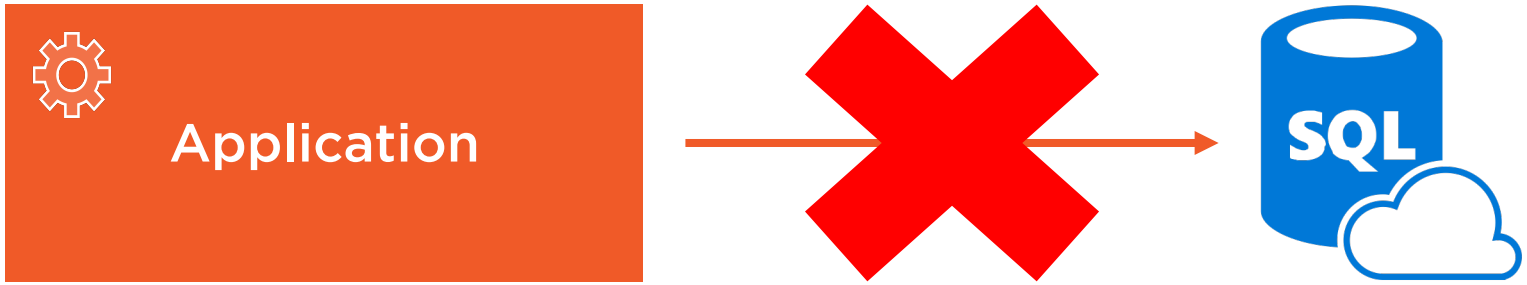
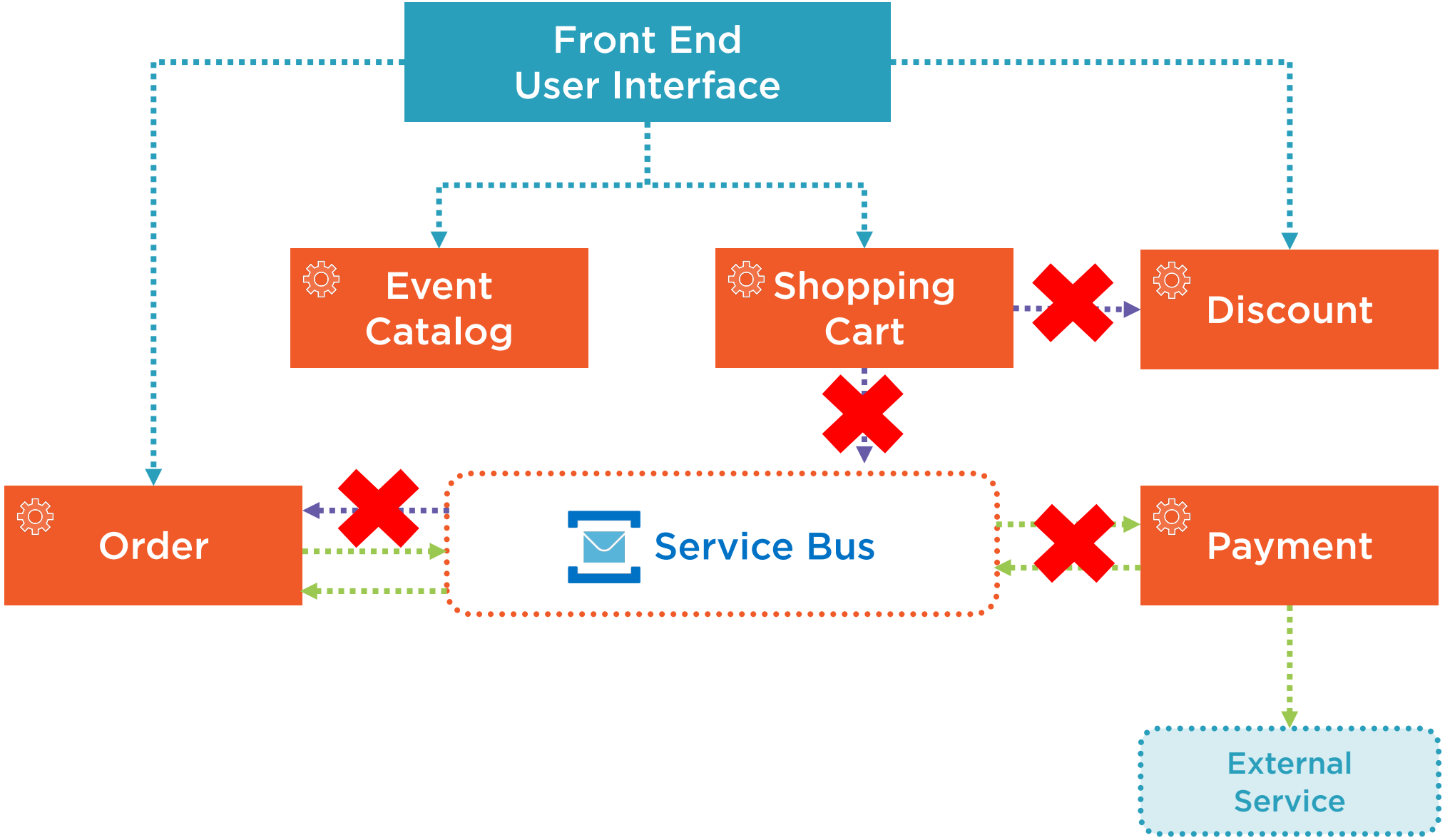
Data Consistency across Microservices

