Harnessing the Durable Execution Pattern:
From Failures to Fault-Tolerance

Alberto Acerbis
Software Architect @Intré S.r.L.

Microsoft® Most Valuable Professional


FAILURE
SUCCCES

Distributed System Happy Path

Sales
Create Order

→

Warehouse
Prepare Order

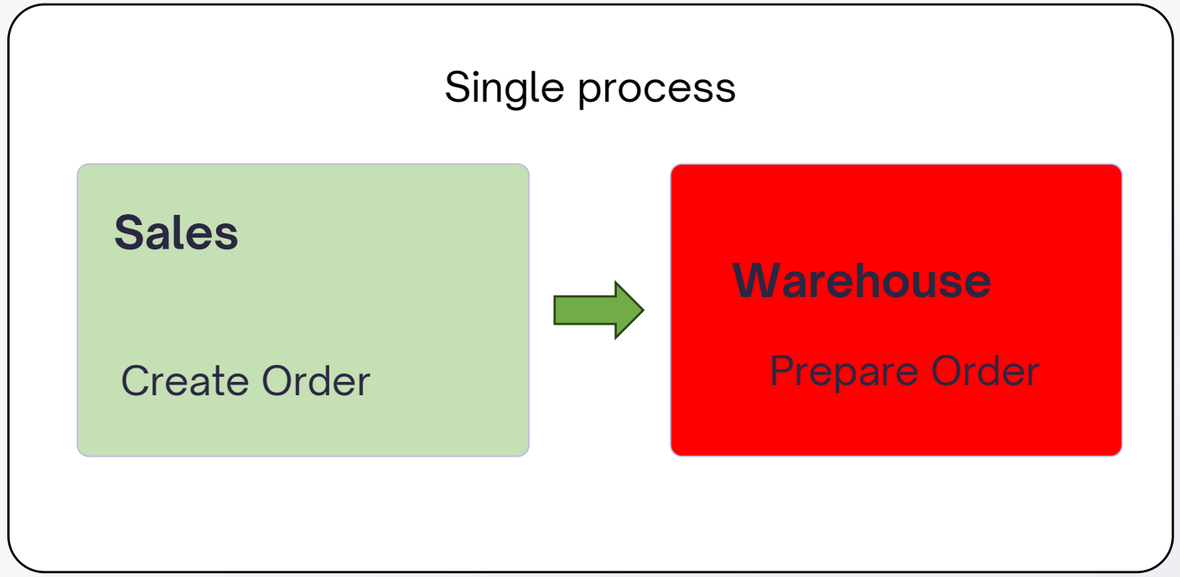
