A guarantee too good to be true

Micah Whitacre
@mkwhit

I was a naive and gullible child...

“Shoes guaranteed to make a kid run faster and jump higher…”

Sandlot (1993)

“Part of a complete breakfast..”

I am now a less naive and less gullible adult...

Definition “Guarantee”:

an assurance for the fulfillment of a condition

to give security to

provide a formal assurance or promise, especially that certain conditions shall be fulfilled relating to a product, service, or transaction.

Vendor Pitches

Conference Talks

Project Documentation

Team Presentations

Fastest Processing Engine

Write Once Run Anywhere

Best Programming Language

Works with a single click

It can solve all of your problems...

Are guarantees just lies?

Are guarantees just lies?

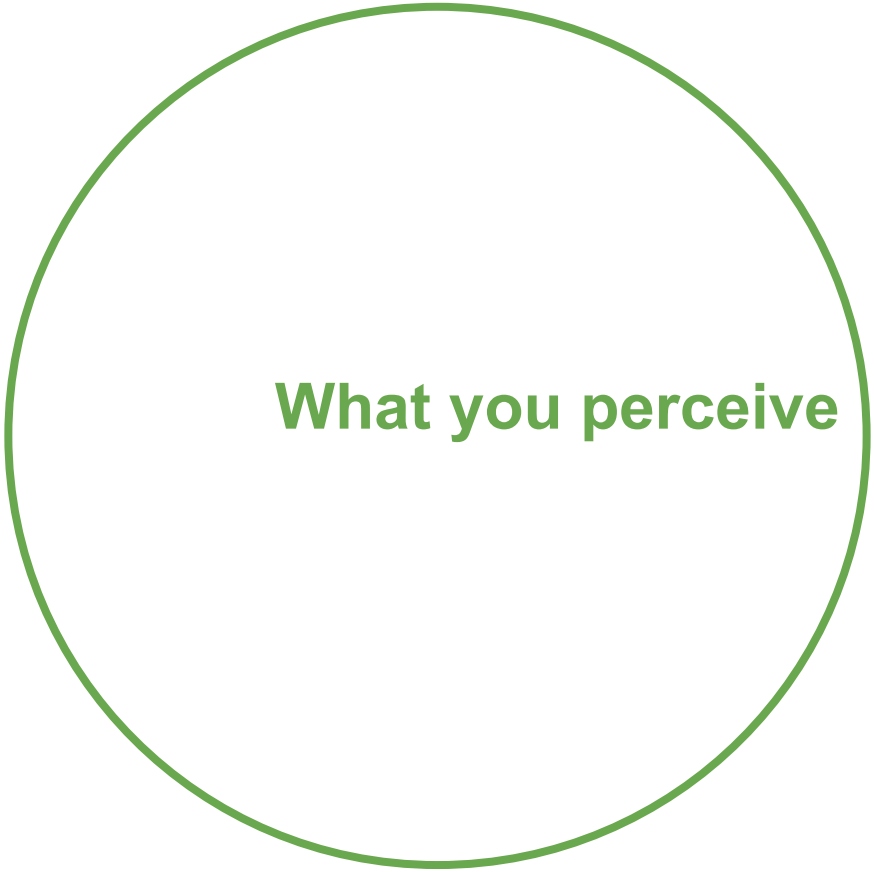
NO!

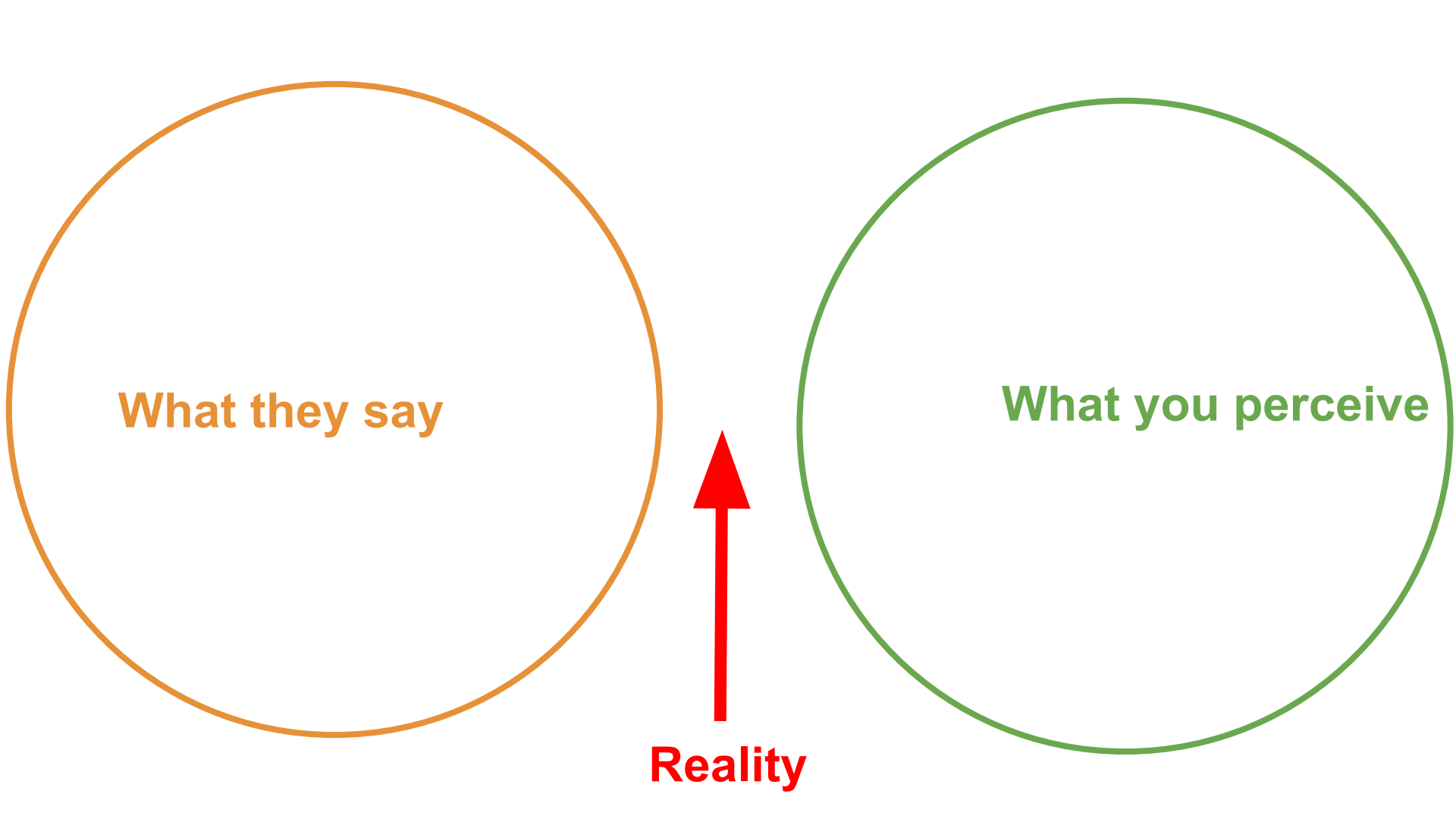
Are guarantees just lies?

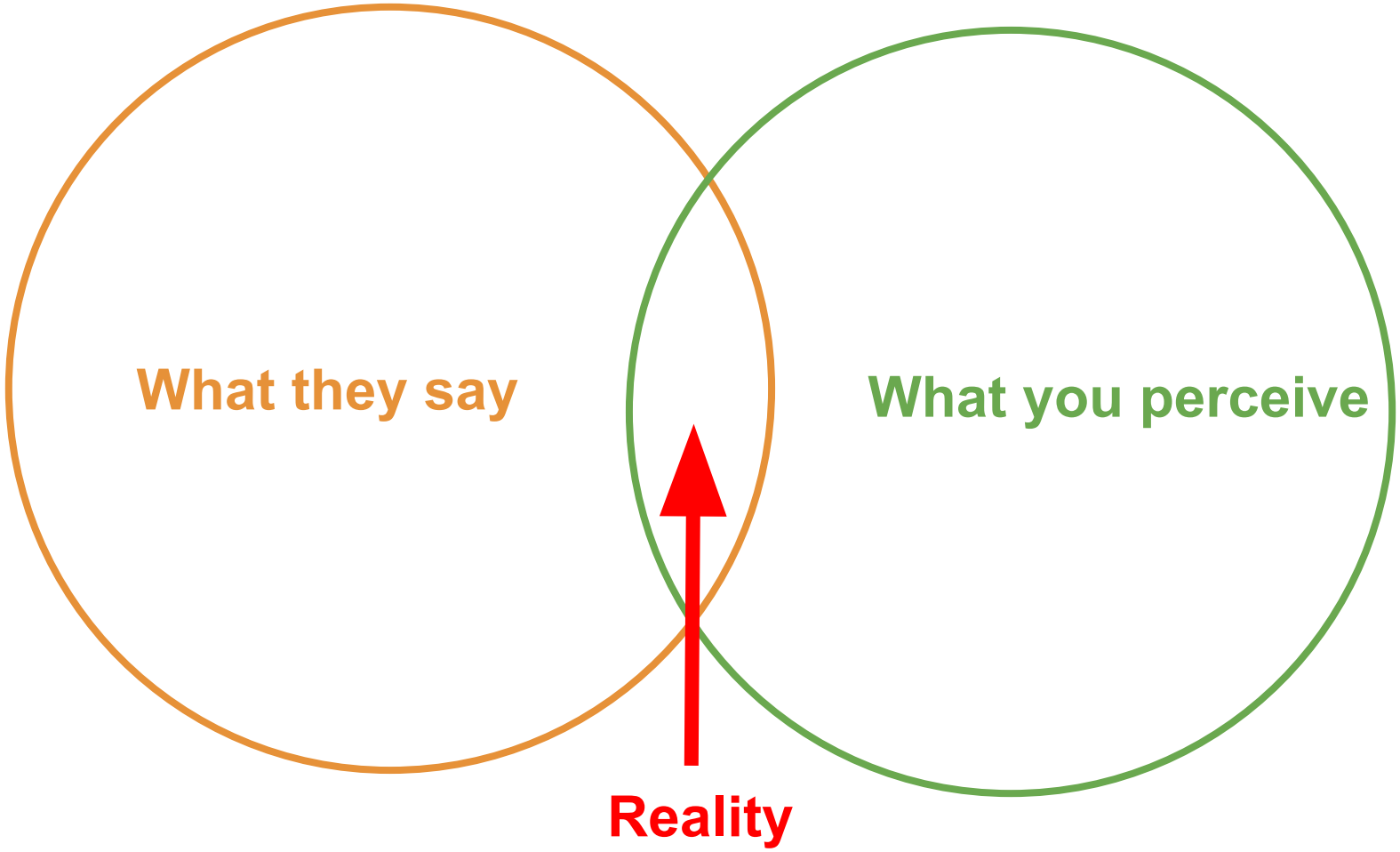
NO! (usually)

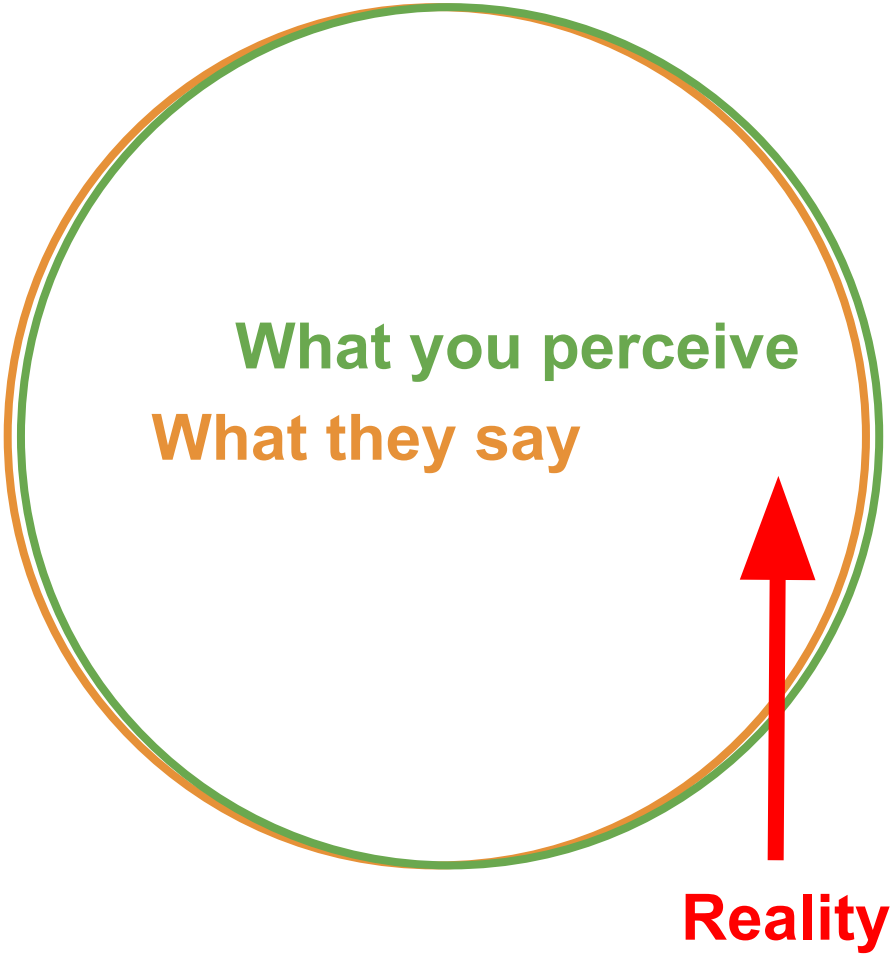
What they say


What they say


What you perceive


What they say
What you perceive
Reality


What they say
What you perceive
Reality


What you perceive
What they say
Reality

Definitions Matter!

I could use a cobbler right now ...

https://www.merriam-webster.com/dictionary/cobbler

I could use a cobbler right now ...

A. a mender or maker of shoes and often of other leather goods
B. a tall iced drink consisting usually of wine, rum, or whiskey and sugar garnished with mint or a slice of lemon or orange
C. a deep-dish fruit dessert with a thick top crust

https://www.merriam-webster.com/dictionary/cobbler

I could use a cobbler right now ...

A. a mender or maker of shoes and often of other leather goods
B. a tall iced drink consisting usually of wine, rum, or whiskey and sugar garnished with mint or a slice of lemon or orange
C. a deep-dish fruit dessert with a thick top crust

https://www.merriam-webster.com/dictionary/cobbler

Fault Tolerant

Fault Tolerant

Web Scale

Fault Tolerant

Web Scale

Cloud First

Define “Consistency”:

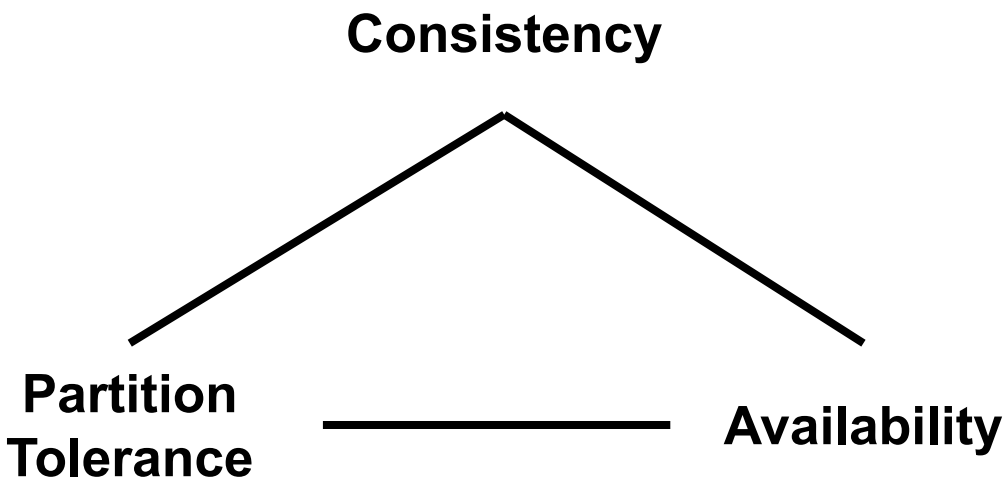
A. The guarantee that any transactions started in the future necessarily see the effects of other transactions committed in the past.
B. The guarantee that operations in transactions are performed accurately, correctly, and with validity, with respect to application semantics.
C. The guarantee that database constraints are not violated, particularly once a transaction commits.
D. An atomic operation is one which cannot be (or is not) interrupted by concurrent operations.

ACID

ACID

Atomicity
Consistency
Isolation
Durability

CAP



CAP

ACID

CAP

ACID

CAP

Both have “Consistency” that’s the same thing right?

Both have “Consistency” that’s the same thing right?

No

Define “Consistency”:

A. The guarantee that any transactions started in the future necessarily see the effects of other transactions committed in the past.
B. The guarantee that operations in transactions are performed accurately, correctly, and with validity, with respect to application semantics.
C. The guarantee that database constraints are not violated, particularly once a transaction commits.
D. An atomic operation is one which cannot be (or is not) interrupted by concurrent operations.

CAP

Every read receives the most recent write or an error.

CAP

Every read receives the most recent write or an error.

Linearizability

Linearizability

If operation B started after operation A successfully completed, then operation B must see the the system in the same state as it was on completion of operation A, or a newer state.

Define “Consistency”:

A. The guarantee that any transactions started in the future necessarily see the effects of other transactions committed in the past.
B. The guarantee that operations in transactions are performed accurately, correctly, and with validity, with respect to application semantics.
C. The guarantee that database constraints are not violated, particularly once a transaction commits.
D. An atomic operation is one which cannot be (or is not) interrupted by concurrent operations.

Linearizability

“Atomic Consistency”

Linearizability

“Atomic Consistency”

Sequential consistency with the real-time constraint.

Linearizability

“Atomic Consistency”

Sequential consistency with the real-time constraint.

A write to a variable does not have to be seen instantaneously, however, writes to variables by different processors have to be seen in the same order by all processors.

The data consistency model specifies a contract between programmer and system, wherein the system guarantees that if the programmer follows the rules, memory will be consistent and the results of reading, writing, or updating memory will be predictable.

https://en.wikipedia.org/wiki/Consistency_model

“One of the reasons these definitions are so confusing is that linearizability hails from the distributed systems and concurrent programming communities, and serializability comes from the database community. Today, almost everyone uses *both* distributed systems and databases, which often leads to overloaded terminology (e.g., “consistency,” “atomicity”).” - Peter Bailis 2014

http://www.bailis.org/blog/linearizability-versus-serializability/

Look the explicit definition

Avoid Vague Terms

Read the Fine Print!