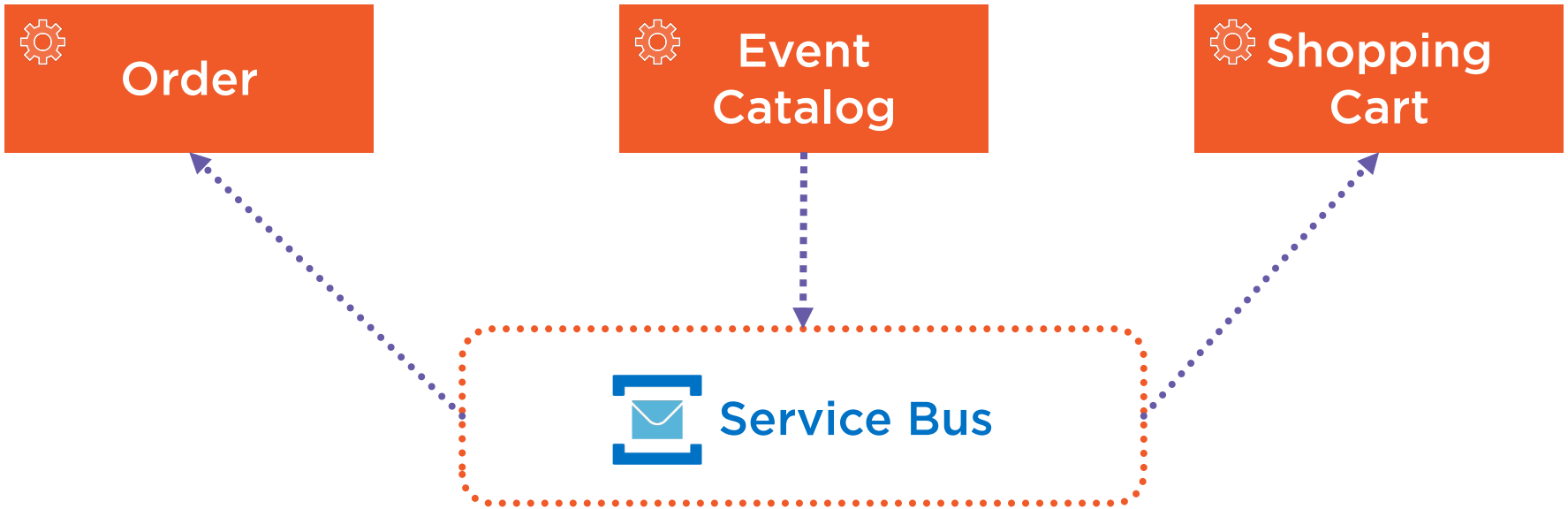
Neil Morrissey
SOLUTIONS ARCHITECT

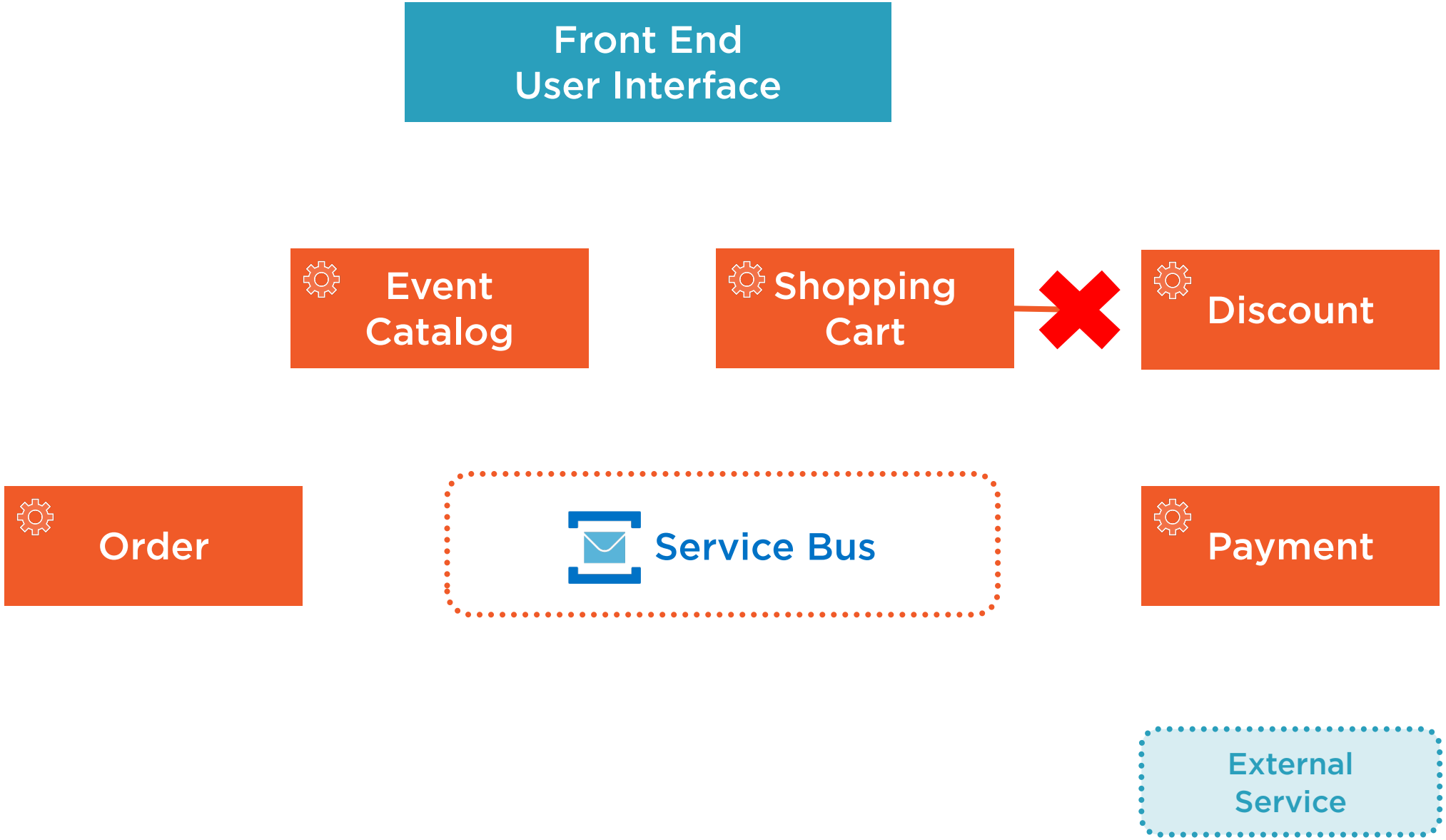
@morrisseycode

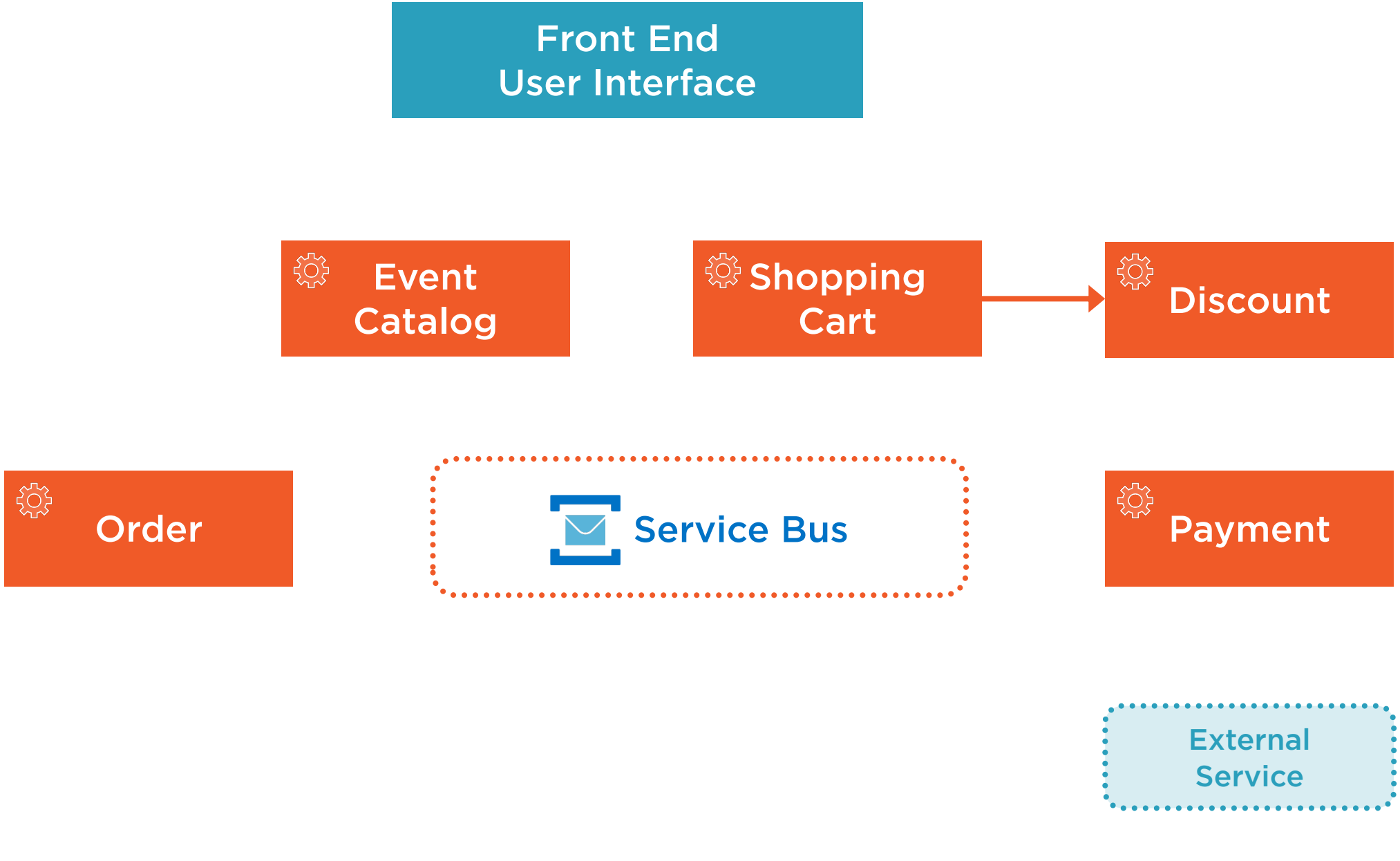
neilmorrissey.net

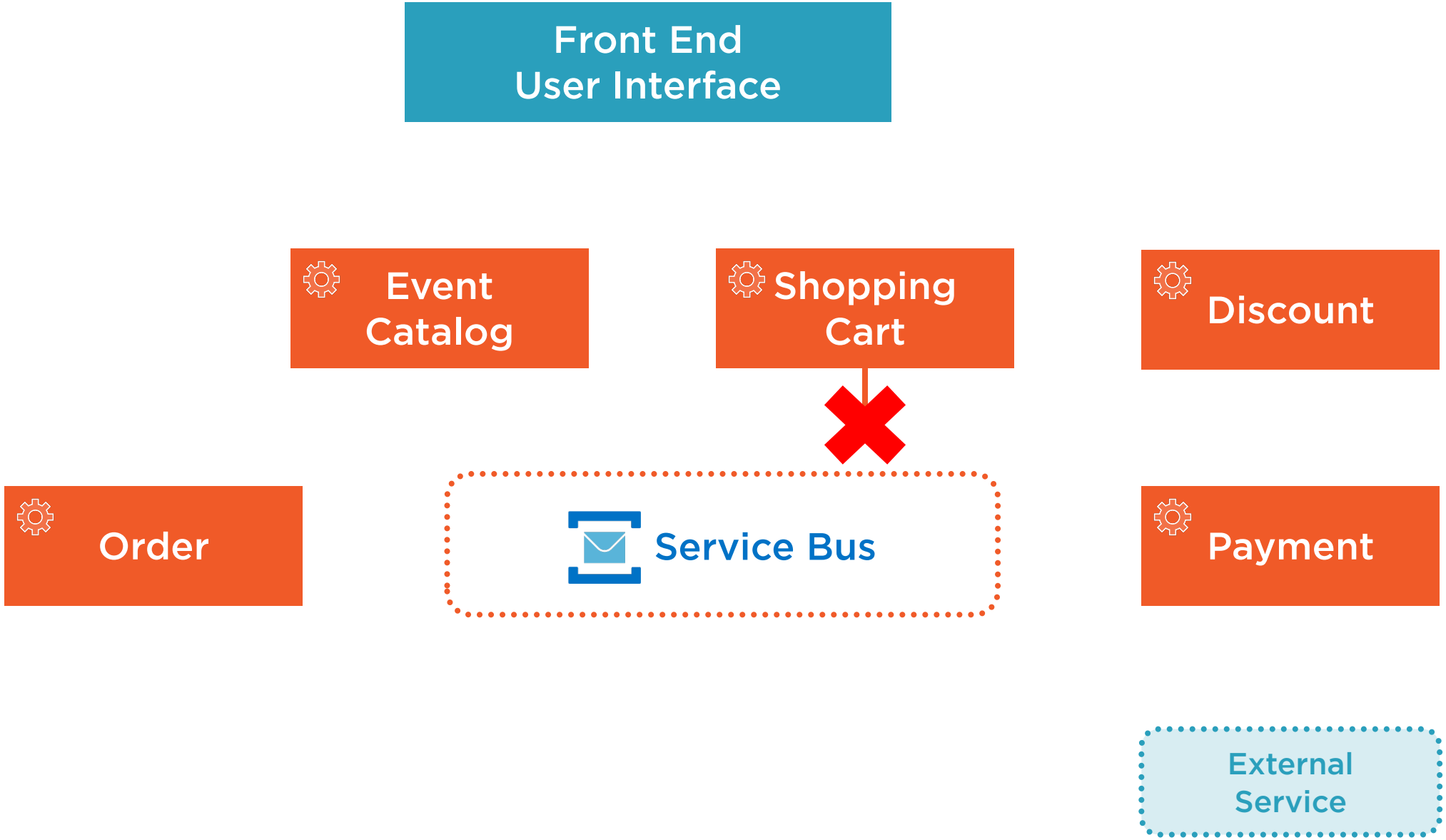

Application
SQL

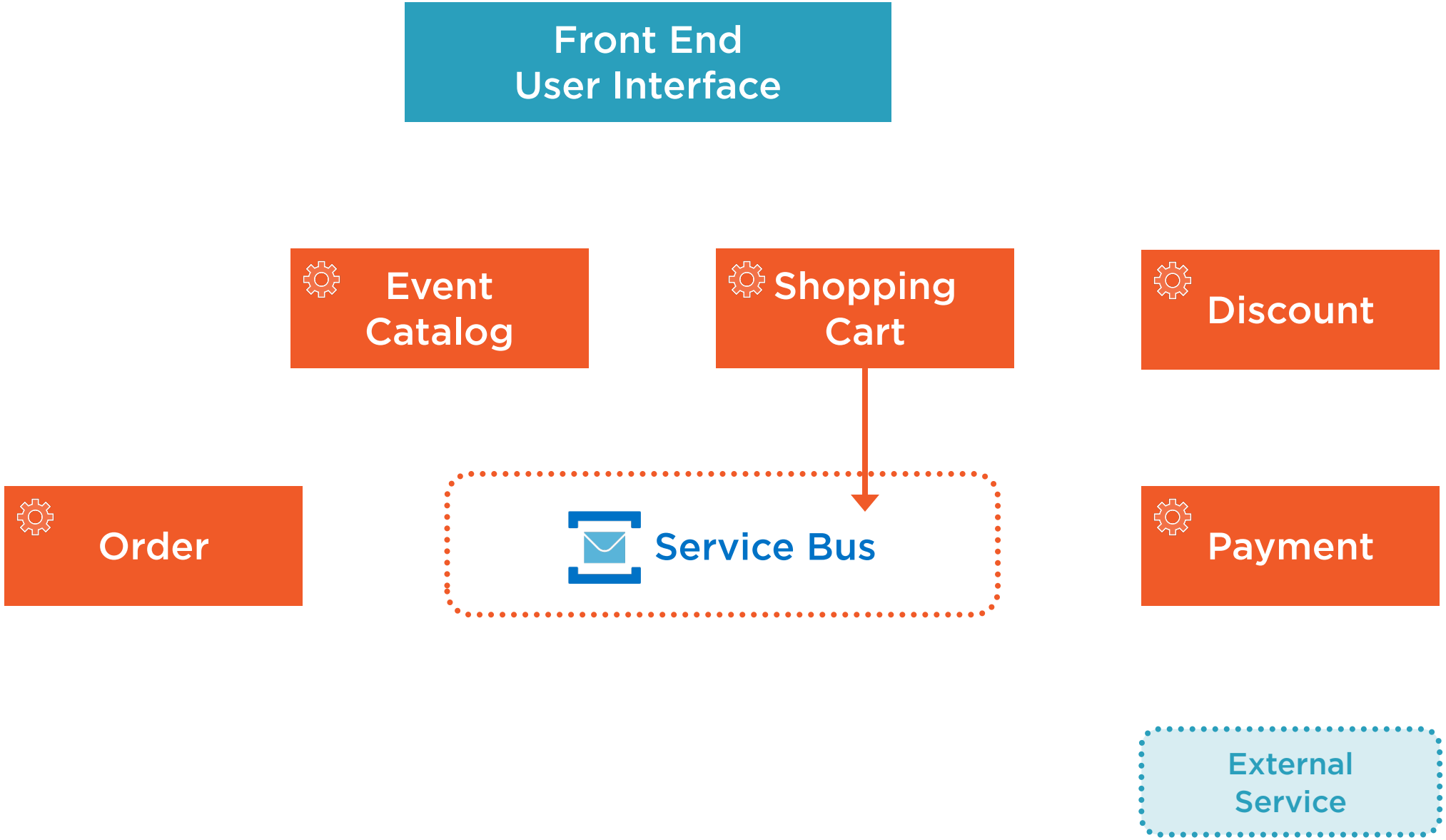

Front End
User Interface
Event
Catalog
Shopping
Cart
Discount
Order
Service Bus
Payment
External
Service

