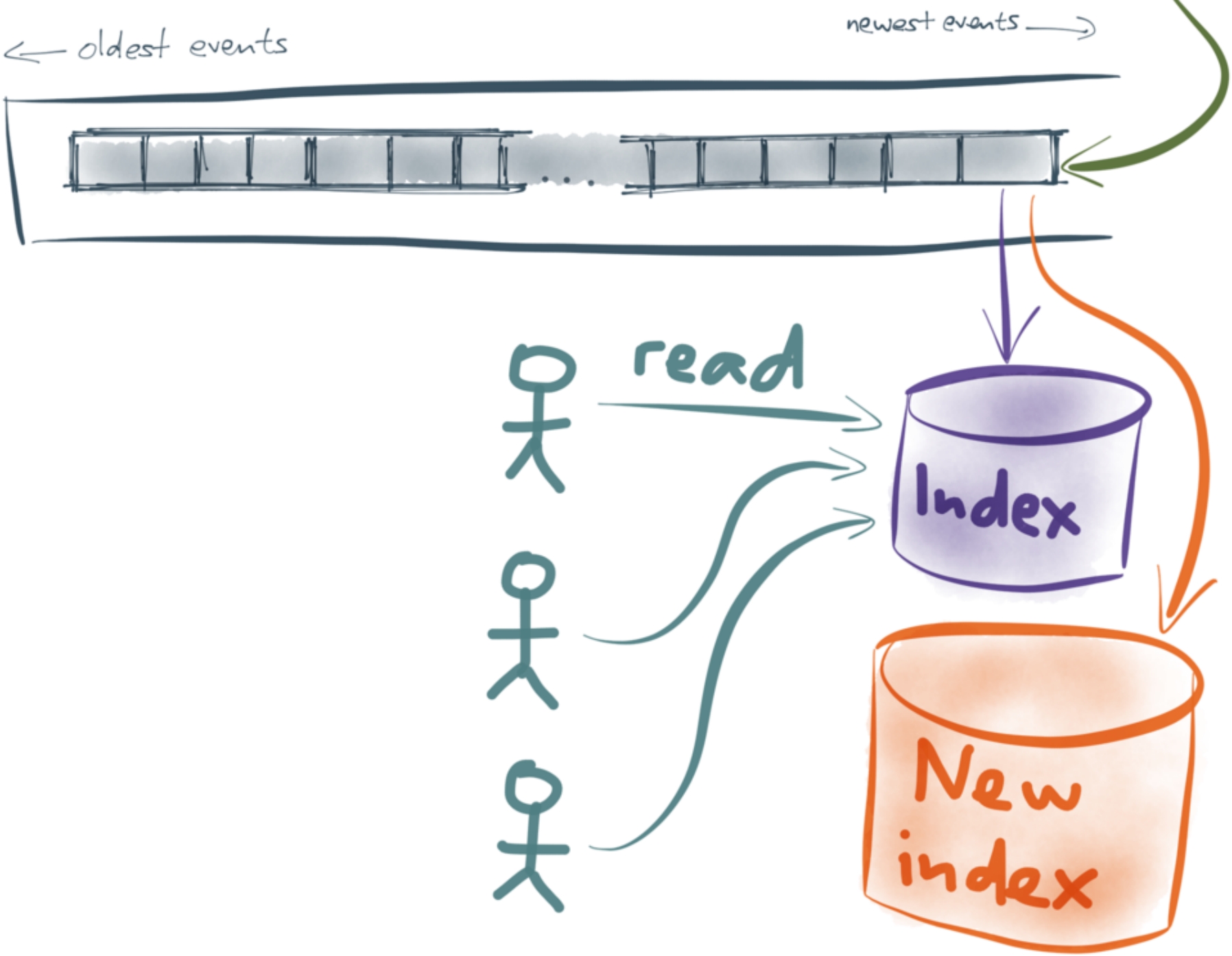
New index
(starts off empty)