Munchies

Potato chips with assorted dips
Nuts
Pretzels
M & M's (WARNING: ABSOLUTELY NO BROWN ONES)
Twelve (12) Reese's peanut butter cups
Twelve (12) assorted Dannon yogurt (on ice)

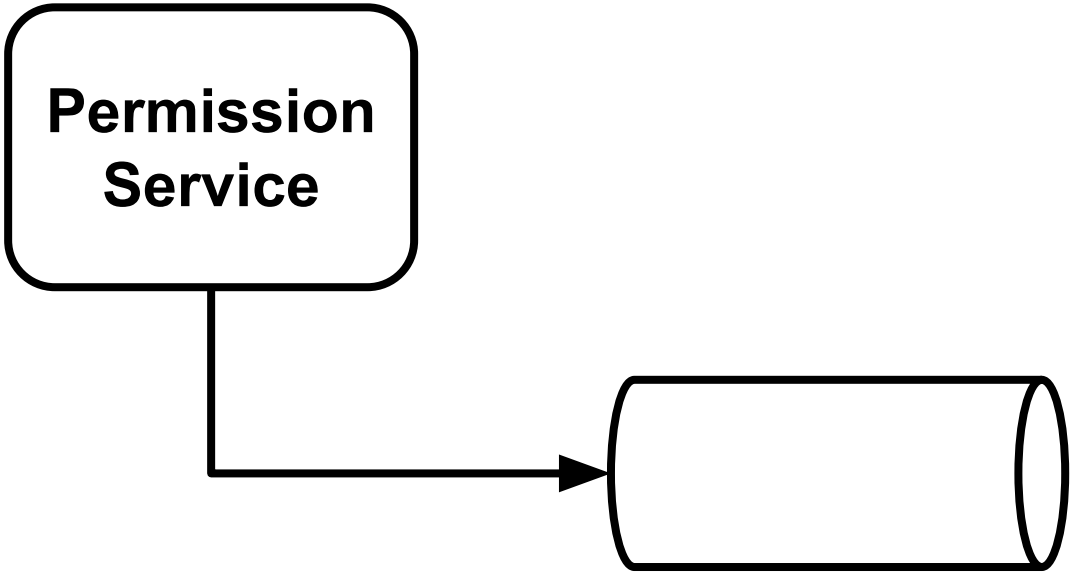
https://www.onelegal.com/blog/fantastic-clauses-hidden-in-contracts-and-eulas/
http://www.thesmokinggun.com/file/van-halens-legendary-mms-rider?page=8

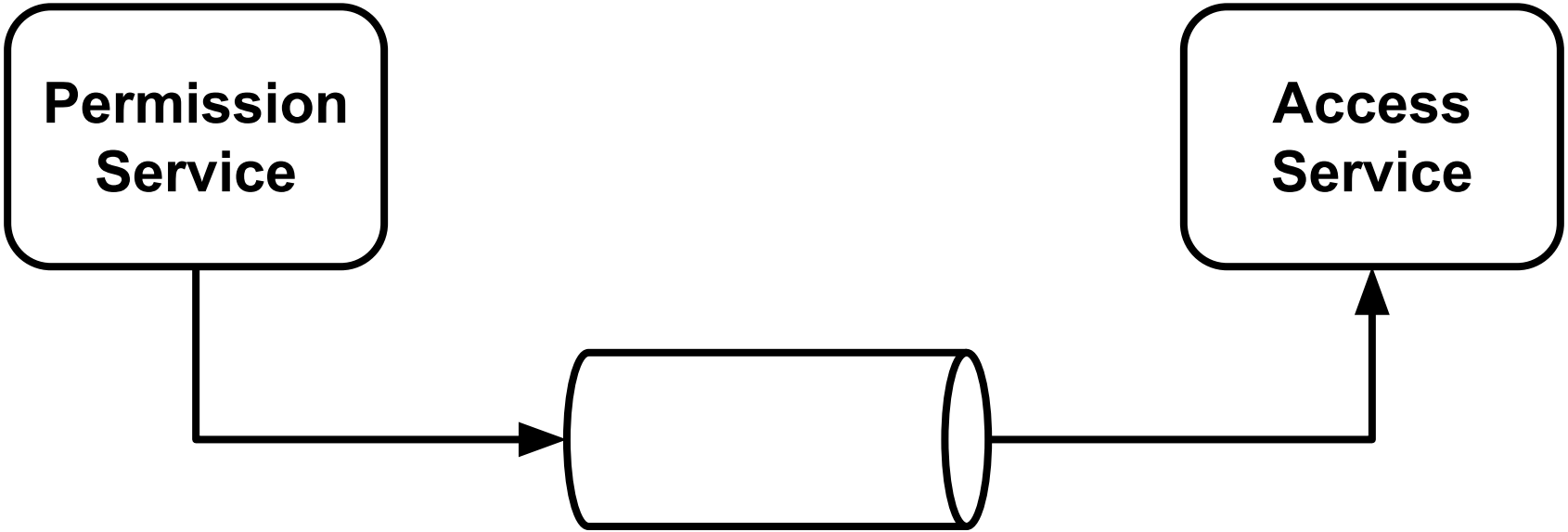
42.10. Acceptable Use; Safety-Critical Systems. Your use of the Lumberyard Materials must comply with the AWS Acceptable Use Policy. The Lumberyard Materials are not intended for use with life-critical or safety-critical systems, such as use in operation of medical equipment, automated transportation systems, autonomous vehicles, aircraft or air traffic control, nuclear facilities, manned spacecraft, or military use in connection with live combat. However, this restriction will not apply in the event of the occurrence (certified by the United States Centers for Disease Control or successor body) of a widespread viral infection transmitted via bites or contact with bodily fluids that causes human corpses to reanimate and seek to consume living human flesh, blood, brain or nerve tissue and is likely to result in the fall of organized civilization.

https://www.onelegal.com/blog/fantastic-clauses-hidden-in-contracts-and-eulas/
https://aws.amazon.com/service-terms/


Sup?
Sup?
Delivered


Permission
Service


Permission
Service


Permission Service
Access Service

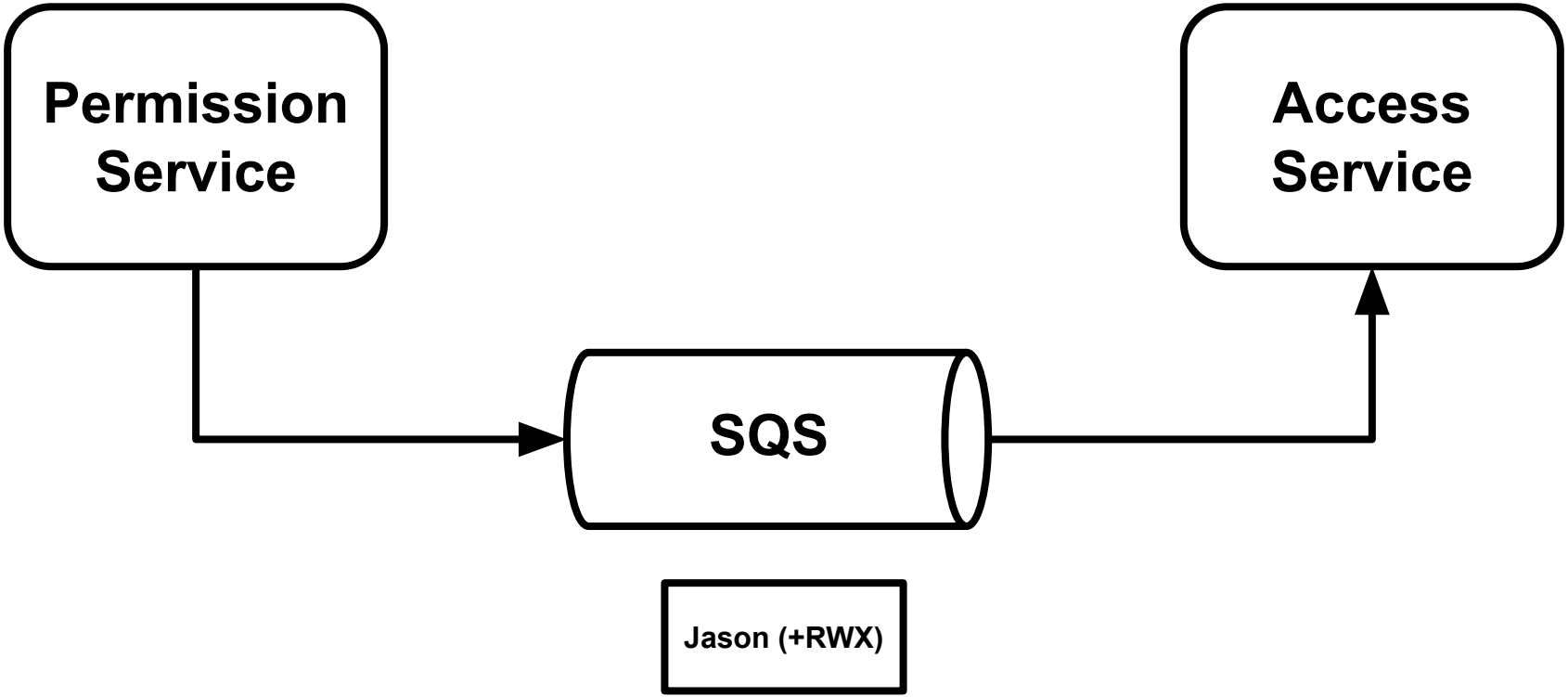
AWS Simple Queue Service

- Nearly Unlimited Throughput*
- Best Effort Ordering
- At-least-once Delivery

* 3000 transactions per second**
** 300 API calls with 10 messages each

Basic Workflow

1. Permission Service produces message to queue
2. Access Service consumes message from queue
3. Message is now hidden for visibility timeout
4. Access Service deletes message


Permission Service
Access Service
SQS
Jason (+RWX)

What about failure?

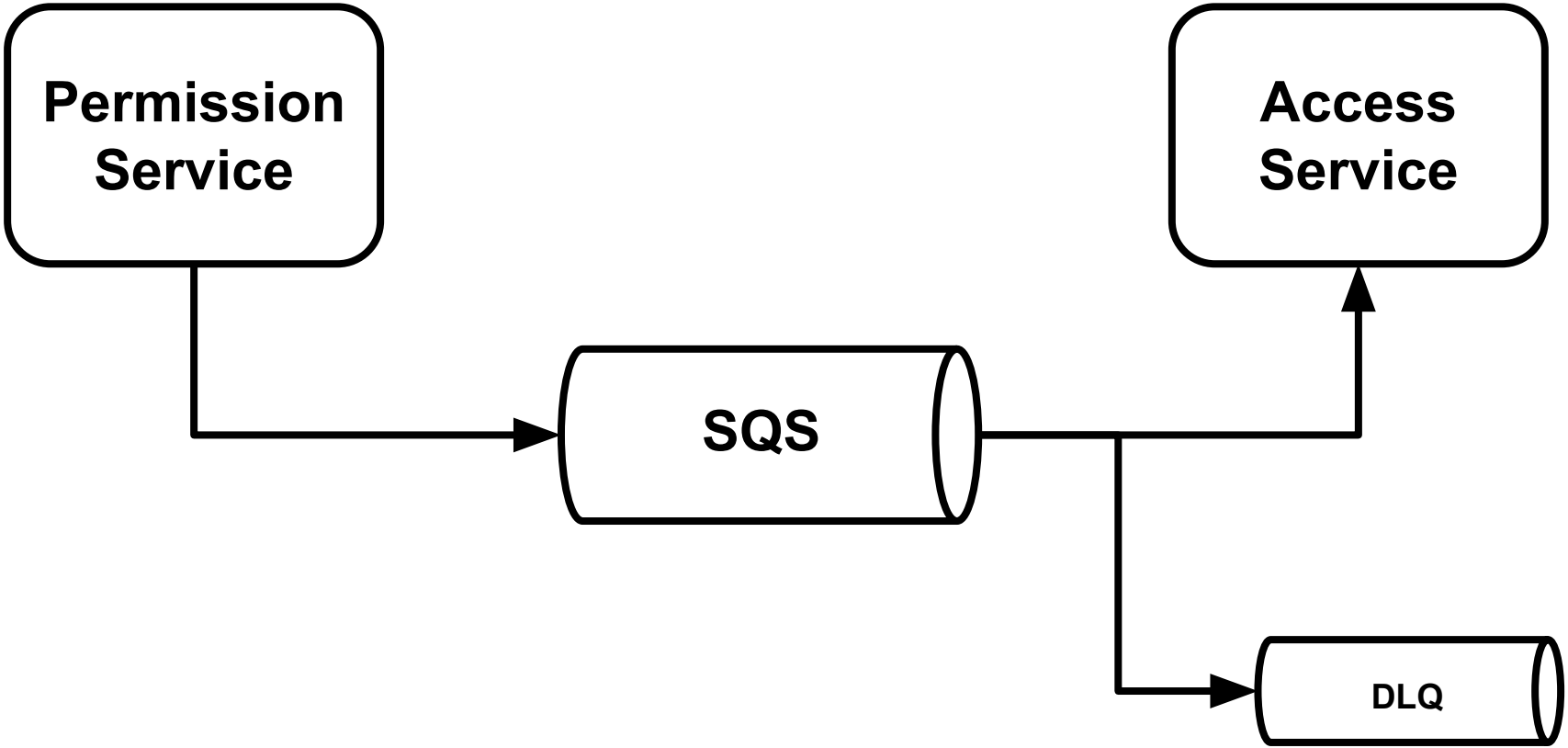
What about failure?

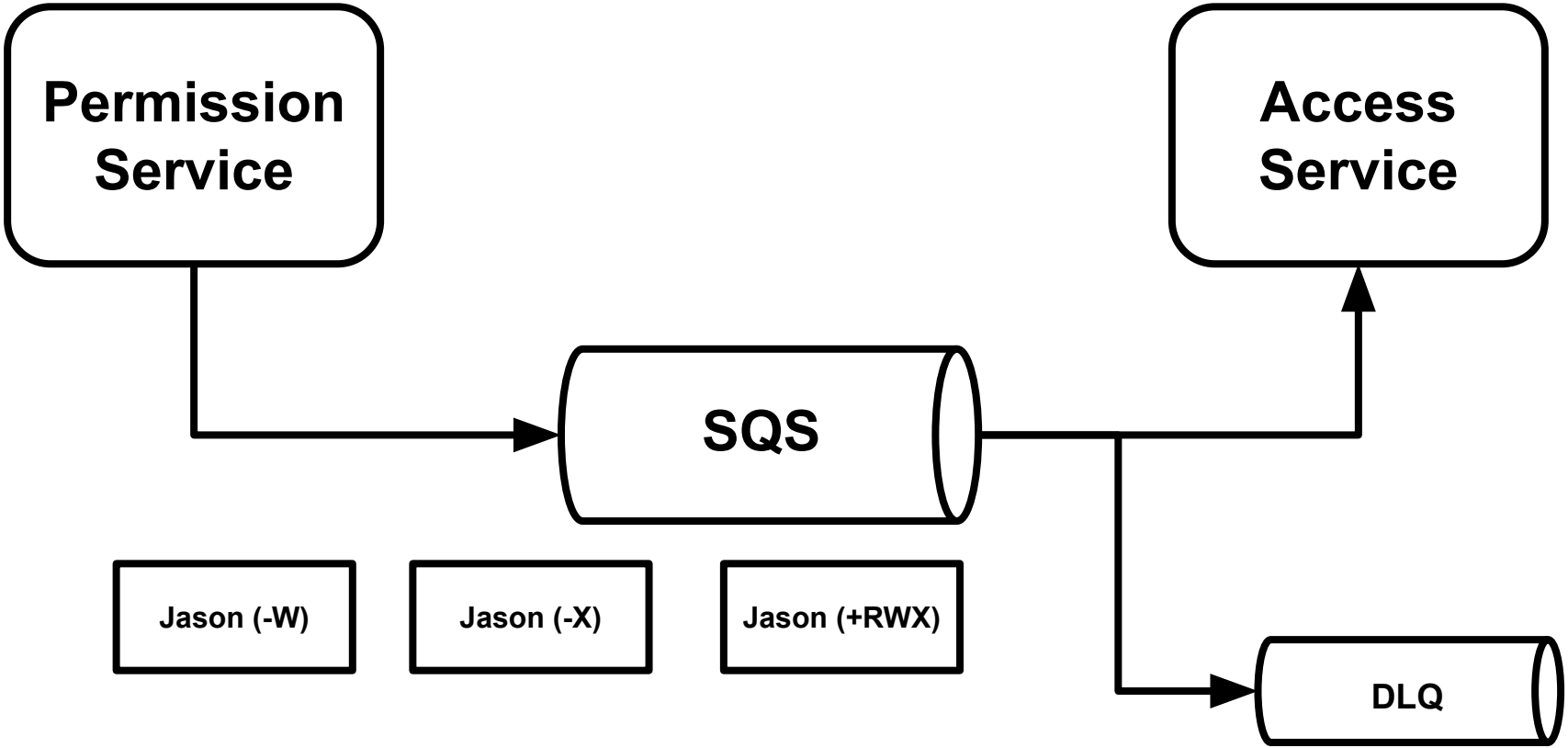
Downstream system down?

What about failure?

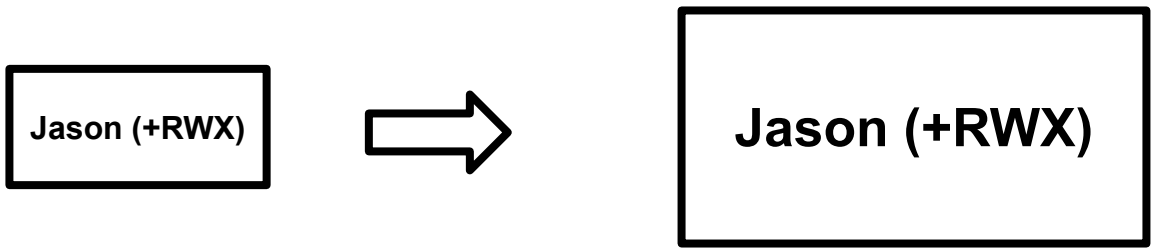
Downstream system down?