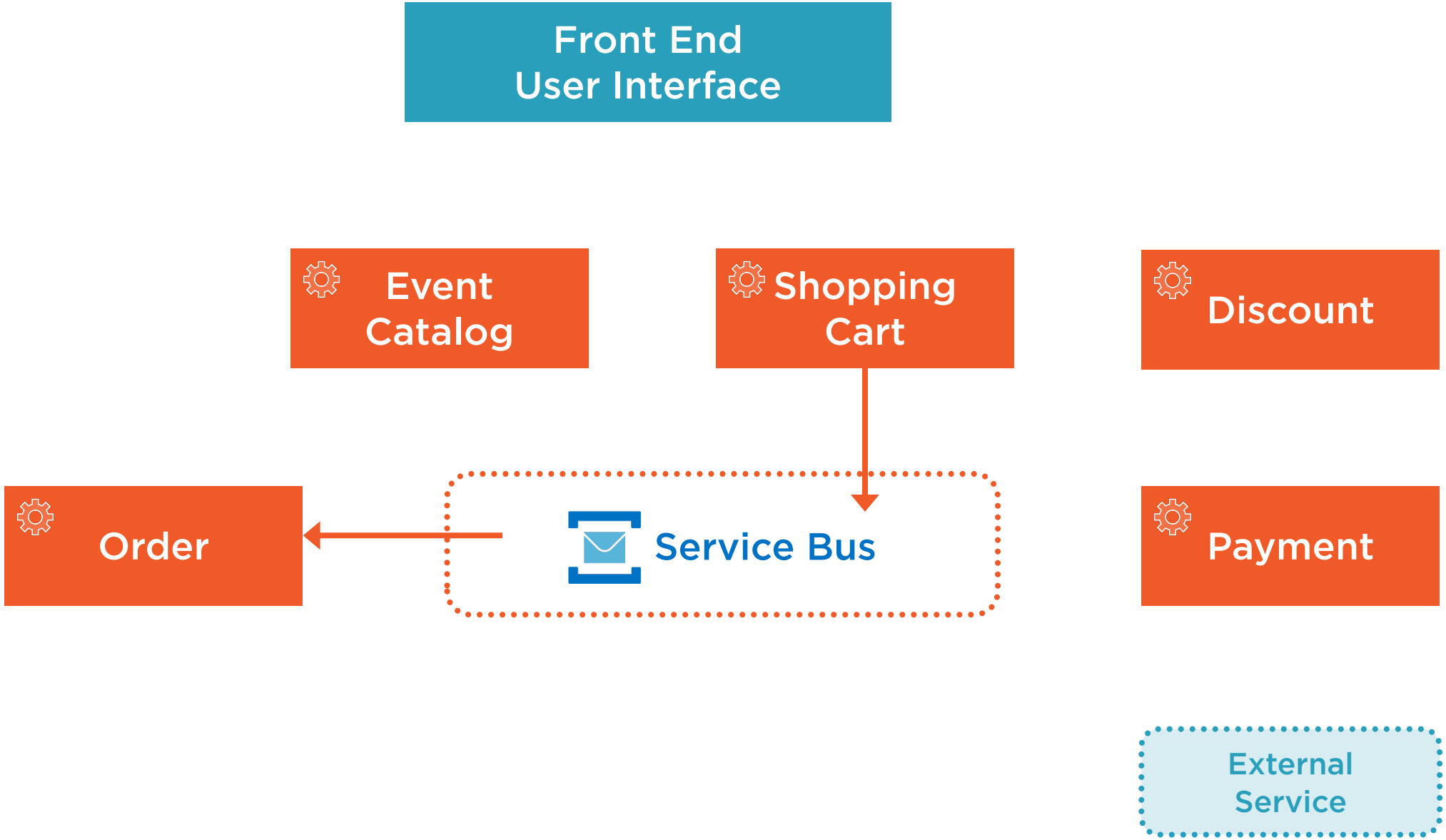

Order
Event Catalog
Shopping Cart
Service Bus

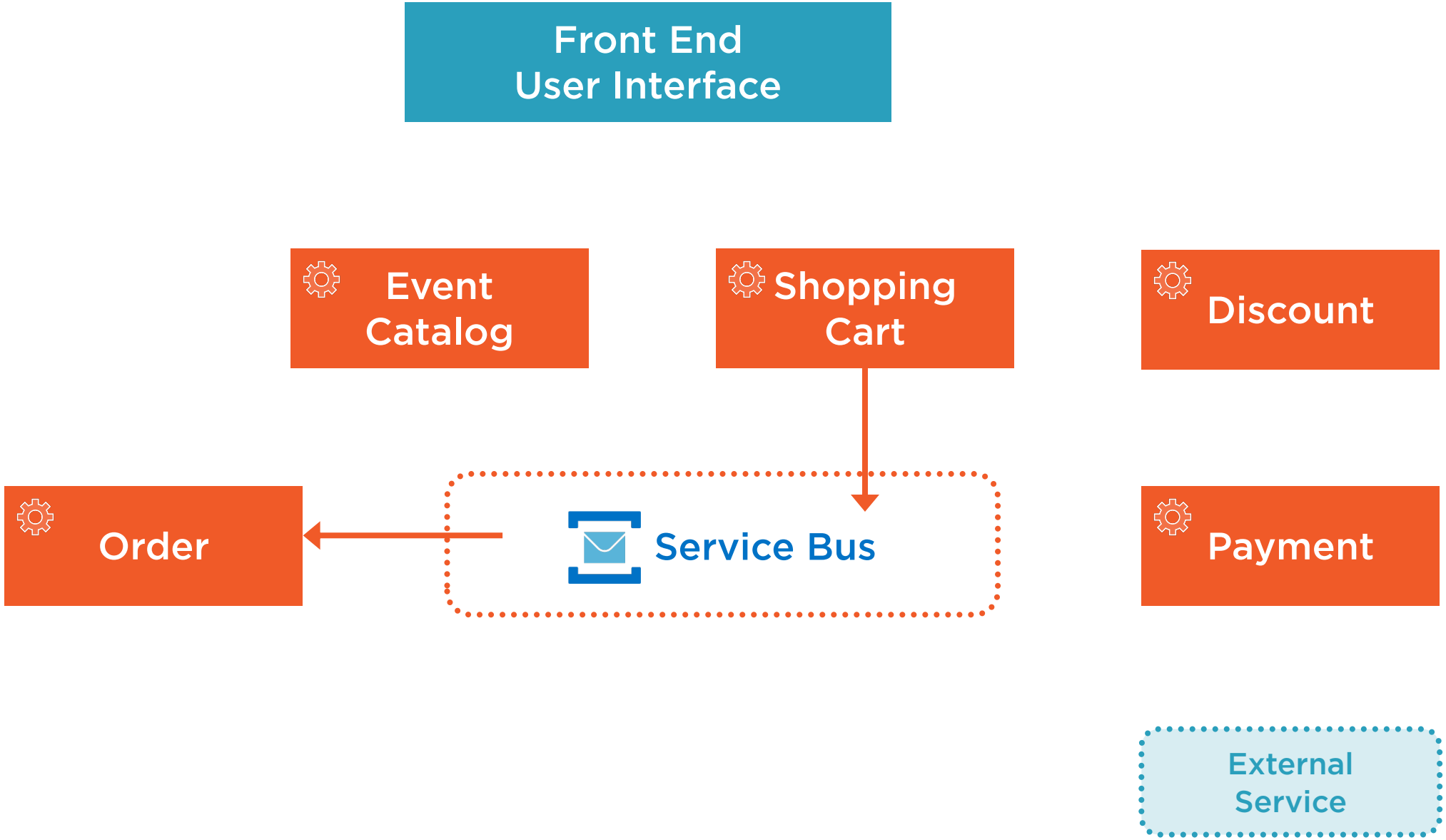

Front End
User Interface
Event
Catalog
Shopping
Cart
Discount
Order
Service Bus
Payment
External
Service


Front End
User Interface
Event
Catalog
Shopping
Cart
Discount
Order
Service Bus
Payment
External
Service

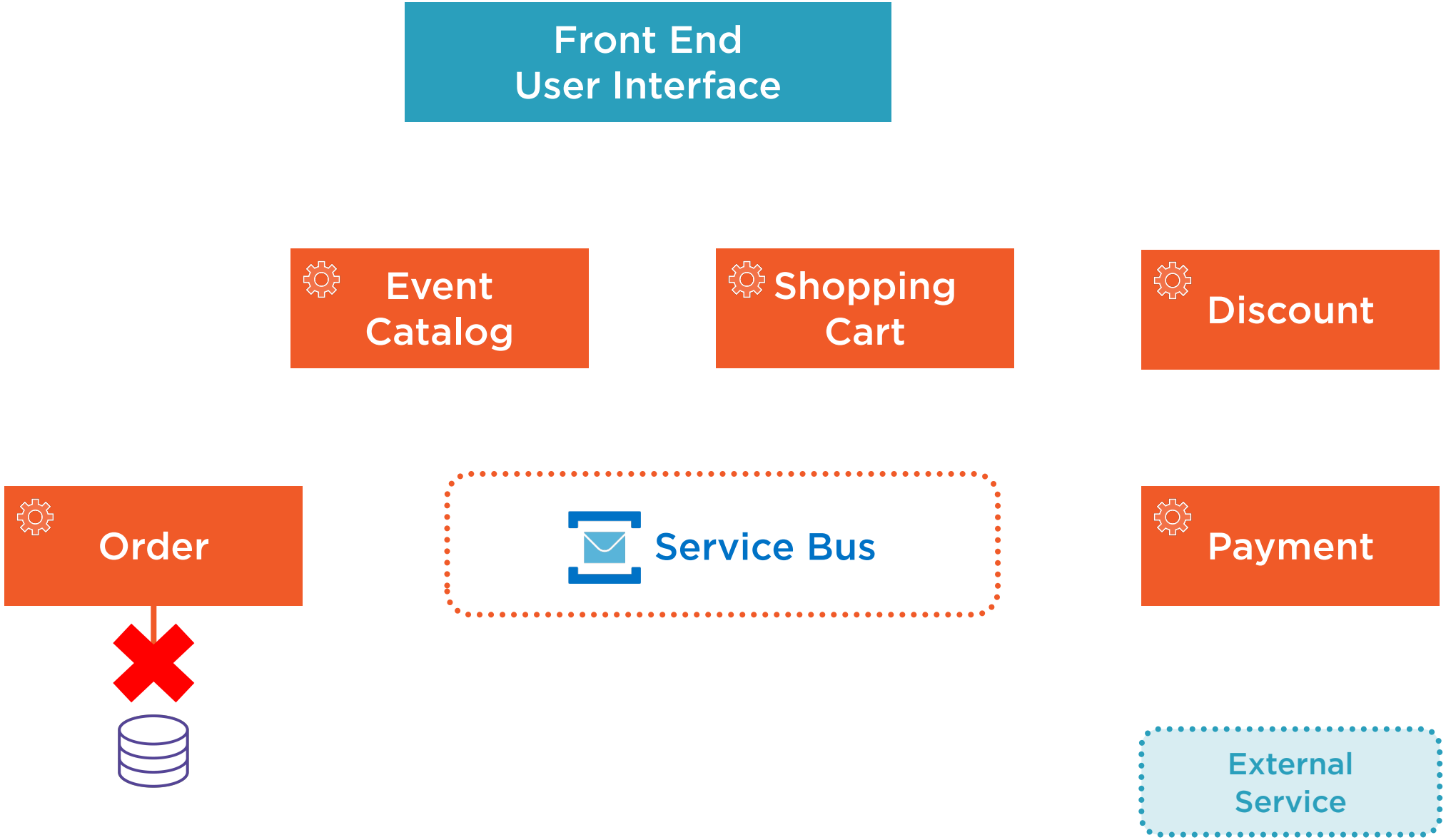

Front End
User Interface
Event
Catalog
Shopping
Cart
Discount
Order
Service Bus
Payment
External
Service

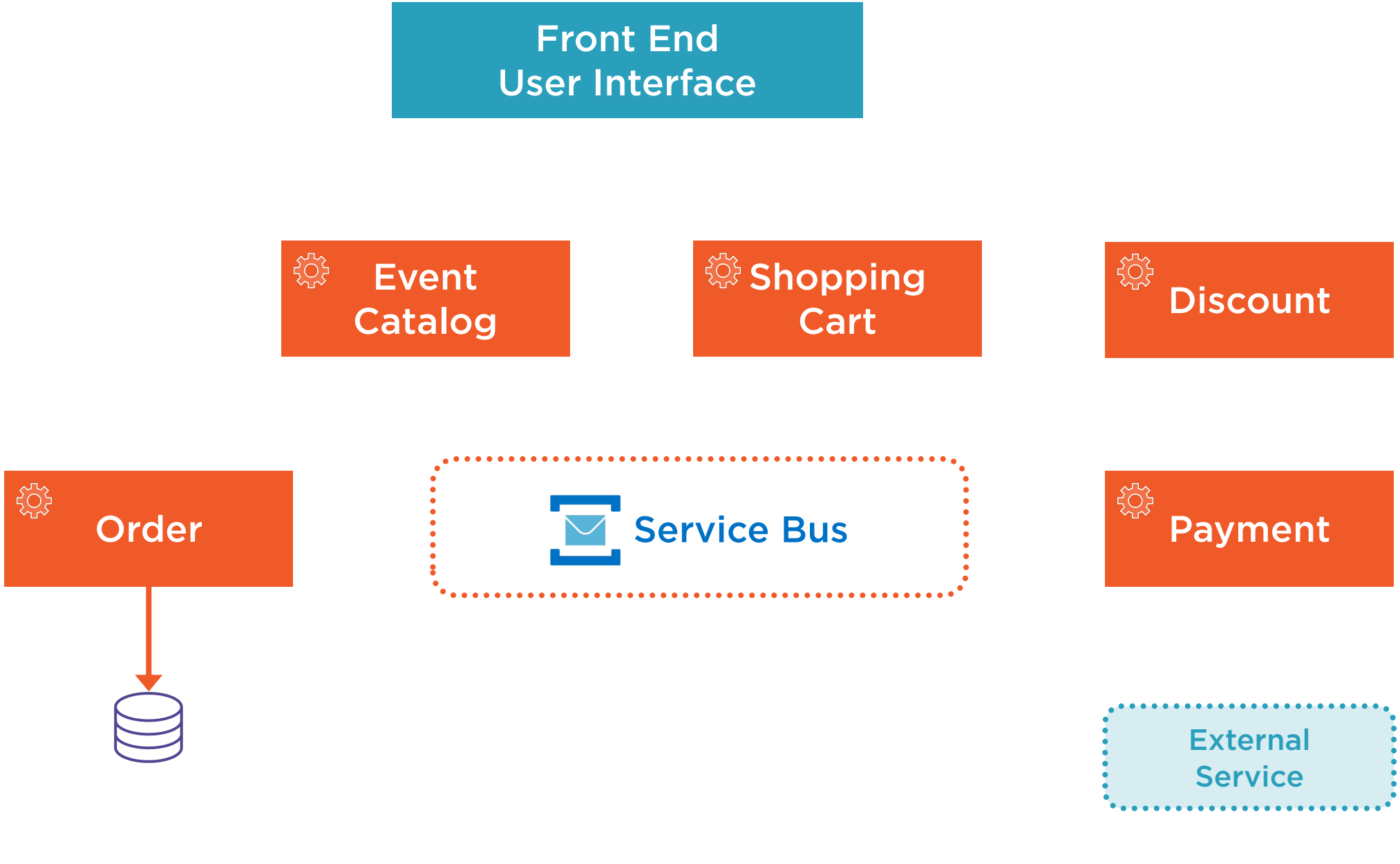

Front End
User Interface
Event
Catalog
Shopping
Cart
Discount
Order
Service Bus
Payment
External
Service


Front End
User Interface
Event
Catalog
Shopping
Cart
Discount
Order
Service Bus
Payment
External
Service


Front End
User Interface
Event
Catalog
Shopping
Cart
Discount
Order
Service Bus
Payment
External
Service

Demo


Front End
User Interface
Event
Catalog
Shopping
Cart
Discount
Order
Service Bus
Payment
External
Service


Front End
User Interface
Event
Catalog
Shopping
Cart
Discount
Order
Service Bus
Payment
External
Service

Module Overview

Integration events for eventual consistency

Event sourcing pattern

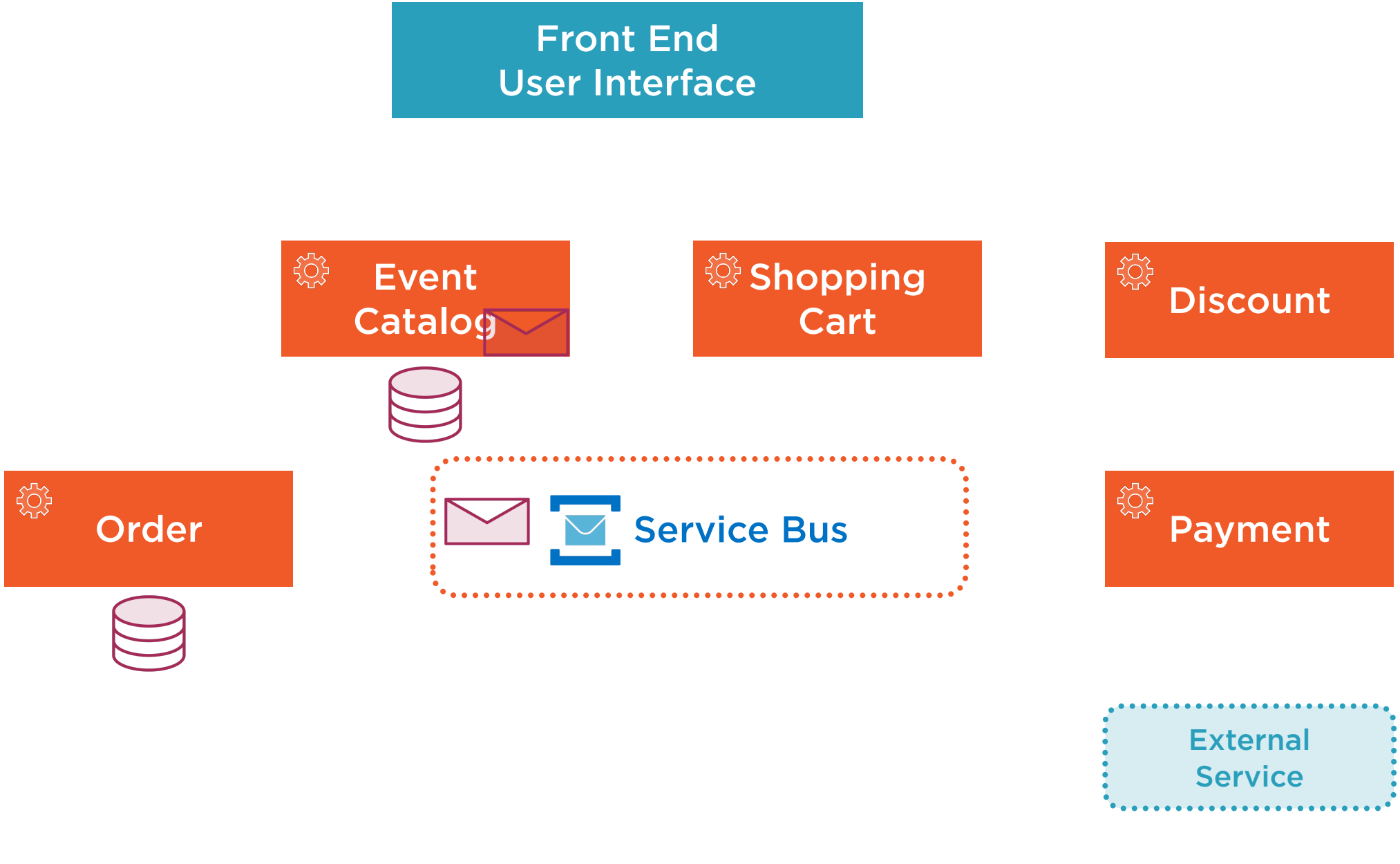
Unit of work pattern

Implementing the event sourcing pattern

EF Core connection resiliency

Handling duplicate messages

Using Integration Events for Eventual Consistency


Front End
User Interface
Event
Catalog
Shopping
Cart
Discount
Order
Service Bus
Payment
External
Service