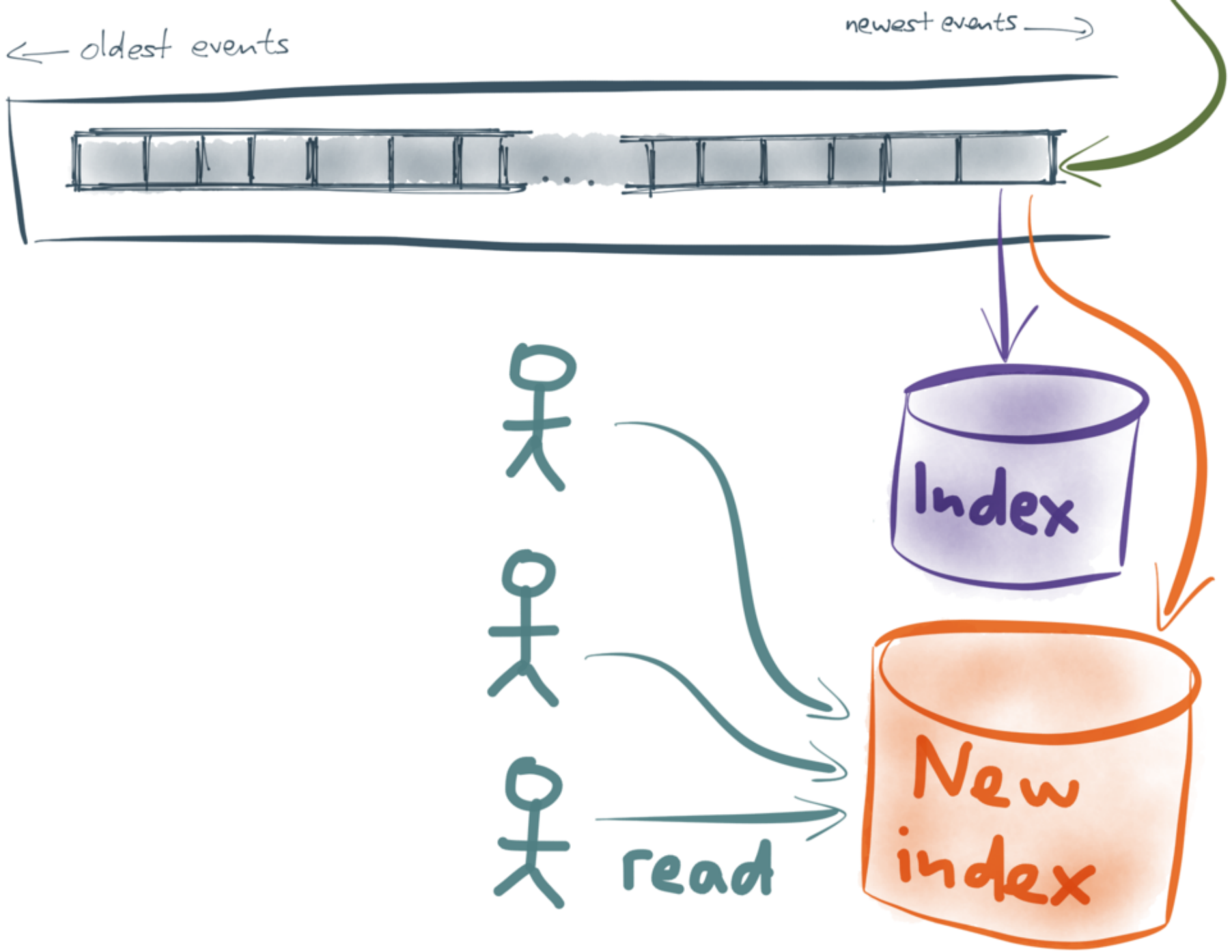

oldest events
newest events
Index
New index
(starts off empty)