→

Payments
Charge Card

→

Shipment
Send Order

Distributed System Failure



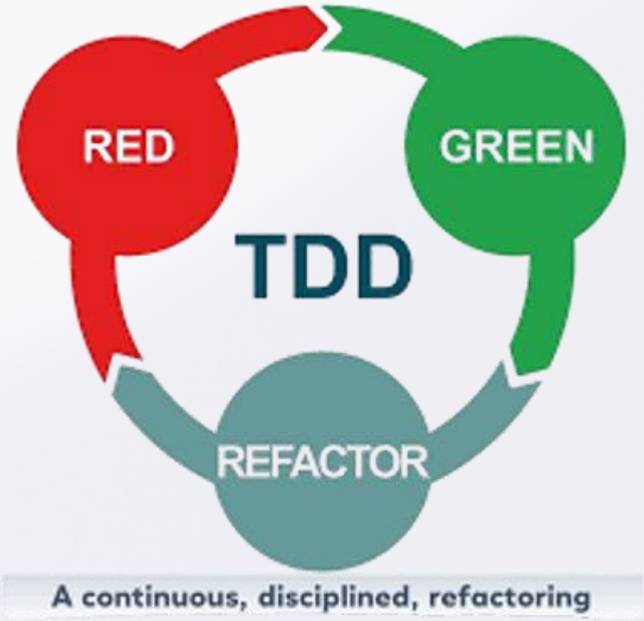

SYSTEM FAILURE

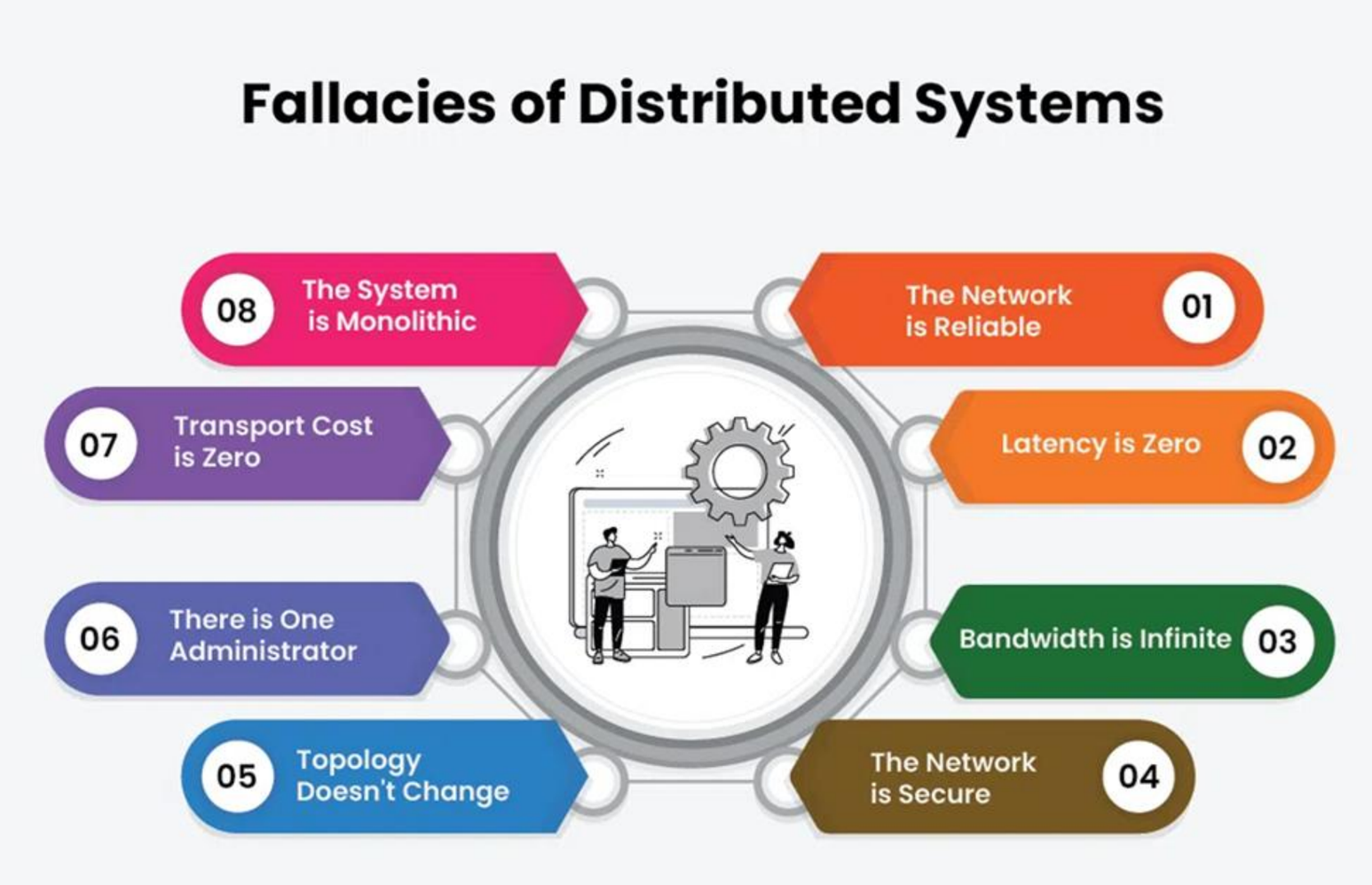
INTR3

HARNESSING THE DURABLE EXECUTION PATTERN

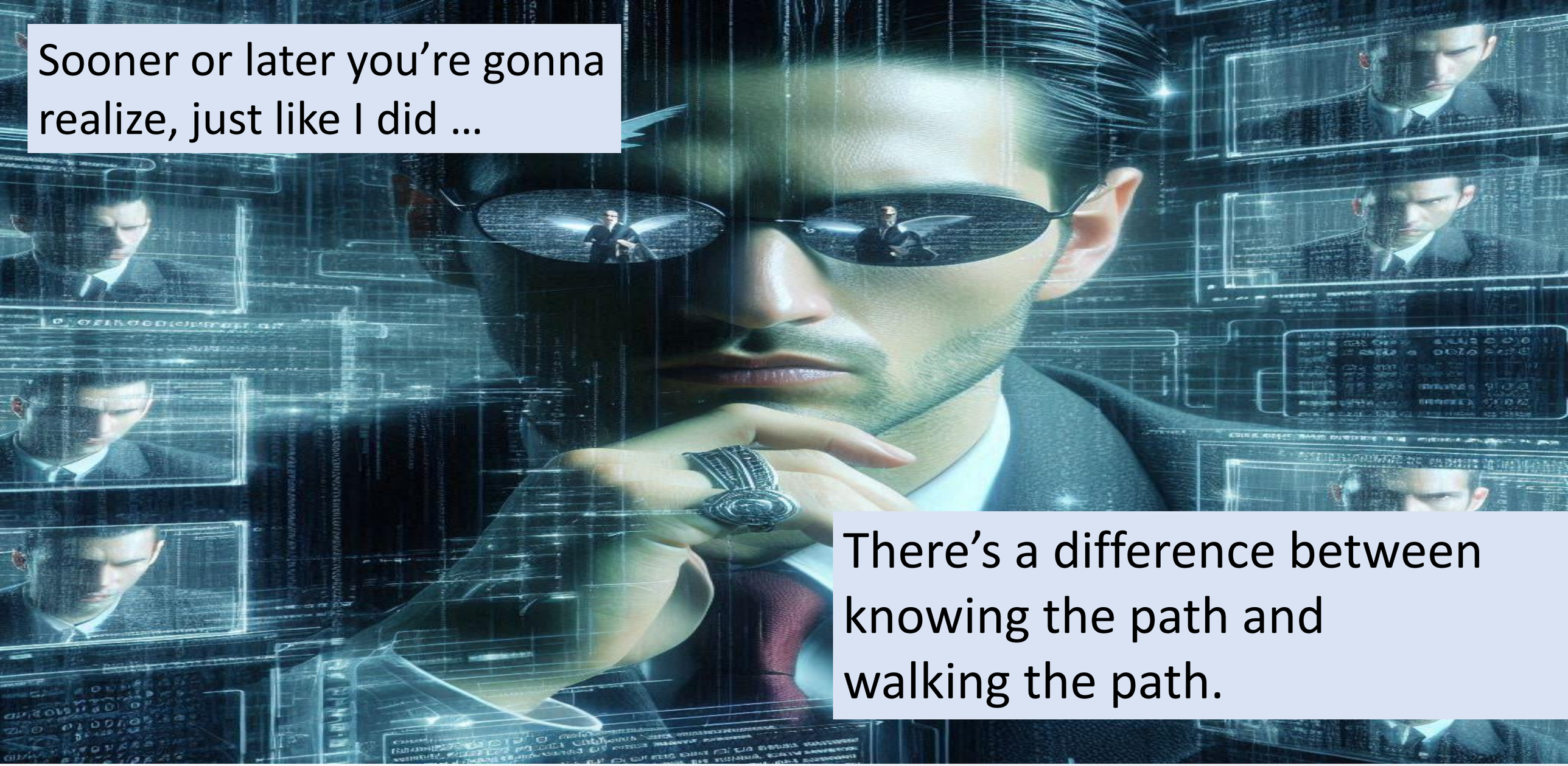
What do you do in this case?