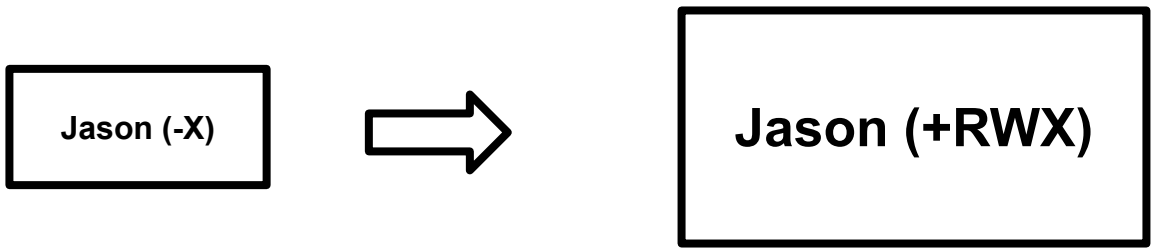
Message unprocessable?

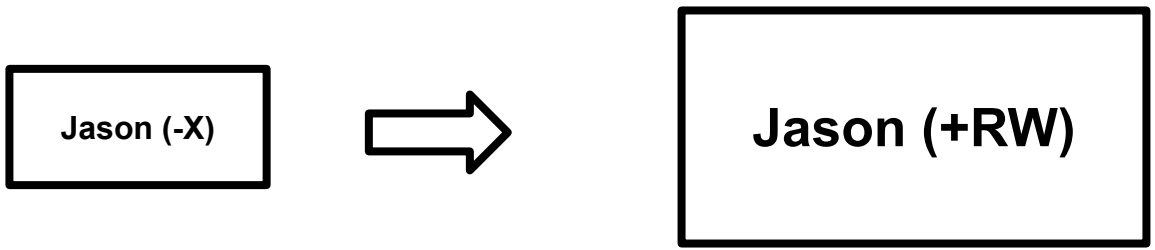

Permission Service
Access Service
SQS
DLQ

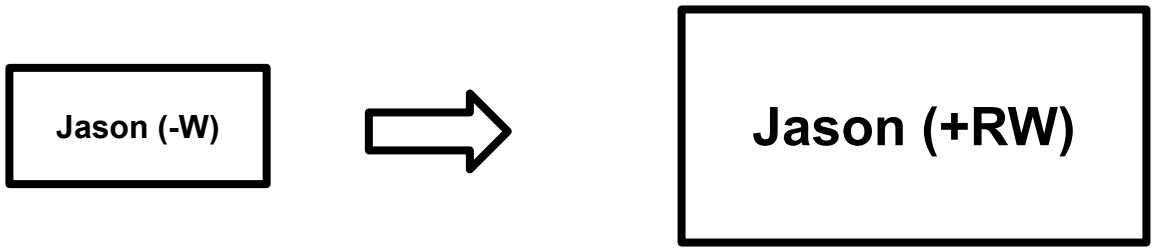

Permission Service
Access Service
SQS
Jason (-W)
Jason (-X)
Jason (+RWX)
DLQ

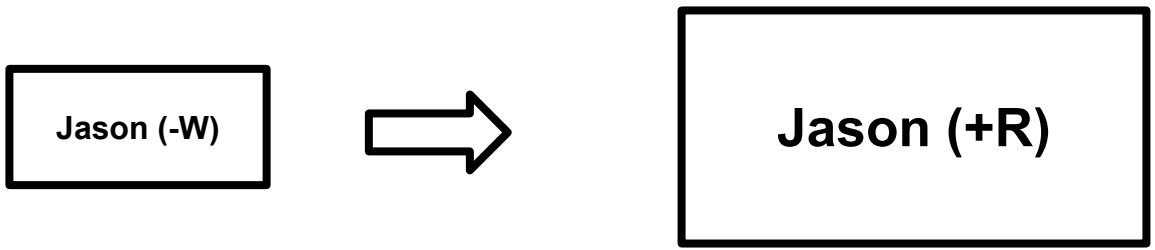

Jason (+RWX)


Jason (+RWX)
Jason (+RWX)


Jason (-X)
Jason (+RWX)


Jason (-X)
Jason (+RW)

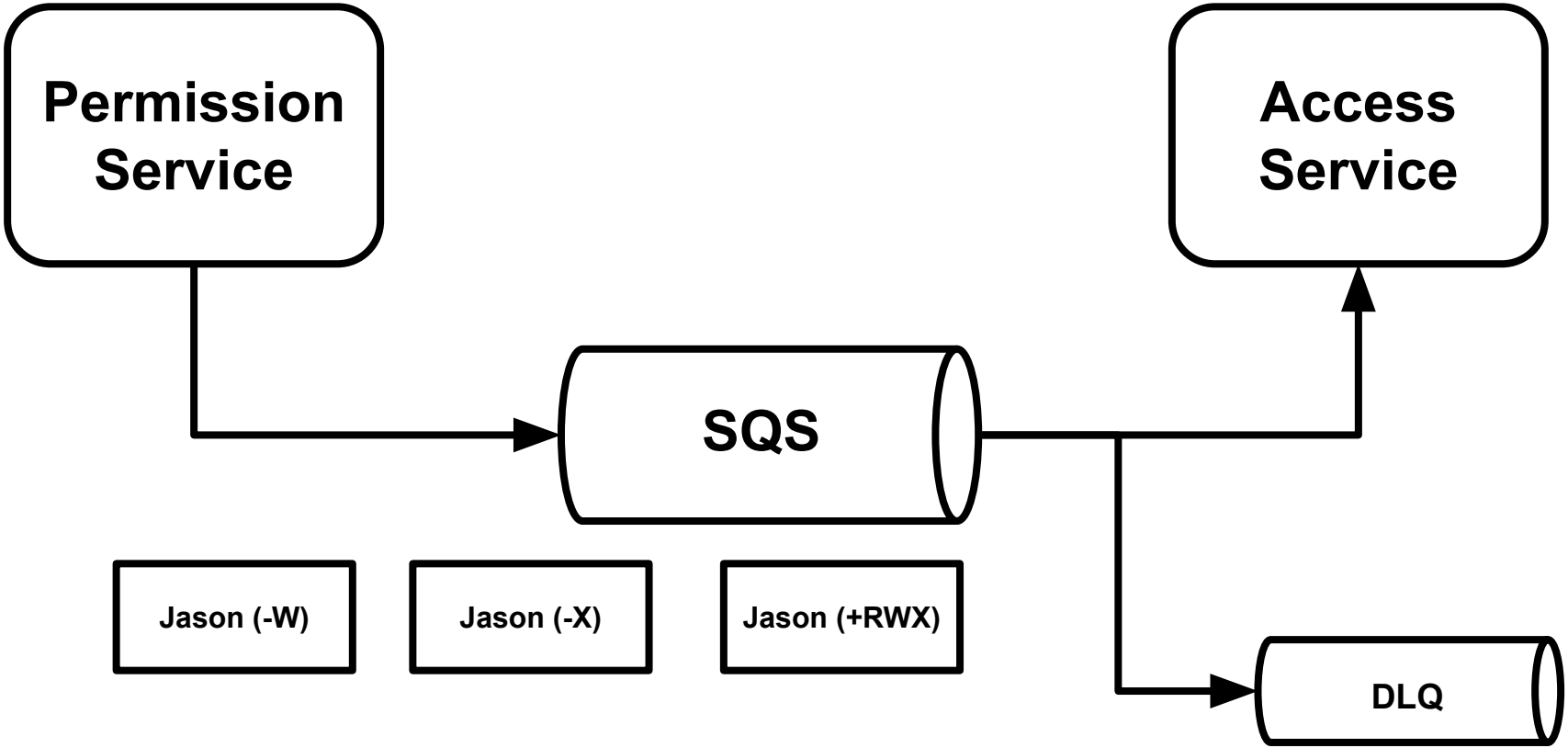

Jason (-W)
Jason (+RW)


Jason (-W)
Jason (+R)

SQS FIFO Queue

- Message order is preserved.*
- Message is delivered once until consumer processes and deletes.**
- Duplicates are not introduced into the queue.***

* Assuming same Message Group Id
** Assuming same Request Receive Attempt Id
*** Assuming same Message Deduplication Id and 5 minute window

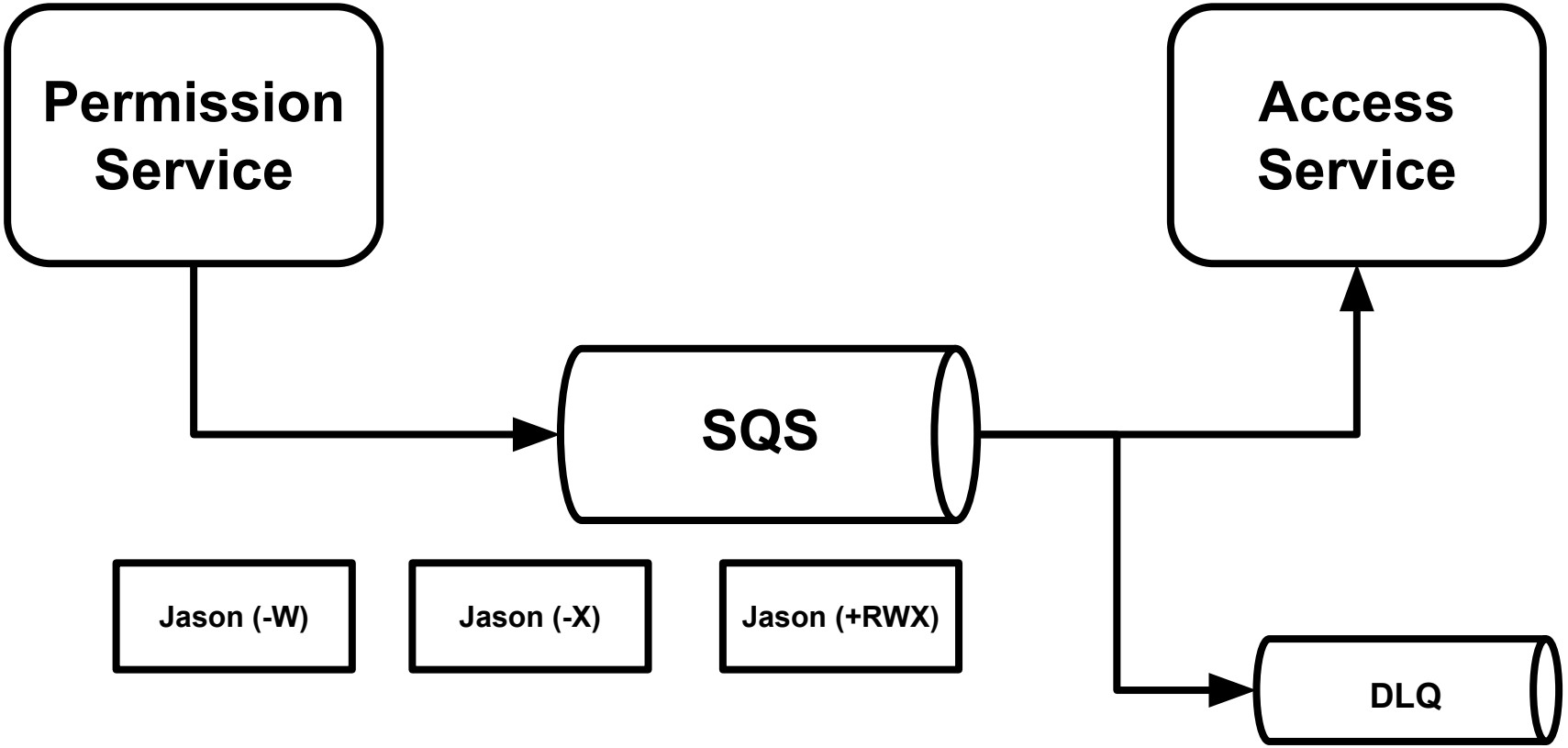

Permission Service
Access Service
SQS
Jason (-W)
Jason (-X)
Jason (+RWX)
DLQ

If you are covering the “exactly once” thing..the exactly once guarantee of SQS FIFO is a giant lie

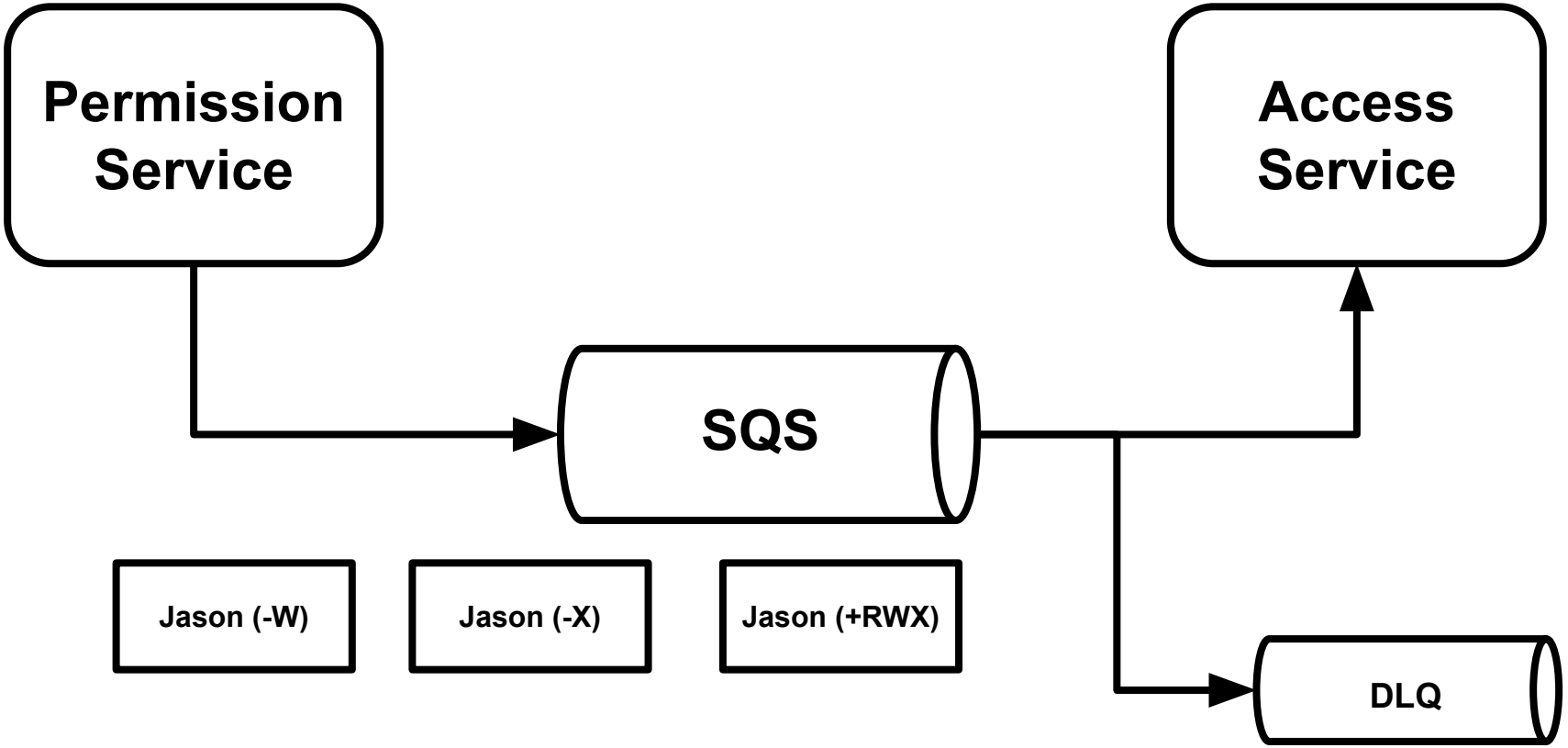
It’s exactly once processing, not exactly once delivery

I gave up on FiFo SQS completely vowing to never touch it again ;)

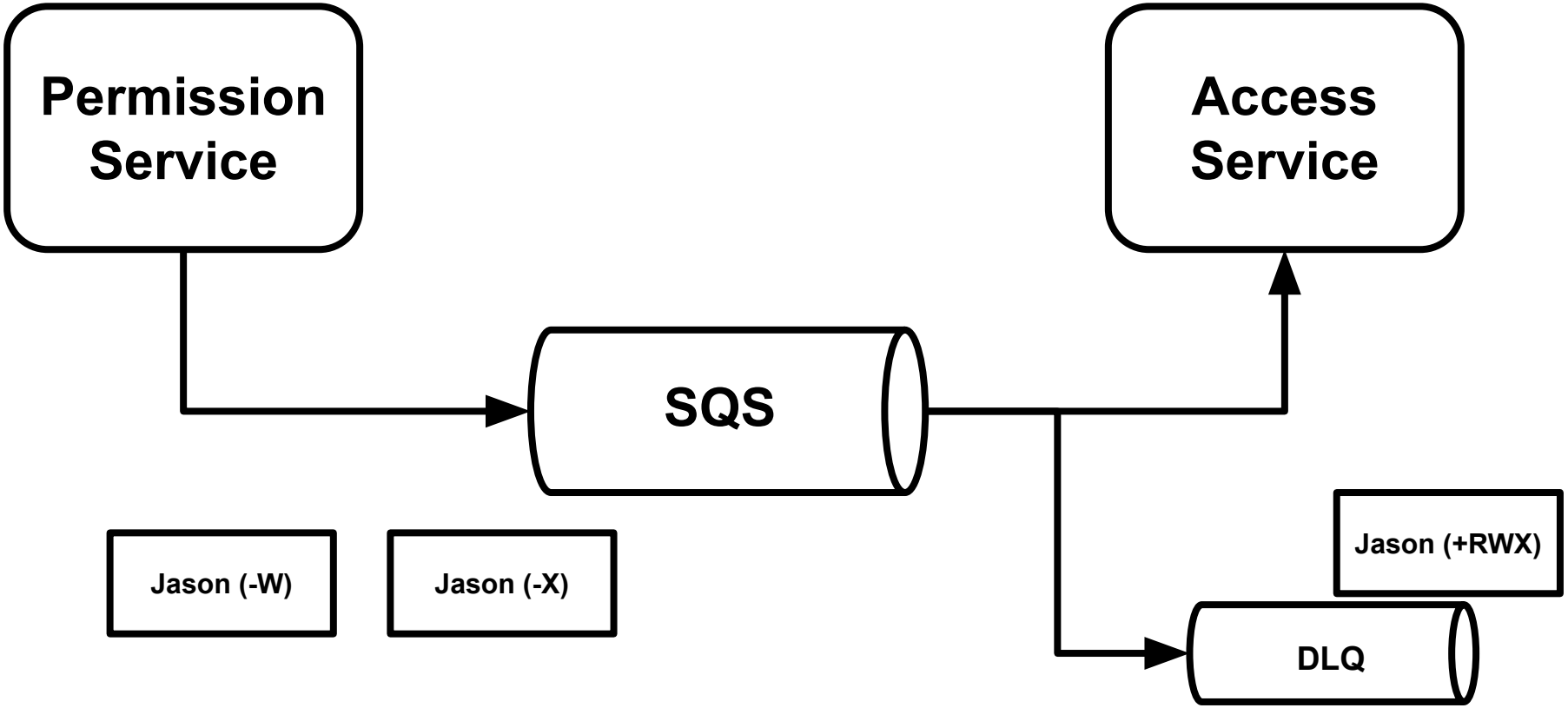
What can go wrong?