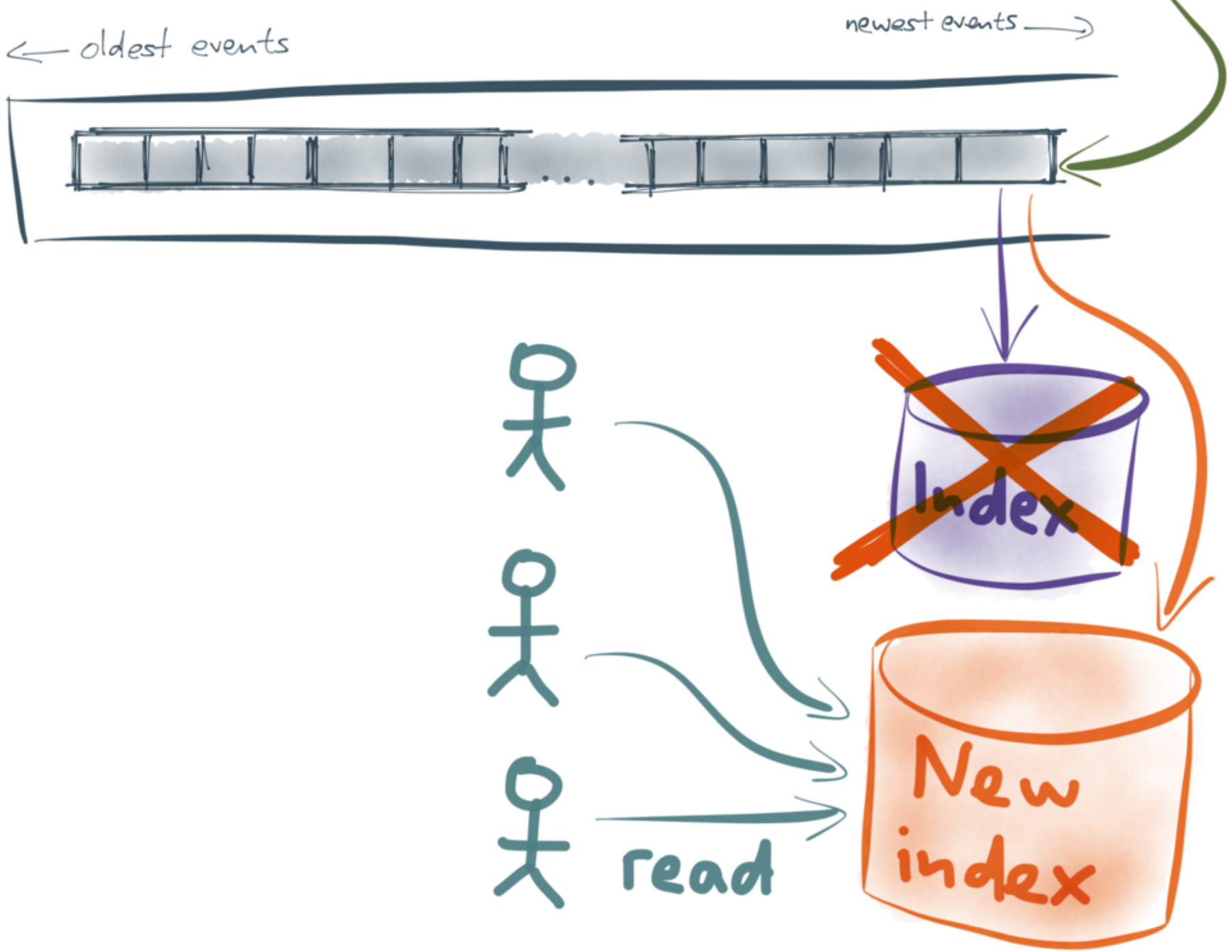

oldest events
newest events
Index

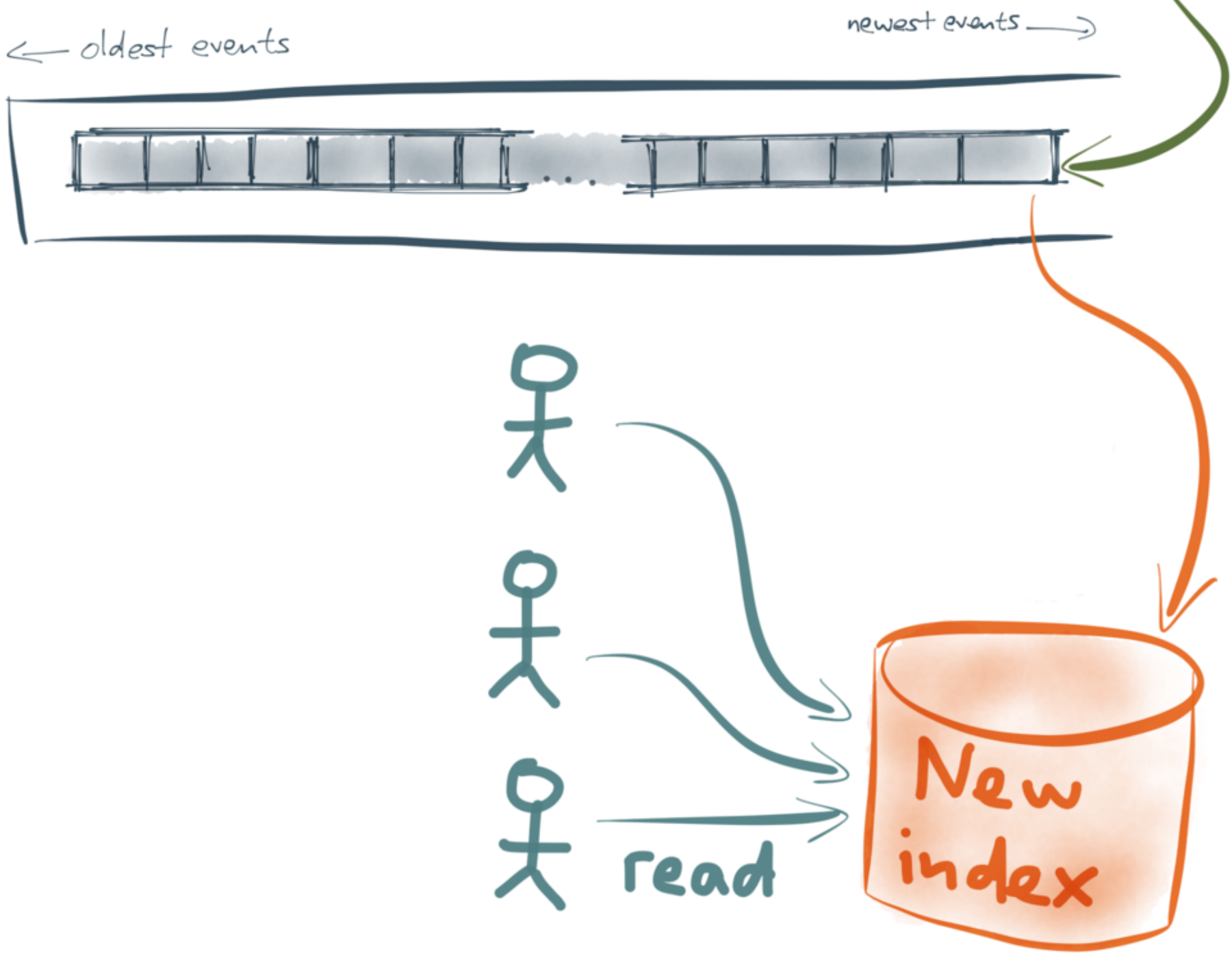

oldest events
newest events
Index


oldest events
