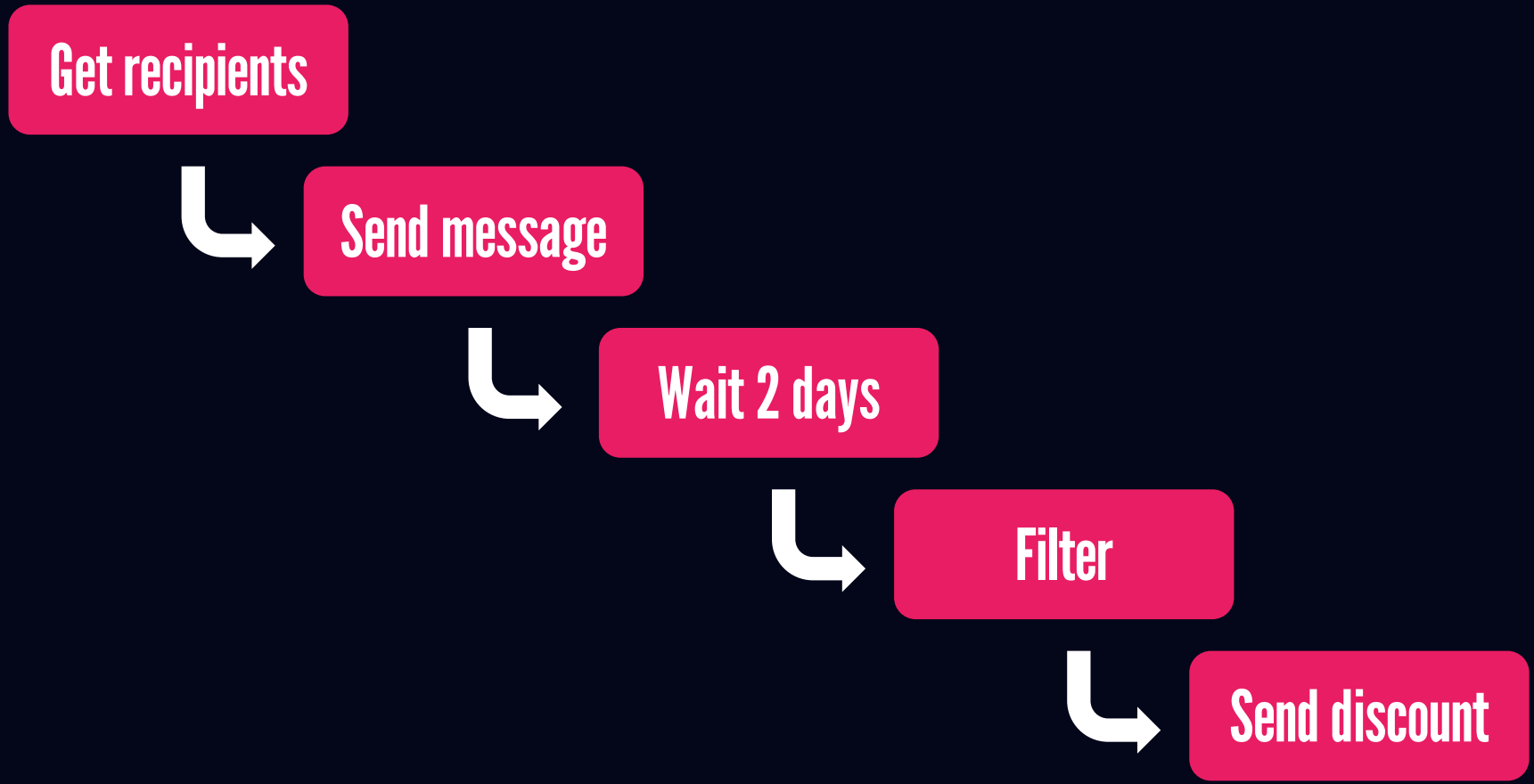
Microservices Will Break... Unless You Do This!

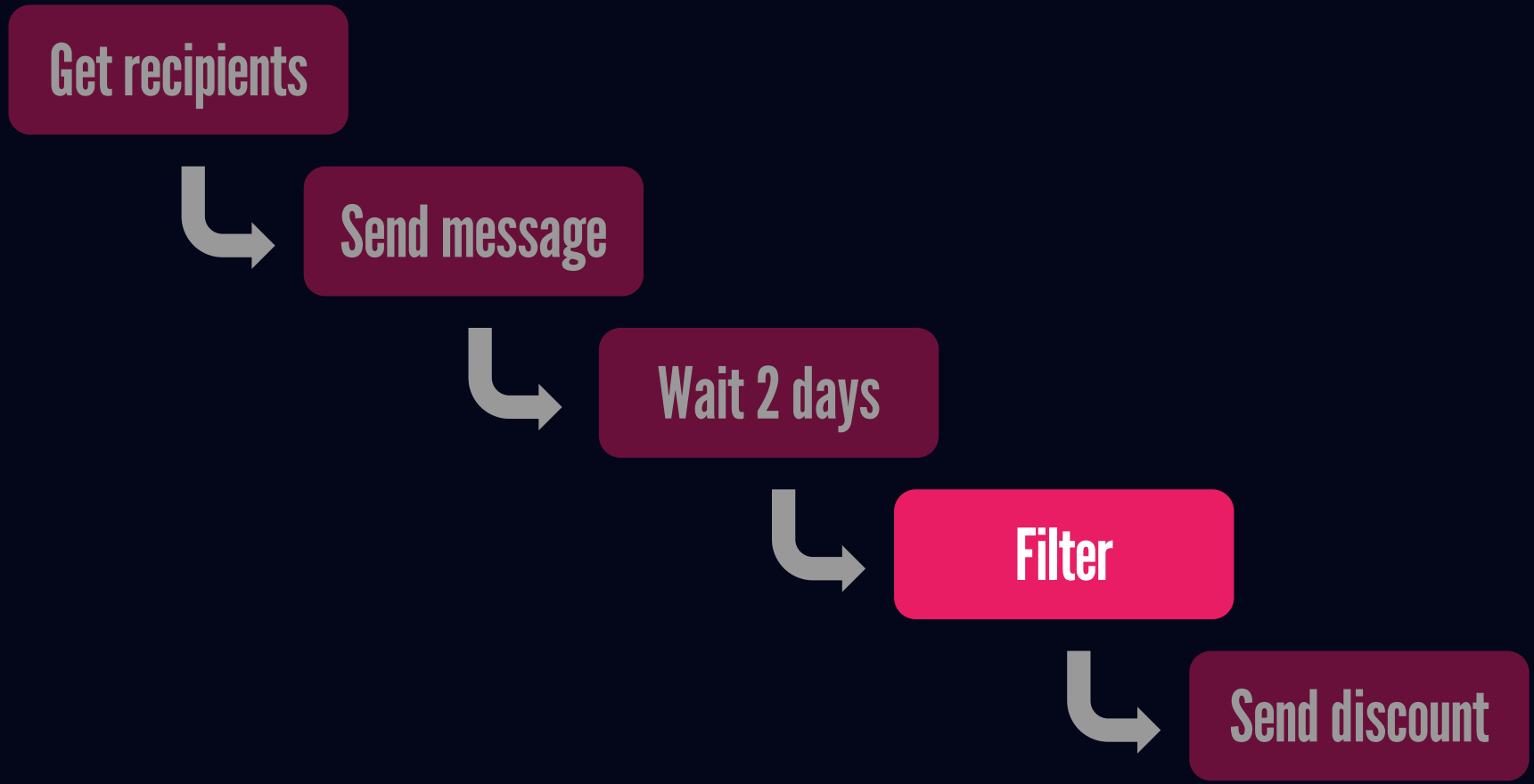
Ivett Ördög
ivettordog.com

BOB Konferenz
2025


ERROR


WORKED FINE IN DEV
OPS PROBLEM NOW
memegenerator.net


Get recipients
Send message
Wait 2 days
Filter
Send discount


Get recipients
Send message
Wait 2 days
Filter
Send discount

LET'S LEARN TO FALL


The Journey
Enlightenment
Digging deeper

The Journey

Check availability

Reserve the car

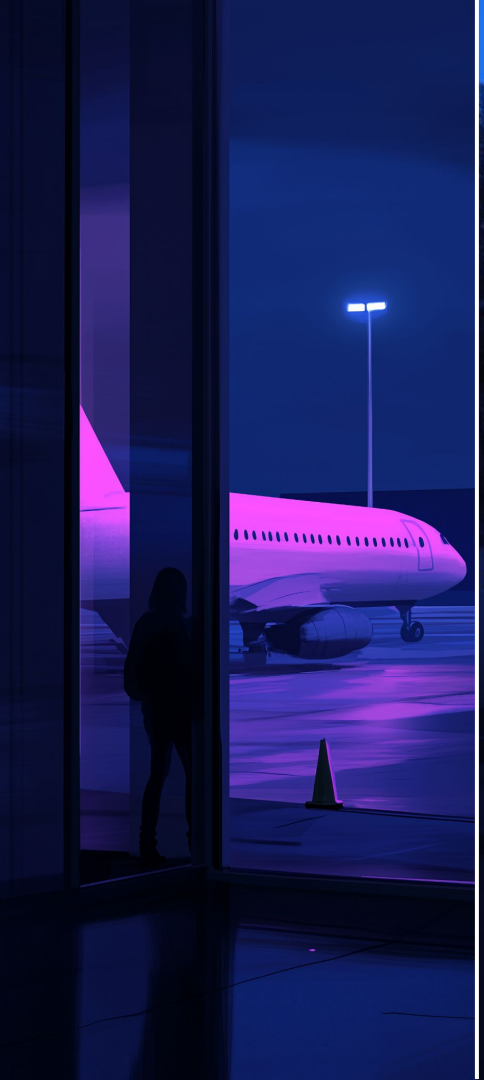
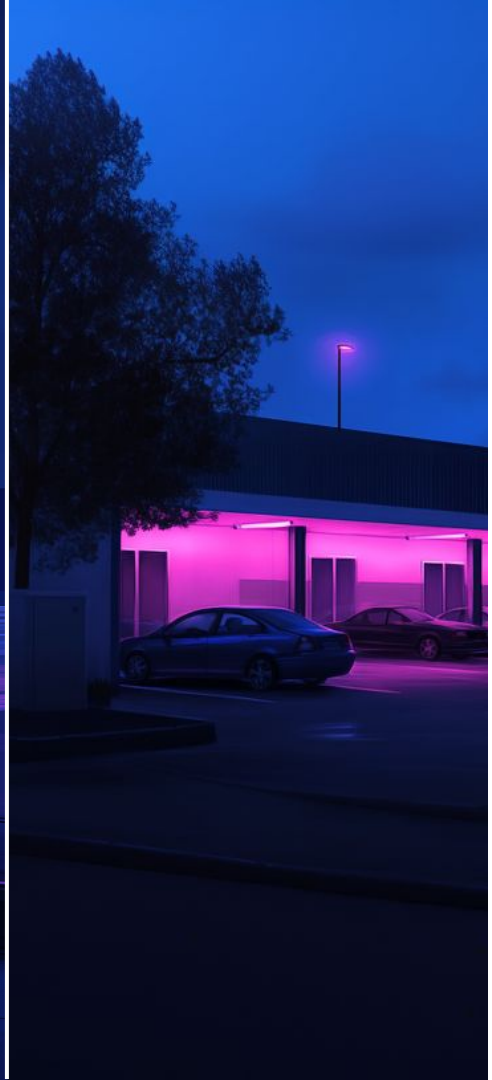
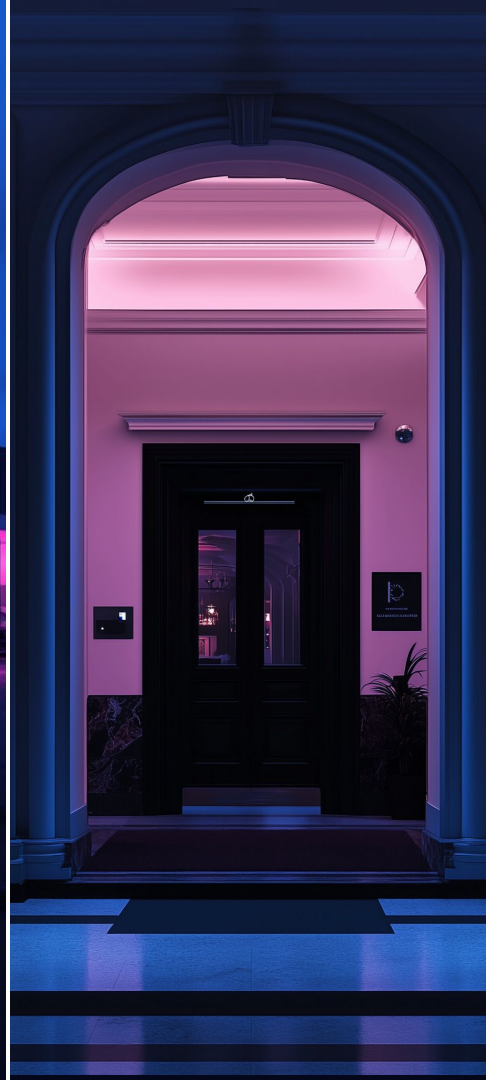
Assign the car