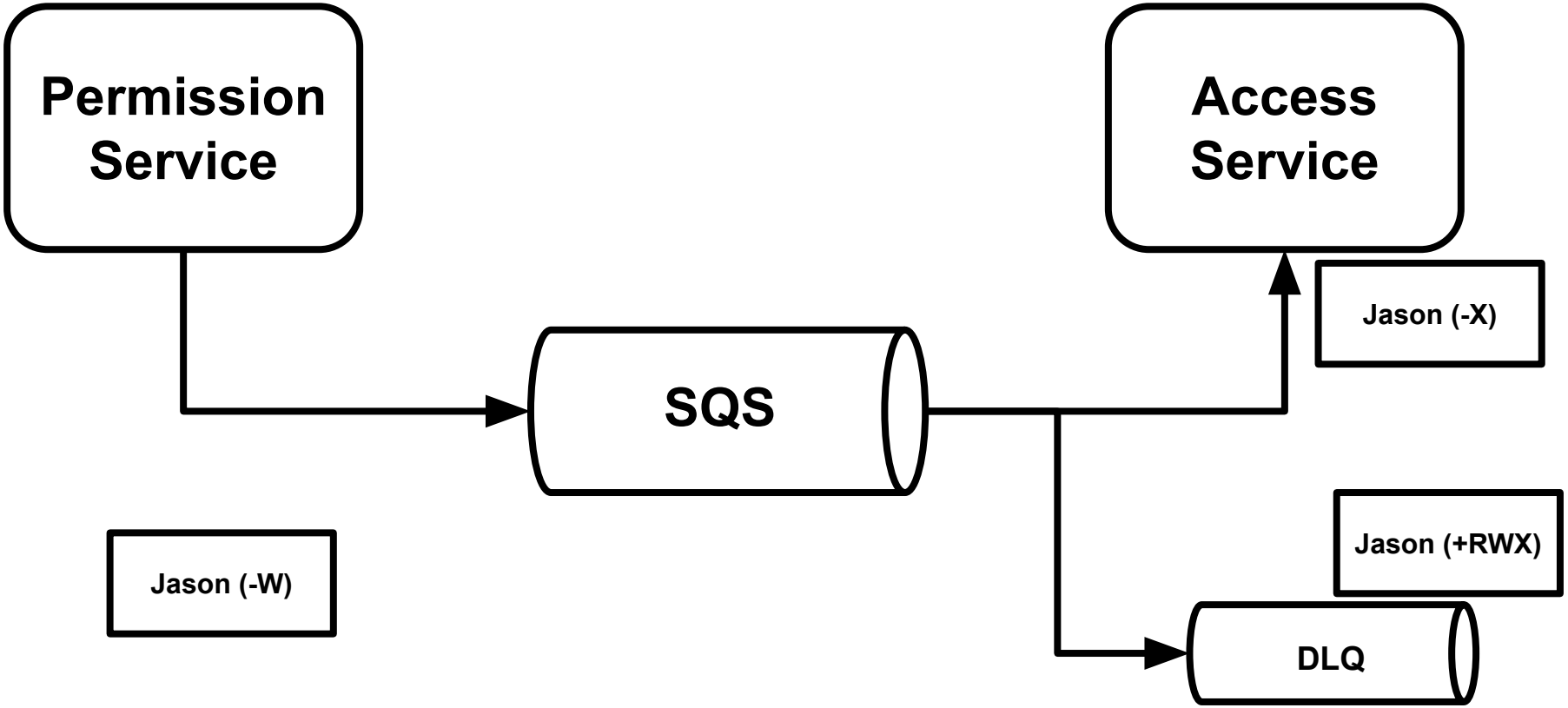
What can go wrong?



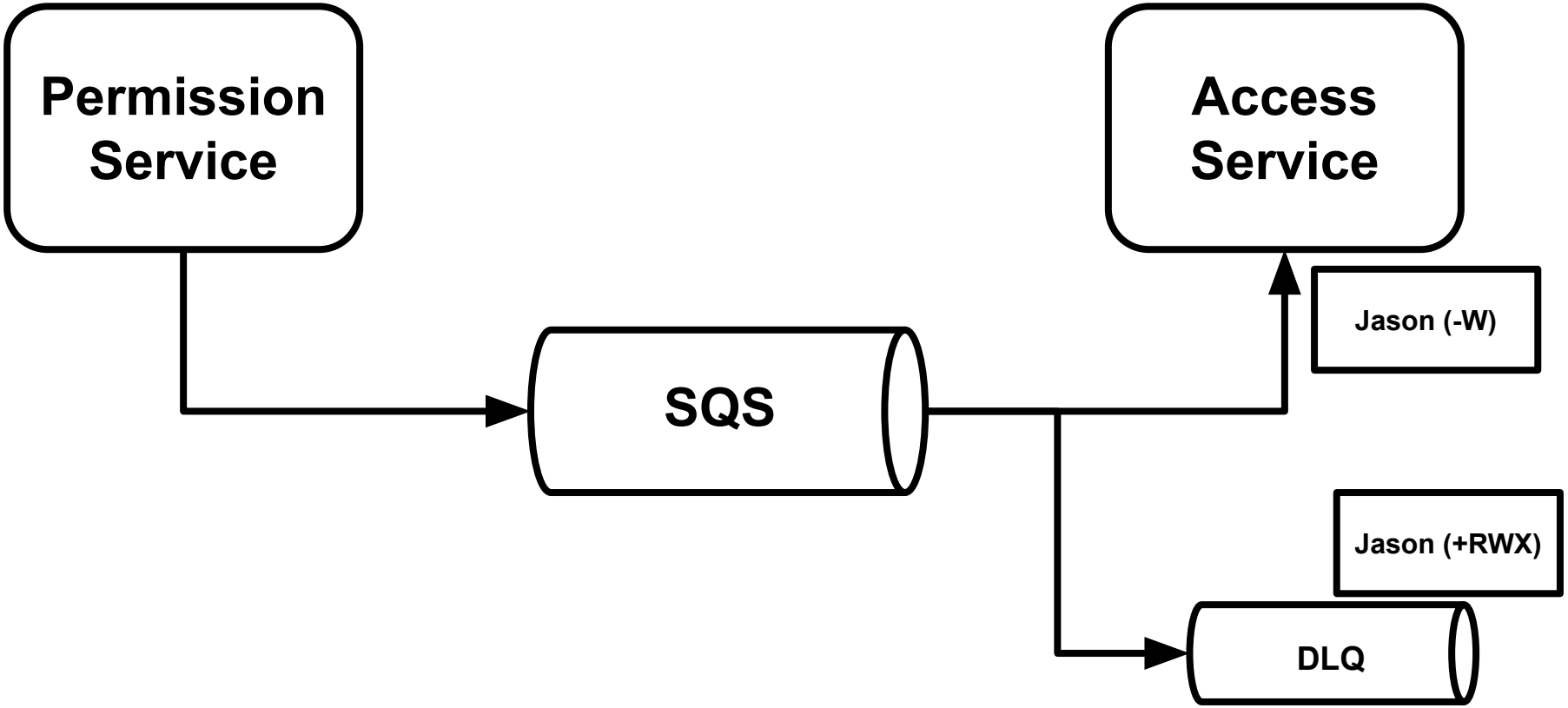
What can go wrong?



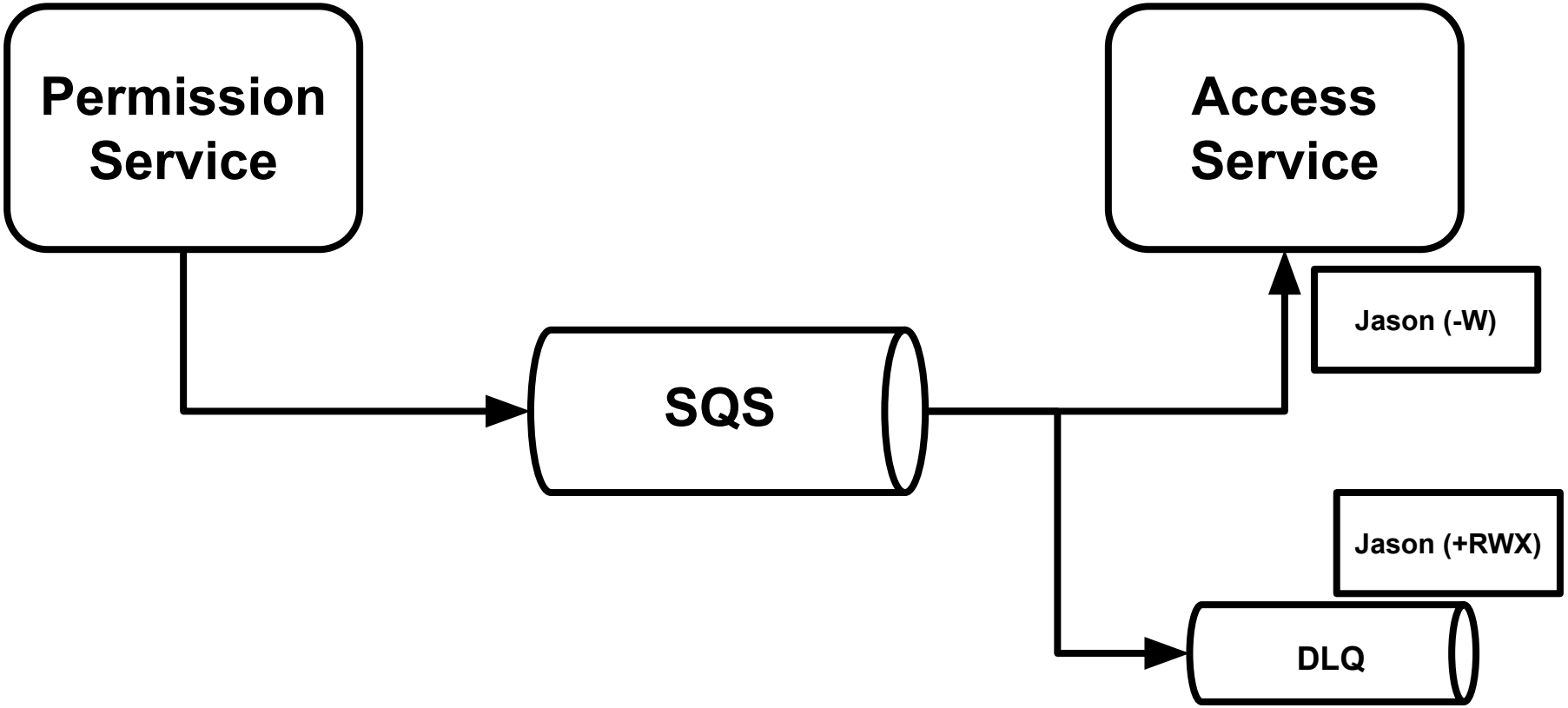
What can go wrong?

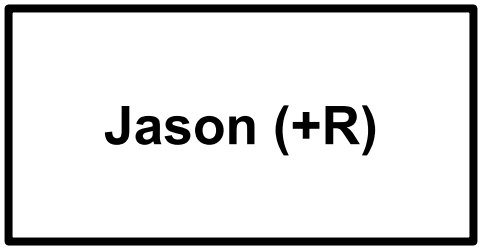


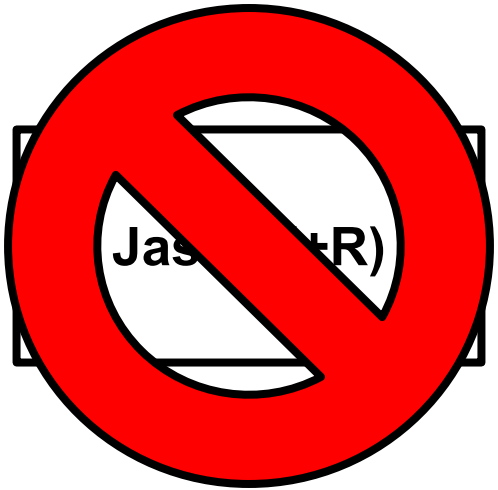
What can go wrong?



What can go wrong?




Jason (+R)


Jas R)

Don’t use a dead-letter queue with a FIFO queue if you don’t want to break the exact order of messages or operations.

https://docs.aws.amazon.com/AWSSimpleQueueService/latest/SQSDeveloperGuide/sqs-dead-letter-queues.html#sqs-dead-letter-queues-when-to-use

The “fine print” helps you to:

See the boundaries of what is possible

Design around the limitations

Avoid stupid mistakes

Hopefully hit horrible bugs in production

Understand the Problem Being Solved!