newest events
Index
New index


oldest events
newest events
read
Index
New index

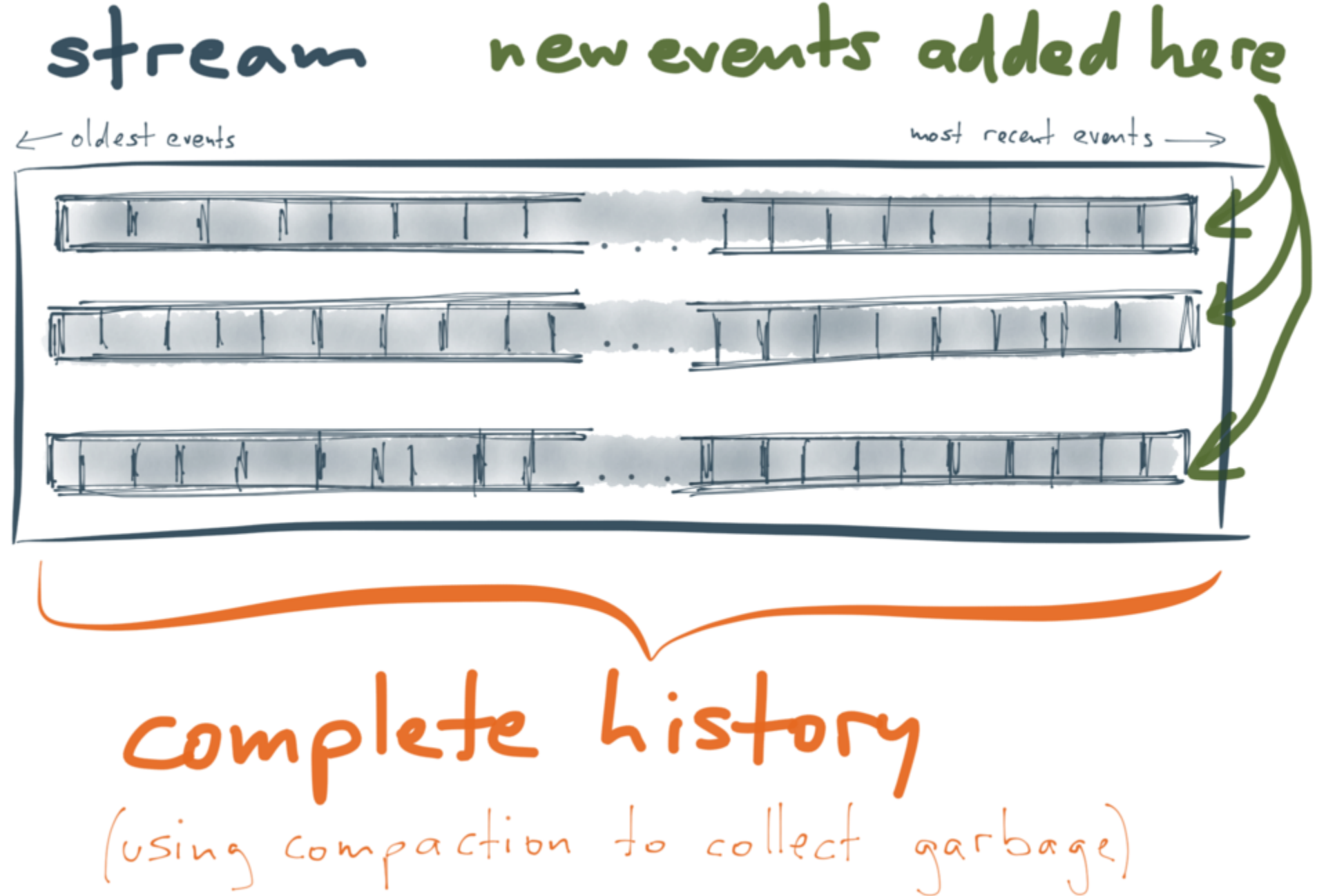

oldest events
newest events
Index
New index