Check availability

Reserve the car

~~Assign the car~~

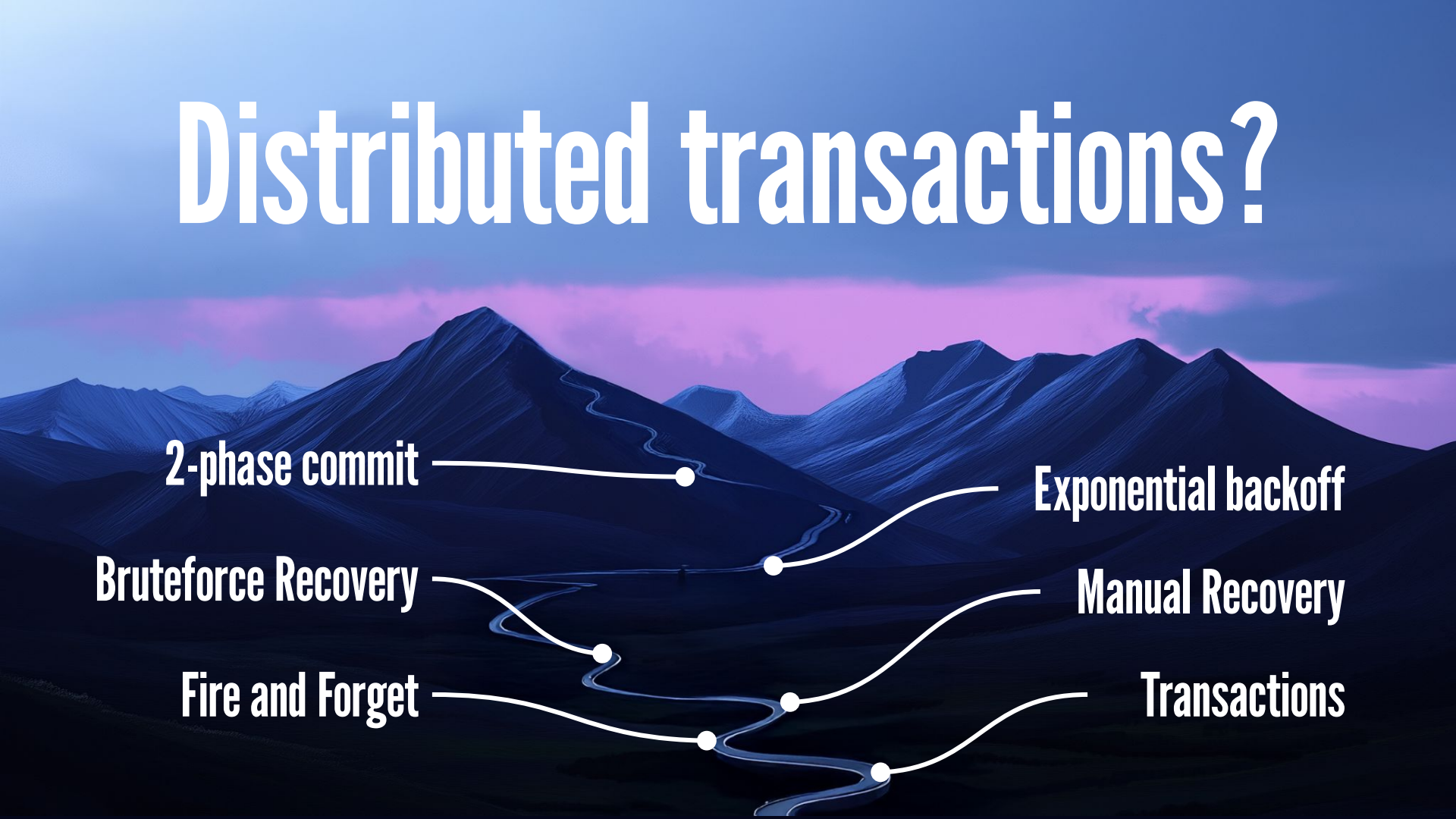

Transactions


#1 Long transactions

#2 No single database

Distributed transactions?



#1 Long transactions

#2 Single point of failure

Enlightenment

What if we didn't stick to ACID?

A

C

I

D

Atomicity

C

I

D

All or nothing transactions

Atomicity

Consistency

I

D

Always in valid state

Atomicity

Consistency

Isolation

D

Transactions not committed are “invisible”