- Examine all the components of the system
- Fix some single points of failure
- Run tests
- Despite this … failures Occur





https://www.geeksforgeeks.org/fallacies-of-distributed-systems/



Sooner or later you're gonna realize, just like I did …

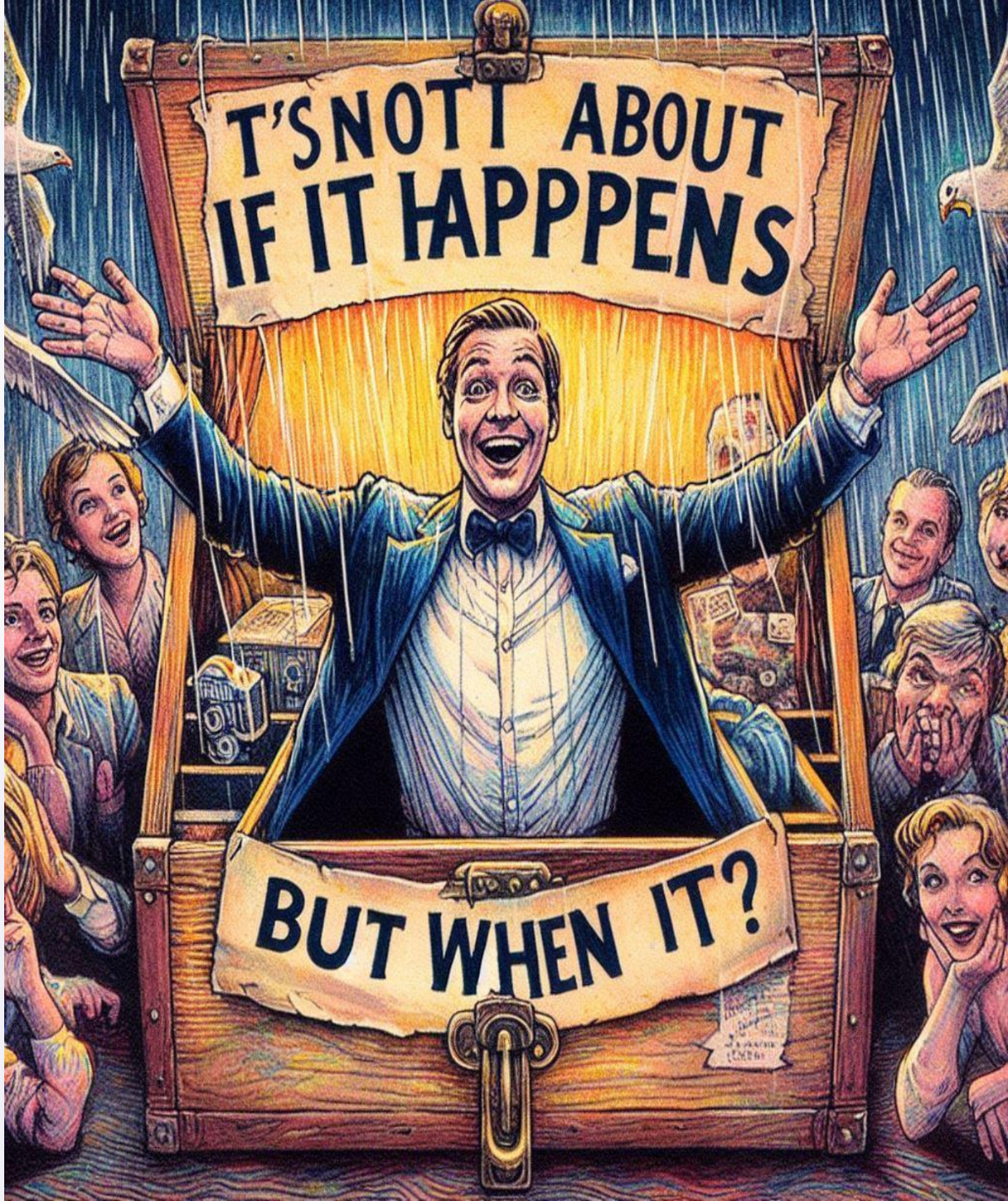
There's a difference between knowing the path and walking the path.

It's not about **if** it happens, but **when** it happens.

In today's distributed computing landscape, **failures are expected**, not anomalies.

Failure is inevitable.

The challenge is **designing systems that handle these inevitabilities** gracefully.



INTR3


Unfortunately, no one truly understand that failure is inevitable.
Until they see a system fail.

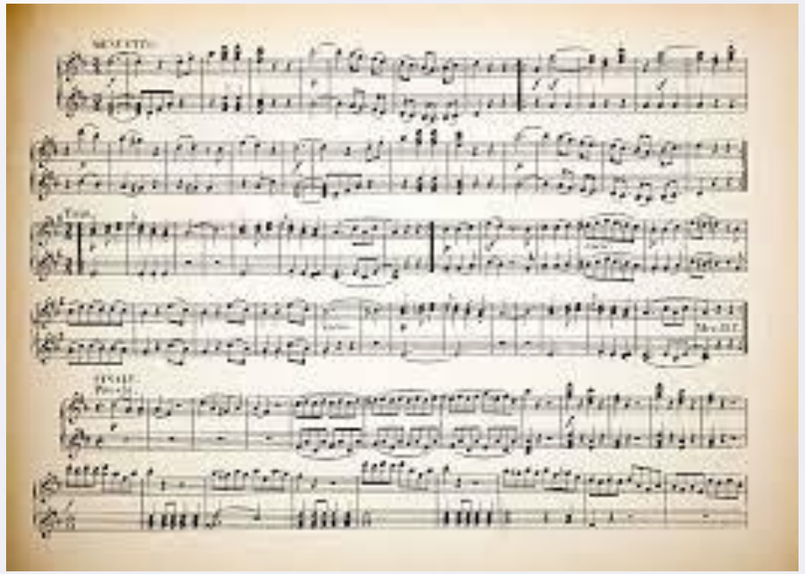
The real challenge is designing for that moment.