read


oldest events
newest events
Index
New index
read


oldest events
newest events
read
New index

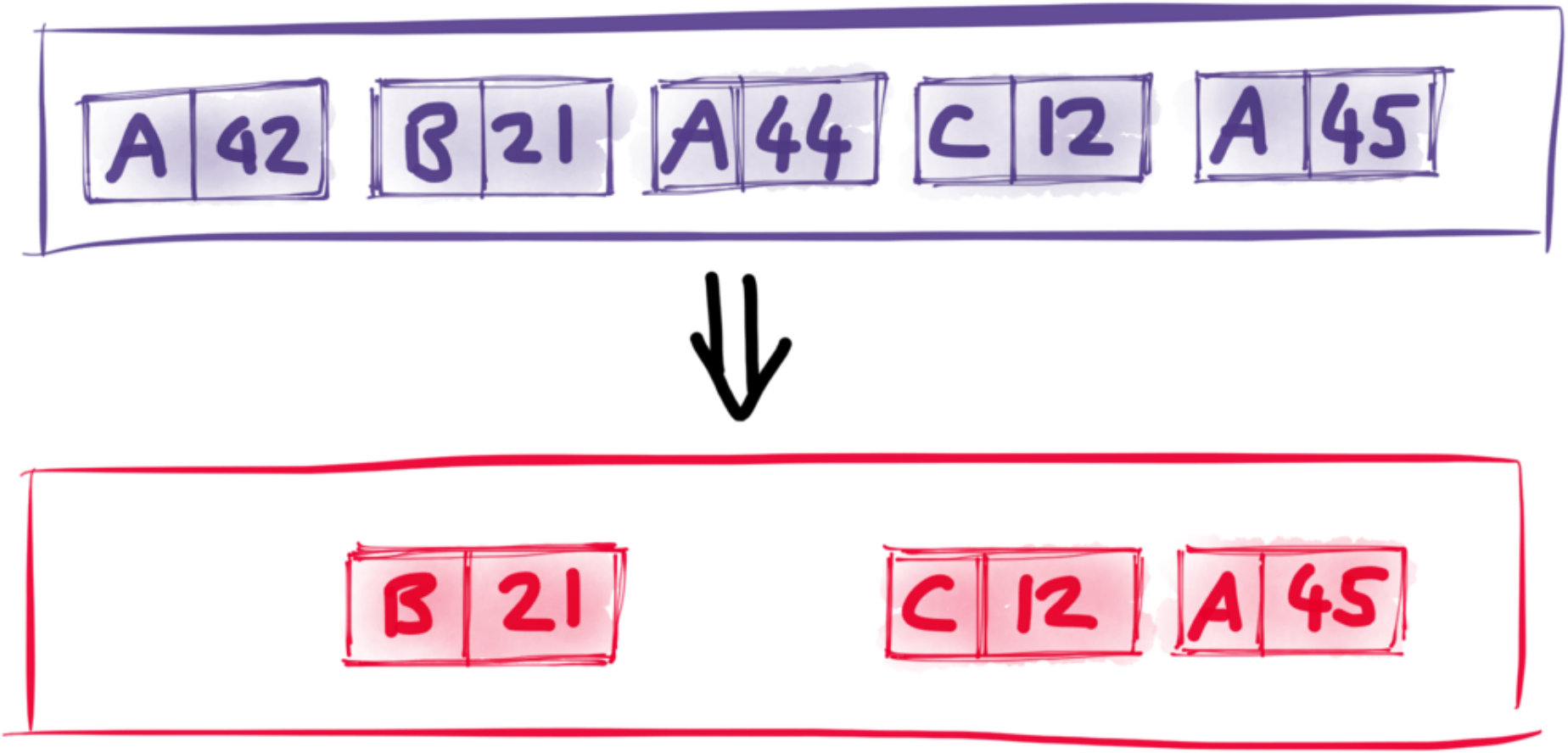
"Reducing irreversibility"