Atomicity

Consistency

Isolation

Durability

Committed transactions written to disk

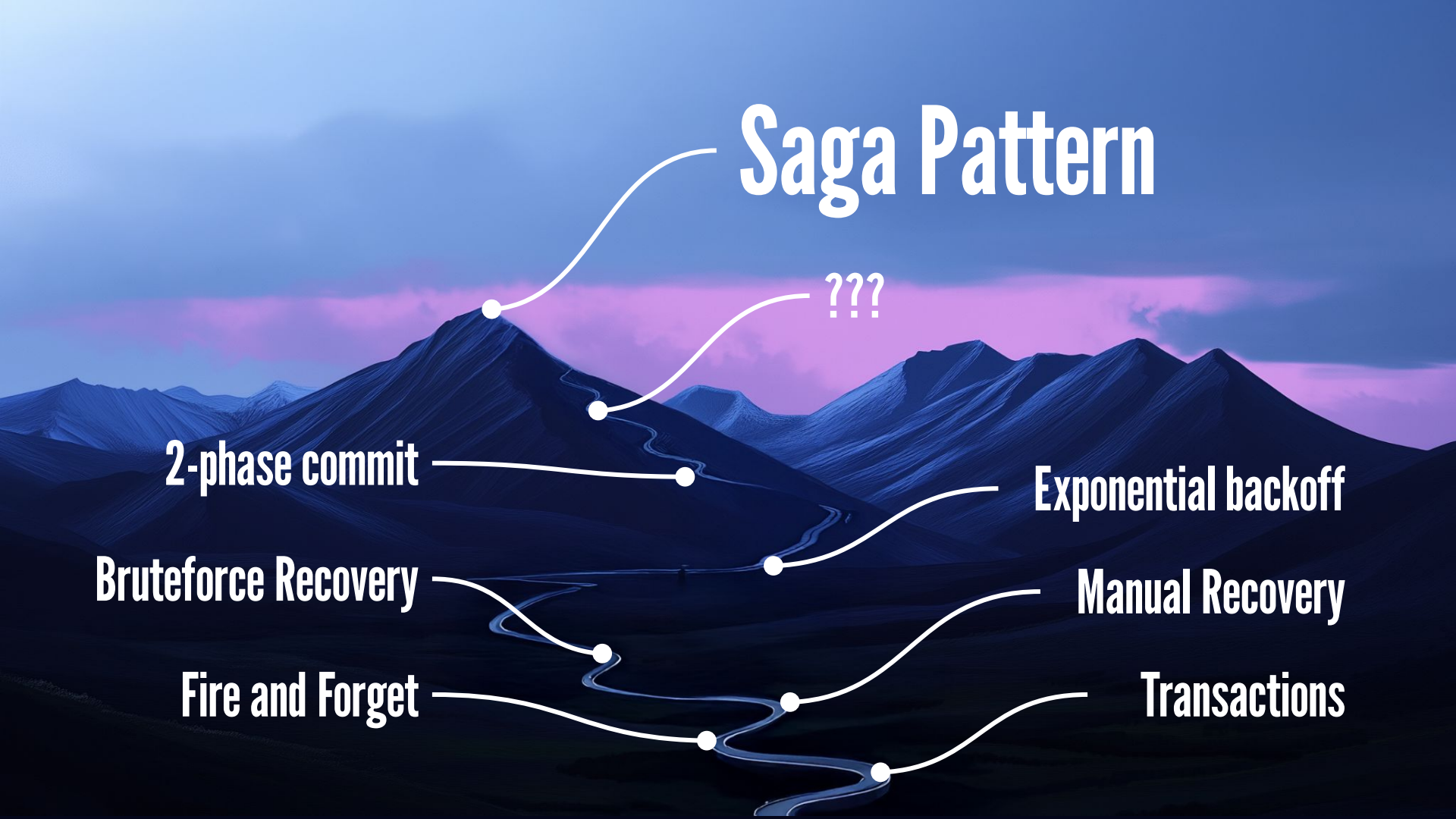
Atomicity

Consistency

Isolation

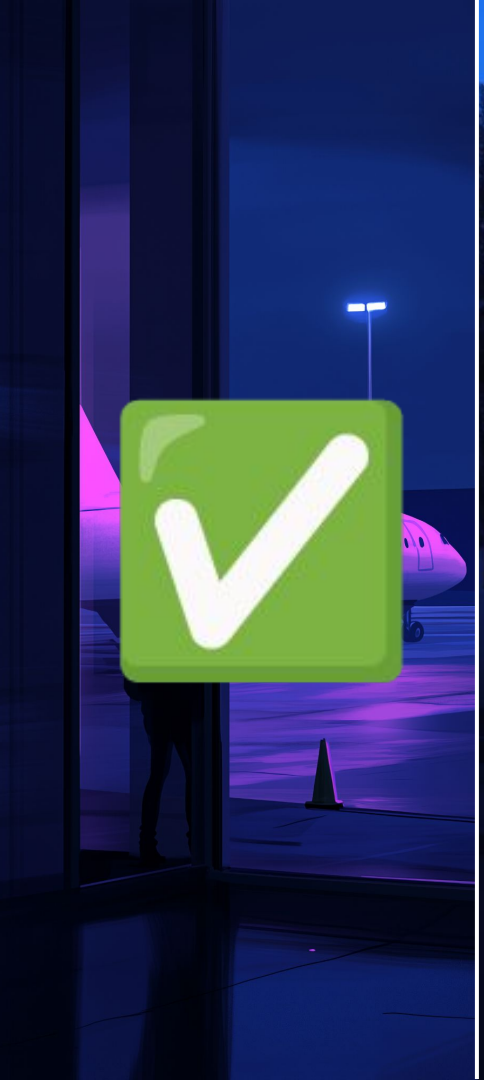
Durability

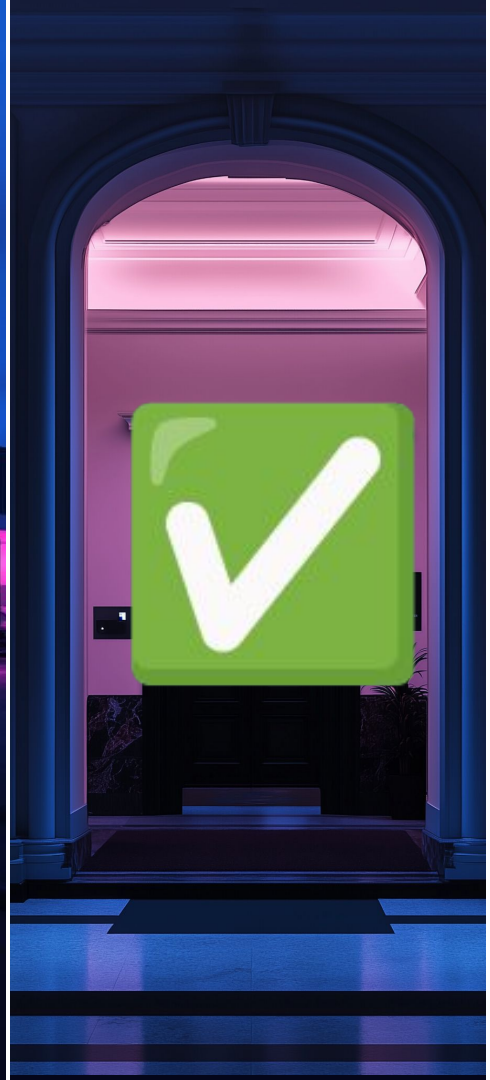
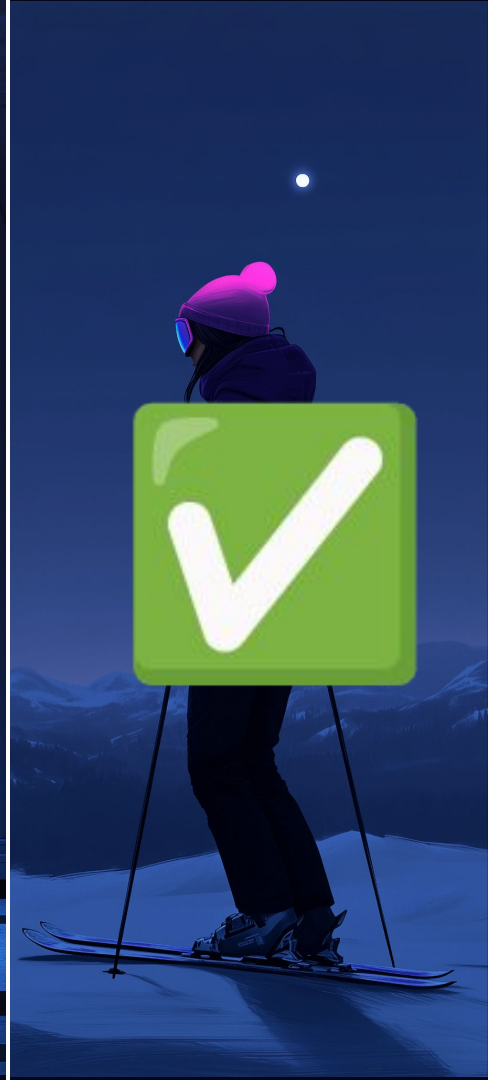
Which one are we sacrificing?


Saga Pattern
???
2-phase commit
Bruteforce Recovery
Fire and Forget
Exponential backoff
Manual Recovery
Transactions

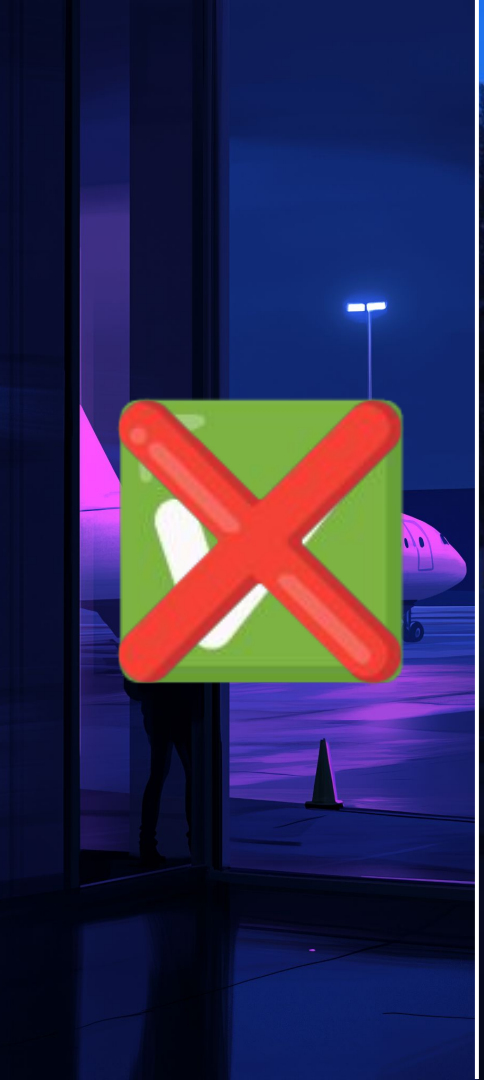
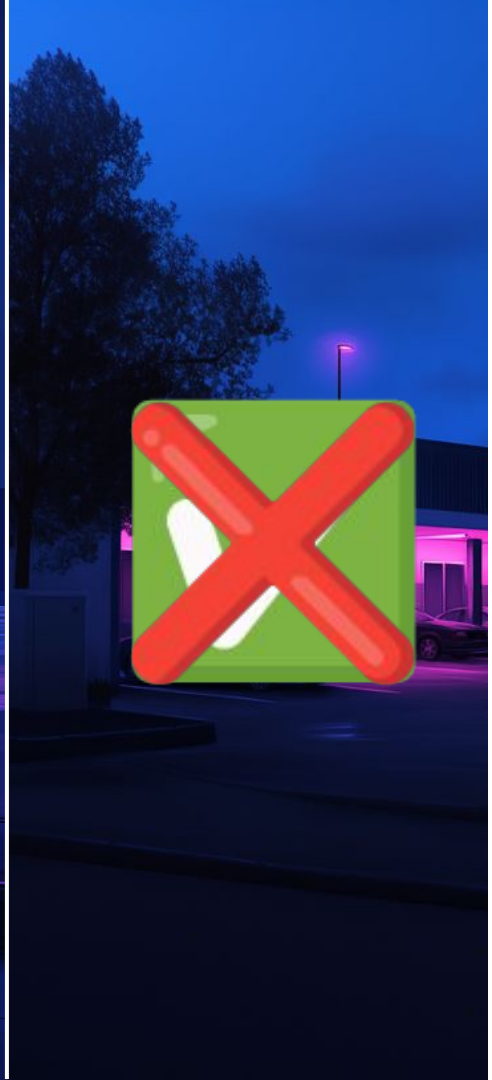
Goal: Break up large transactions

Let’s organise a ski trip!

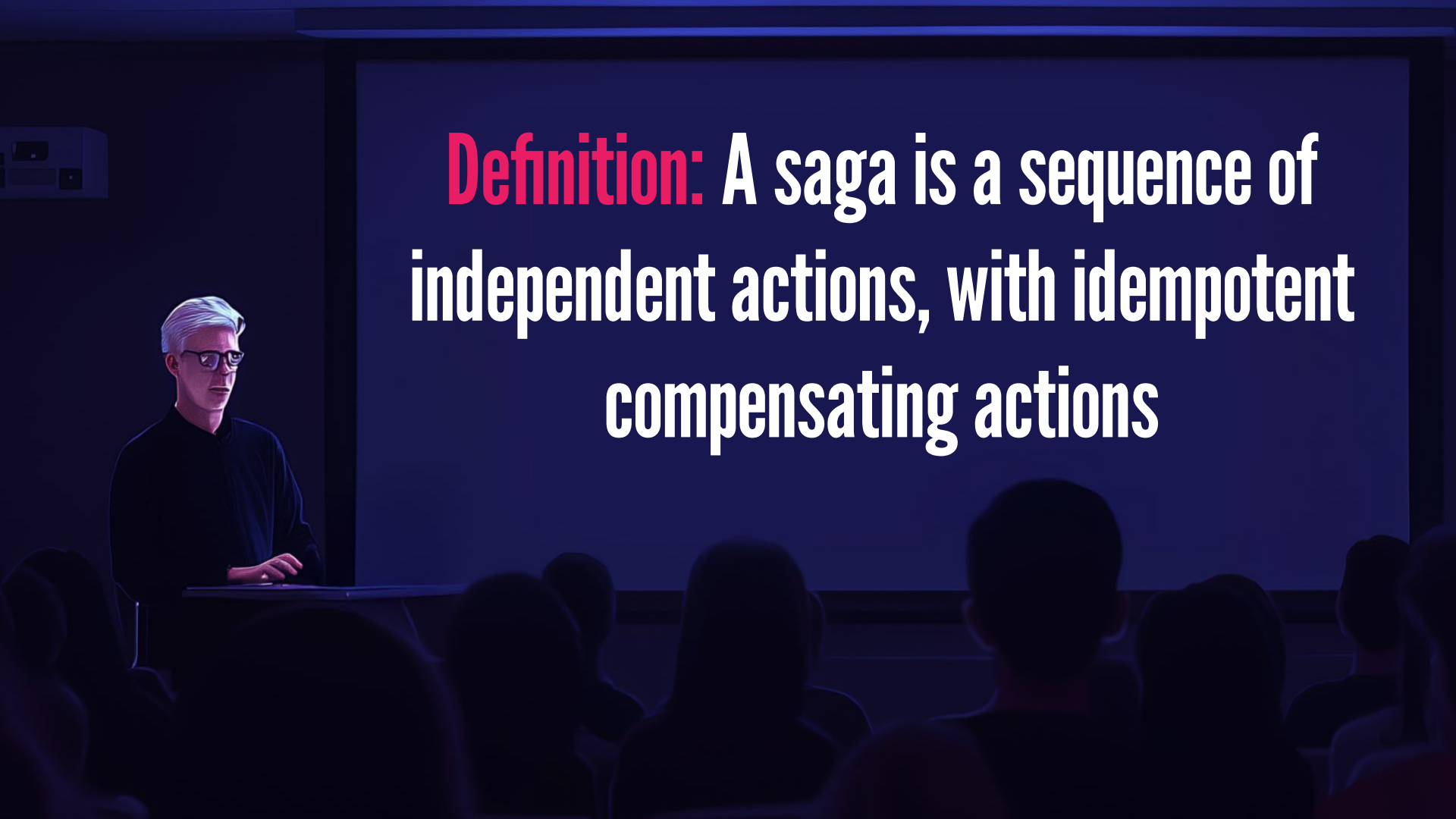


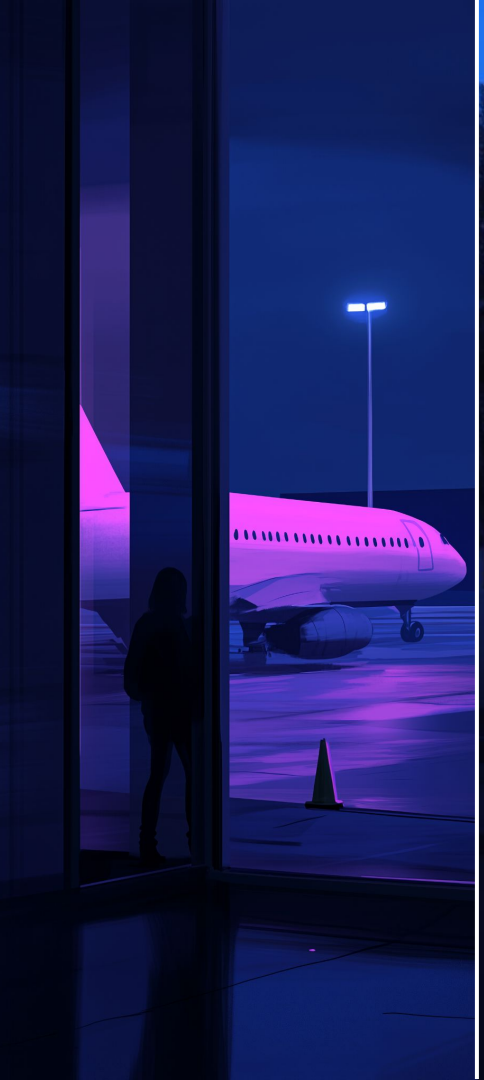
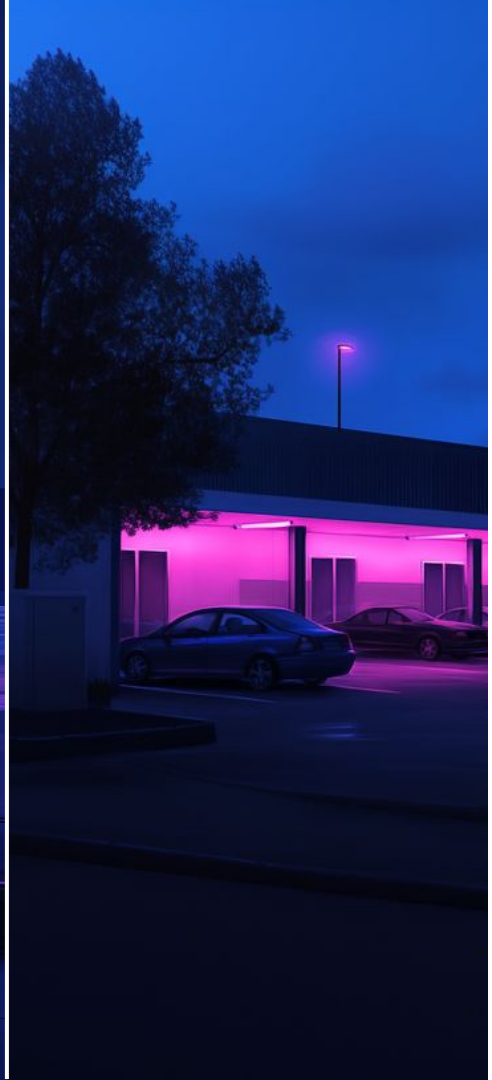
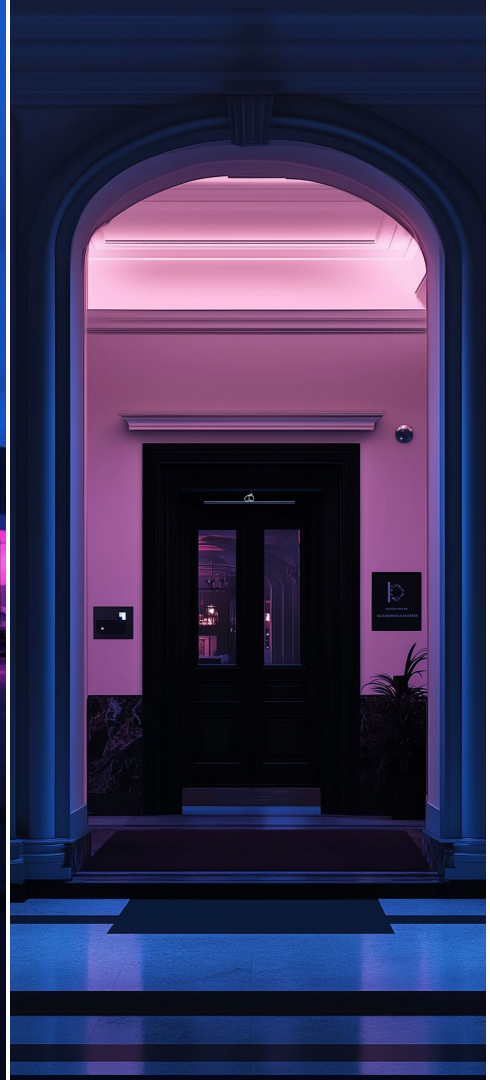
What if the hotel is full?