Software is like a music sheet

```
public async Task<T> GetByIdAsync(string id, CancellationToken cancellationToken)
{
    var collection = Database.GetCollection<T>(typeof(T).Name);
    var filter = Builders<T>.Filter.Eq("_id", id);
    return (await collection.CountDocumentsAsync(filter) > 0 ? (await collection.FindAsync(filter)).First() : null)!;
}

public async Task<PagedResult<T>> GetByFilterAsync(Expression<Func<T, bool>>? query, int page, int pageSize, Cancellat
{
    if (--page < 0)
        page = 0;

    var collection = Database.GetCollection<T>(typeof(T).Name);
    var queryable = query != null
        ? collection.AsQueryable().Where(query)
        : collection.AsQueryable();
```



What is Durable Execution?

Durable Execution

Refers to the capability of a system to ensure that a program's execution can withstand failures and continue seamlessly.

Durable Execution

Refers to a programming model where your application's logic **continues reliably**, **even in the face of infrastructure failures** with all state.

How Durable Execution **Works**

Start Process

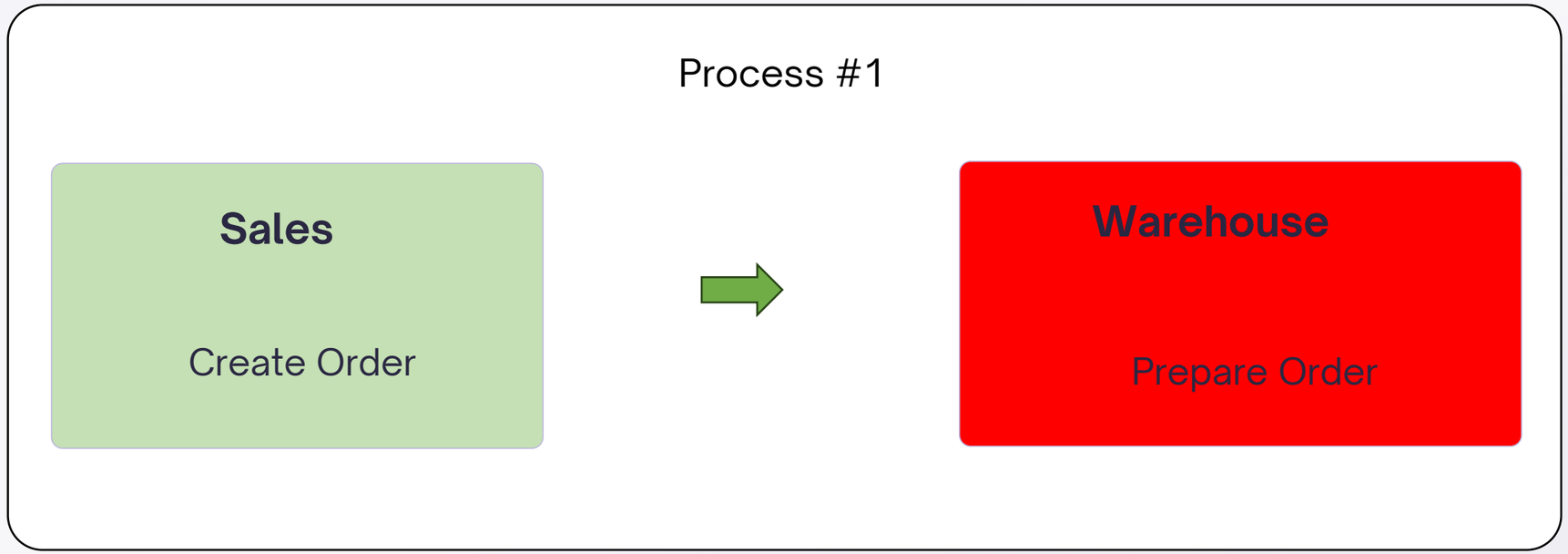
Persist State

Failure Occurs

Resume Process

INTR3

Durable Execution



Durable Execution

Durable Execution

Process #1

Sales
Create Order

→

Warehouse
Prepare Order

Process #2

Warehouse
Prepare Order

→

Payments
Charge C.C.