(Bartlett and Fowler, 2015)


stream
new events added here
← oldest events
most recent events →
complete history
(using compaction to collect garbage)


A 42
B 21
A 44
C 12
A 45

Kafka changelog compaction



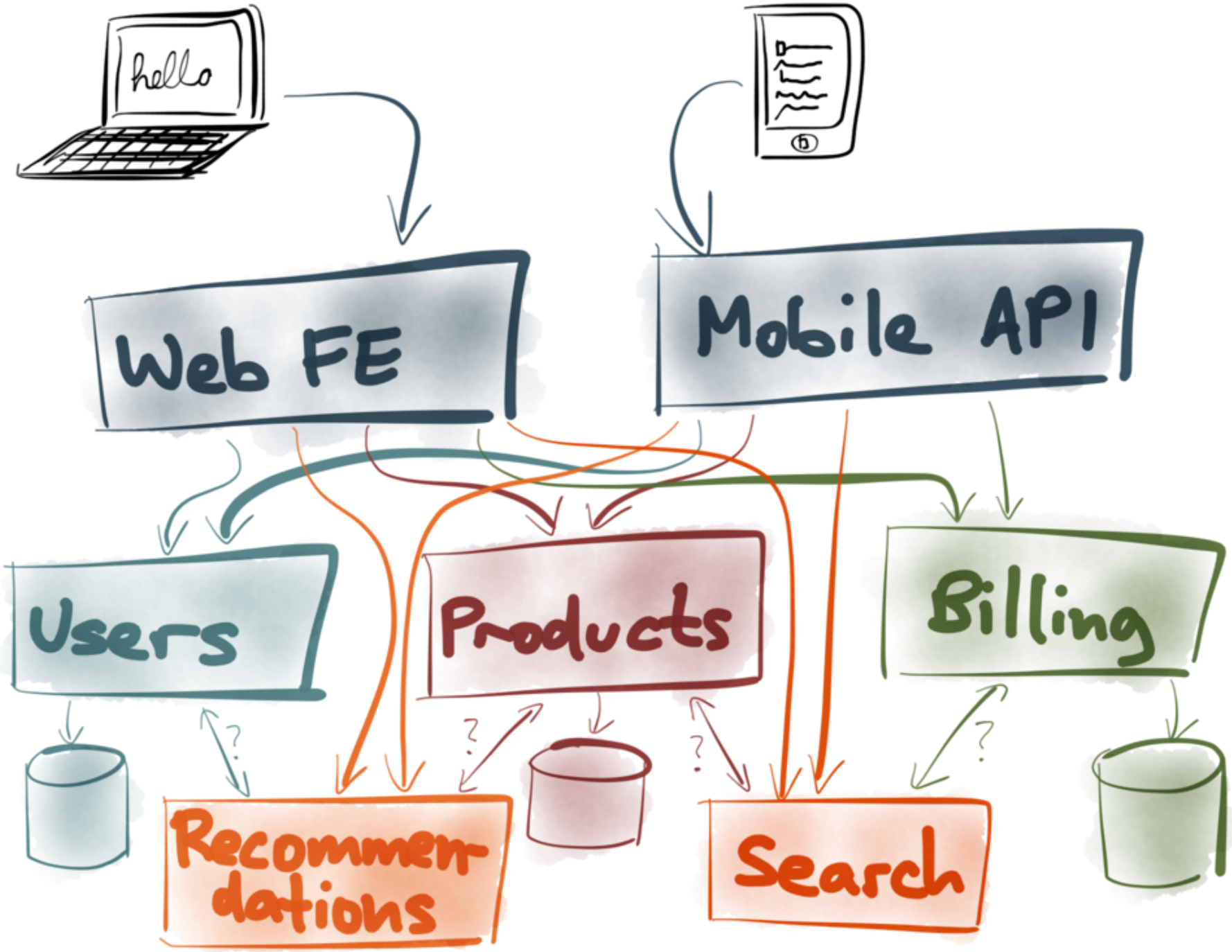
Tools :