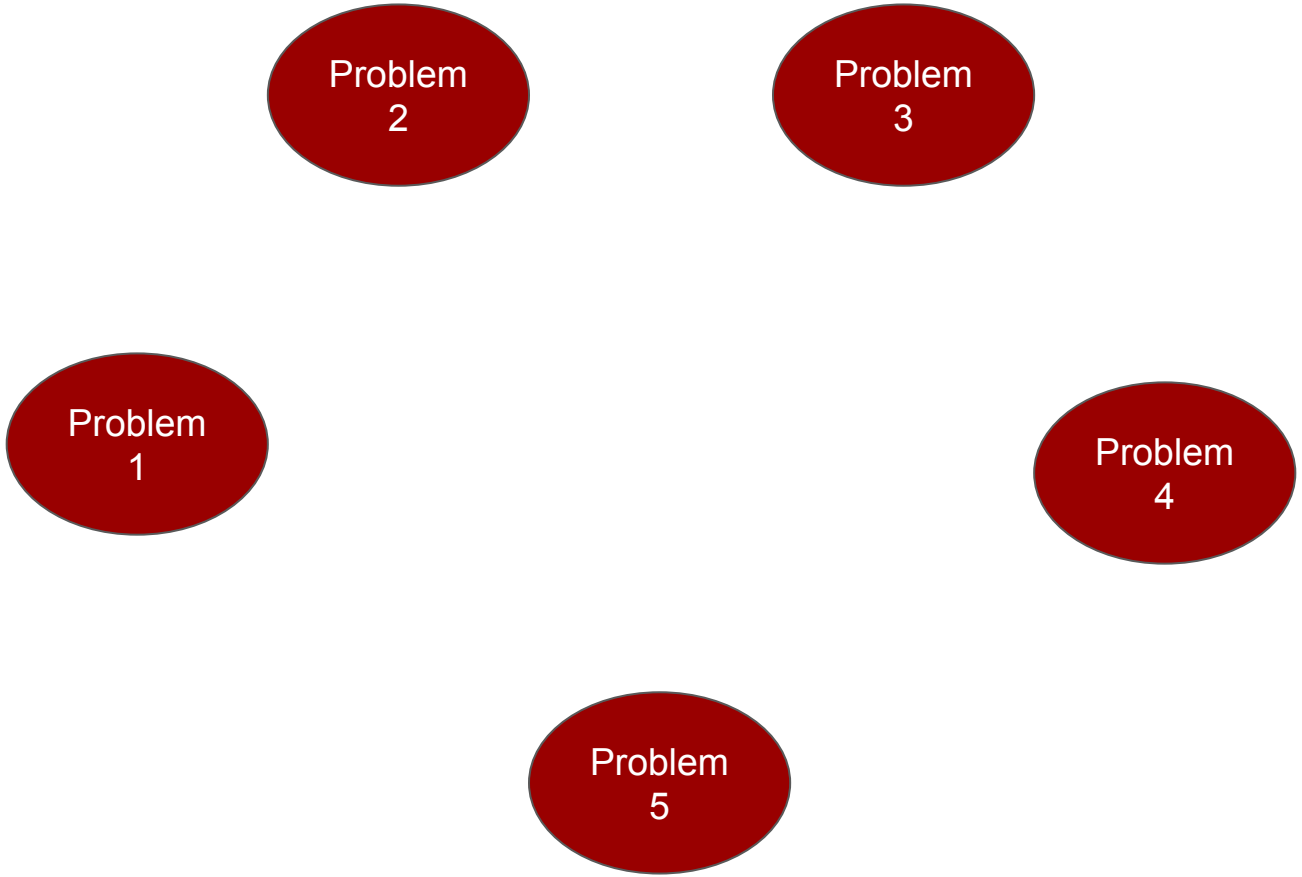
Use the right tool for the job

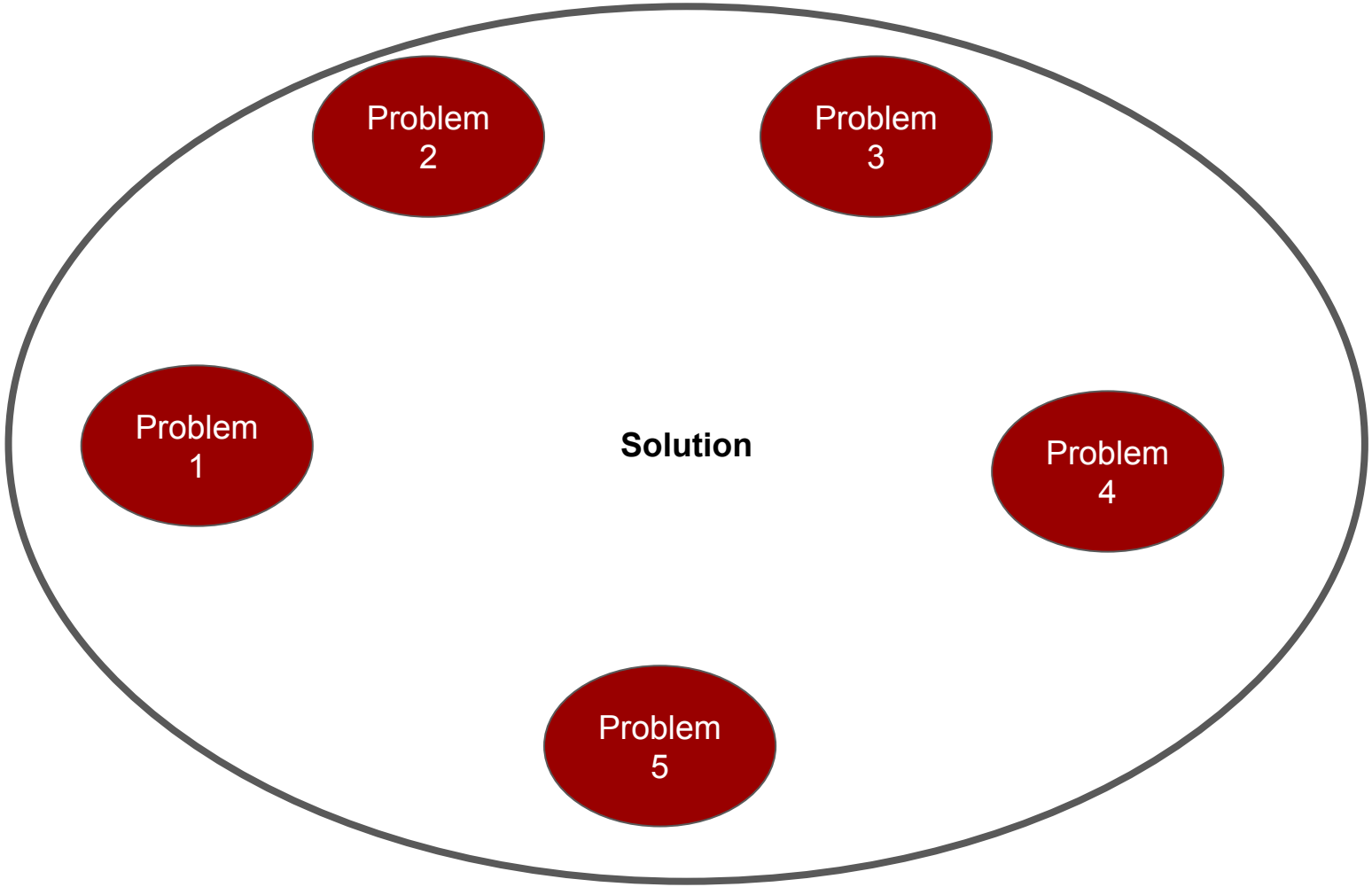
Use the right tool for the job

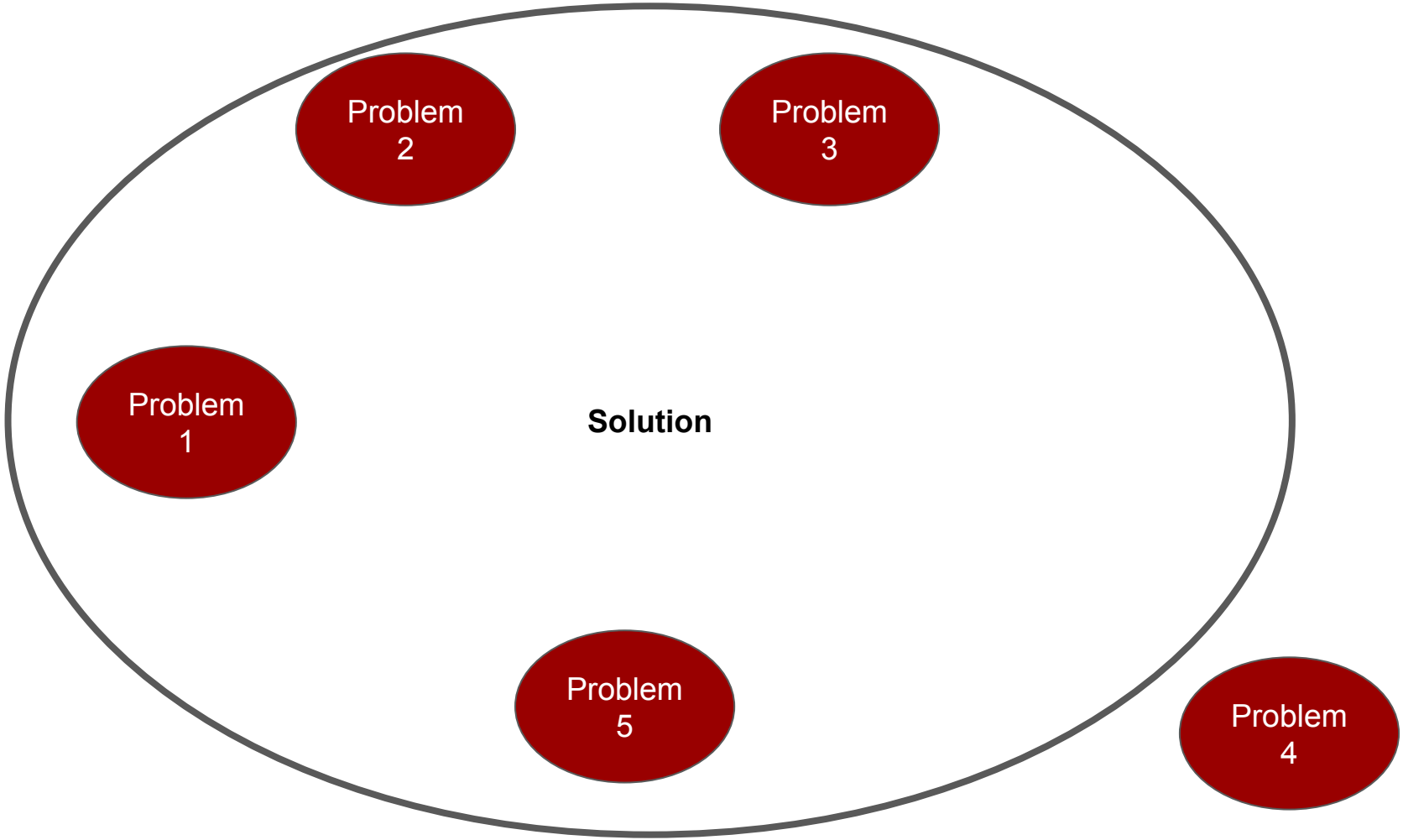
You can != You should

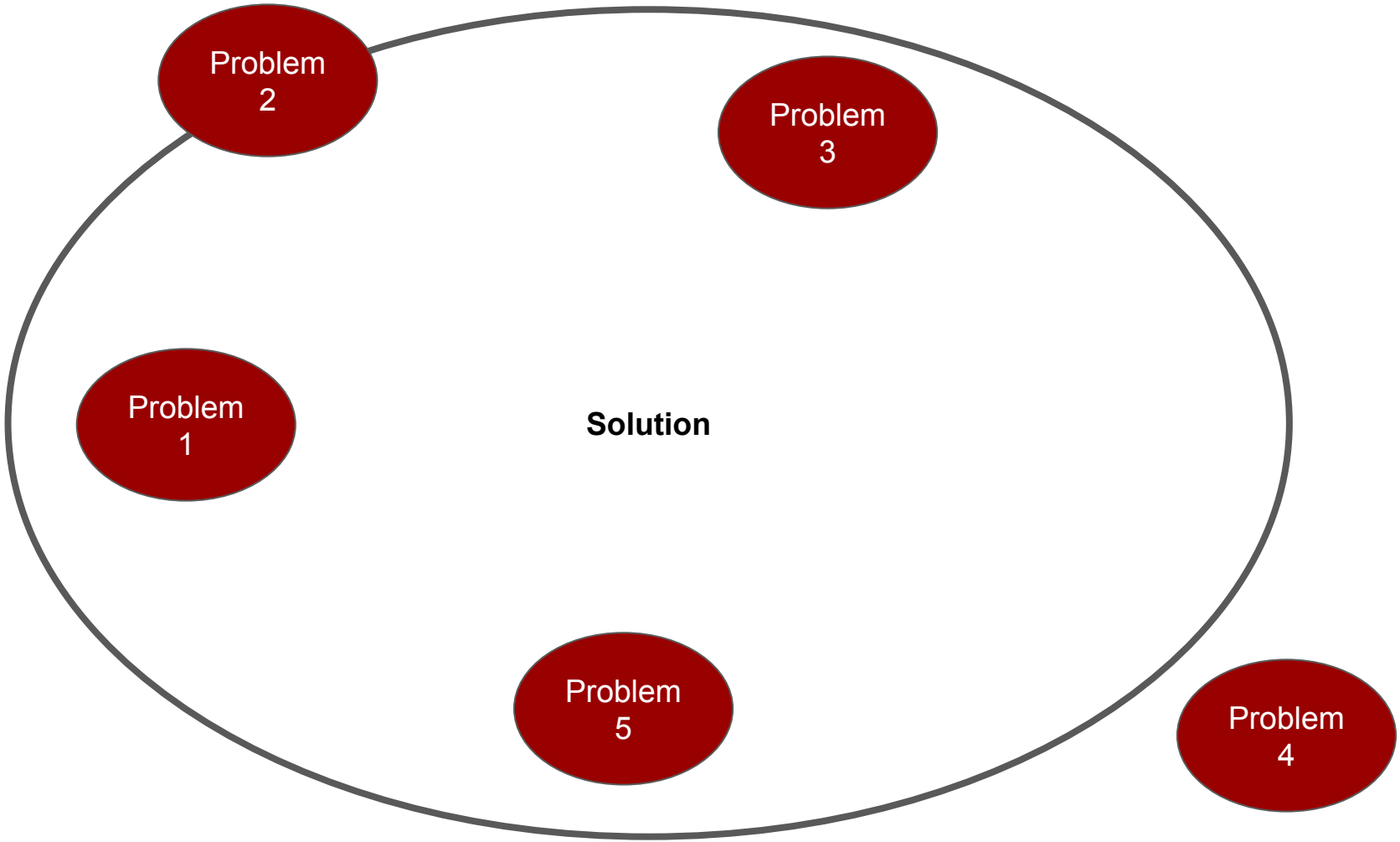
Building a REST Service in Bash?

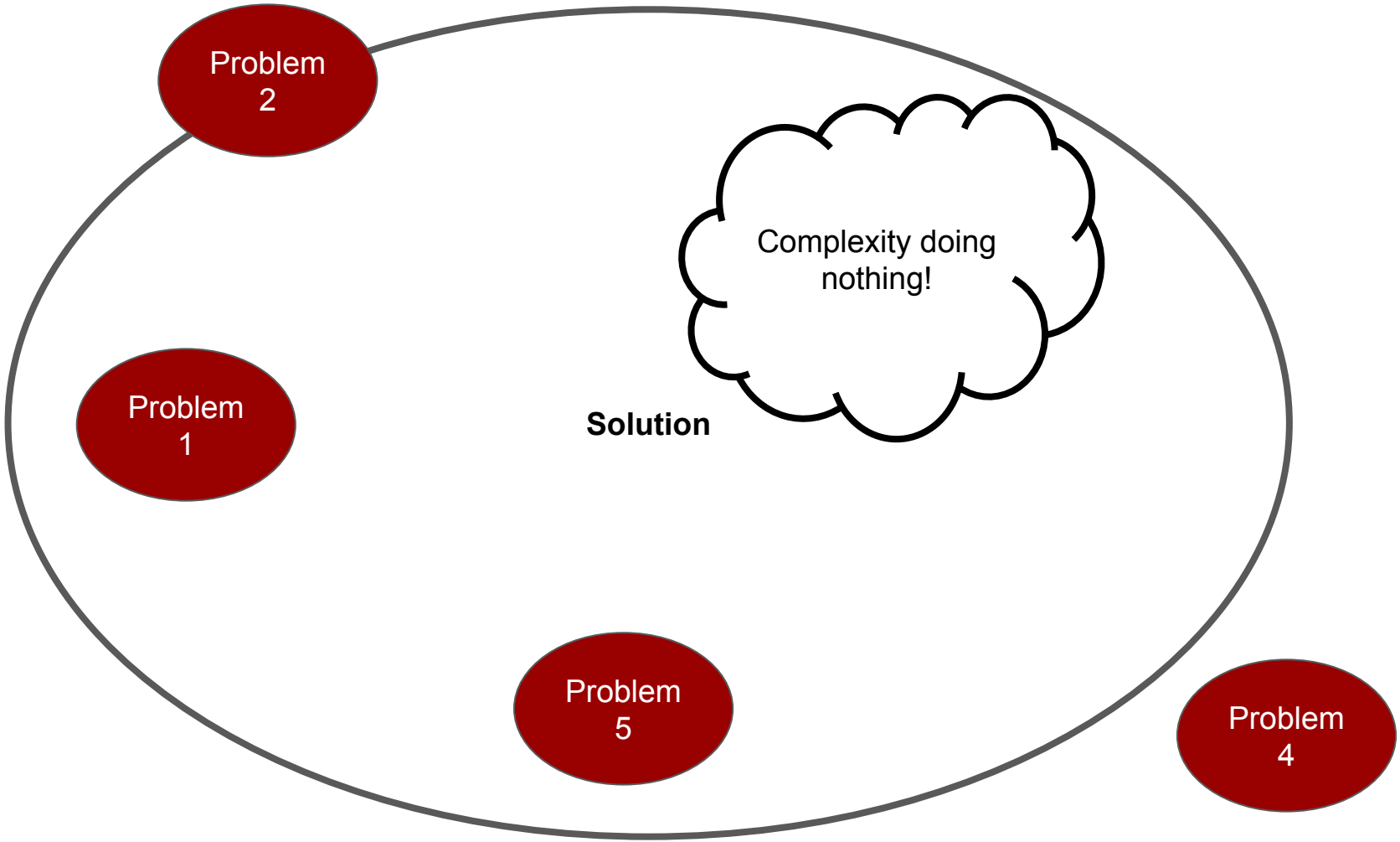
https://www.youtube.com/watch?v=jsXK11f4yE0