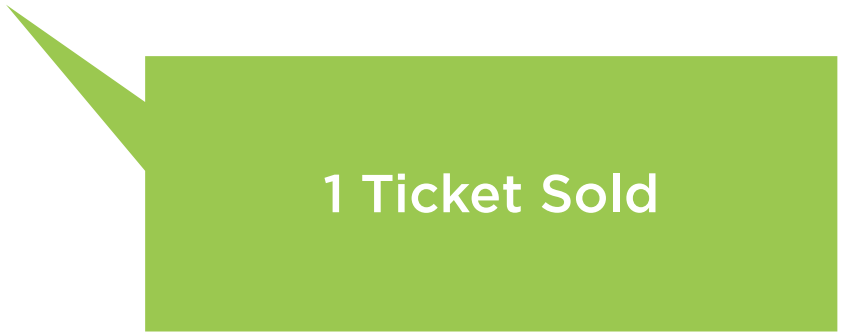
The Event Sourcing Pattern

Traditional Database Approach

ID	Name	Price	AvailableTickets
254	Conference1	$50	100

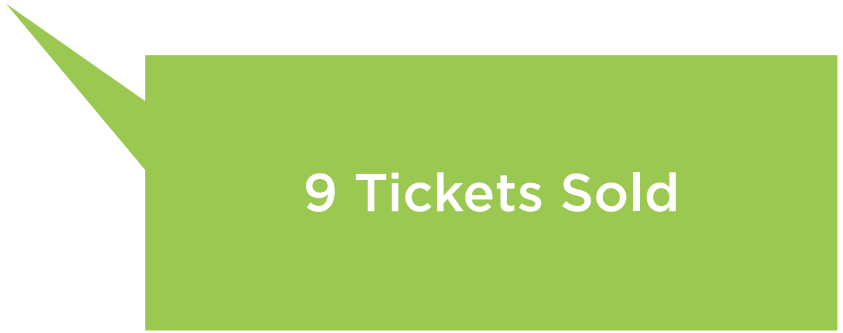
Traditional Database Approach

ID	Name	Price	AvailableTickets
254	Conference1	$50	99



Traditional Database Approach

ID	Name	Price	AvailableTickets
254	Conference1	$50	90



Traditional Database Approach

ID	Name	Price	AvailableTickets
254	Conference1	$50	92

2 Tickets Returned

Traditional Database Approach

ID	Name	Price	AvailableTickets
254	Conference1	$55	92

Traditional Database Approach

ID	Name	Price	AvailableTickets
254	Conference1	$55	92

Conference1
Price $55
92 Tickets Available

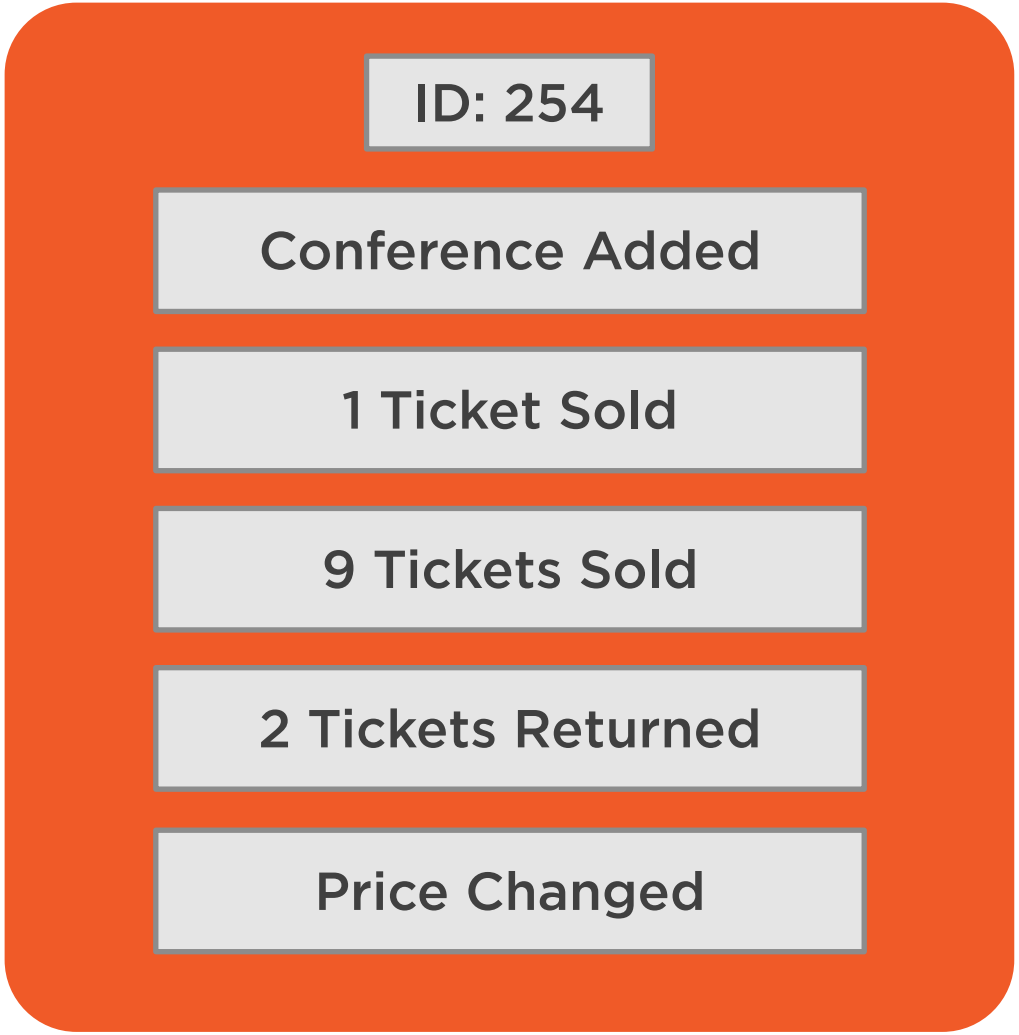
Traditional Database Approach

ID	Name	Price	AvailableTickets
254	Conference1	$55	92

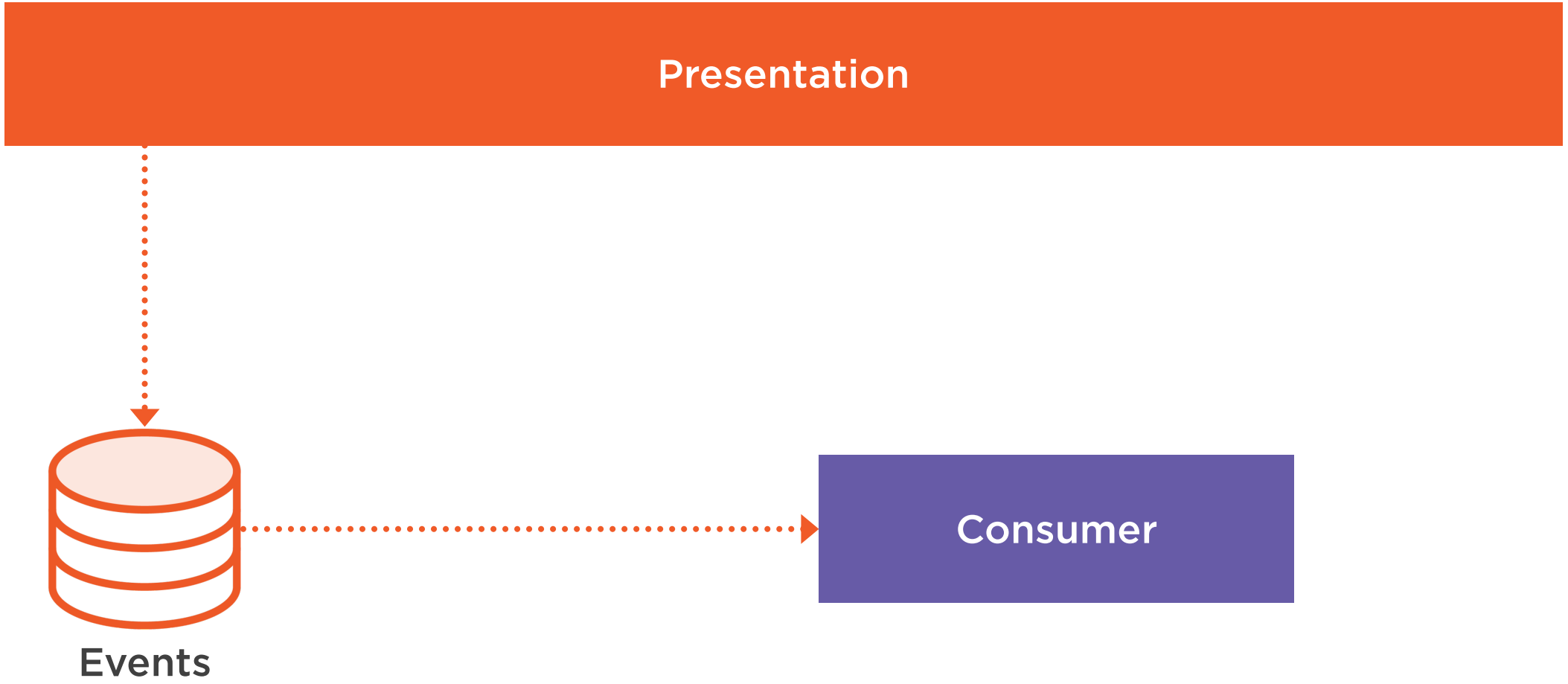
History Table

Id	EntityId	Name	Price	Tickets
1	254	Conference1	$50	100
2	254	Conference1	$50	99
3	254	Conference1	$50	90
4	254	Conference1	$50	92
5	254	Conference1	$55	92