Kafka Connect

Bottled Water (PostgreSQL)

Stream processing:

Kafka Streams / Samza / Storm / Spark / Flink / ...


hello
Web FE
Mobile API
Users
Products
Billing
Recommendations
Search

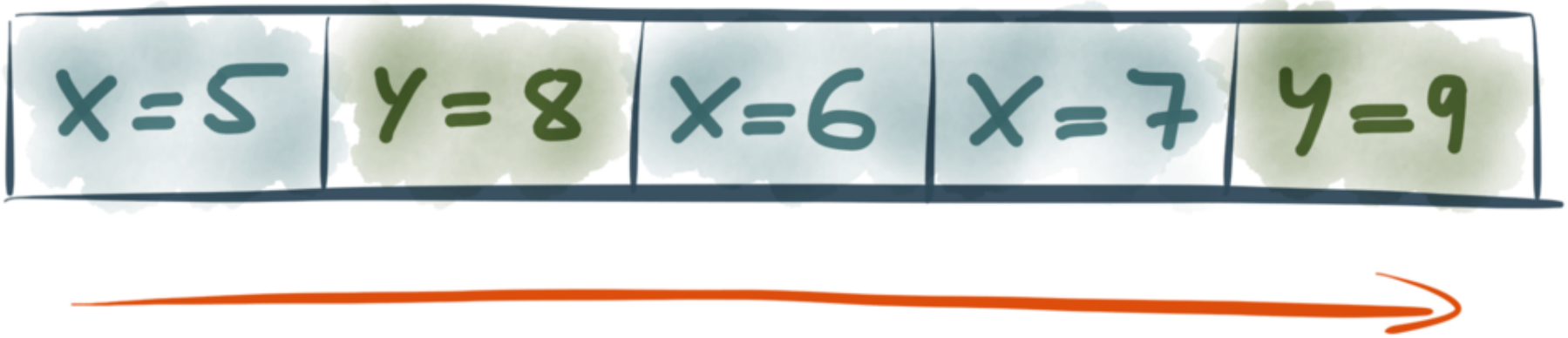
Interactive R/W transactions

Lots of network round trips
Lots of locking — or poor consistency
Global coordination, 2-phase commit

Ordered event log

All consumers see events in the same order
Pipelined, non-blocking
Idempotent operations for fault tolerance