What is a Saga?


Definition: A saga is a sequence of independent actions, with idempotent compensating actions

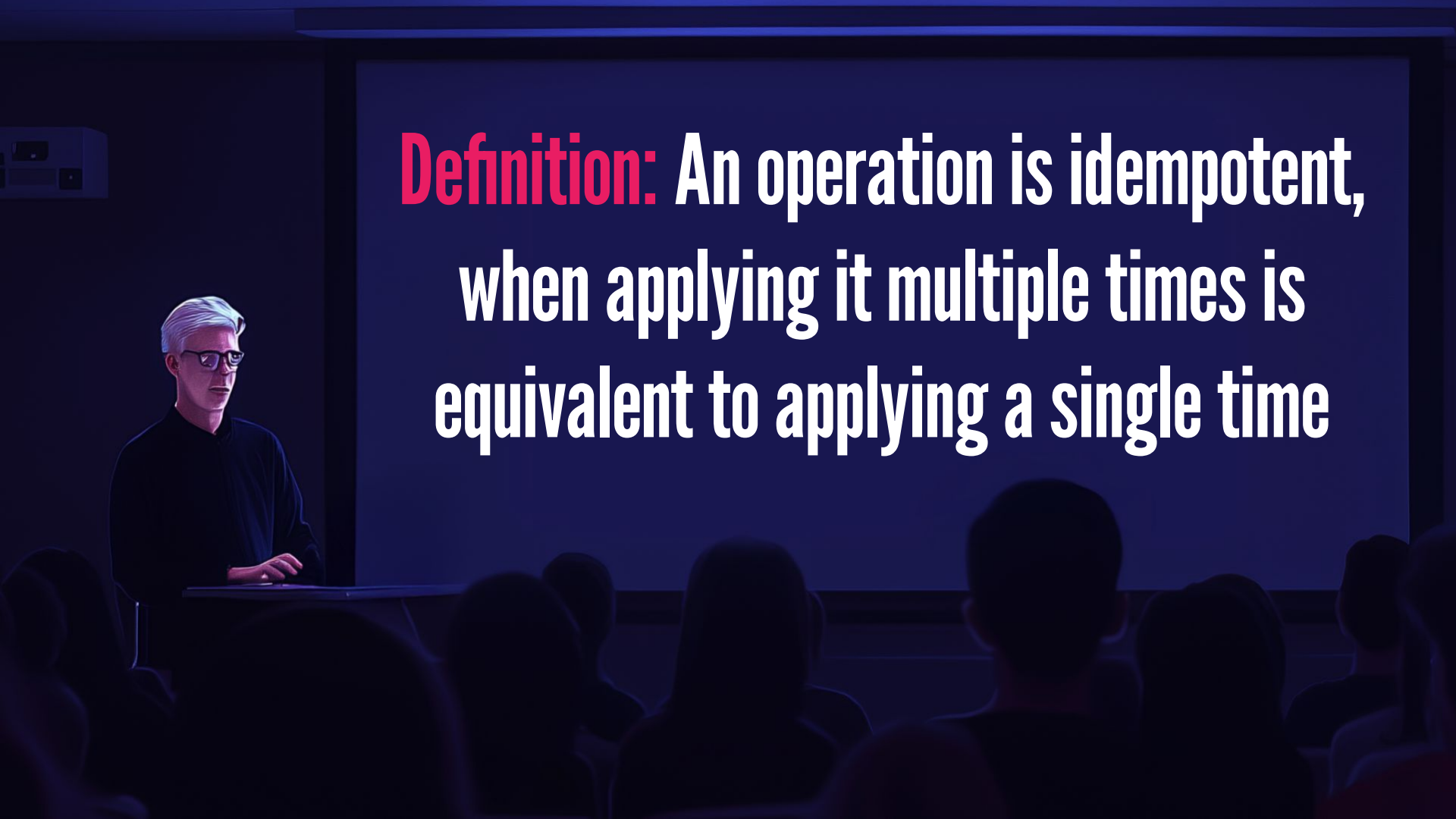
#1 Sequence of actions


#2 Compensating actions

Idempotent

Compensating actions


Definition: An operation is idempotent, when applying it multiple times is equivalent to applying a single time

Cancel plane ticket by ID


Order pizza

Eat the re  ng pizza

Reserve a hotel room

Reserve roor✅ for tonight

Cancel the la  reservation


Cancel reser ID I8P1ZZ4

What did we lose from ACID?