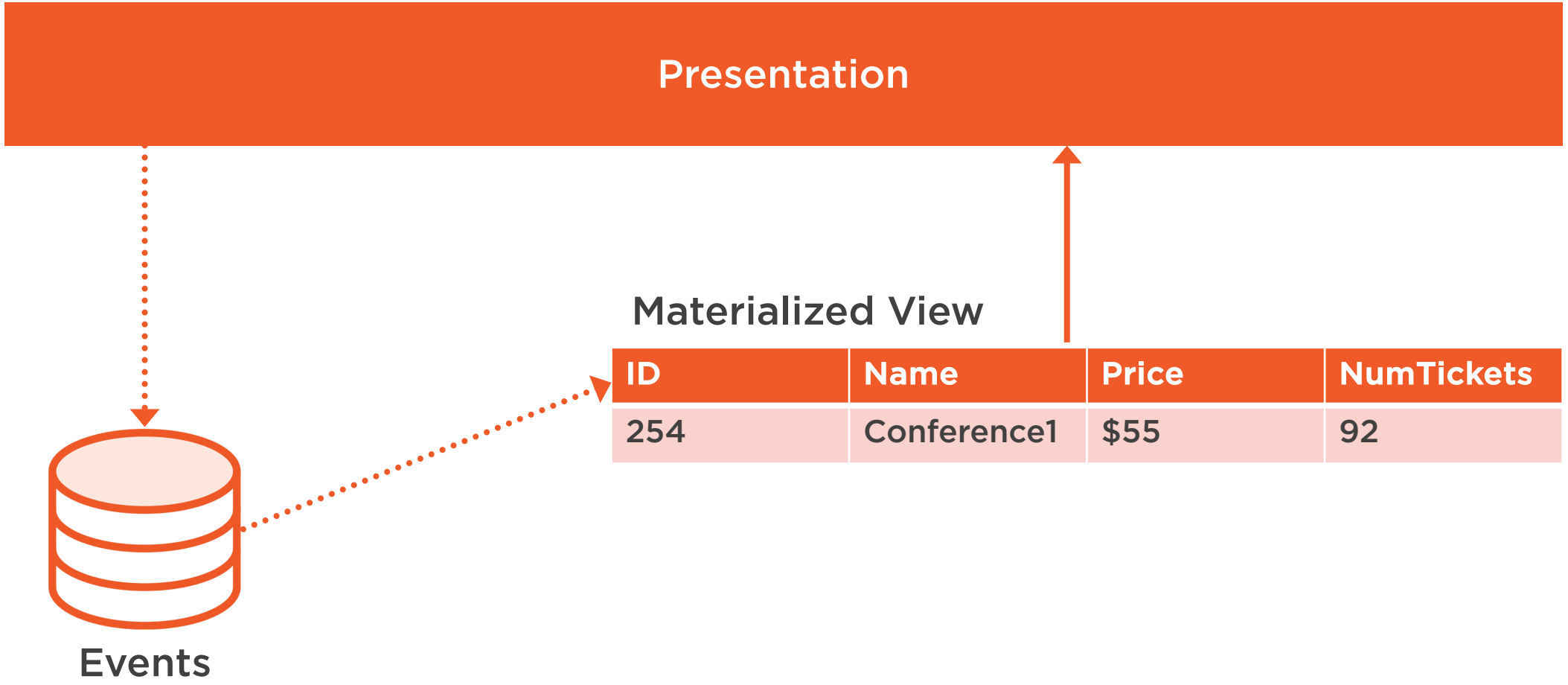
Event Sourcing



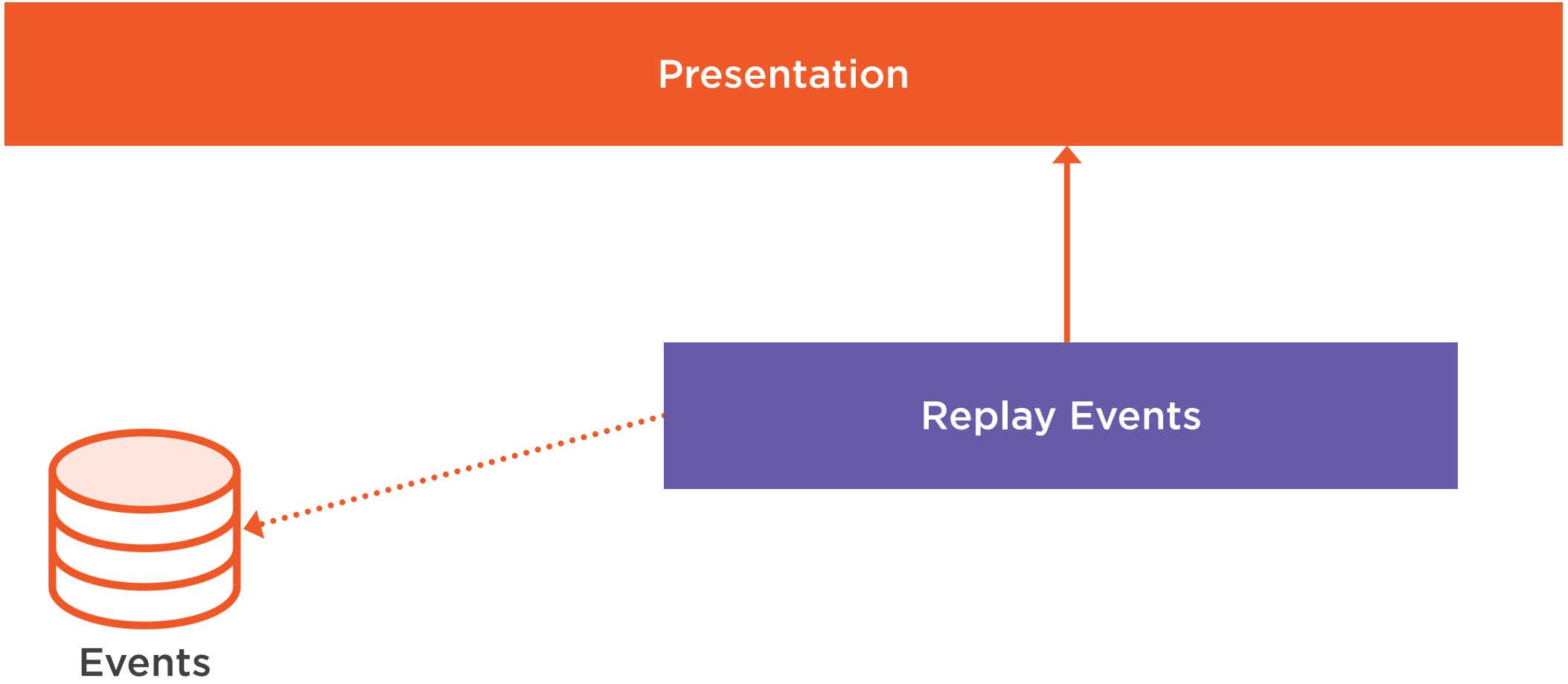
Event Sourcing Pattern



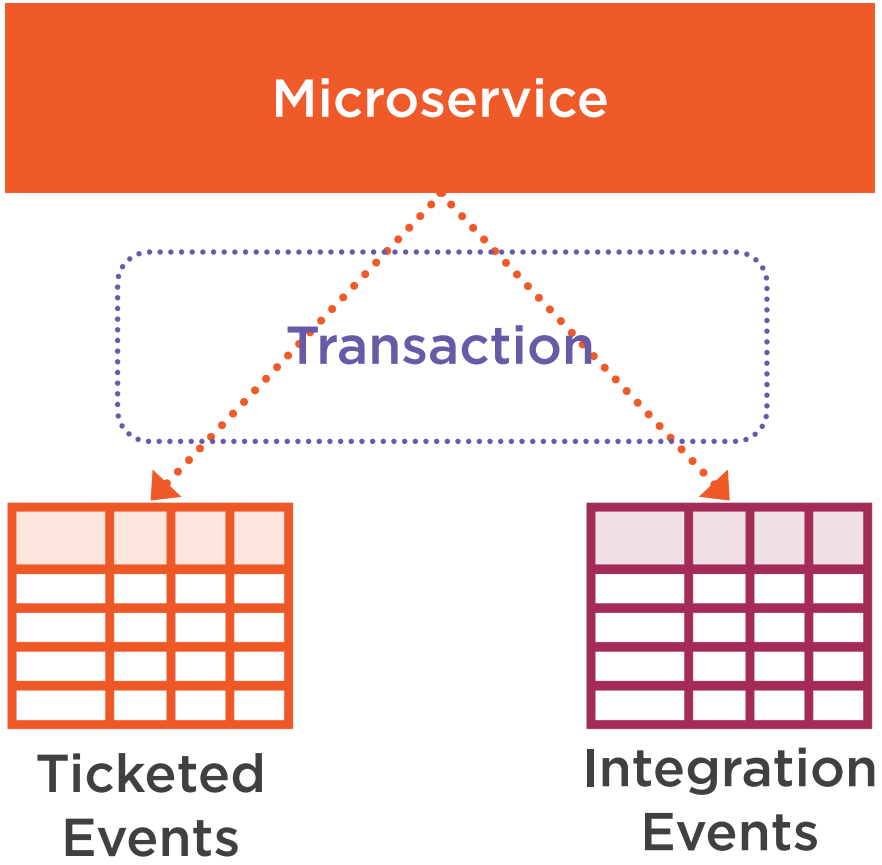
Event Sourcing Pattern



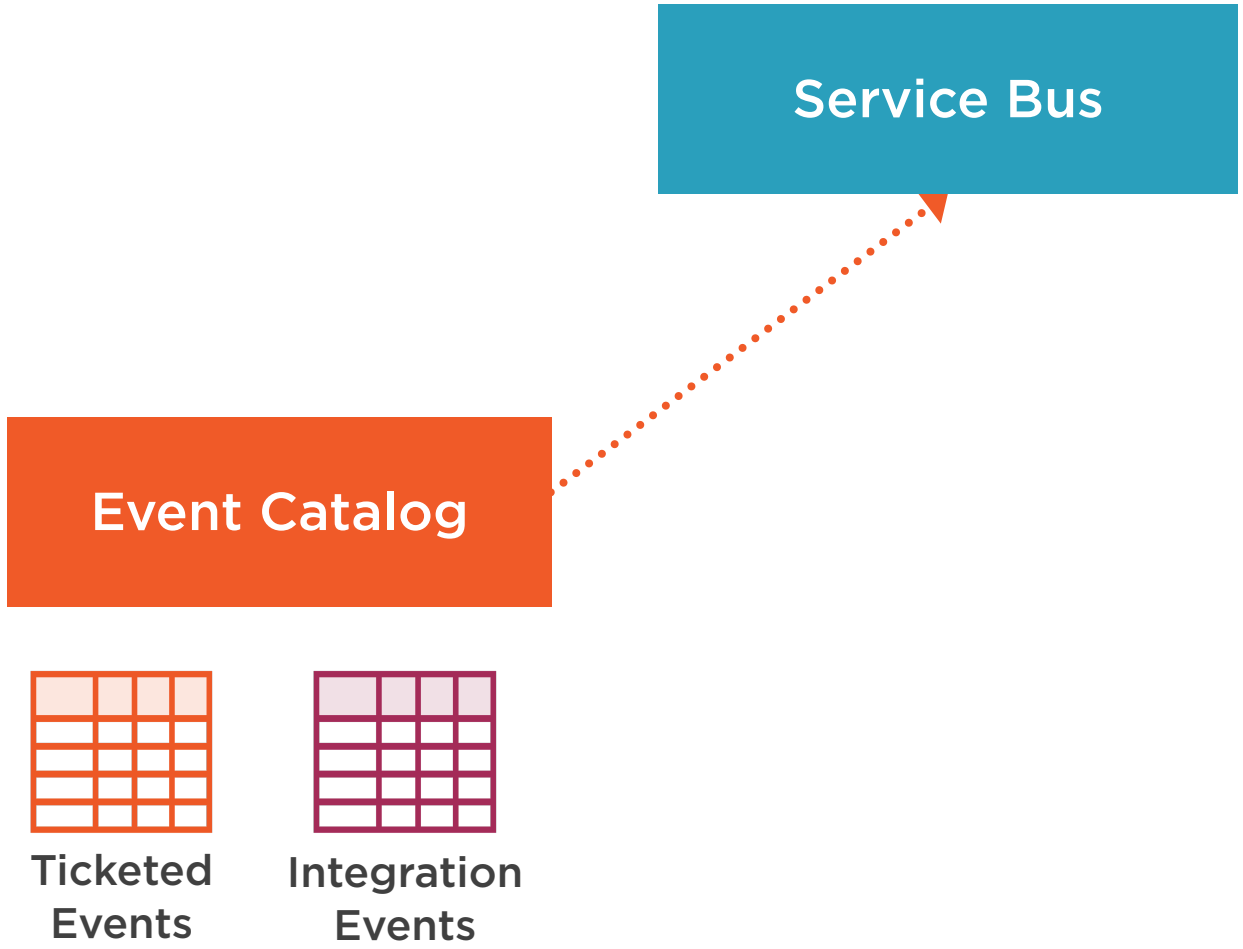
Event Sourcing Pattern



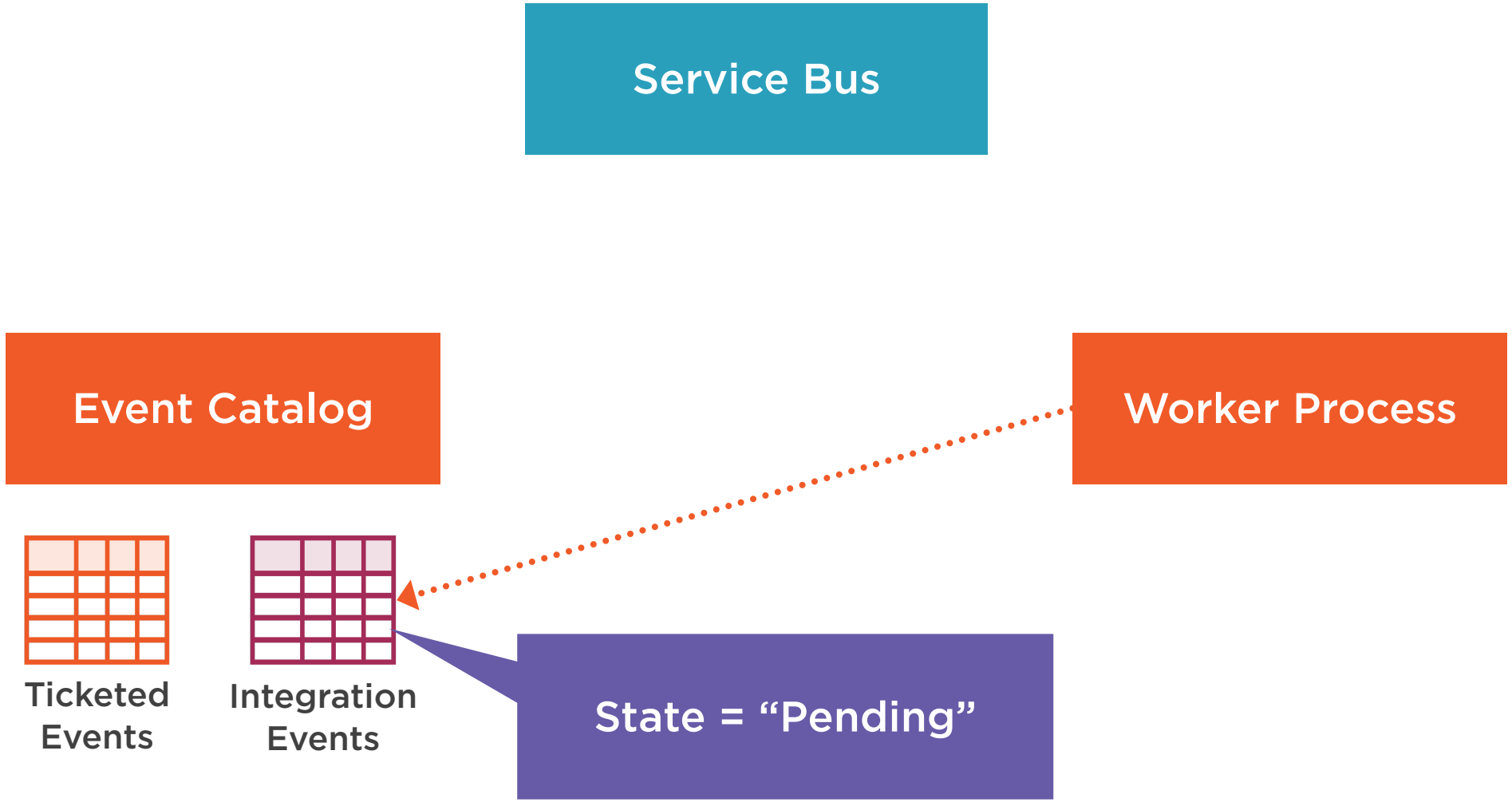
Event Sourcing Implementation



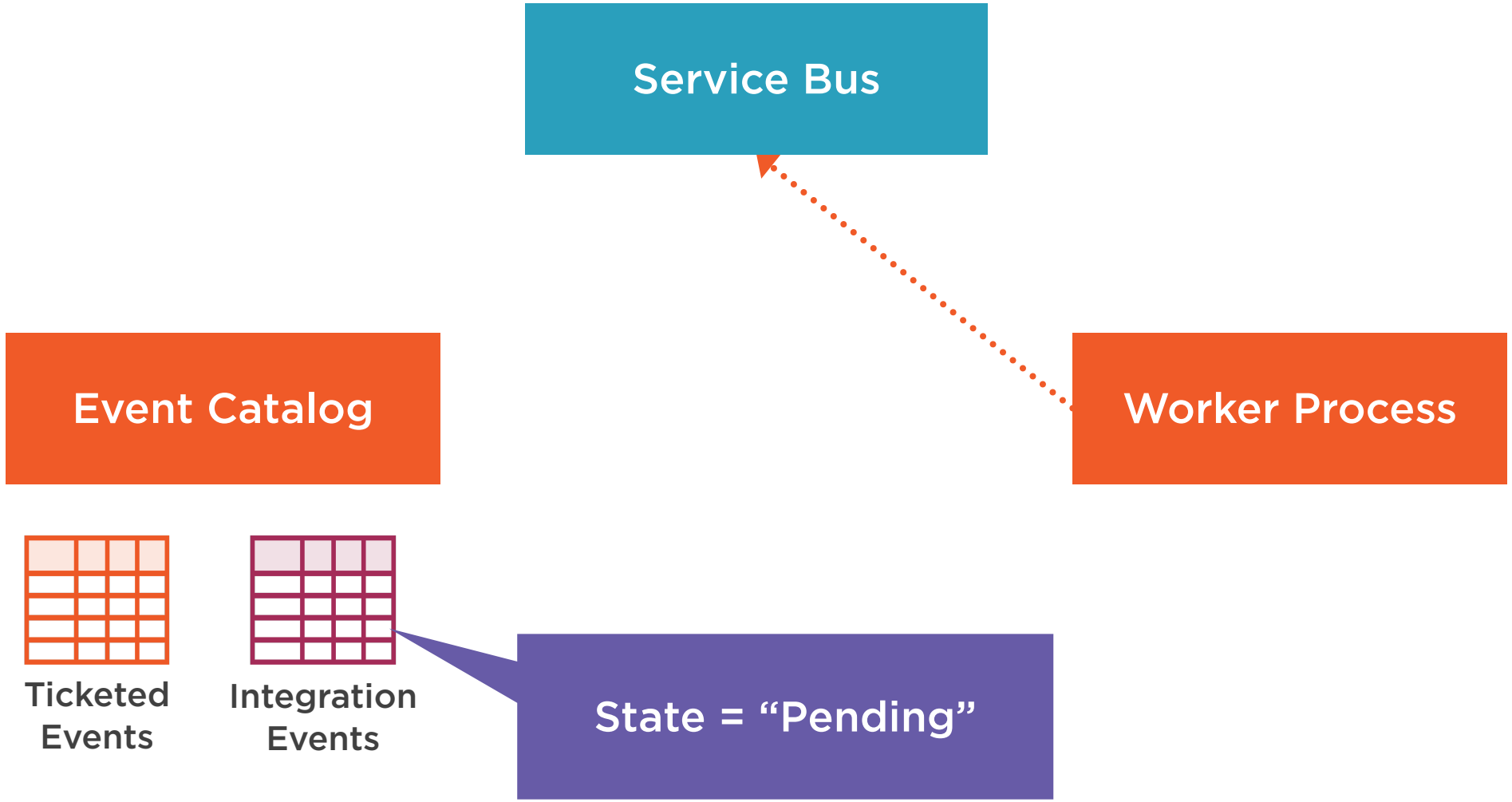