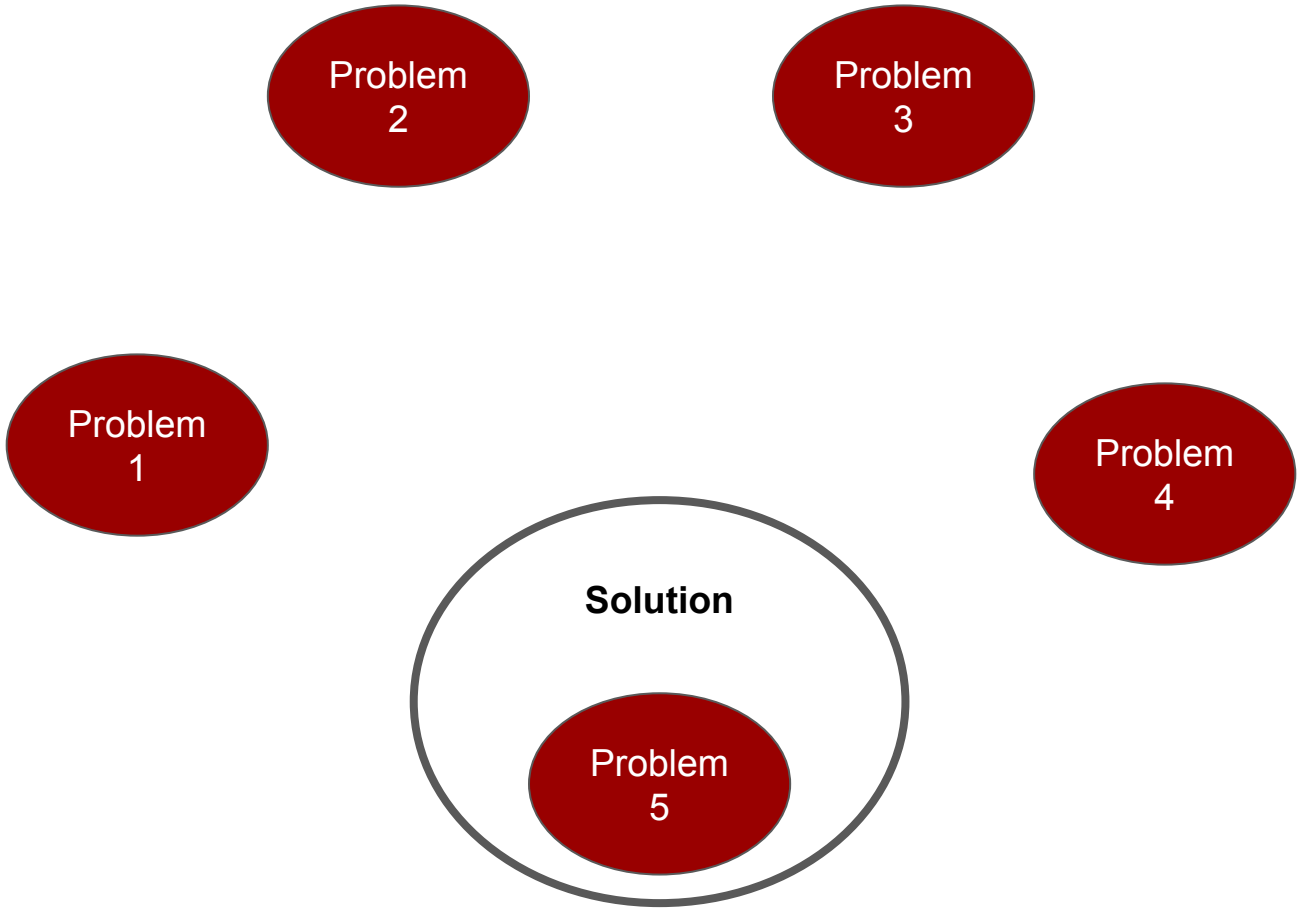

Problem 2
Problem 3
Problem 1
Problem 4
Problem 5

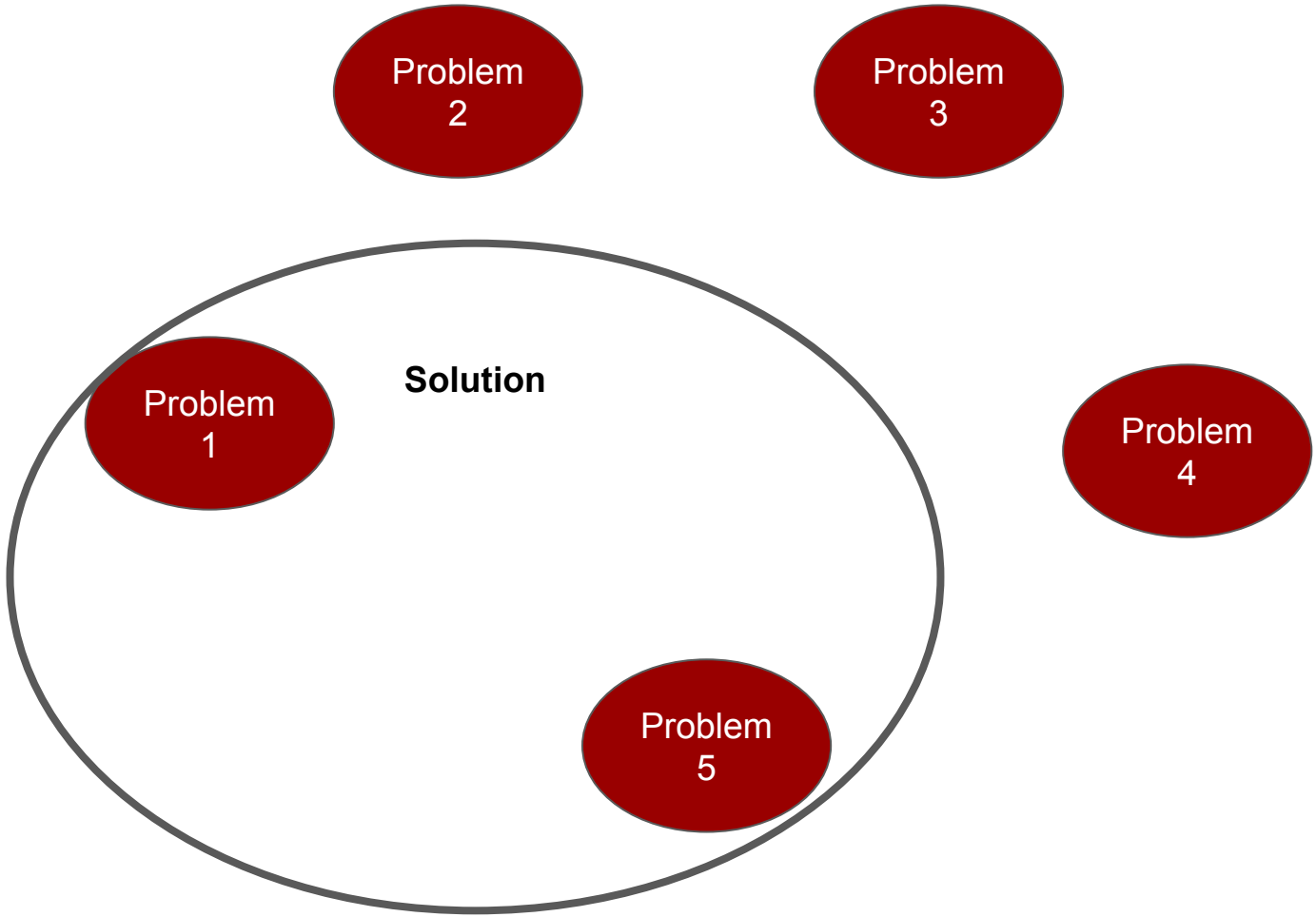

Problem 2
Problem 3
Problem 1
Solution
Problem 4
Problem 5

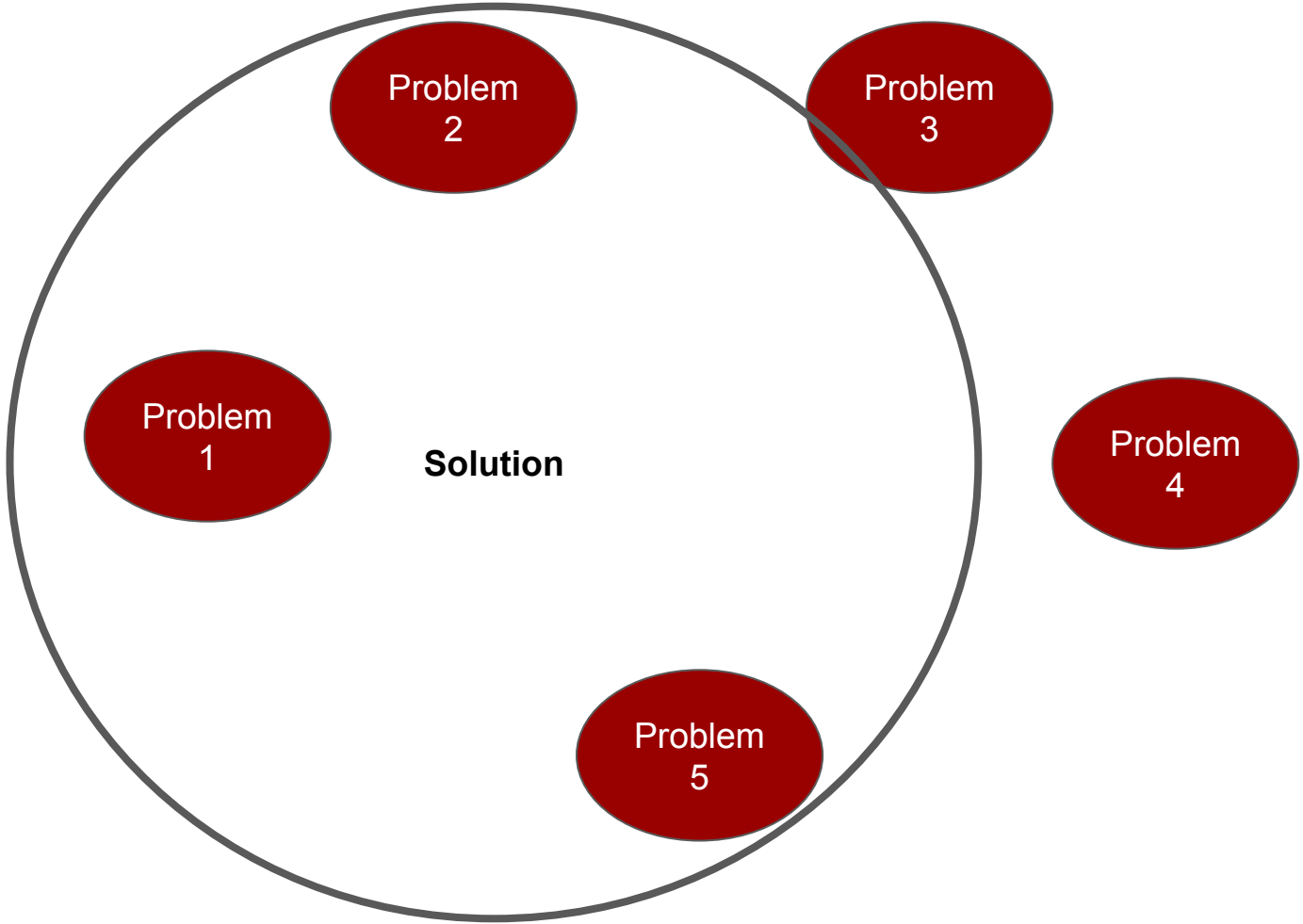

Problem 2
Problem 3
Problem 1
Solution
Problem 5
Problem 4

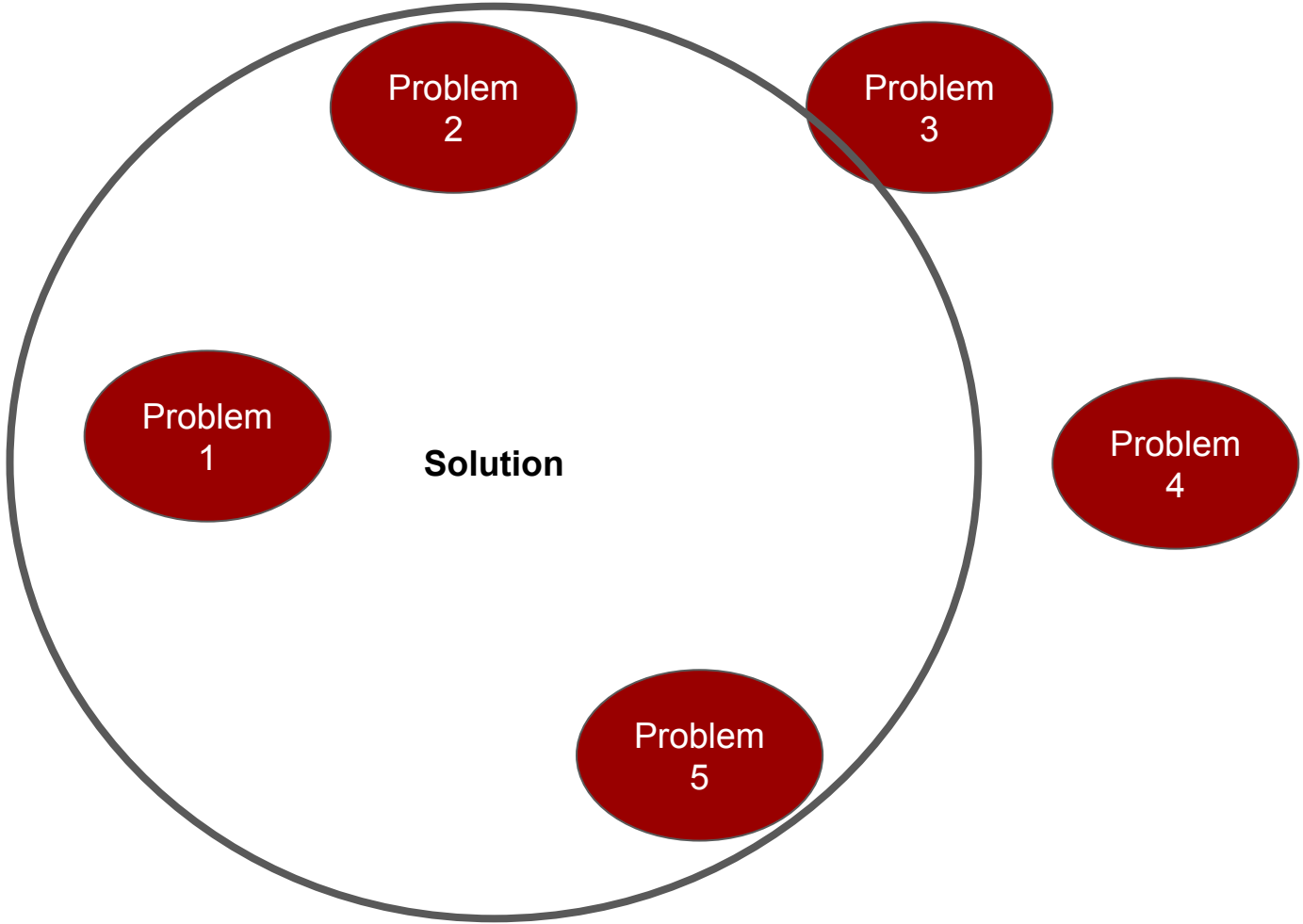

Problem 2
Problem 3
Problem 1
Solution
Problem 5
Problem 4

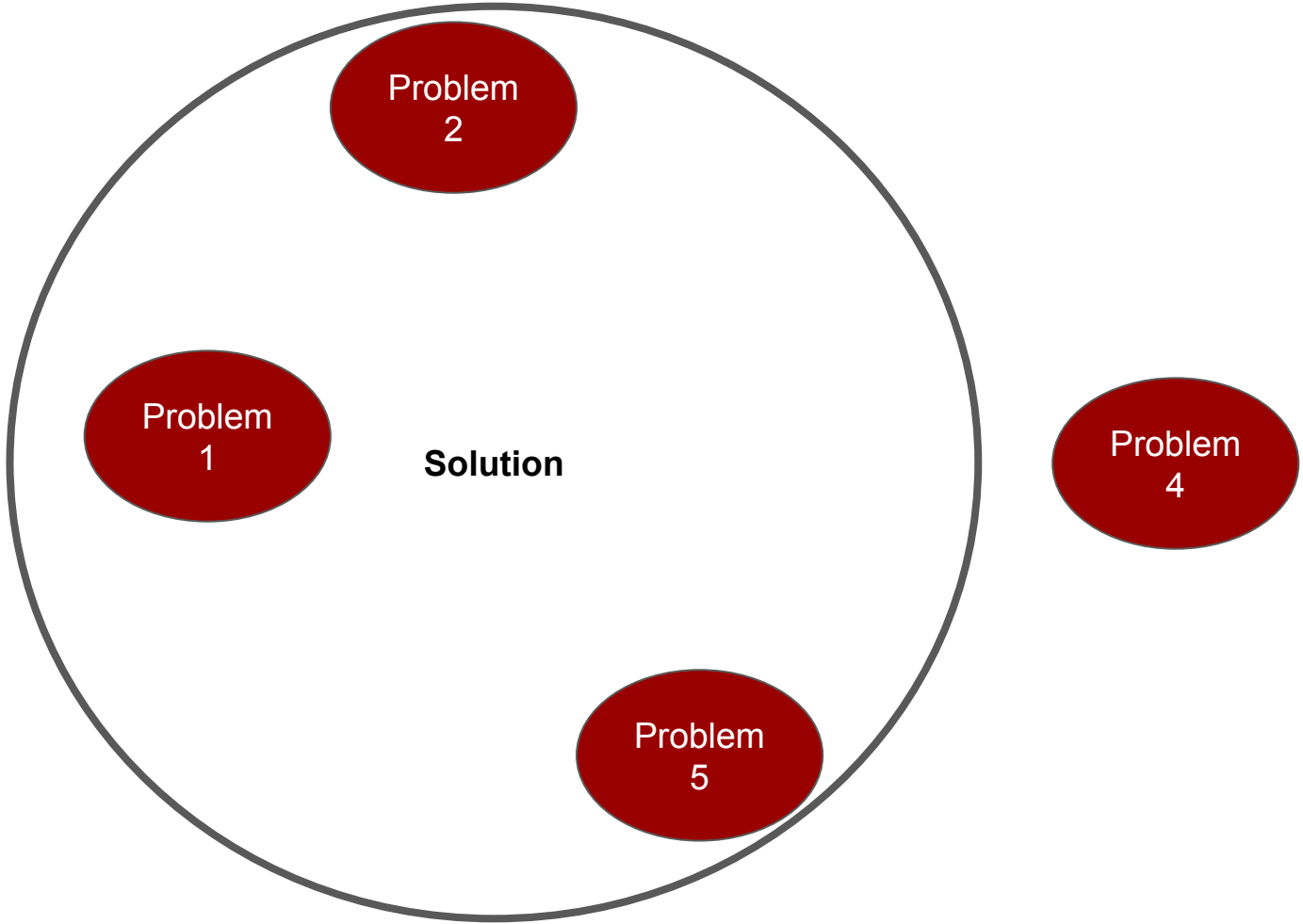

Problem 2
Complexity doing nothing!
Problem 1
Solution
Problem 5
Problem 4

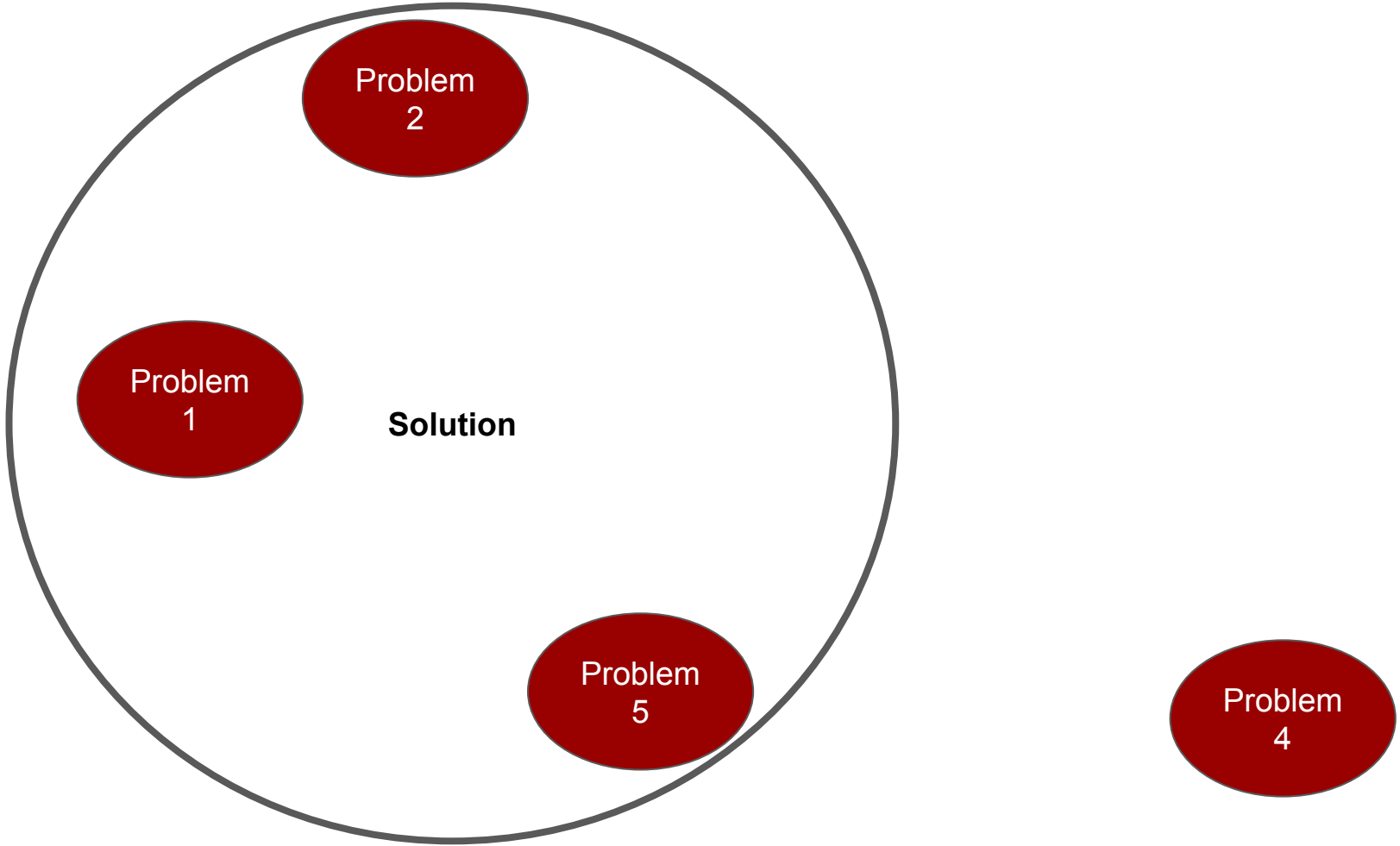

Problem 2
Problem 3
Problem 1
Problem 4
Solution
Problem 5

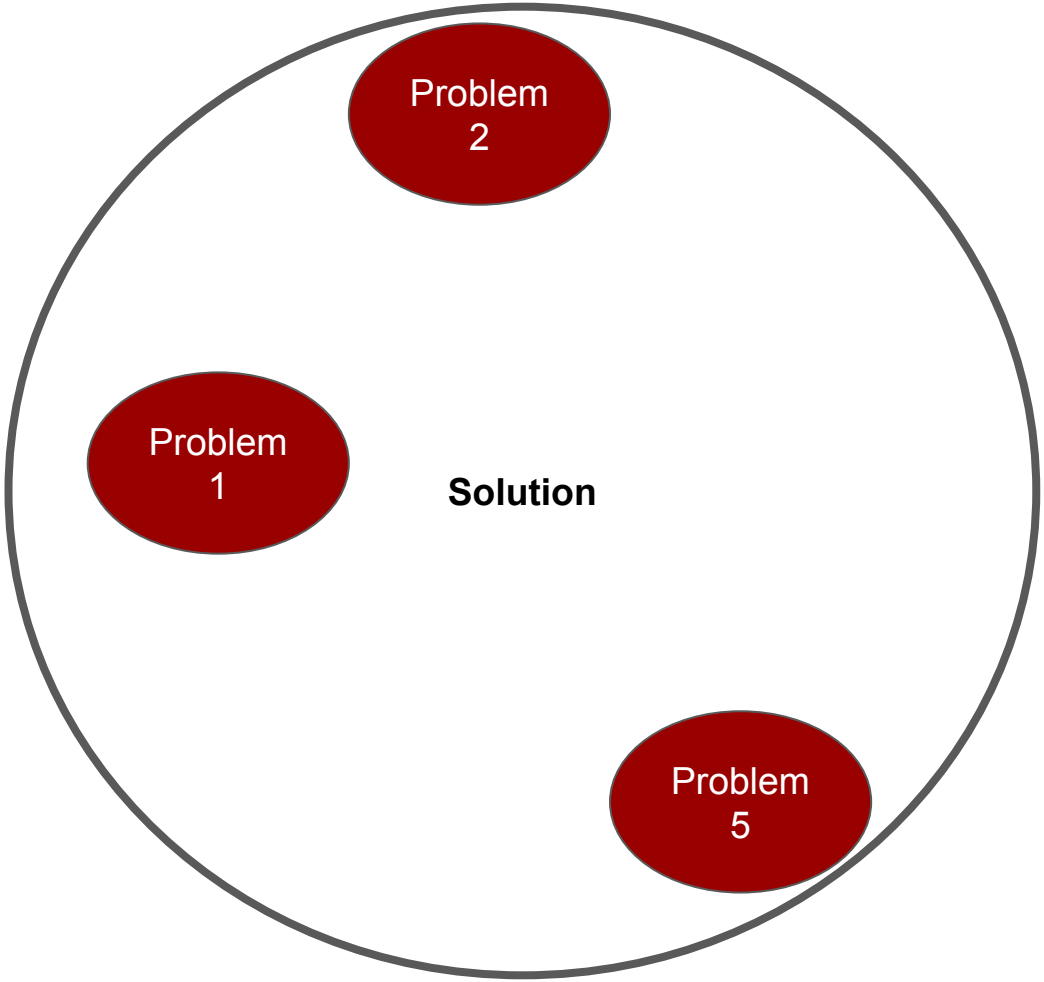

Problem 2
Problem 3
Solution
Problem 1
Problem 4
Problem 5

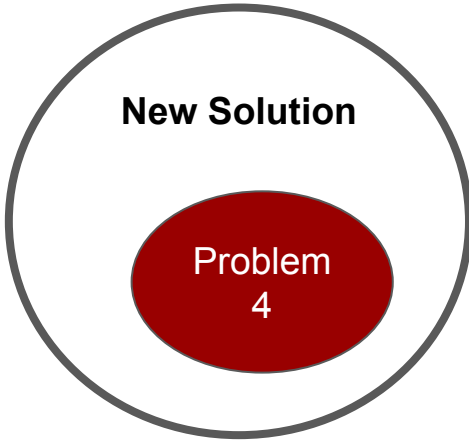

Problem
2
Problem
3
Problem
1
Solution
Problem
4
Problem
5


Problem
2
Problem
3
Problem
1
Solution
Problem
4
Problem
5


Problem 2
Problem 1
Solution
Problem 4
Problem 5


Problem 2
Problem 1
Solution
Problem 5
Problem 4

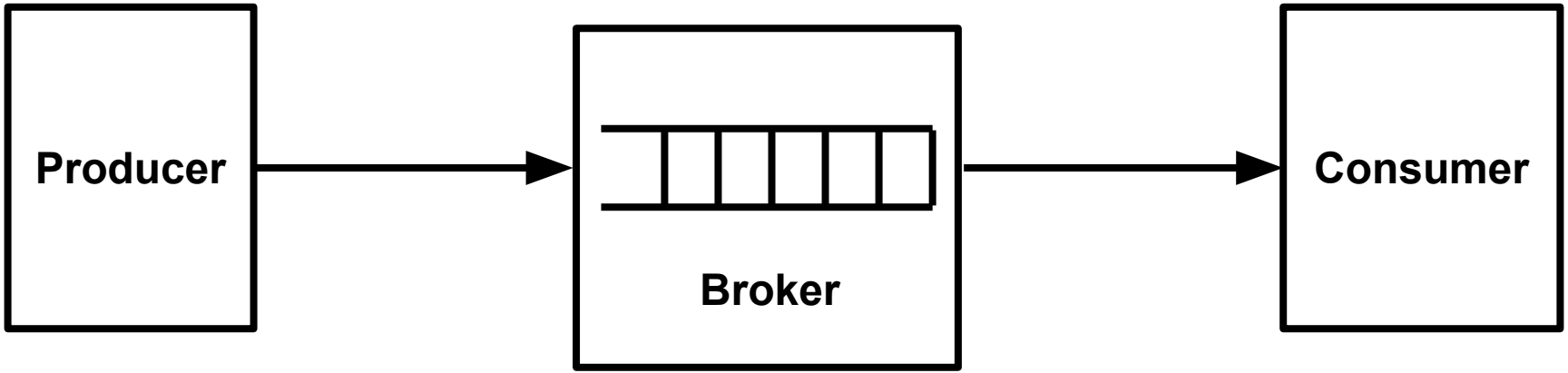

Problem
2
Problem
1
Solution
Problem
5


New Solution
Problem
4

A complex system that works is invariably found to have evolved from a simple system that worked. The inverse proposition also appears to be true: a complex system designed from scratch never works and cannot be made to work. You have to start over, beginning with a simple system.

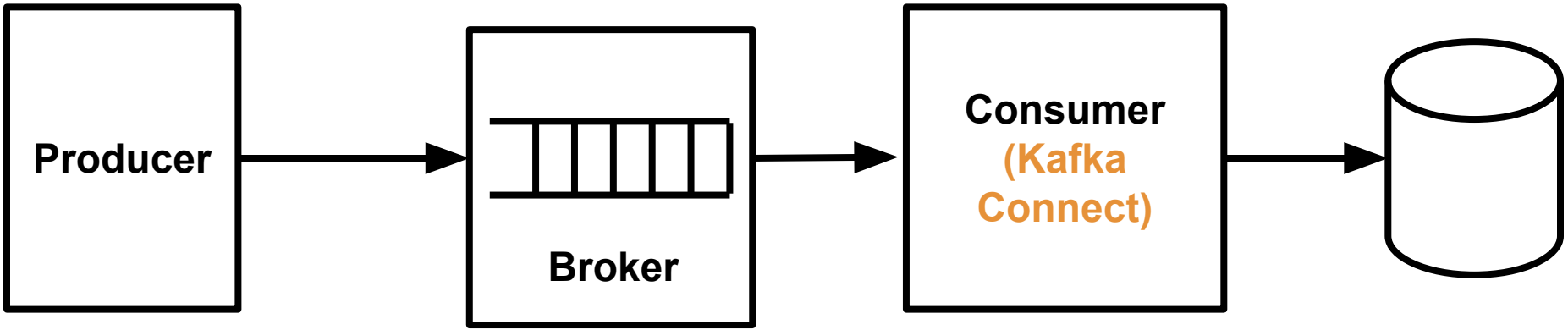
- John Gall (Gall's Law)

Apache Kafka



Kafka in 30 seconds

- Messages are sent to Topics
- Topics are divided into Partitions
- Partitions are hosted and stored on Brokers
- Partitions can be replicated for resiliency
- Message ordering is only guaranteed in a single Partition
- Producer asynchronously sends messages to Broker
- Broker replicates messages and sends ack back to Producer
- Consumer asynchronously reads messages from Broker
- Consumer moves offset when it consumes a message
- If a Consumer fails/restarts it starts over from the last offset it saved per partition.


Producer
Broker
Consumer
(Kafka
Connect)

Guarantees

At Most Once

At Least Once

Exactly Once

Guarantees

At Most Once

At Least Once

Exactly Once

Guarantees

At Most Once

At Least Once

~~Exactly Once~~