Atomicity

Consistency

Isolation

Durability

Atomicity

Consistency

Isolation

Durability

~~Atomicity~~

Consistency

Isolation

Durability

Entirely lost

~~Atomicity~~

Consistency

Isolation

Durability

~~Atomicity~~

Consistency*

Isolation

Durability

Eventual consistency

~~Atomicity~~

Consistency*

Isolation

Durability

~~Atomicity~~

Consistency*

~~Isolation~~

Durability

Entirely lost

~~Atomicity~~

Consistency*

~~Isolation~~

Durability

~~Atomicity~~

Consistency*

~~Isolation~~

Durability

Remaizns true

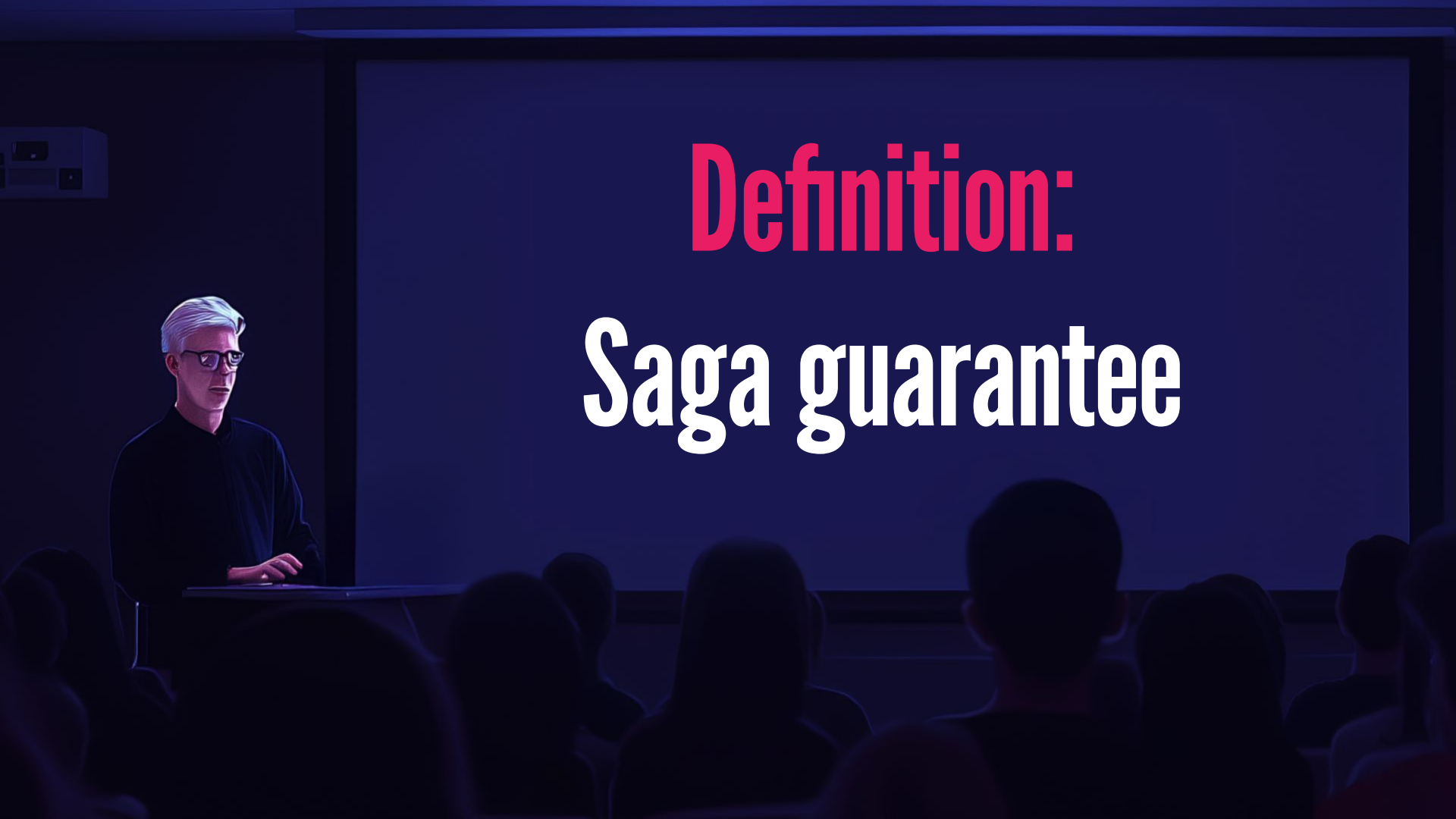
~~Atomicity~~

Consistency*

~~Isolation~~

Durability

We gained availability

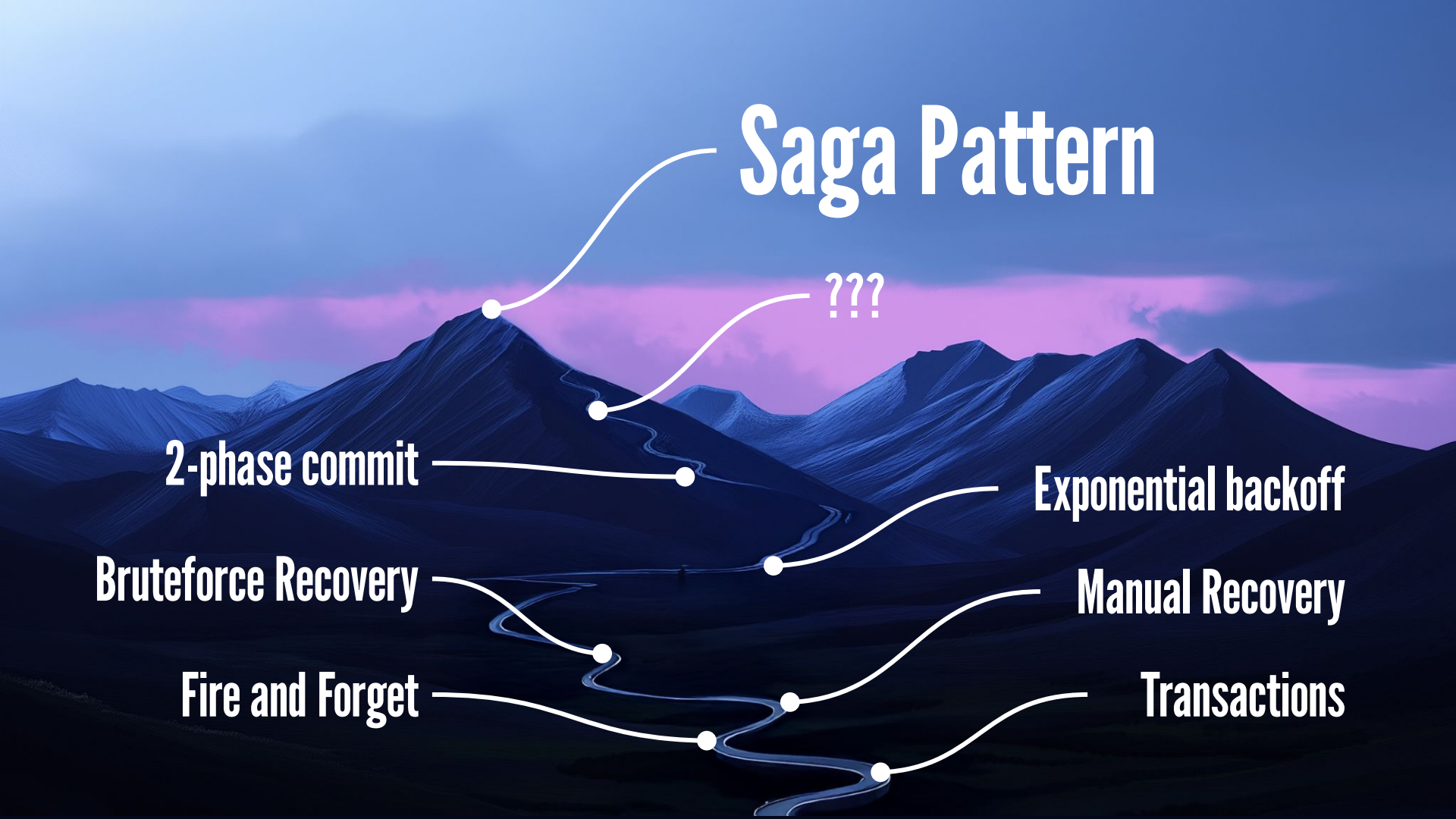

Definition:
Saga guarantee

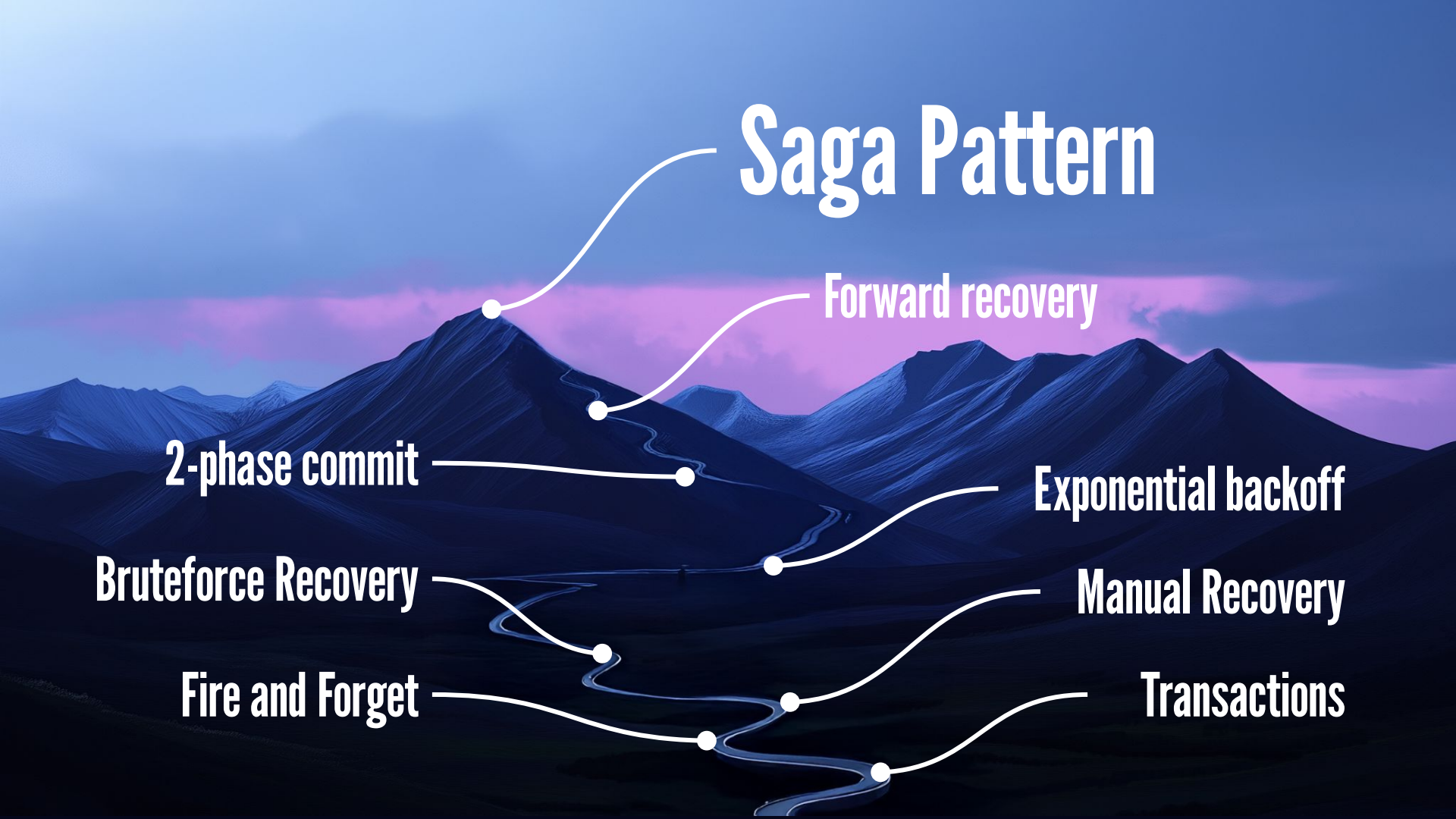
Either everything is successful

Or successful actions are compensated

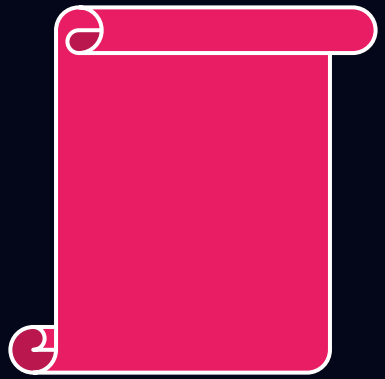
Or we are in the process of executing the actions

What if an action can’t be compensated?

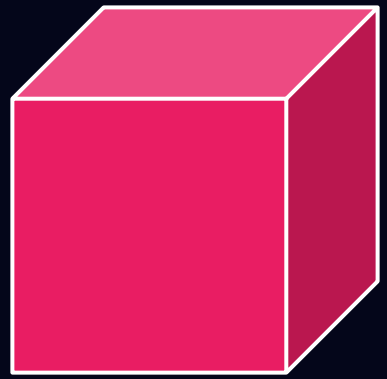

Saga Pattern
???
2-phase commit
Bruteforce Recovery
Fire and Forget
Exponential backoff
Manual Recovery
Transactions


Saga Pattern
Forward recovery
2-phase commit
Exponential backoff
Bruteforce Recovery
Manual Recovery
Fire and Forget
Transactions

Digging deeper

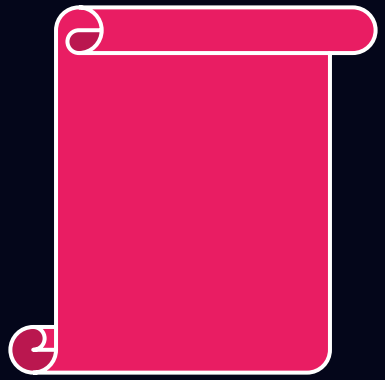


Saga Log



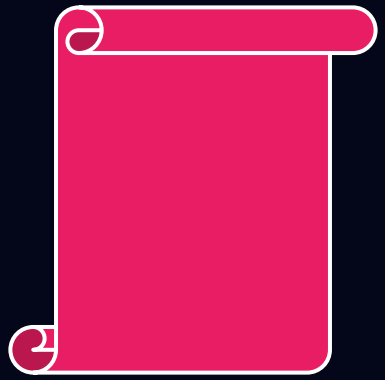
S.E.C.

Saga Execution Coordinator



Saga Log

#1 Durable record of independent actions

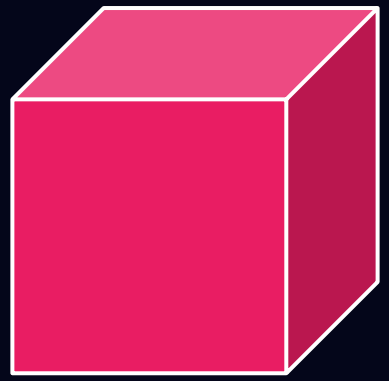


Saga Log

#2

Single source with fall back

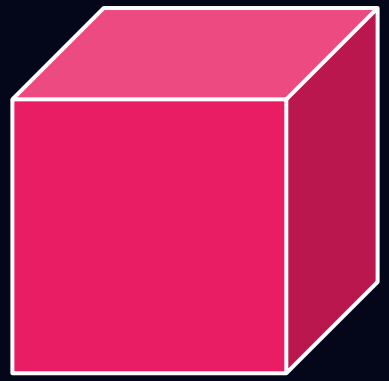
(Sharded DB)



S.E.C.

Saga Execution Coordinator

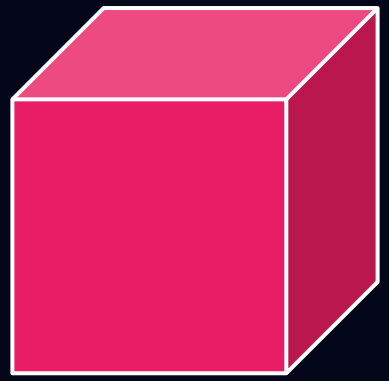
#1 Stateless & fungible process



S.E.C.

Saga Execution Coordinator

#2 Processes the Saga Logs



S.E.C.