Event Sourcing Implementation



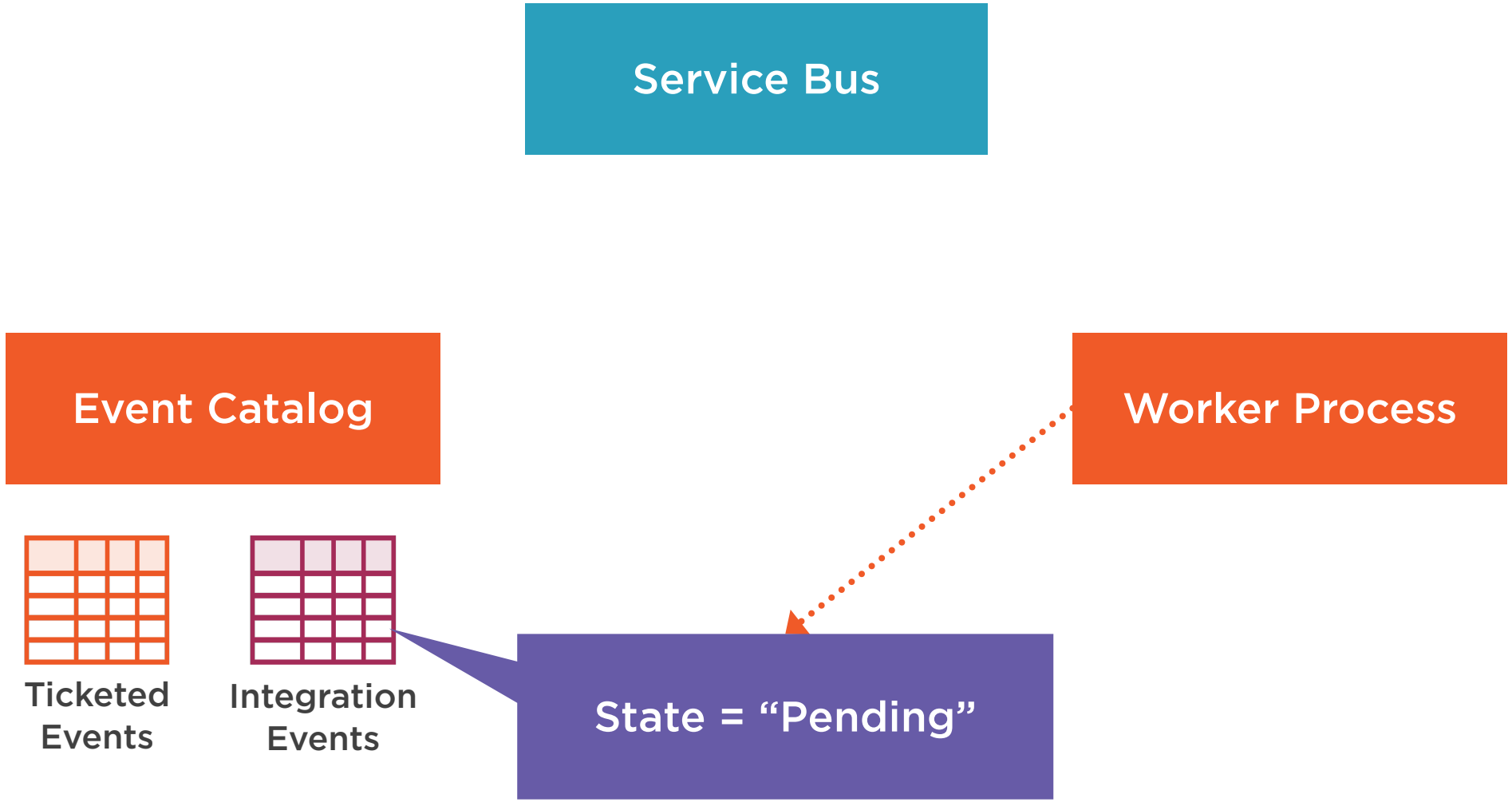
Event Sourcing Implementation



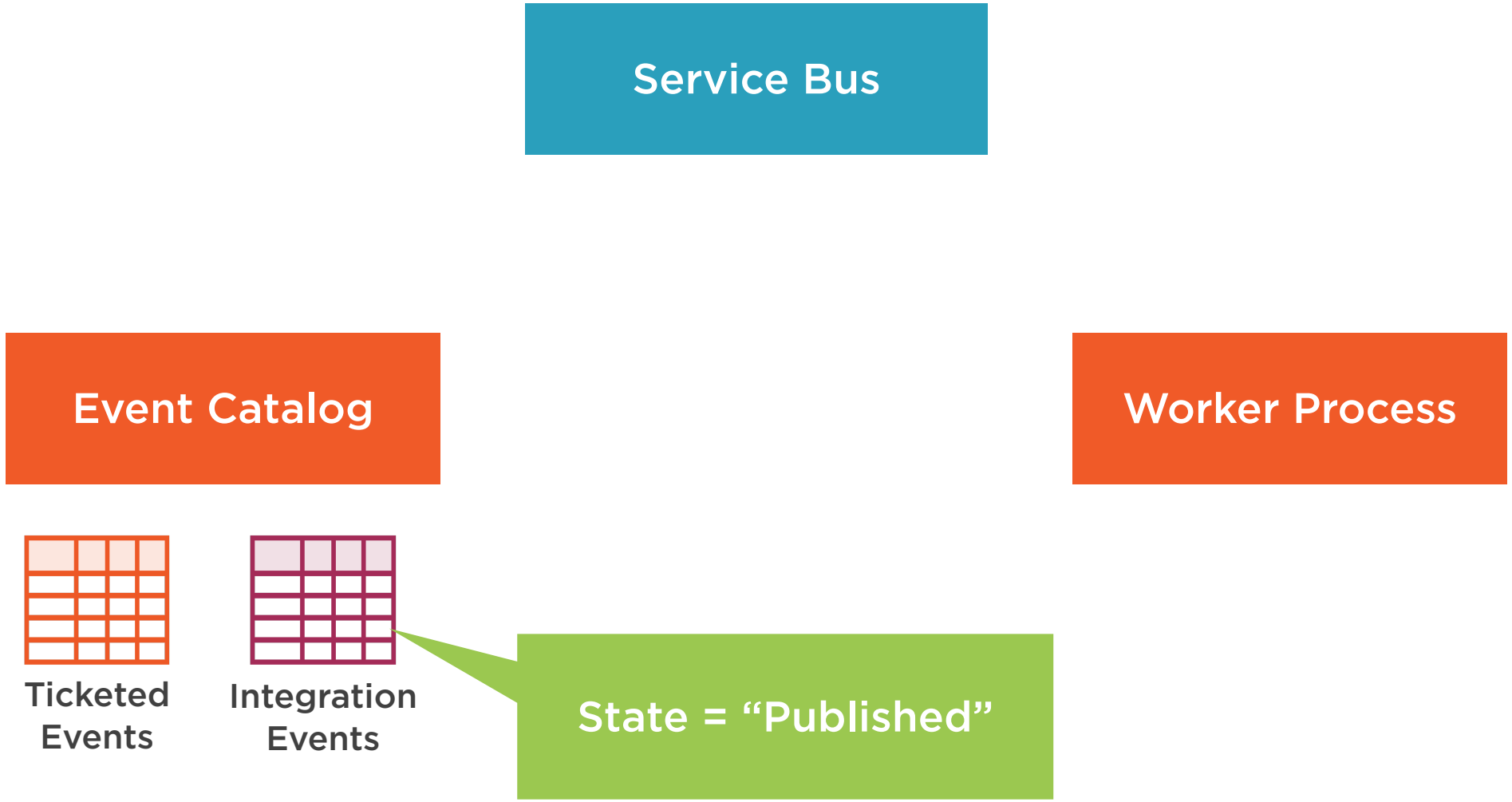
Event Sourcing Implementation



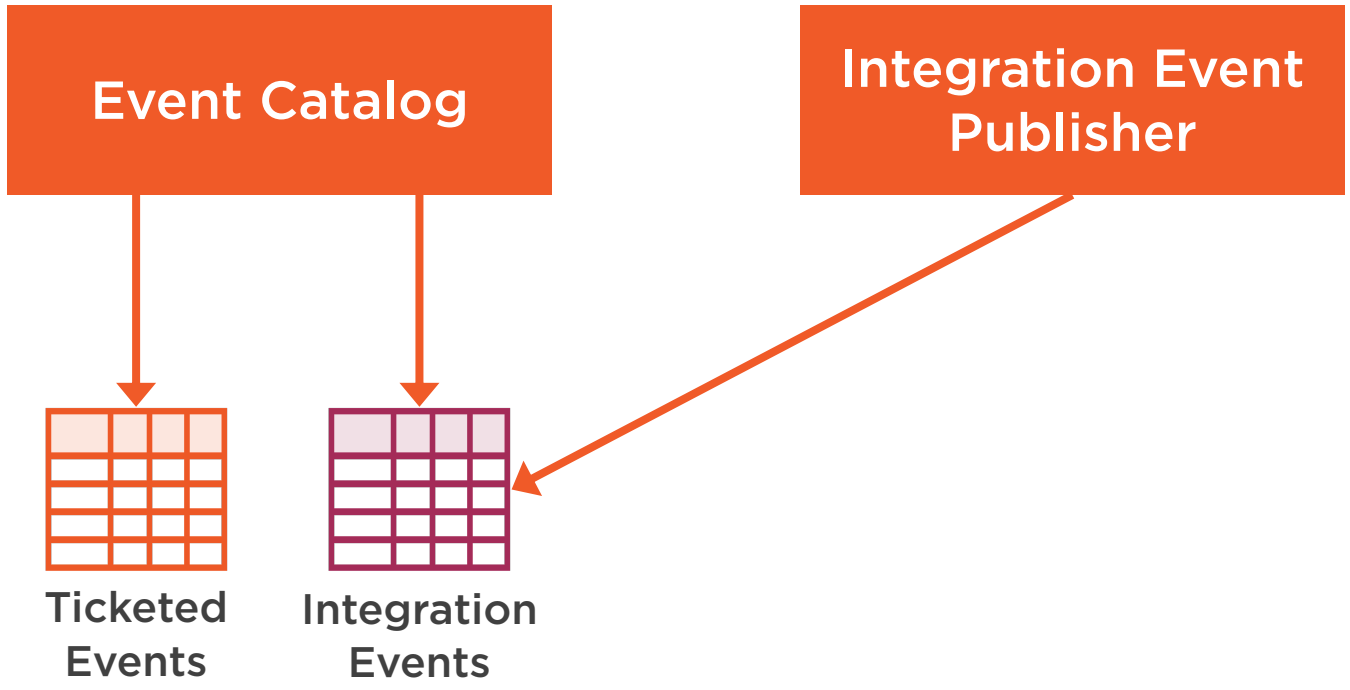
Event Sourcing Implementation



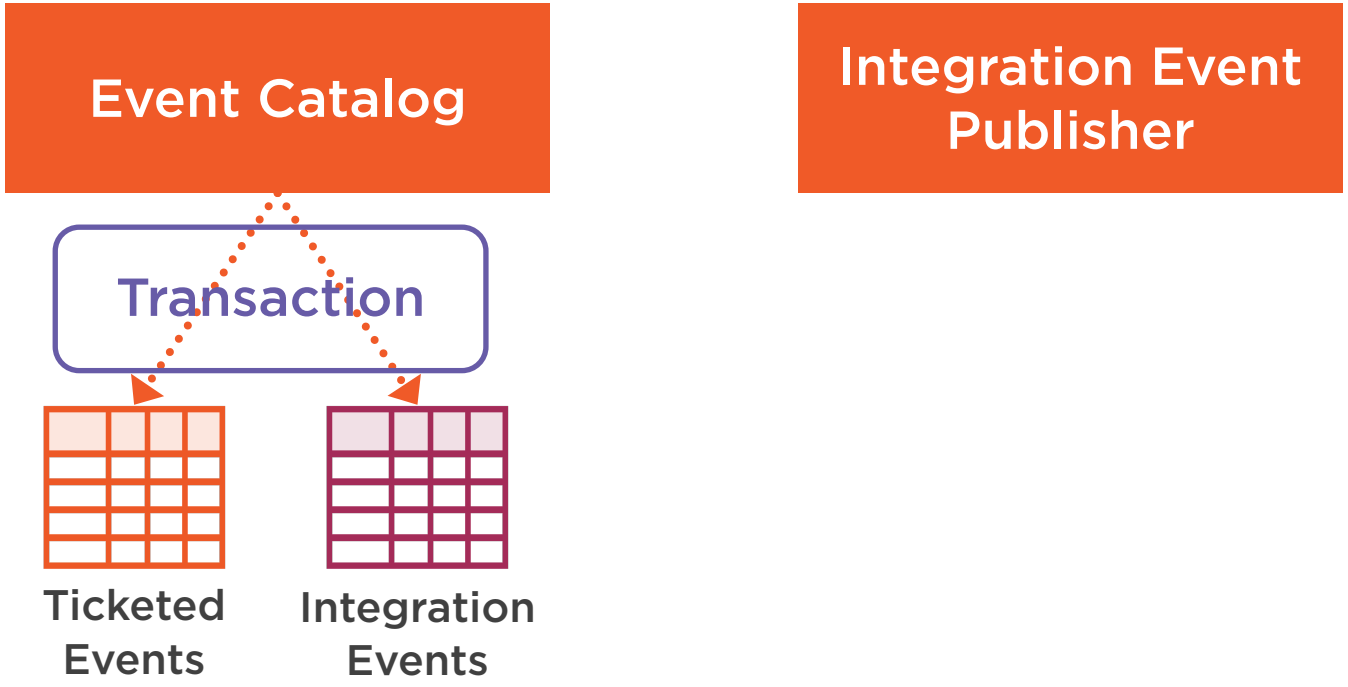
Event Sourcing Implementation