https://bravenewgeek.com/you-cannot-have-exactly-once-delivery/

Exactly-Once Semantics Are Possible: Here's How Kafka Does it

NEHA NARKHEDE

JUNE 30, 2017

https://www.confluent.io/blog/exactly-once-semantics-are-possible-heres-how-apache-kafka-does-it/

Kafka in 30 seconds

- Messages are sent to Topics
- Topics are divided into Partitions
- Partitions are hosted and stored on Brokers
- Partitions can be replicated for resiliency
- Message ordering is only guaranteed in a single Partition
- Producer asynchronously sends messages to Broker
- Broker replicates messages and sends ack back to Producer
- Consumer asynchronously reads messages from Broker
- Consumer moves offset when it consumes a message
- If a Consumer fails/restarts it starts over from the last offset it saved per partition.

Kafka in 30 seconds

- Messages are sent to Topics
- Topics are divided into Partitions
- Partitions are hosted and stored on Brokers
- Partitions can be replicated for resiliency
- Message ordering is only guaranteed in a single Partition
- Producer asynchronously sends messages to Broker
- Broker replicates messages and sends ack back to Producer
- Consumer asynchronously reads messages from Broker
- Consumer moves offset when it consumes a message
- If a Consumer fails/restarts it starts over from the last offset it saved per partition.

Exactly Once

Guarantees

At Most Once

At Least Once

~~Exactly Once~~

https://bravenewgeek.com/you-cannot-have-exactly-once-delivery/

Delivery Guarantees

At Most Once

At Least Once

~~**Exactly Once**~~

https://bravenewgeek.com/you-cannot-have-exactly-once-delivery/

Exactly Once: Delivery

Exactly Once: ~~Delivery~~

Processing

Observably

Exactly Once: ~~Delivery~~

Processing

```
Y = 5
X = 5 + Y
```

```
Y = 5
X = 5 + Y
```

```
X = 10
```

Exactly Once Processing

```
Y = 5
X = 5 + Y
```

```
X = 10
```

Exactly Once Processing

```
Y = 5
X = 5 + Y
```

```
Y = 5
for(i=0;i<10;i++)
    X = 5 + Y
```

```
X = 10
```

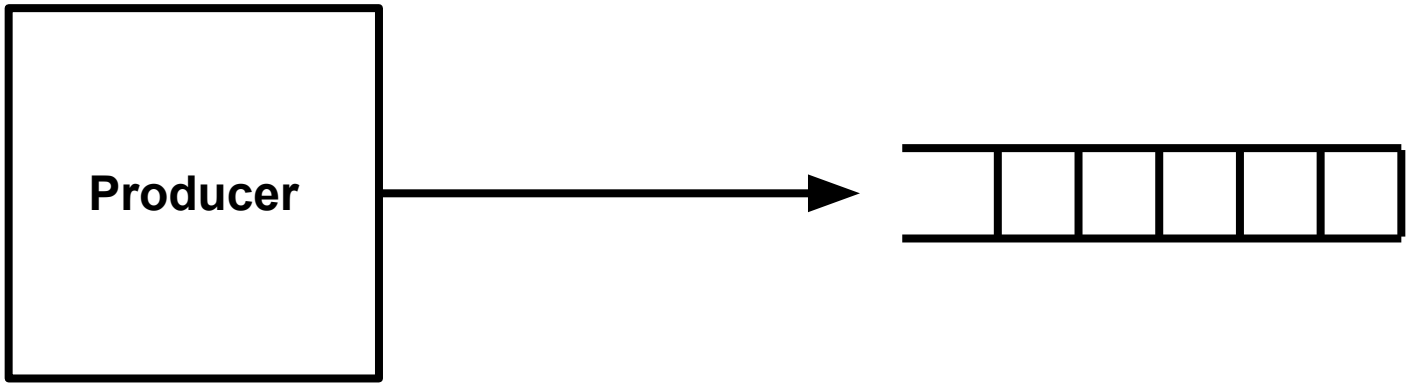
Exactly Once Processing

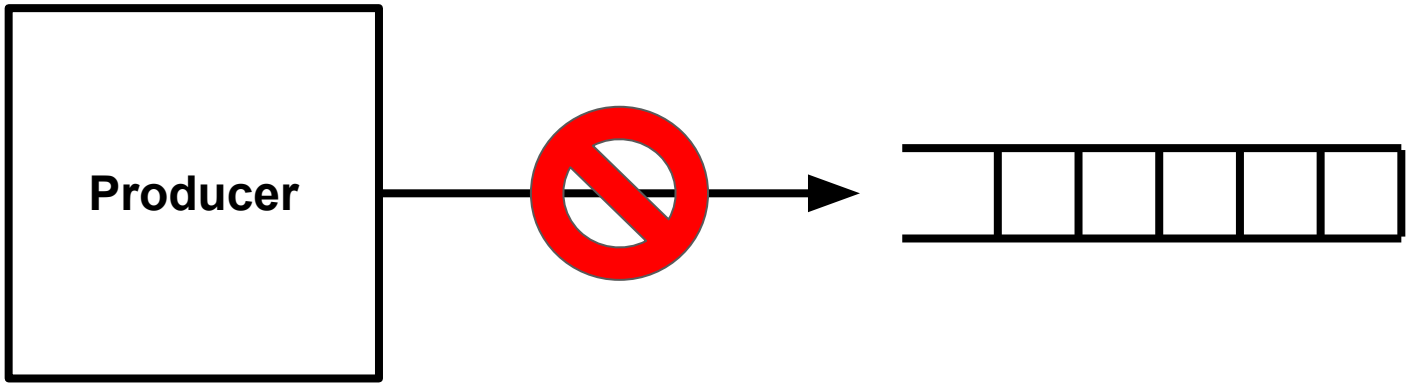
```
Y = 5
X = 5 + Y
```

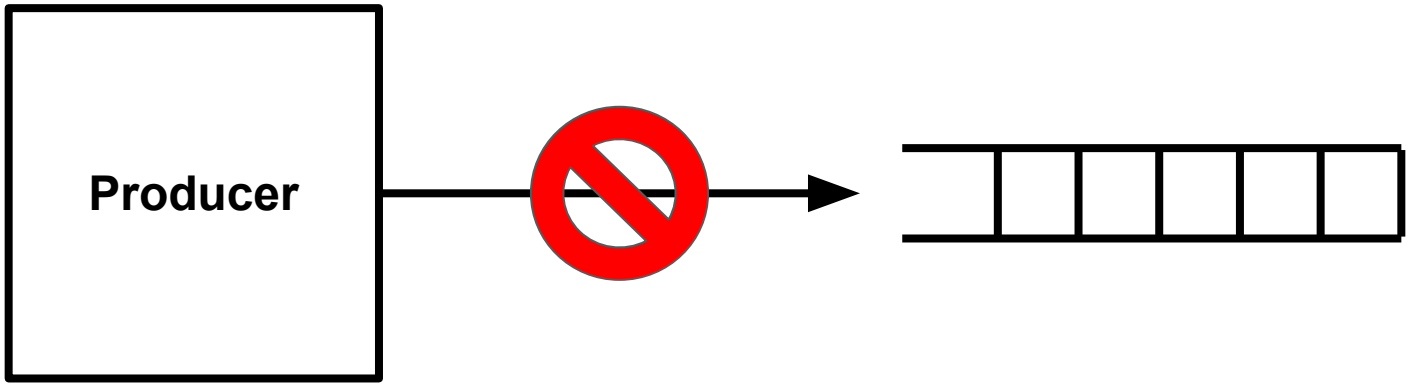
Observably Exactly Once Processing

```
Y = 5
for(i=0;i<10;i++)
   X = 5 + Y
```