Saga Execution Coordinator

#3
May fail and restart at any time

Happy path

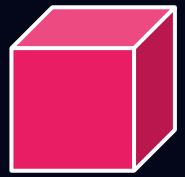
Task list:

#1 Buy plane ticket

#2 Reserve car

#3 Reserve hotel

#4 Buy ski pass



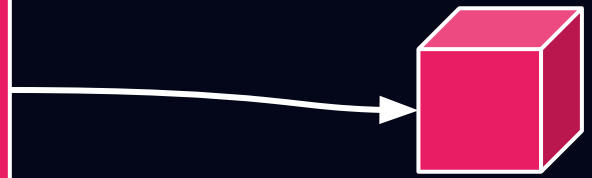
Task list:

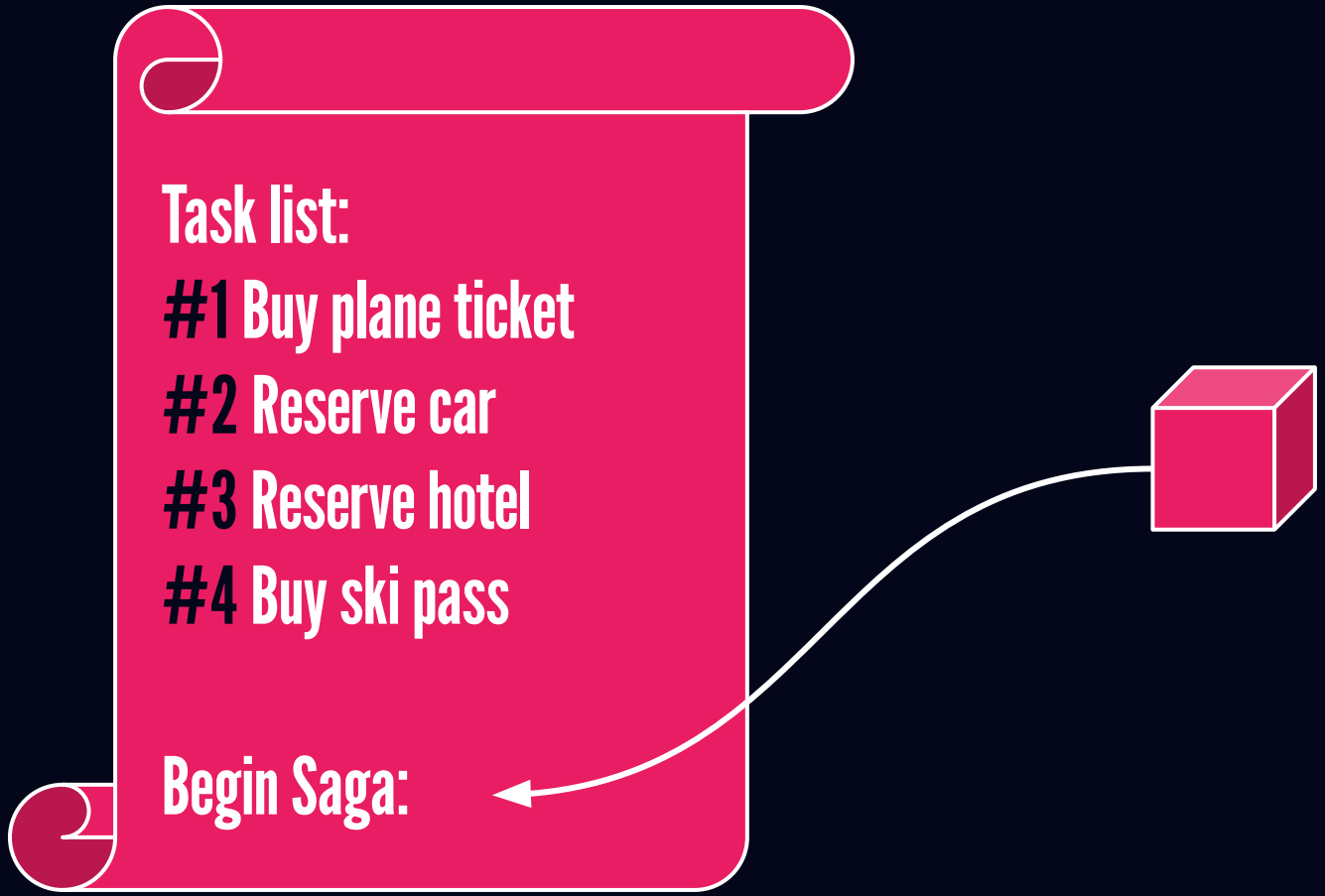
#1 Buy plane ticket

#2 Reserve car

#3 Reserve hotel

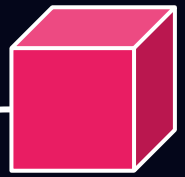
#4 Buy ski pass




Task list:
#1 Buy plane ticket
#2 Reserve car
#3 Reserve hotel
#4 Buy ski pass
Begin Saga:

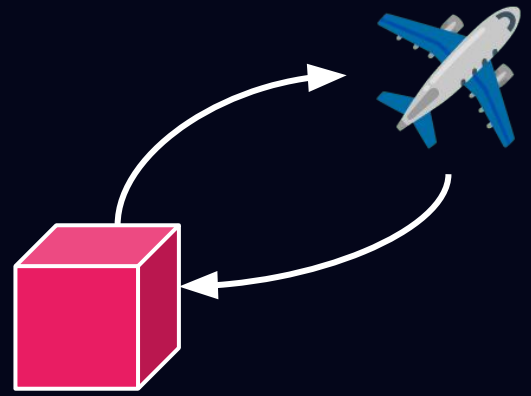
#1 Buy plane ticket
#2 Reserve car
#3 Reserve hotel
#4 Buy ski pass

Begin Saga:
Begin #1



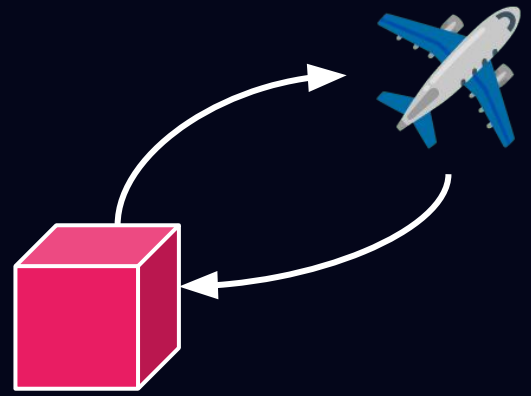
#1 Buy plane ticket
#2 Reserve car
#3 Reserve hotel
#4 Buy ski pass

Begin Saga:
Begin #1



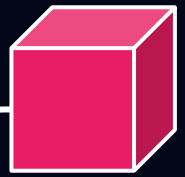
#1 Buy plane ticket
#2 Reserve car
#3 Reserve hotel
#4 Buy ski pass

Begin Saga:
Begin #1



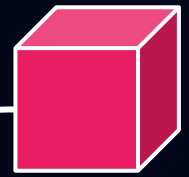
#2 Reserve car
#3 Reserve hotel
#4 Buy ski pass

Begin Saga:
Begin #1
Done #1



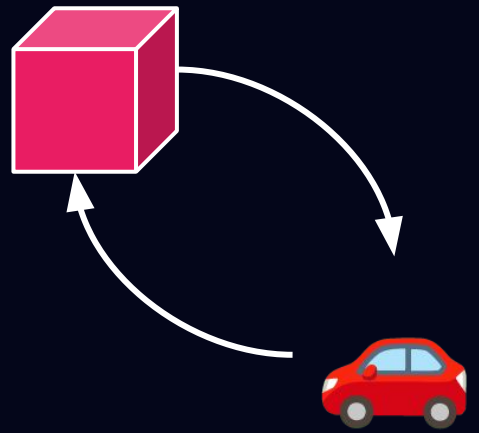
#3 Reserve hotel
#4 Buy ski pass

Begin Saga:
Begin #1
Done #1
Begin #2



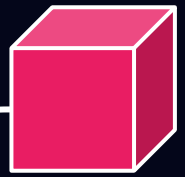
#3 Reserve hotel
#4 Buy ski pass

Begin Saga:
Begin #1
Done #1
Begin #2

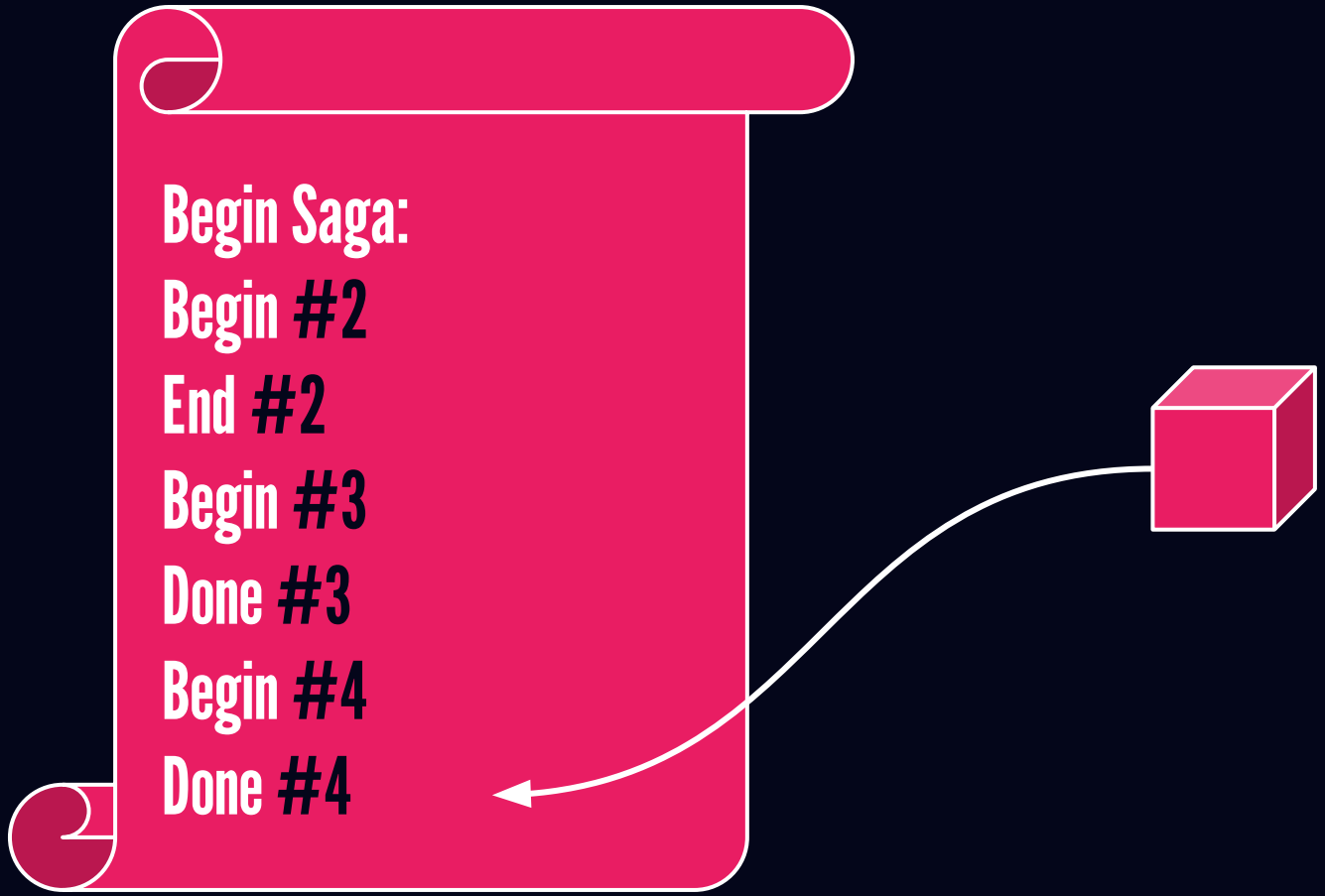


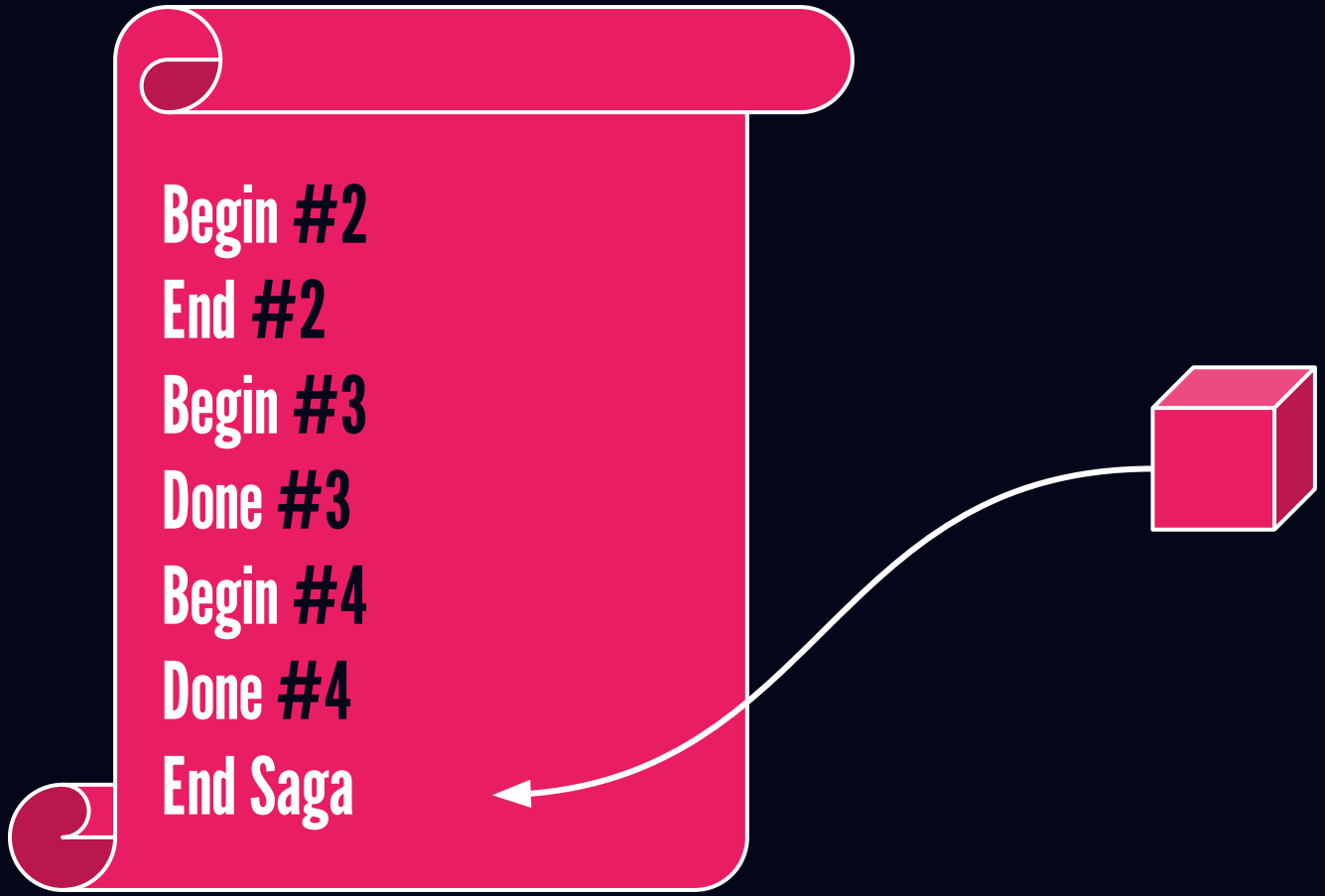
#4 Buy ski pass

Begin Saga:
Begin #1
Done #1
Begin #2
End #2



And so on...