Event Sourcing Implementation



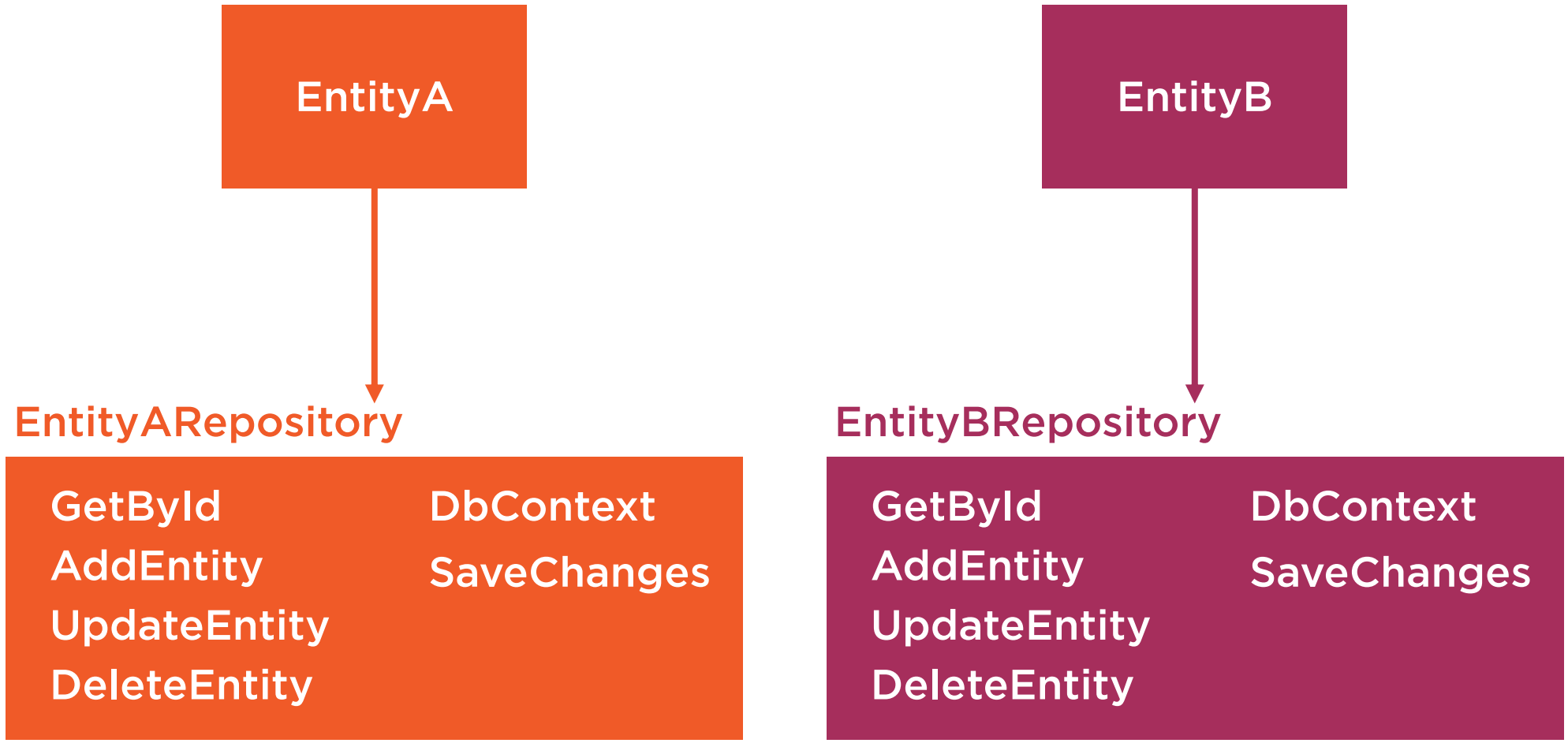
Event Sourcing Implementation

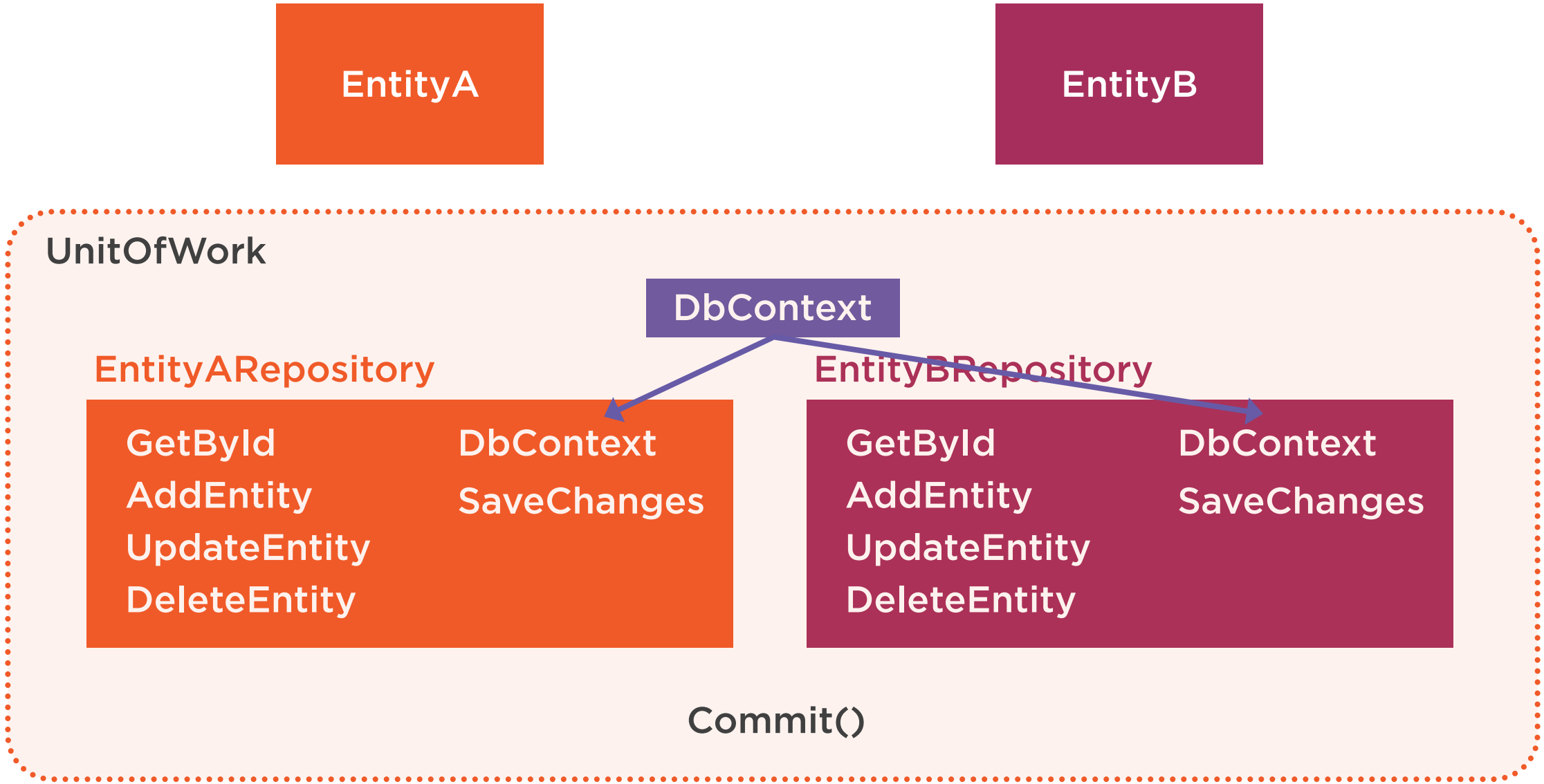


Implementing the Unit of Work Pattern

Entity Framework Core

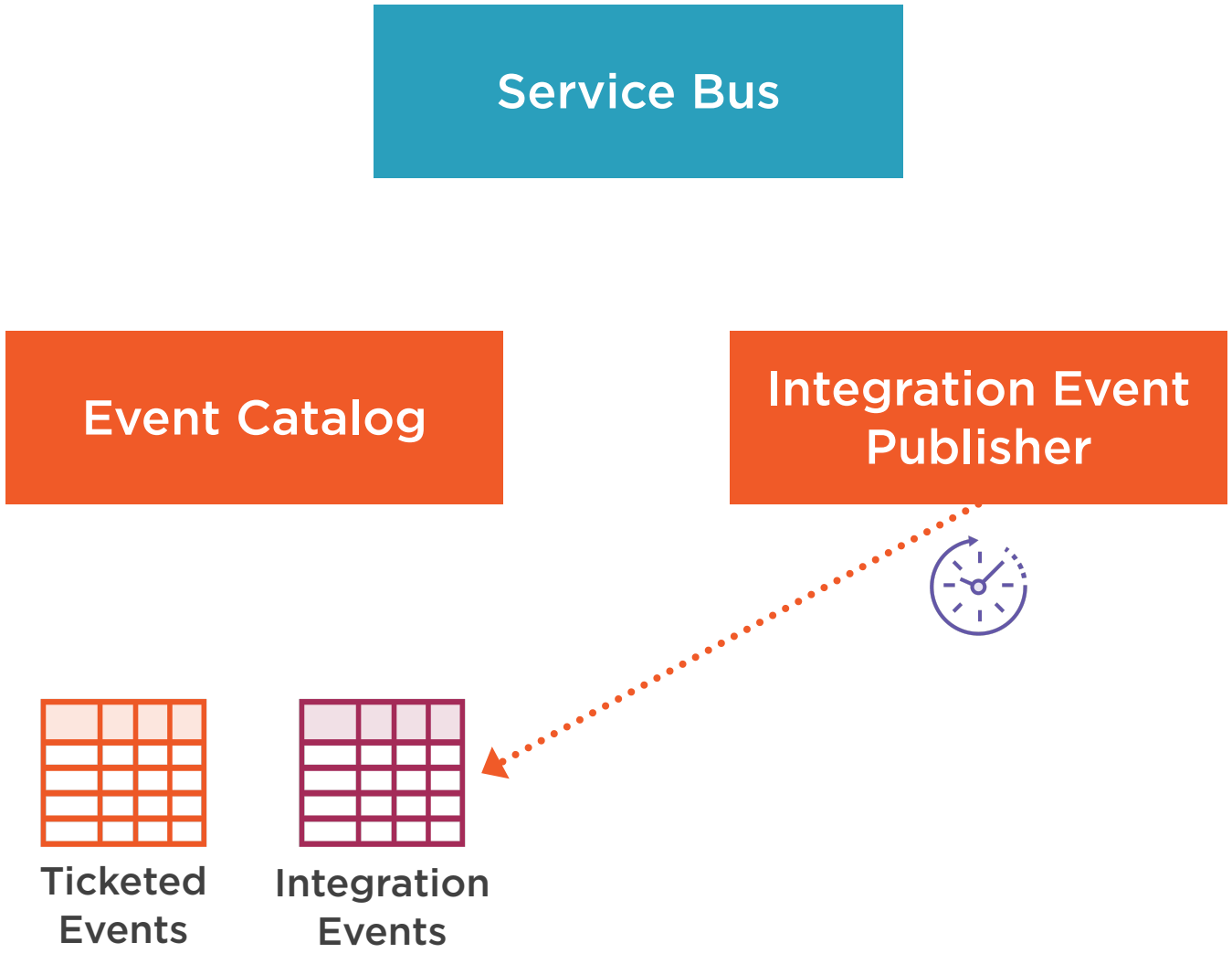
- DbContext.SaveChanges();
- Operations occur within a transaction
- Multiple contexts can share transaction