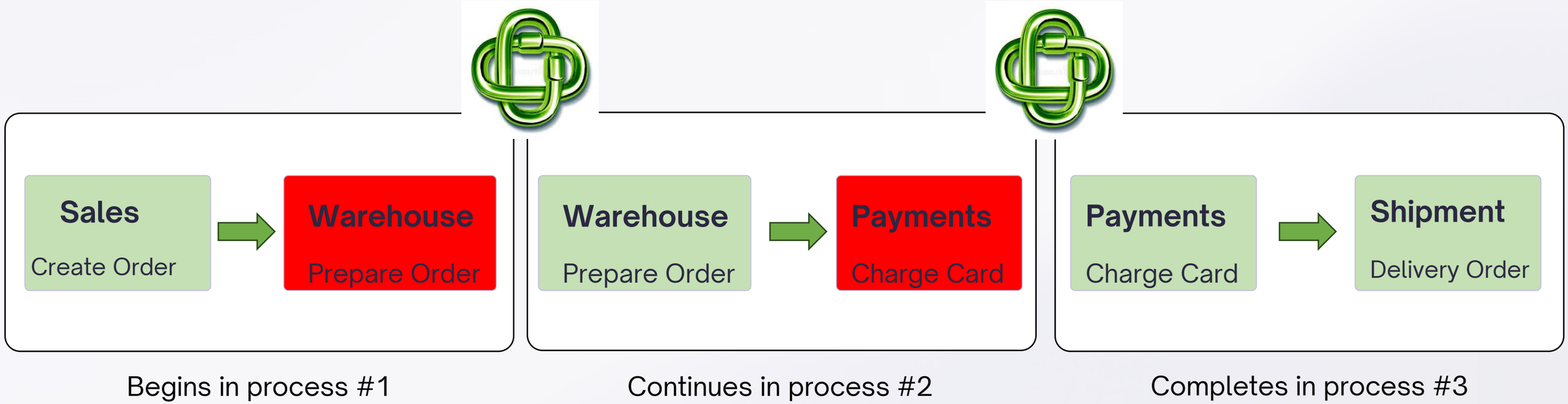
Process #3

Payments
Charge Card

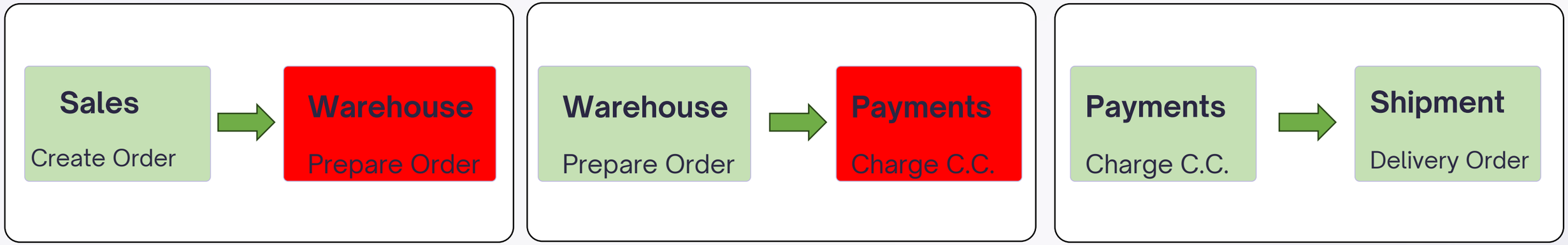
→

Shipment
Delivery Order

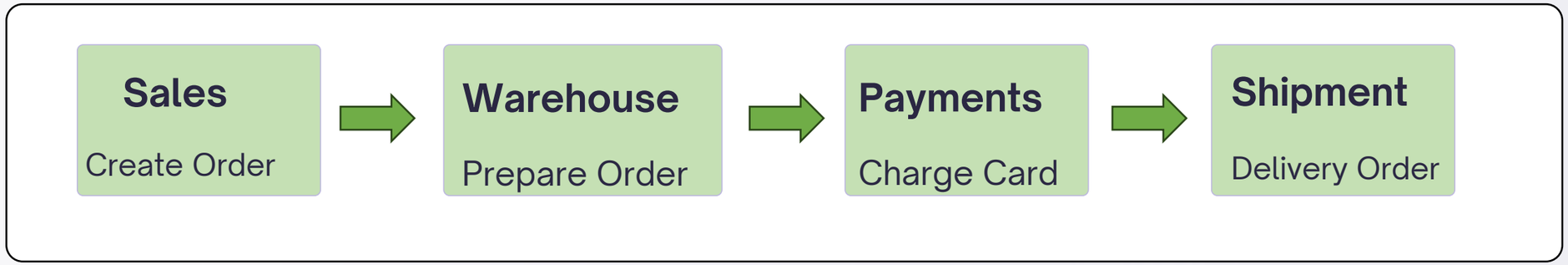
Role of Durable Execution



Physical representation of the processes



Developer experience



```
public class SalesOrderTasks
{
    public Task<string> CreateOrderAsync(SalesOrderJson salesOrder)
    {
        SalesOrderJson order = salesOrder with { SalesOrderId = Guid.NewGuid().ToString("N") };
        return Task.FromResult(order.SalesOrderId);
    }

    public Task<SalesOrderJson> PrepareOrderAsync(SalesOrderJson salesOrder)
    {
        // Do Something
        return Task.FromResult(order);
    }

    public Task<SalesOrderJson> ChargeCustomerCreditCardAsync(SalesOrderJson salesOrder)
    {
        // Do Something
        return Task.FromResult(order);
    }

    public Task<SalesOrderJson> ShipOrderToCustomerAsync(SalesOrderJson salesOrder)
    {
        // Do Something
        return Task.FromResult(order);
    }
}
```

```
public class SalesOrderTasks
{
    [Activity]
    public Task<string> CreateOrderAsync(SalesOrderJson salesOrder)
    {
        SalesOrderJson order = salesOrder with { SalesOrderId = Guid.NewGuid().ToString("N") };
        return Task.FromResult(order.SalesOrderId);
    }

    [Activity]
    public Task<SalesOrderJson> PrepareOrderAsync(SalesOrderJson salesOrder)
    {
        // Do Something
        return Task.FromResult(order);
    }

    [Activity]
    public Task<SalesOrderJson> ChargeCustomerCreditCardAsync(SalesOrderJson salesOrder)
    {
        // Do Something
        return Task.FromResult(order);
    }

    [Activity]
    public Task<SalesOrderJson> ShipOrderToCustomerAsync(SalesOrderJson salesOrder)
    {
        // Do Something
        return Task.FromResult(order);
    }
}
```

intr3

No Time Constraints With **Durable Execution**

INTR3

No inherent time limits

- Durable Execution ensures application resilience in the face of crashes
- Think about how you would design an application to handle a 20-year loan
- Makes it possible to design apps that focus on what the system is about, not just what it does.

Durable Execution diminishes the need to manually preserve state

INTR3

Manually preserving application state

- Consider the time and resources consumed just to keep data consistent between your app and the DB
- Often, the real goal isn't to use a database – it's to manage state effectively.
- Durable Execution is crash-proof execution.