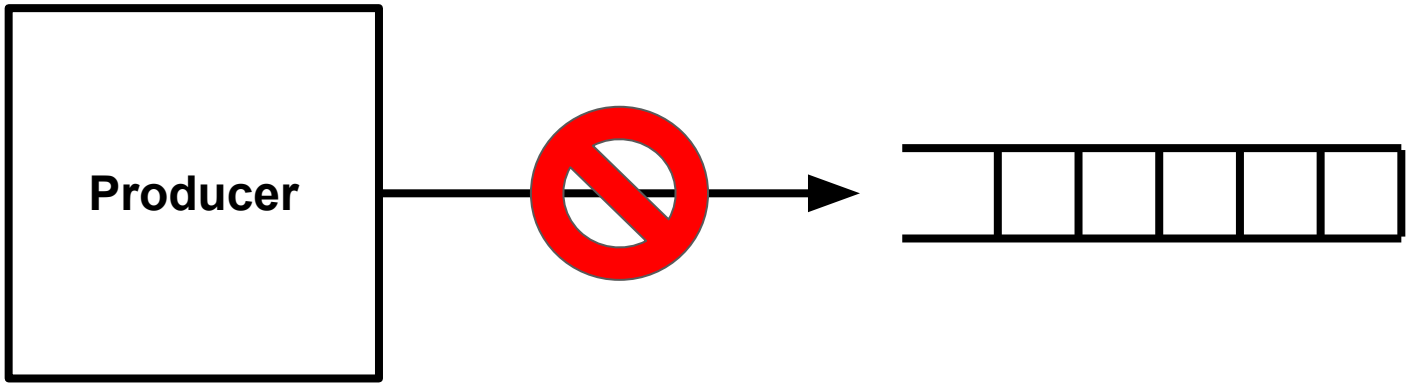
```
X = 10
```


Producer


Producer



Idempotent Producers

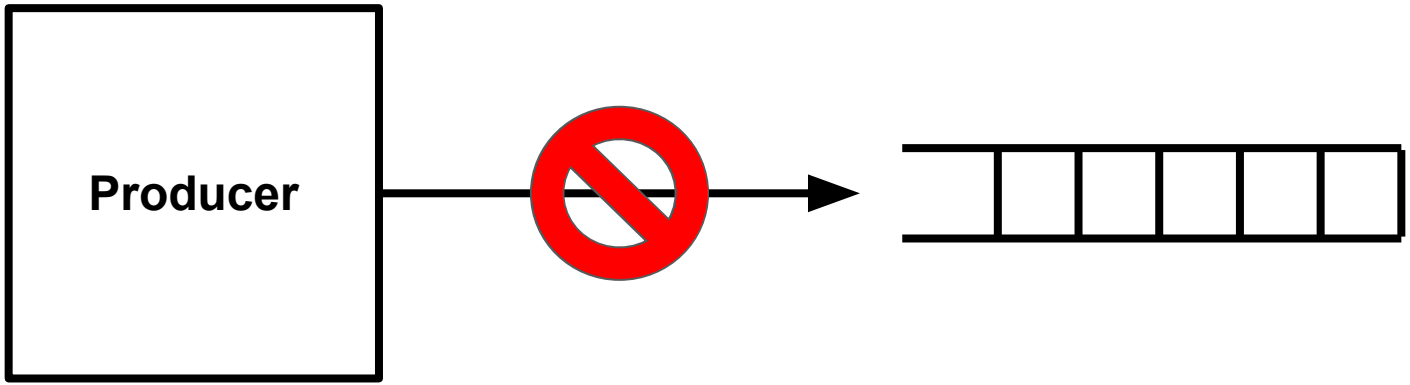


Idempotent Producers

Tied to single producer session

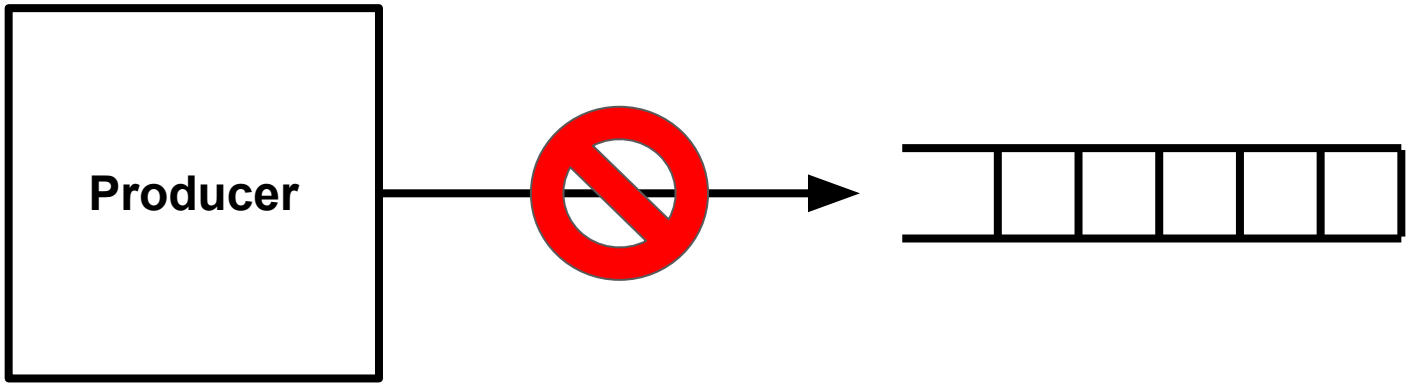
Tied to a single partition

https://cwiki.apache.org/confluence/display/KAFKA/Idempotent+Producer



Idempotent Producers

Transactions

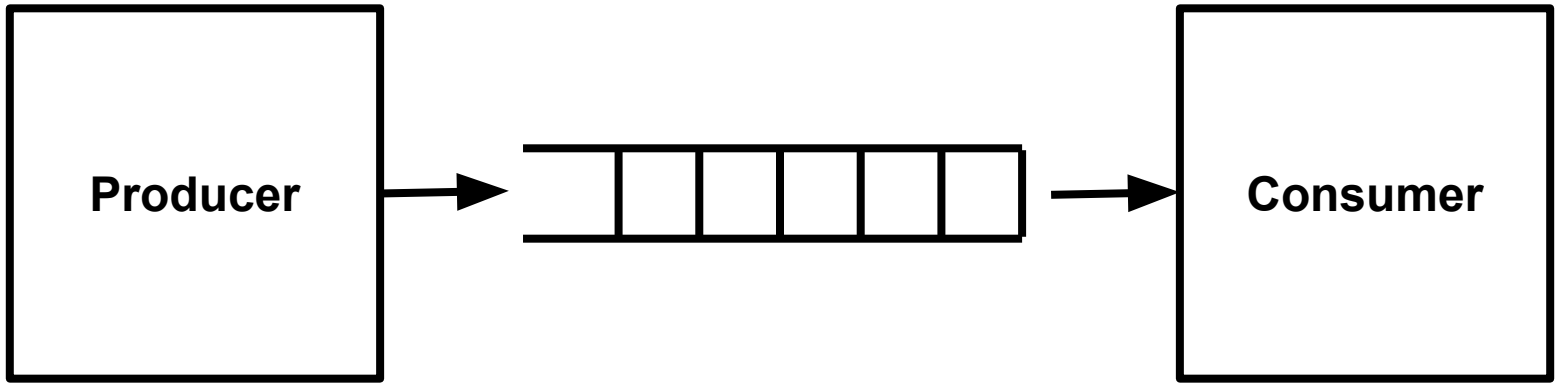


Transactions

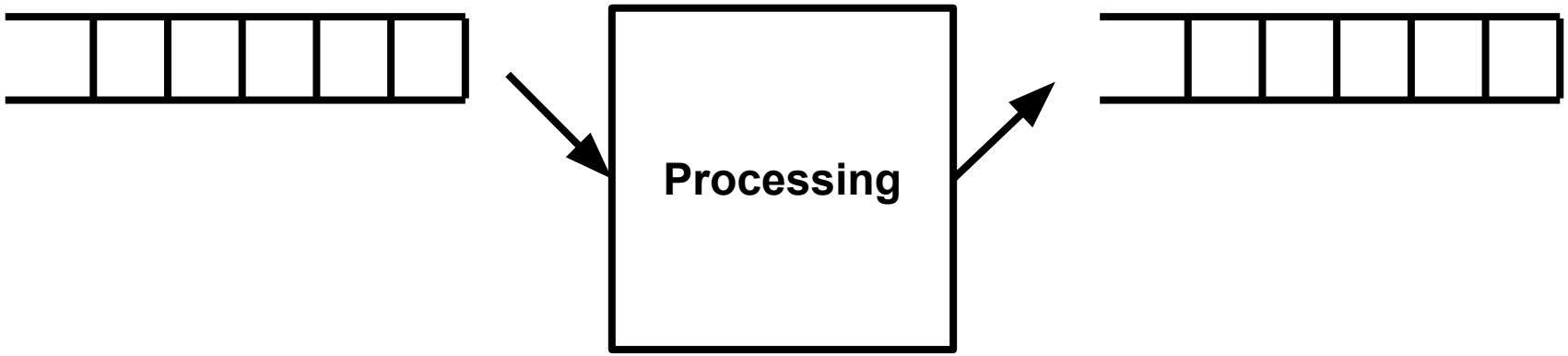
Writes start/stop messages for transaction

Requires consumer configuration

https://cwiki.apache.org/confluence/display/KAFKA/KIP-98+-+Exactly+Once+Delivery+and+Transactional+Messaging

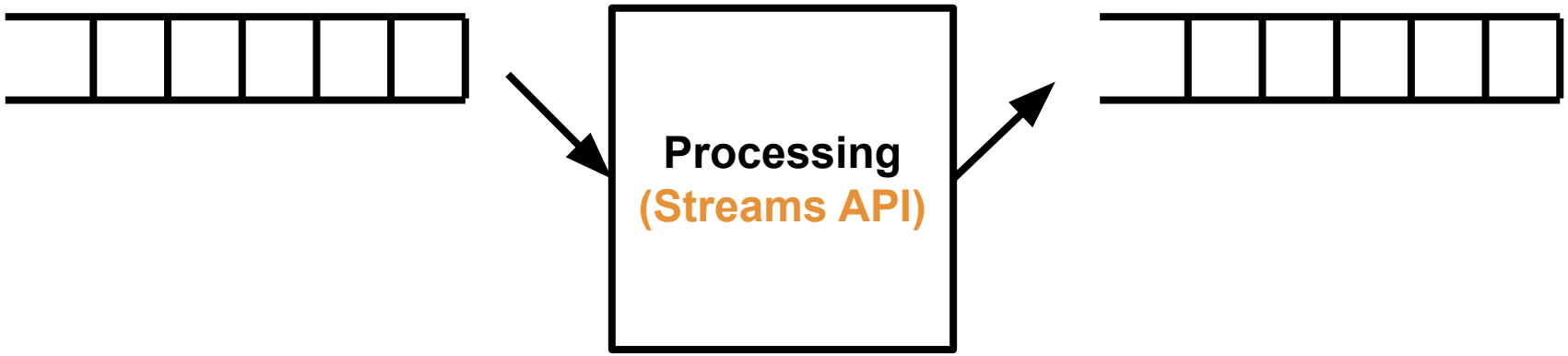


Tighter Coupling
Producer, Broker, and Consumer



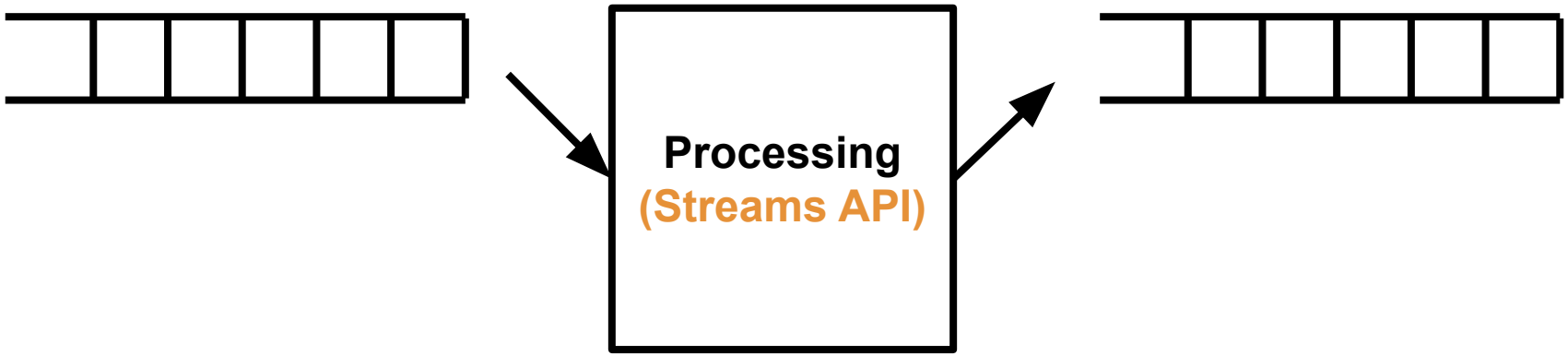
“stream processing is nothing but a read-process-write operation on a Kafka topic”

https://www.confluent.io/blog/exactly-once-semantics-are-possible-heres-how-apache-kafka-does-it/



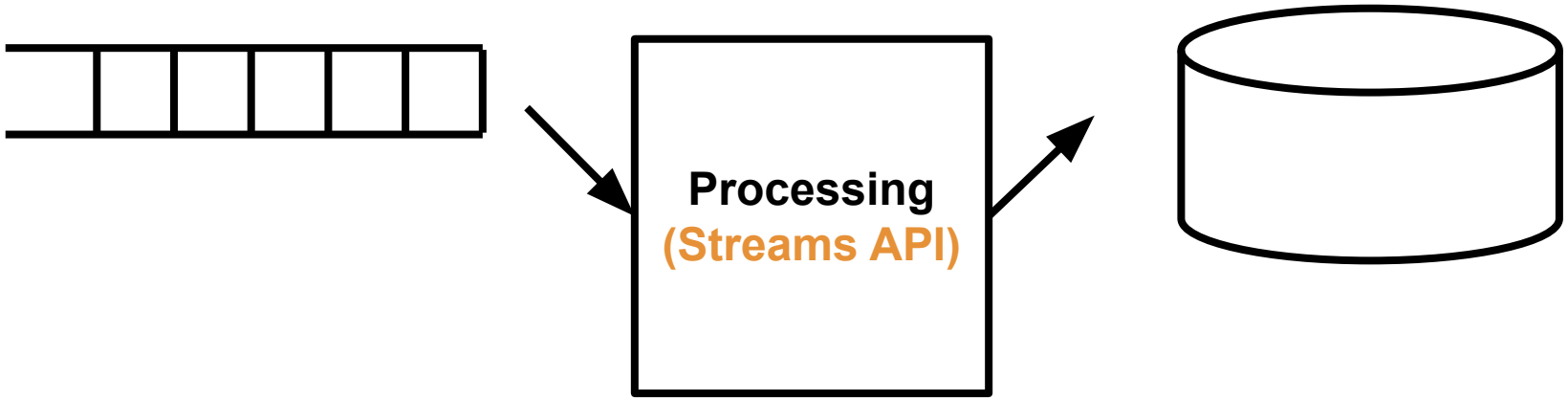
“stream processing is nothing but a read-process-write operation on a Kafka topic”

https://www.confluent.io/blog/exactly-once-semantics-are-possible-heres-how-apache-kafka-does-it/



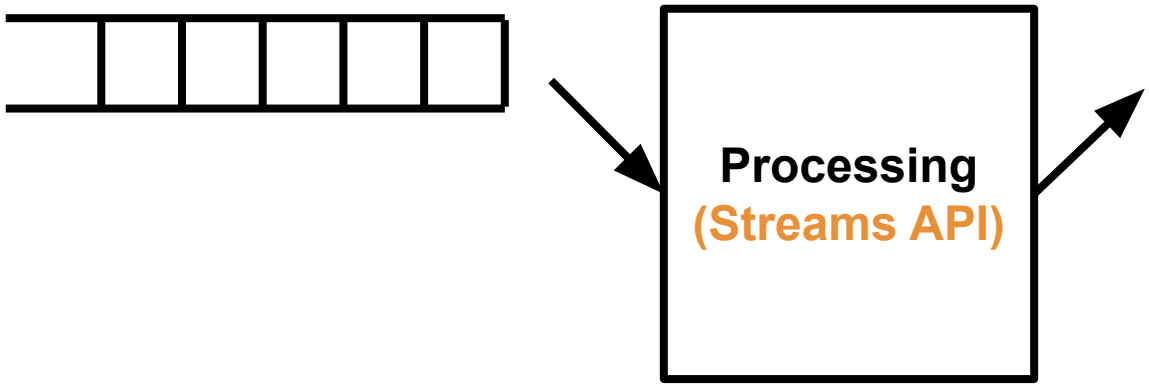
So we must qualify the phrase “exactly once stream processing.” It refers to consuming from a topic, materializing intermediate state in a Kafka topic and producing to one, not all possible computations done on a message using the Streams API.

https://www.confluent.io/blog/exactly-once-semantics-are-possible-heres-how-apache-kafka-does-it/



Stream processing systems that only rely on external data systems to materialize state support weaker guarantees for exactly-once stream processing.

https://www.confluent.io/blog/exactly-once-semantics-are-possible-heres-how-apache-kafka-does-it/



Exactly-once processing is an end-to-end guarantee and the application has to be designed to not violate the property as well ...You still need to get the data out of Kafka, though.

https://www.confluent.io/blog/exactly-once-semantics-are-possible-heres-how-apache-kafka-does-it/

Do the problem statements overlap?

Have you identified the discrepancies?

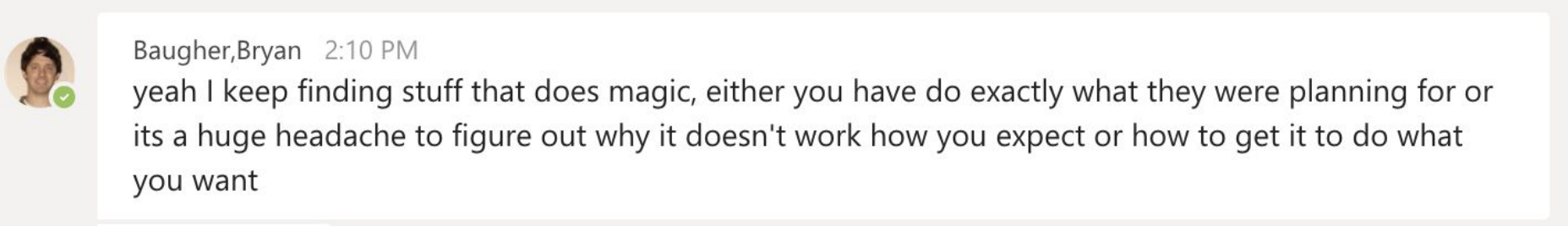
Can you live or compensate for them?

Don’t worry all is not lost!

Before You Believe a Guarantee...

1. Definitions Matter!
2. Read the Fine Print!
3. Understand the Problem Being Solved!

Guaranteed
Certified* Industry** Expert***



*** He showed me a certificate he printed off the internet**
**** He is employed**
***** Ask him anything about Sonic the Hedgehog.******
****** He's smart about other things too.**

References

- https://en.wikipedia.org/wiki/Consistency_(database_systems)
- https://jepsen.io/consistency/models/linearizable
- https://cs.brown.edu/~mph/HerlihyW90/p463-herlihy.pdf
- http://www.bailis.org/blog/linearizability-versus-serializability/
- https://martin.kleppmann.com/2015/05/11/please-stop-calling-databases-cp-or-ap.html
- https://www.cl.cam.ac.uk/~pes20/weakmemory/cacm.pdf
- https://bravenewgeek.com/you-cannot-have-exactly-once-delivery/
- https://bravenewgeek.com/you-cannot-have-exactly-once-delivery-redux/
- https://medium.com/@jaykreps/exactly-once-support-in-apache-kafka-55e1fdd0a35f
- https://medium.com/@jaykreps/exactly-once-one-more-time-901181d592f9
- https://www.confluent.io/blog/exactly-once-semantics-are-possible-heres-how-apache-kafka-does-it/