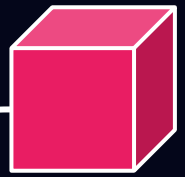

Begin Saga:
Begin #2
End #2
Begin #3
Done #3
Begin #4
Done #4


Begin #2
End #2
Begin #3
Done #3
Begin #4
Done #4
End Saga

Failure case

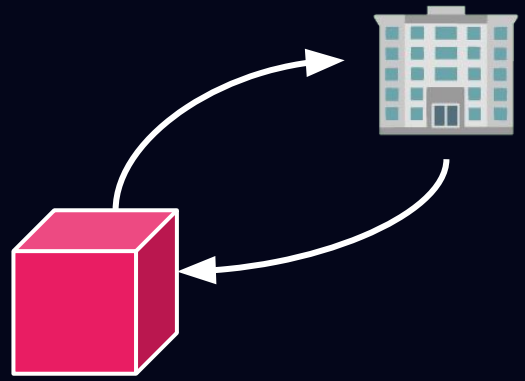
#4 Buy ski pass

Begin Saga:
Begin #1
Done #1
Begin #2
End #2



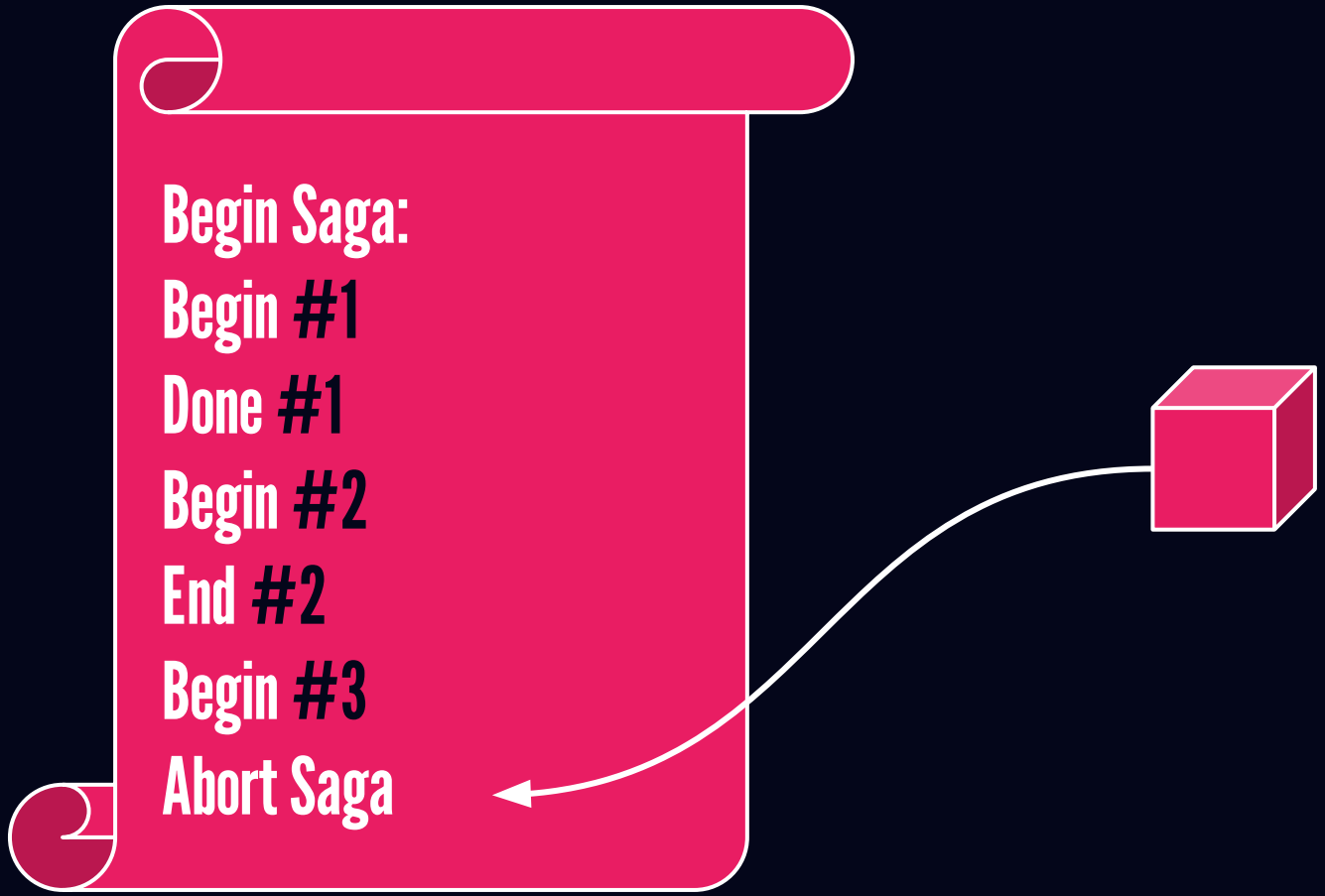

Begin Saga:
Begin #1
Done #1
Begin #2
End #2
Begin #3

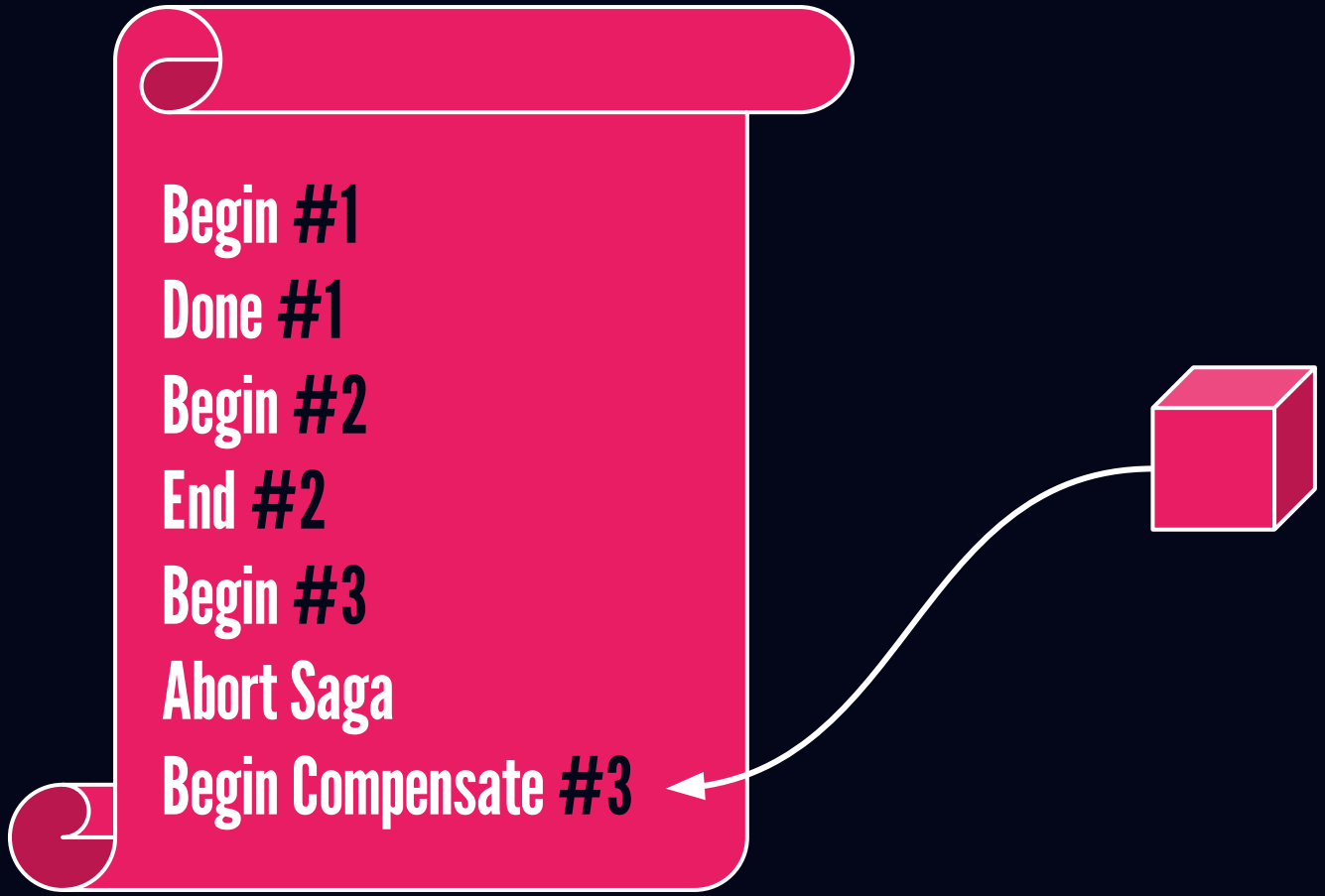
Begin Saga:
Begin #1
Done #1
Begin #2
End #2
Begin #3



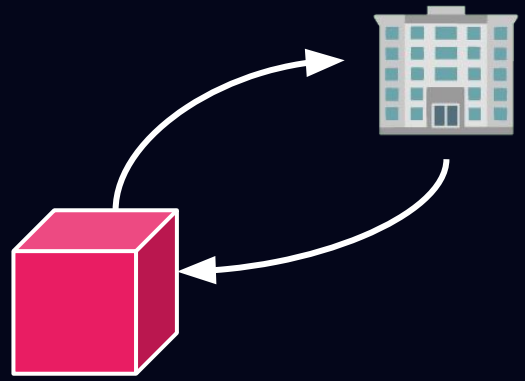
Begin Saga:
Begin #1
Done #1
Begin #2
End #2
Begin #3