INTR3

Durable Execution is hardware agnostic

INTR3

Comparing two approaches to fault tolerance

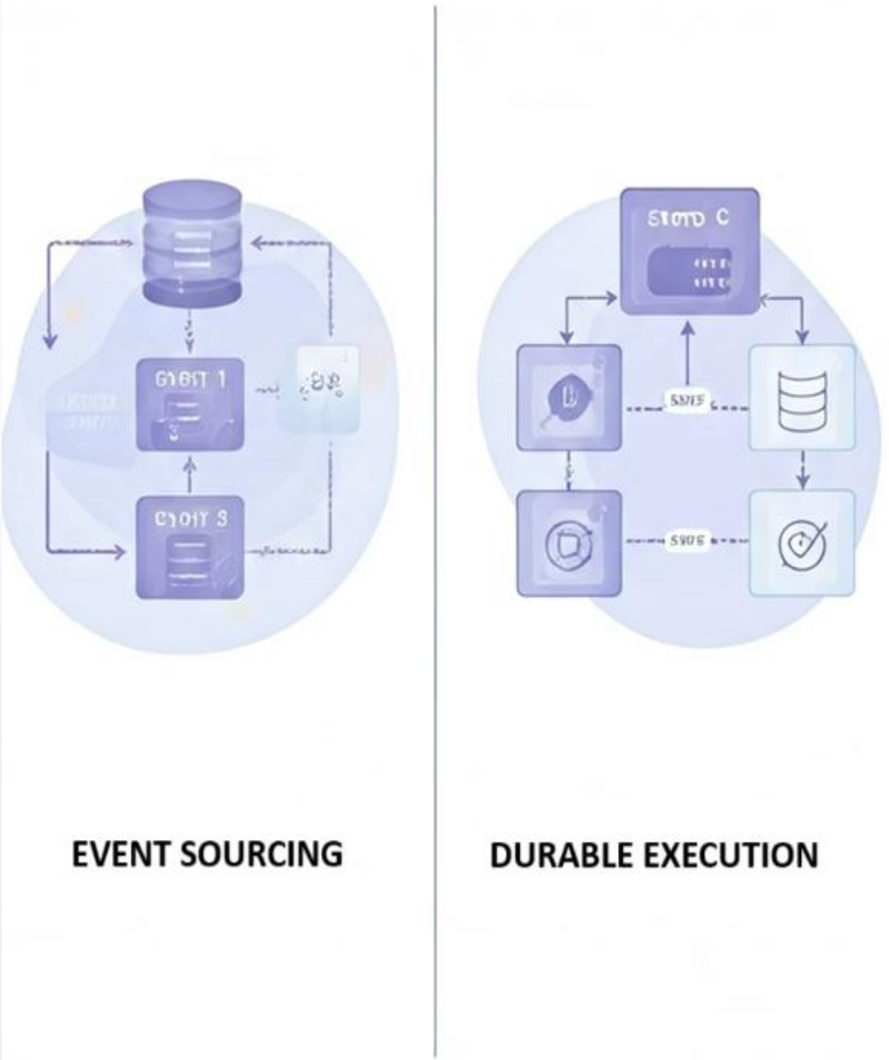
- Reliable systems traditionally emphasized fault-tolerant hardware
- This can reduce downtime and the budget should reflect its cost.
- Durable Execution is a software-based approach to fault tolerance.


INTR3

Durable Execution and Event Sourcing

Concept	Durable Execution	Event Sourcing
Definition	A mechanism for persisting the progress of a long-running workflow.	A pattern where state is stored as a sequence of events.
Goal	Ensure reliable, resumable execution of workflows or processes.	Capture every change to an application's state.
Persistence unit	Workflow state, task checkpoints, or orchestrator state.	Domain Events.
Core idea	Resume execution after failures or interruptions.	Rebuild current state from past events.

Both share a philosophy: capturing state transitions as immutable records to ensure resilience.



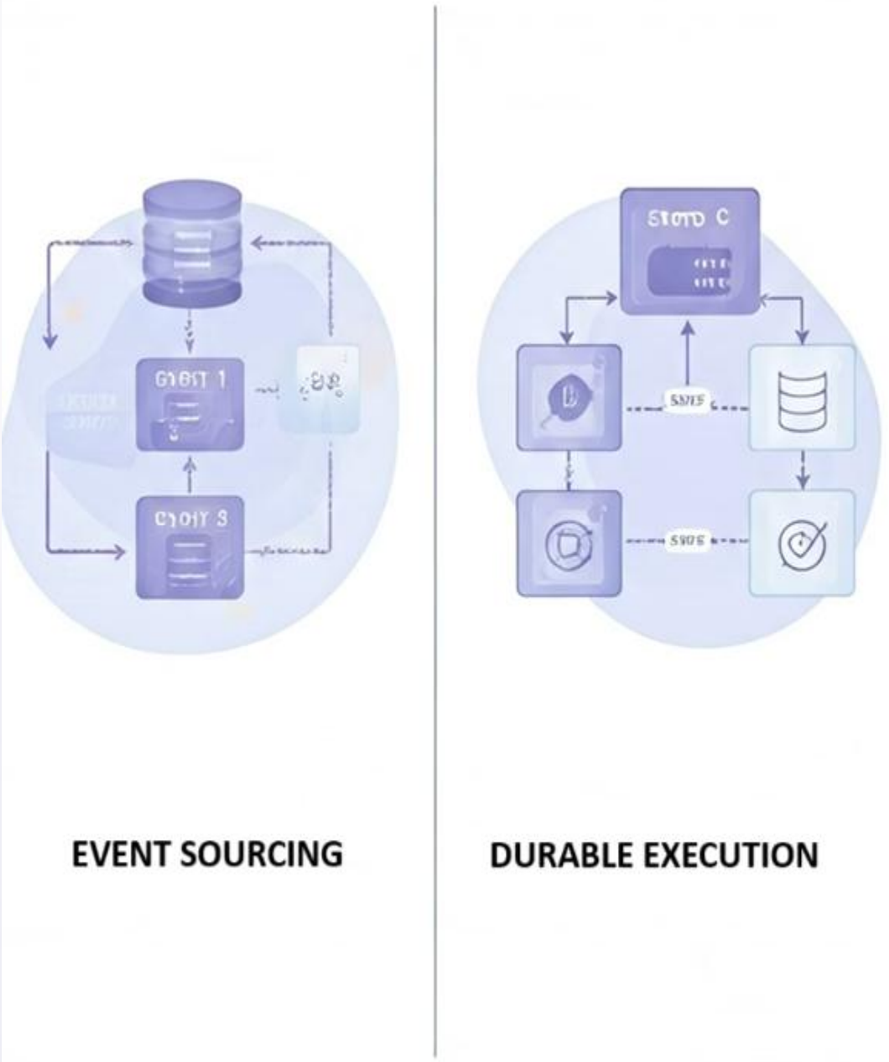
Conceptual Difference

Event Sourcing is about storing system state changes as long as a log of immutable events

- The source of truth is the event log.
- State can be reconstructed by replacing those events.
- Focused on auditability, consistency, and replayability.