STUPIDLY SIMPLE SOLUTIONS ARE THE BEST



totally ordered sequence

append-only, persistent

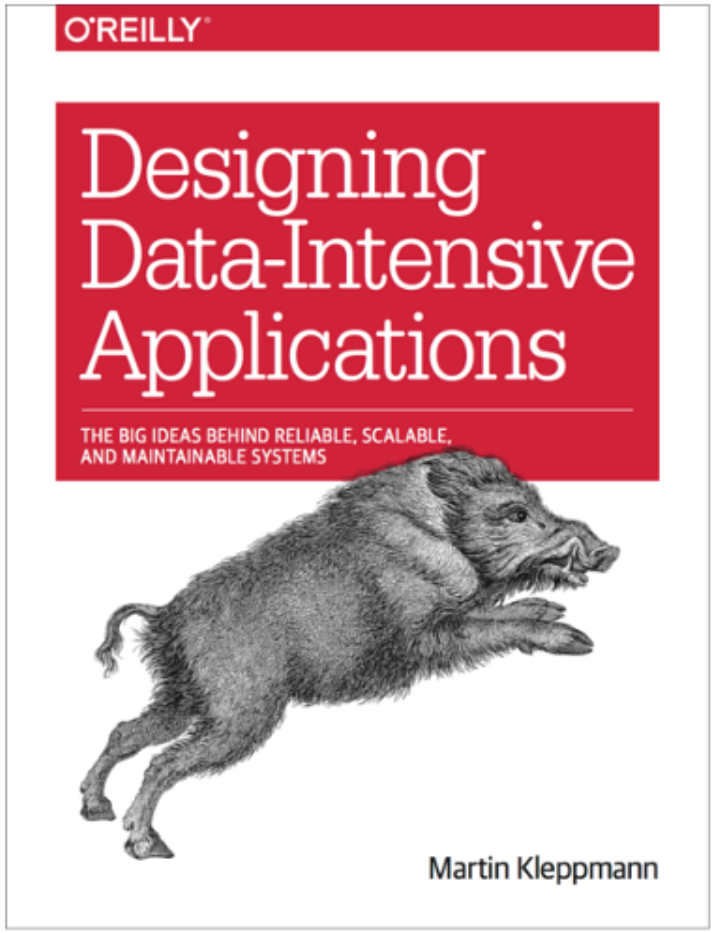
References (1)

1. Mahesh Balakrishnan, Dahlia Malkhi, Ted Wobber, et al.: "Tango: Distributed Data Structures over a Shared Log," at *24th ACM Symposium on Operating Systems Principles* (SOSP), pages 325–340, November 2013. http://research.microsoft.com/pubs/199947/Tango.pdf
2. Molly Bartlett Dishman and Martin Fowler: "Agile Architecture," at *O'Reilly Software Architecture Conference*, March 2015. http://conferences.oreilly.com/software-architecture/sa2015/public/schedule/detail/40388
3. Shirshanka Das, Chavdar Botev, Kapil Surlaker, et al.: "All Aboard the Databus!," at *ACM Symposium on Cloud Computing* (SoCC), October 2012. http://www.socc2012.org/s18-das.pdf
4. Pat Helland: "Life beyond Distributed Transactions: an Apostate's Opinion," at *3rd Biennial Conference on Innovative Data Systems Research* (CIDR), pages 132–141, January 2007. http://www-db.cs.wisc.edu/cidr/cidr2007/papers/cidr07p15.pdf
5. Pat Helland: "Immutability Changes Everything," at *7th Biennial Conference on Innovative Data Systems Research* (CIDR), January 2015. http://www.cidrdb.org/cidr2015/Papers/CIDR15_Paper16.pdf
6. Martin Kleppmann: "Designing Data-Intensive Applications." O'Reilly Media, to appear. http://dataintensive.net/
7. Jay Kreps: "I ♥ Logs." O'Reilly Media, September 2014. http://shop.oreilly.com/product/0636920034339.do
8. Jay Kreps: "Putting Apache Kafka to use: A practical guide to building a stream data platform." 25 February 2015. http://blog.confluent.io/2015/02/25/stream-data-platform-1/

References (2)

9. Leslie Lamport: "Time, Clocks, and the Ordering of Events in a Distributed System," *Communications of the ACM*, volume 21, number 7, pages 558–565, July 1978. http://research.microsoft.com/en-US/um/people/Lamport/pubs/time-clocks.pdf
10. Neha Narkhede: "Announcing Kafka Connect: Building large-scale low-latency data pipelines." 18 February 2016. http://www.confluent.io/blog/announcing-kafka-connect-building-large-scale-low-latency-data-pipelines
11. Fred B Schneider: "Implementing Fault-Tolerant Services Using the State Machine Approach: A Tutorial," *ACM Computing Surveys*, volume 22, number 4, pages 299–319, December 1990. http://www.cs.cornell.edu/fbs/publications/smsurvey.pdf
12. Yogeshwer Sharma, Philippe Ajoux, Petchean Ang, et al.: "Wormhole: Reliable Pub-Sub to Support Geo-replicated Internet Services," at *12th USENIX Symposium on Networked Systems Design and Implementation* (NSDI), May 2015. https://www.usenix.org/system/files/conference/nsdi15/nsdi15-paper-sharma.pdf
13. Martin Thompson: "Single Writer Principle." 22 September 2011. http://mechanical-sympathy.blogspot.co.uk/2011/09/single-writer-principle.html
14. Vaughn Vernon: *Implementing Domain-Driven Design*. Addison-Wesley Professional, February 2013.

dataintensive.net



@martinkl