Begin Saga:
Begin #1
Done #1
Begin #2
End #2
Begin #3

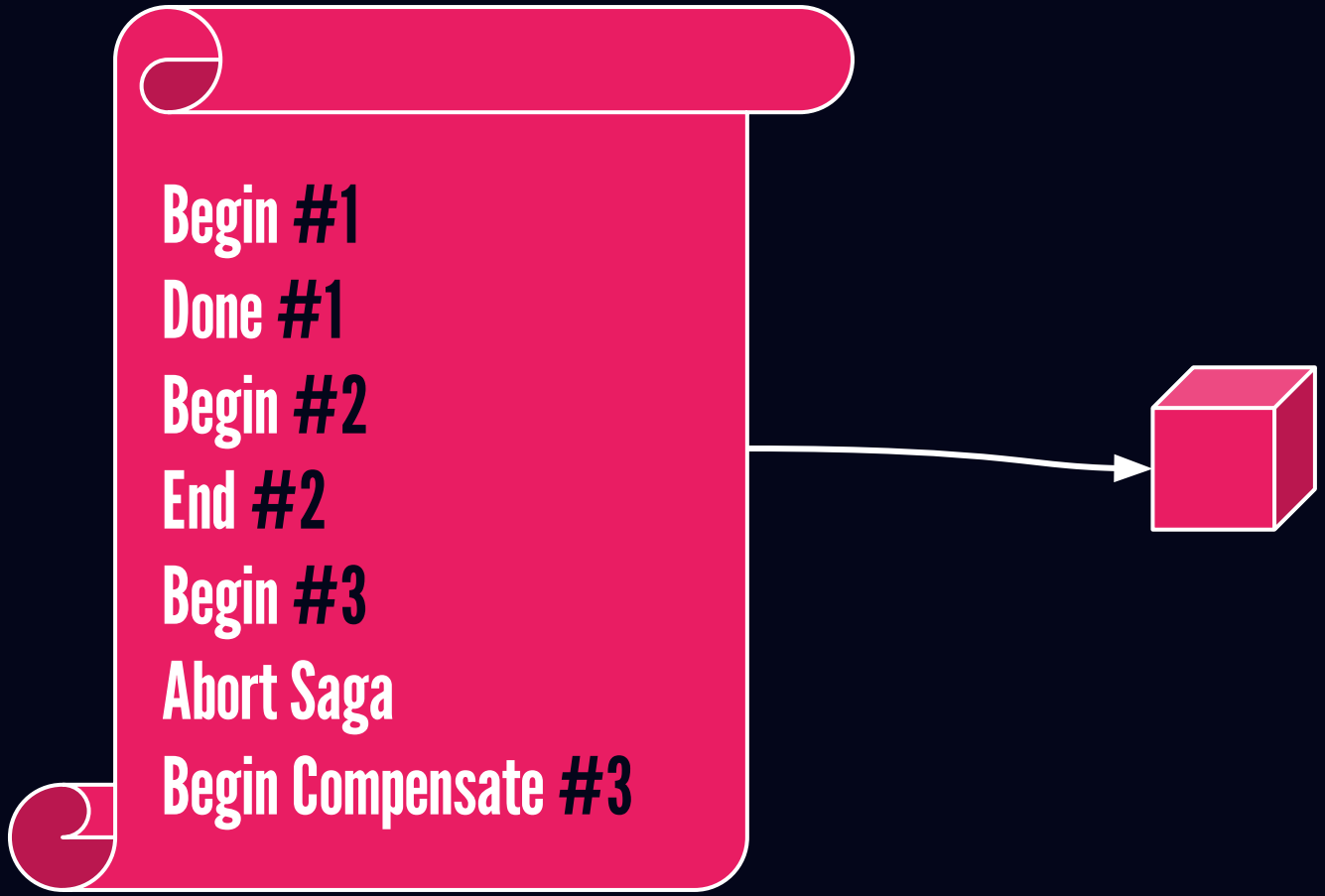

Begin Saga:
Begin #1
Done #1
Begin #2
End #2
Begin #3
Abort Saga


Begin #1
Done #1
Begin #2
End #2
Begin #3
Abort Saga
Begin Compensate #3

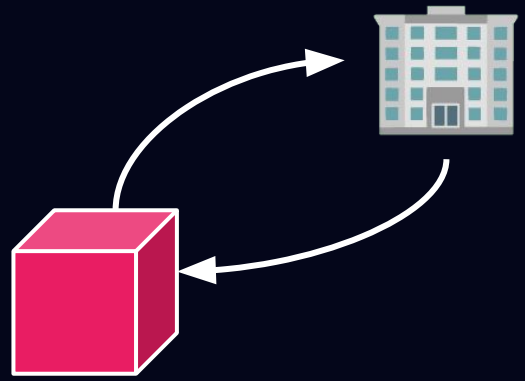
Begin #1
Done #1
Begin #2
End #2
Begin #3
Abort Saga
Begin Compensate #3

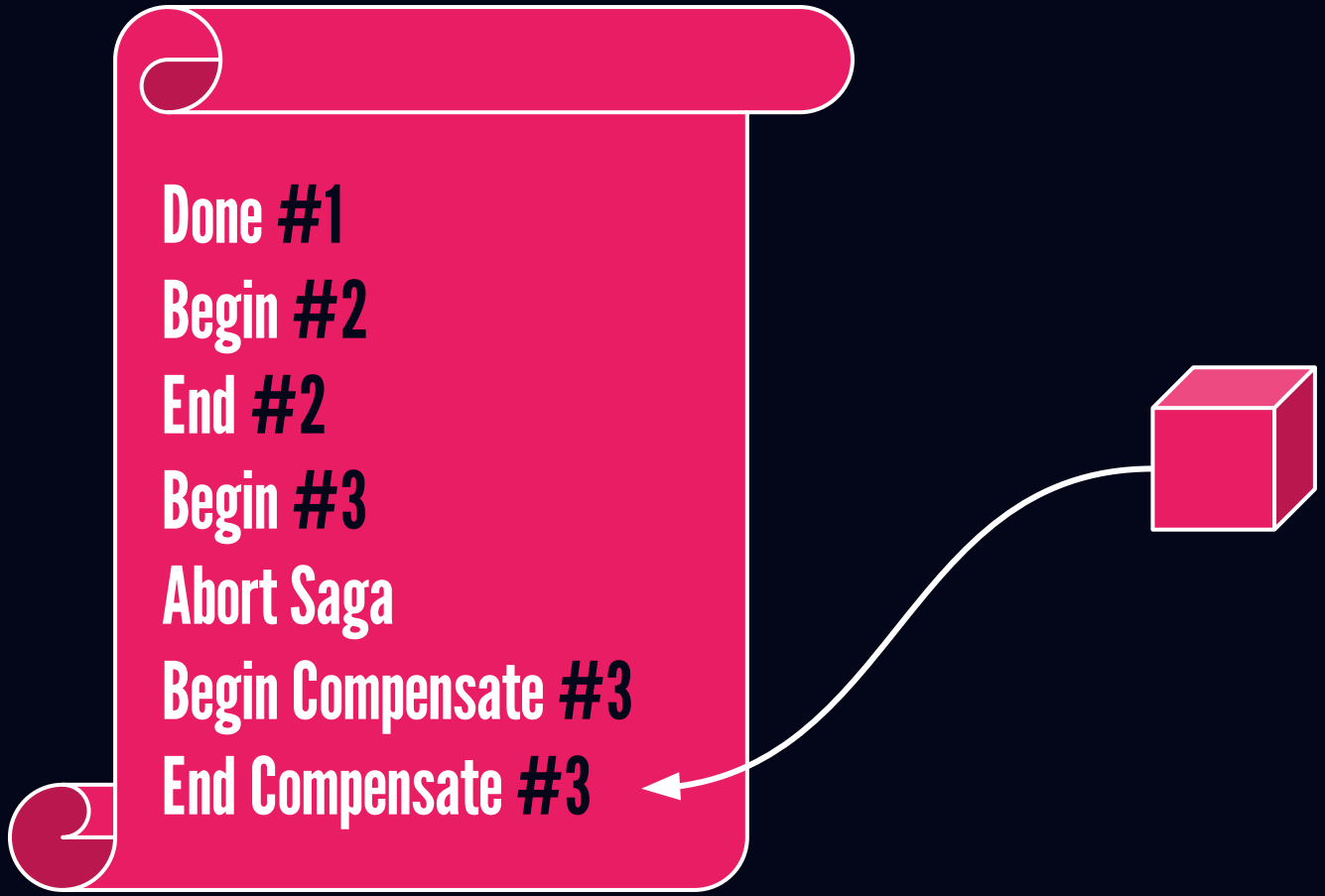


Begin #1
Done #1
Begin #2
End #2
Begin #3
Abort Saga
Begin Compensate #3


Begin #1
Done #1
Begin #2
End #2
Begin #3
Abort Saga
Begin Compensate #3

Begin #1
Done #1
Begin #2
End #2
Begin #3
Abort Saga
Begin Compensate #3




Done #1
Begin #2
End #2
Begin #3
Abort Saga
Begin Compensate #3
End Compensate #3

And so on...

When should we apply compensating actions?

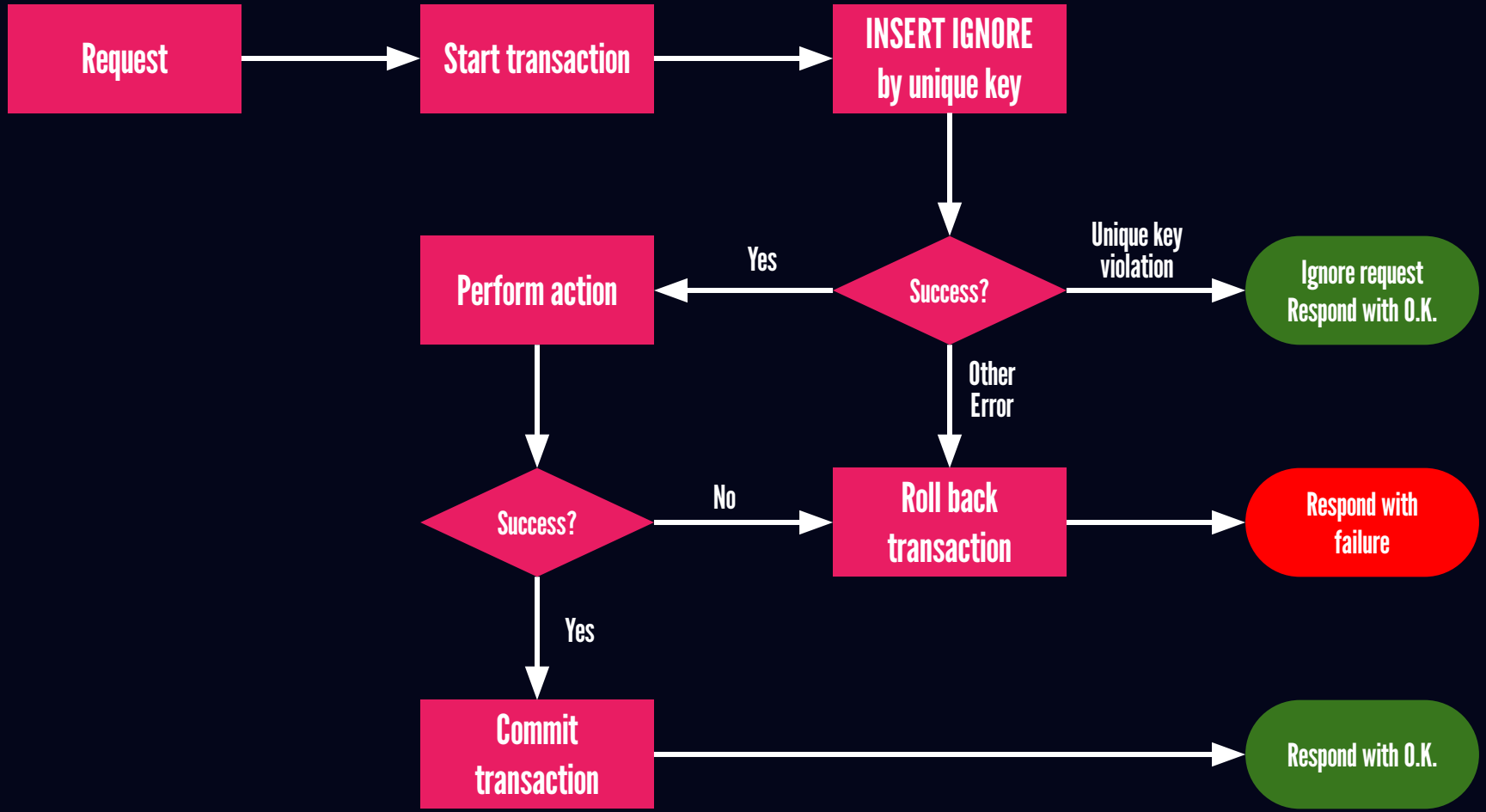
#1 Saga log ends with “Begin”

#2 Saga log contains “Abort”

What if an action isn't idempotent?

We can use transactions and unique action IDs

Action ID registry


Request
Start transaction
INSERT IGNORE by unique key
Success?
Yes
Perform action
Unique key violation
Ignore request Respond with O.K.
Other Error
Success?
No
Roll back transaction
Respond with failure
Yes
Commit transaction
Respond with O.K.

Conclusion

#1 Use Saga for long running distributed processes

#2 Make all API endpoints idempotent

NEXT INCREMENT

LEANPOKER.

IVETTORDOG.COM