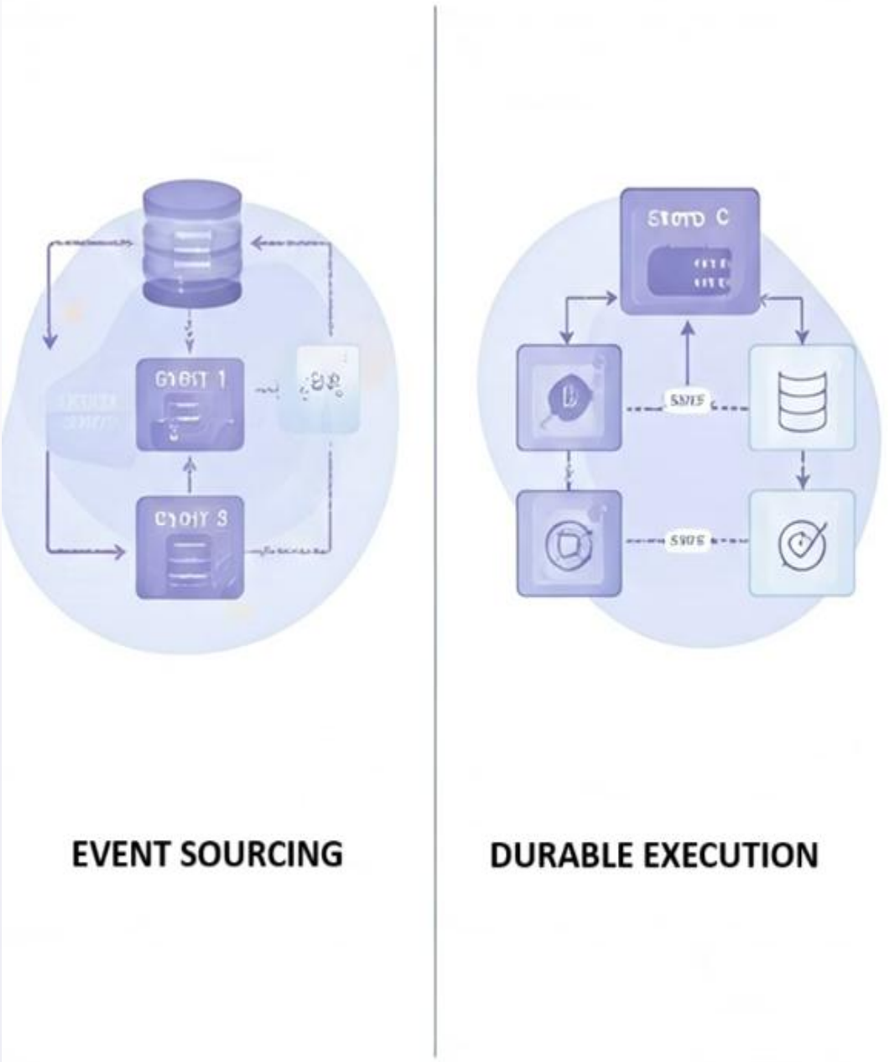
Durable Execution is about tracking the execution of workflows over time

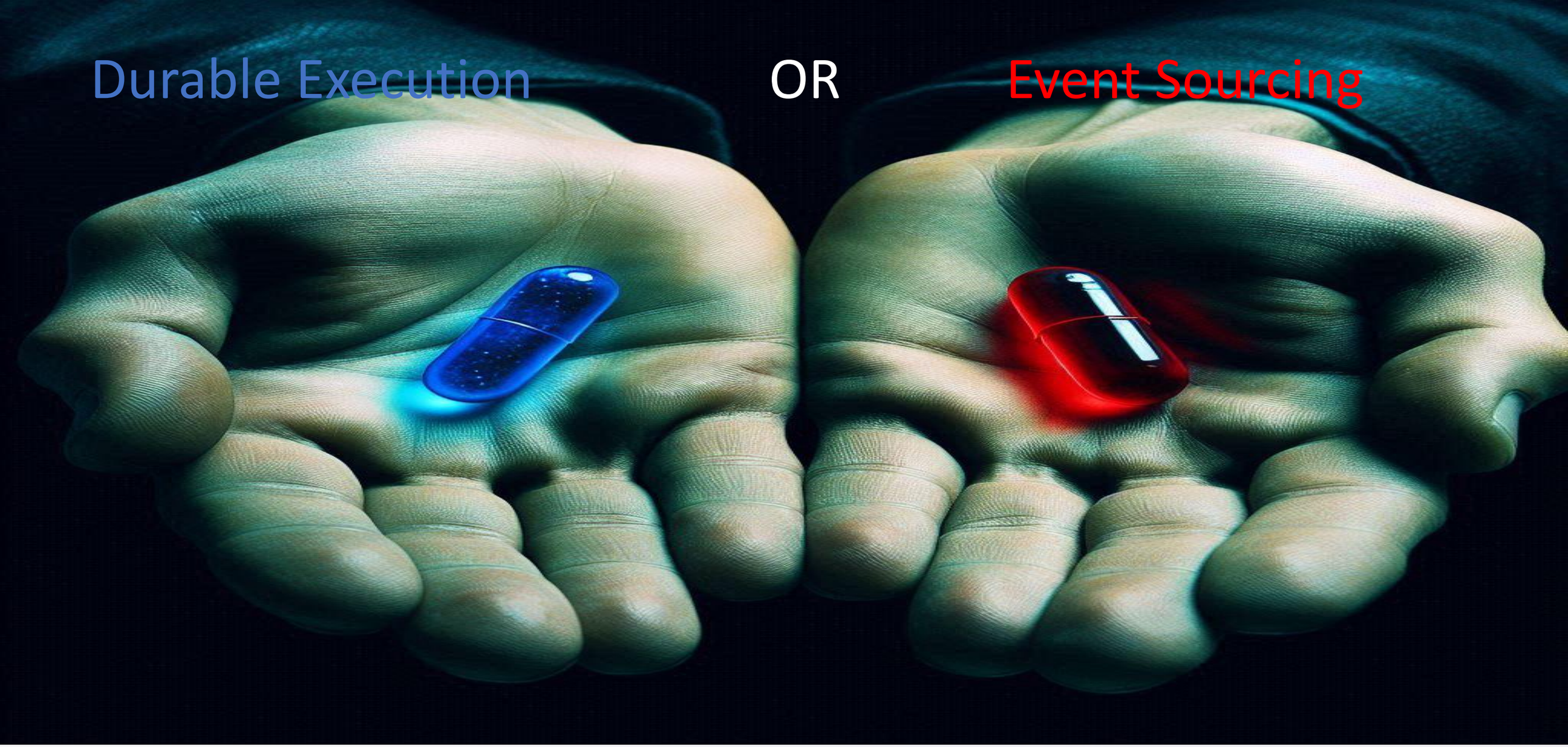
- The focus is on reliability of execution, not just data.
- Involves orchestration, step tracking, and automatic retries.
- Often used for long-running, distributed operations.



What they have in common?

Aspect	Description
State persistence	Both persist state in a durable way, ensuring recoverability
Reliability focus	Each is used to improve system reliability and fault tolerance
Immutability concept	Both embrace immutability
Useful in distributed systems	Both patterns are common in distributed architectures
Auditable execution path	Both allow inspection of what happened

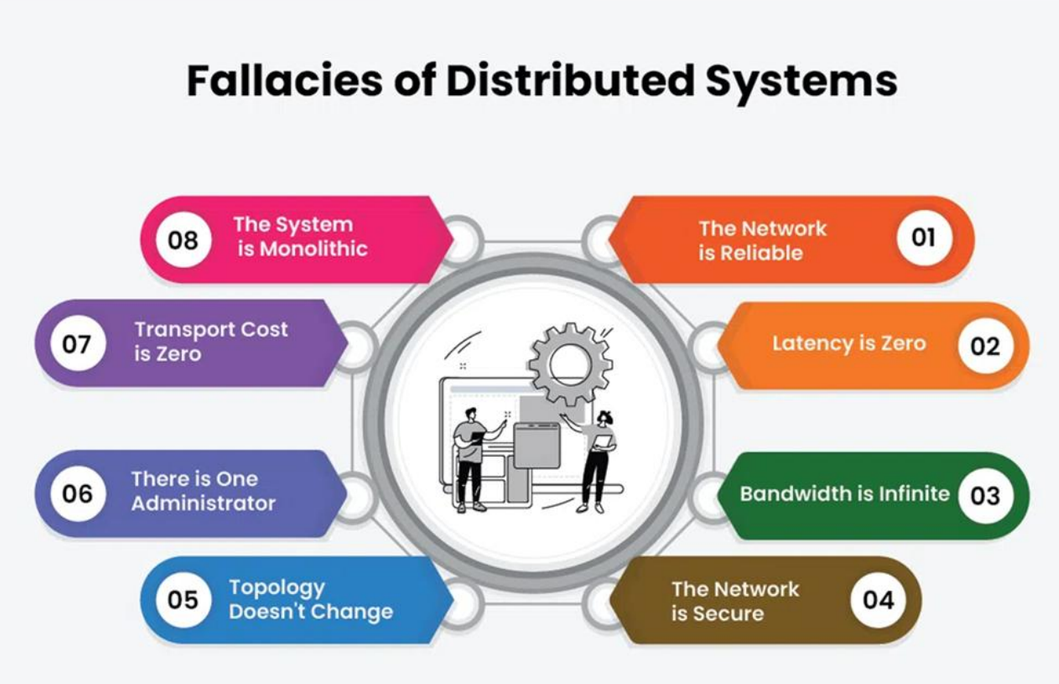



Durable Execution
OR
Event Sourcing

Event Sourcing or Durable Execution?

Need	Choose This
Replay system state from history	Event Sourcing
Orchestrate multi-step workflow	Durable Execution
Implements audit trails	Event Sourcing
Manage retries, timeouts, and waiting	Durable Execution
Combine both	Reliability & Traceability

Why Distributed Systems Fail?



- When **the network isn't reliable**
  - Durable Execution ensures tasks can resume after failure
- When **latency isn't zero**
  - Persisted state and retries help maintain progress without loss
- When **topology changes**
  - The system can relocate execution without manual intervention

History of Durable Execution

Azure Durable Tasks

Temporal

aws

AWS Step Functions

restate

INTR3

- Distillation pattern
- Cost of implementation
- Use platforms to solve your problems

INTR3

Conclusion

01.

Accept Failure Reality

System failures are a matter of 'when,' not 'if.'

INTR3

Conclusion

01.

Accept Failure Reality

System failures are a matter of 'when,' not 'if.'

02.

Implement Durable Execution

Persist state and automate recovery for resilient applications.

INTR3

Conclusion

01.

Accept Failure Reality

System failures are a matter of 'when,' not 'if.'

02.

Implement Durable Execution

Persist state and automate recovery for resilient applications.

03.

Focus on Value Delivery

Let infrastructure handle uncertainties while developers build features.

Thank you!

Alberto Acerbis

Microsoft® Most Valuable Professional

alberto.acerbis@intre.it

https://intre.it/en

Use ***DDD20*** for a 20% discount during the conference days