EntityA
EntityB
EntityARepository
GetById
AddEntity
UpdateEntity
DeleteEntity
DbContext
SaveChanges
EntityBRepository
GetById
AddEntity
UpdateEntity
DeleteEntity
DbContext
SaveChanges

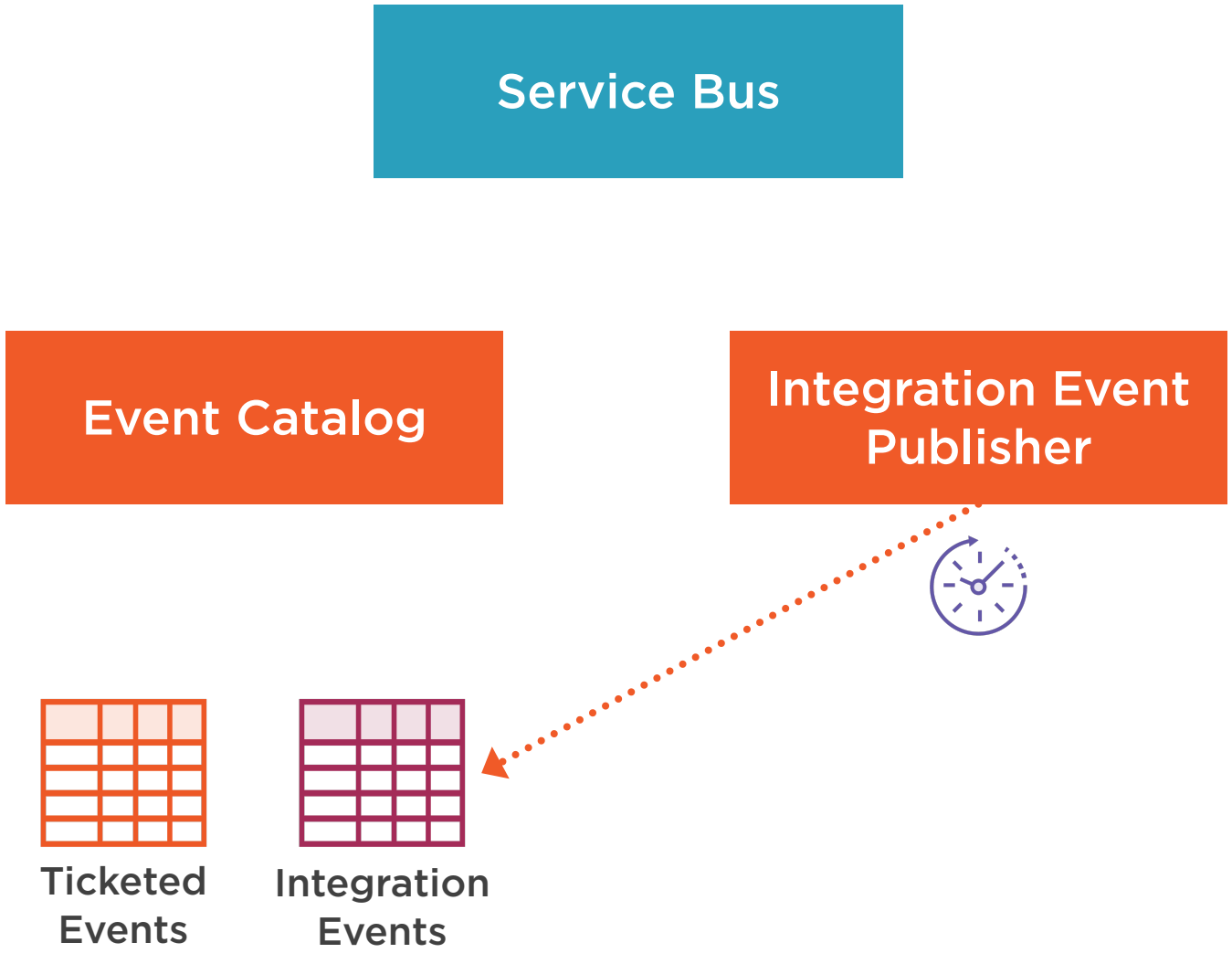

EntityA
EntityB
UnitOfWork
DbContext
EntityARepository
GetById
AddEntity
UpdateEntity
DeleteEntity
DbContext
SaveChanges
EntityBRepository
GetById
AddEntity
UpdateEntity
DeleteEntity
DbContext
SaveChanges
Commit()

Publishing Events with the Event Sourcing Pattern

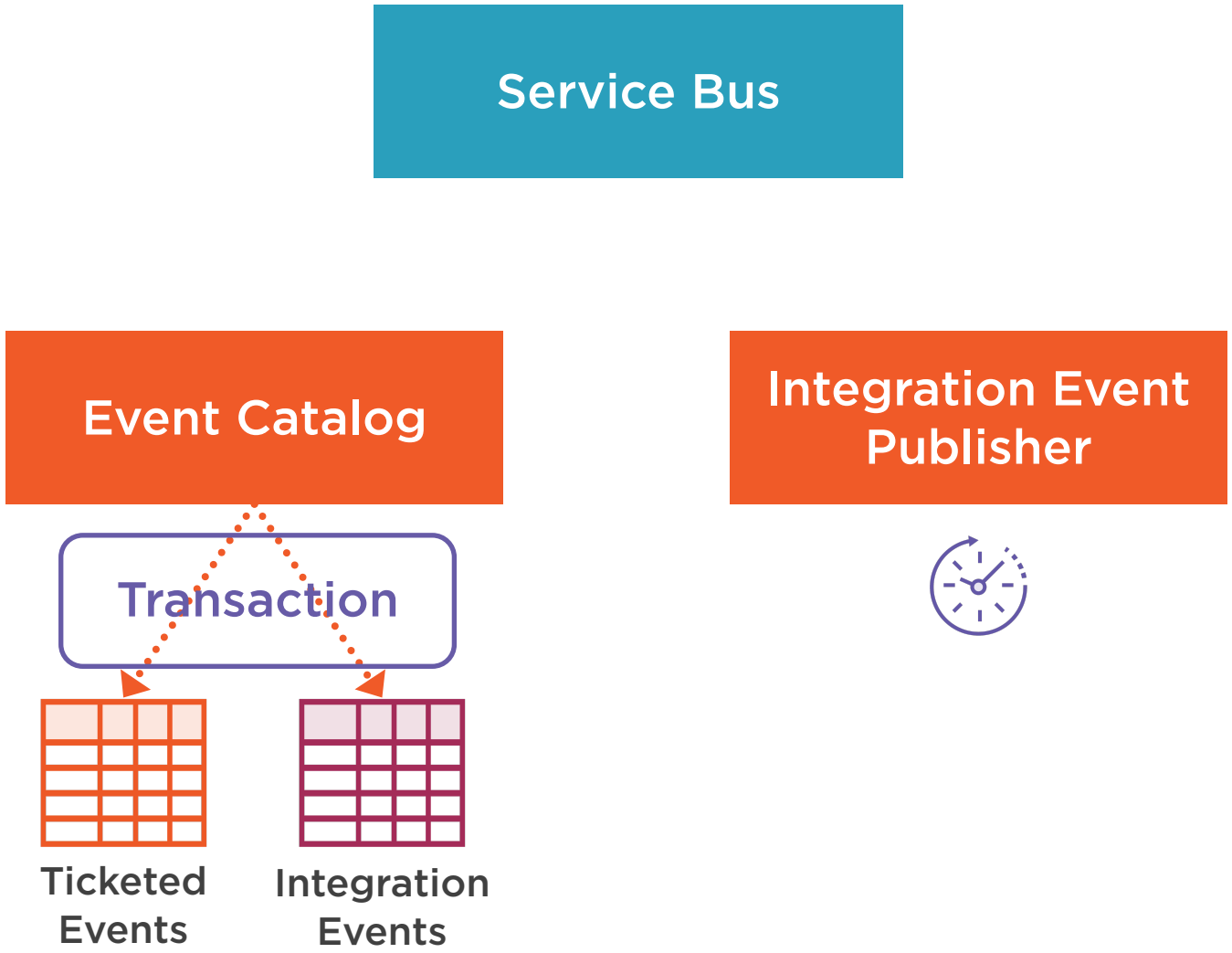
Event Sourcing Implementation



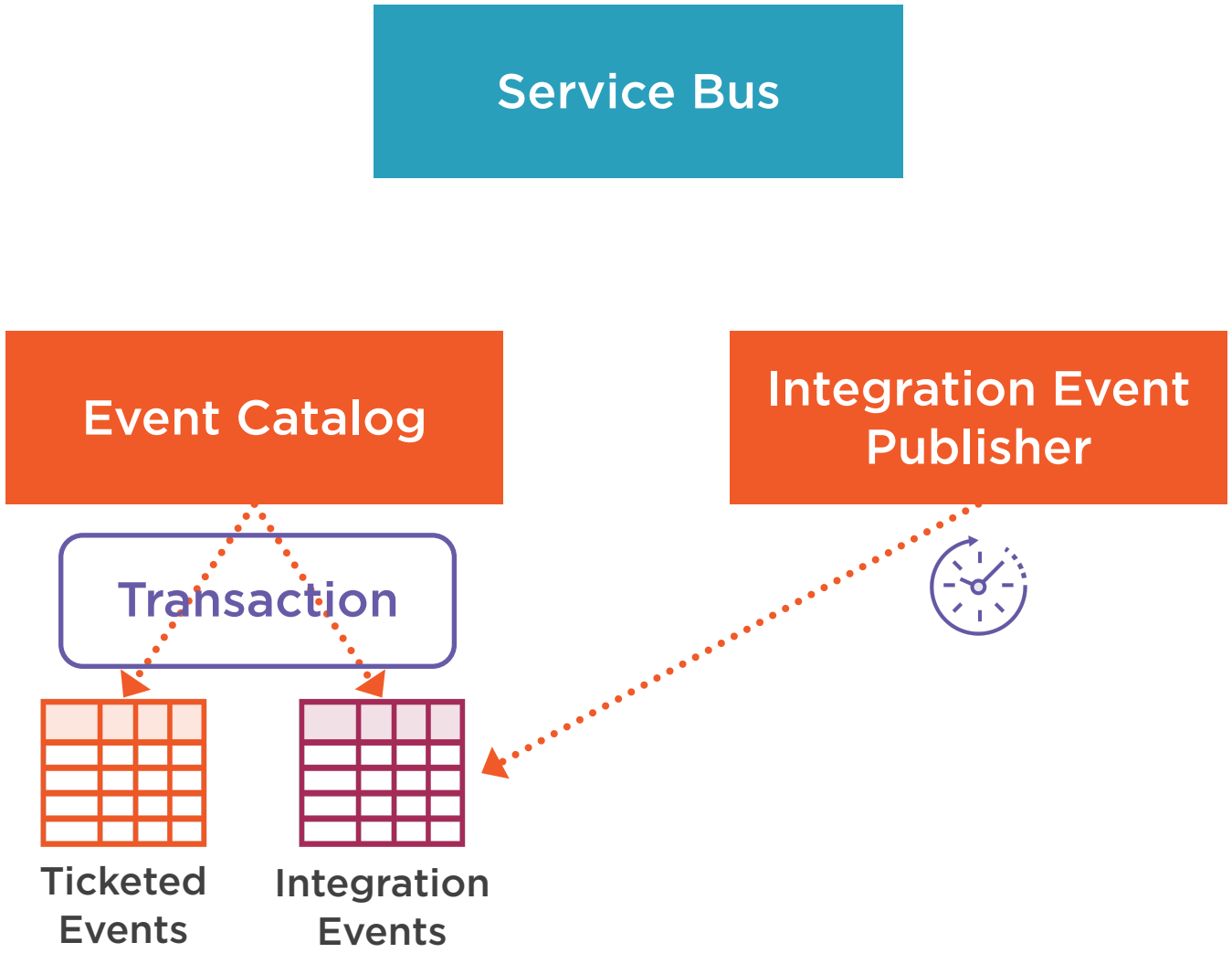
Event Sourcing Implementation



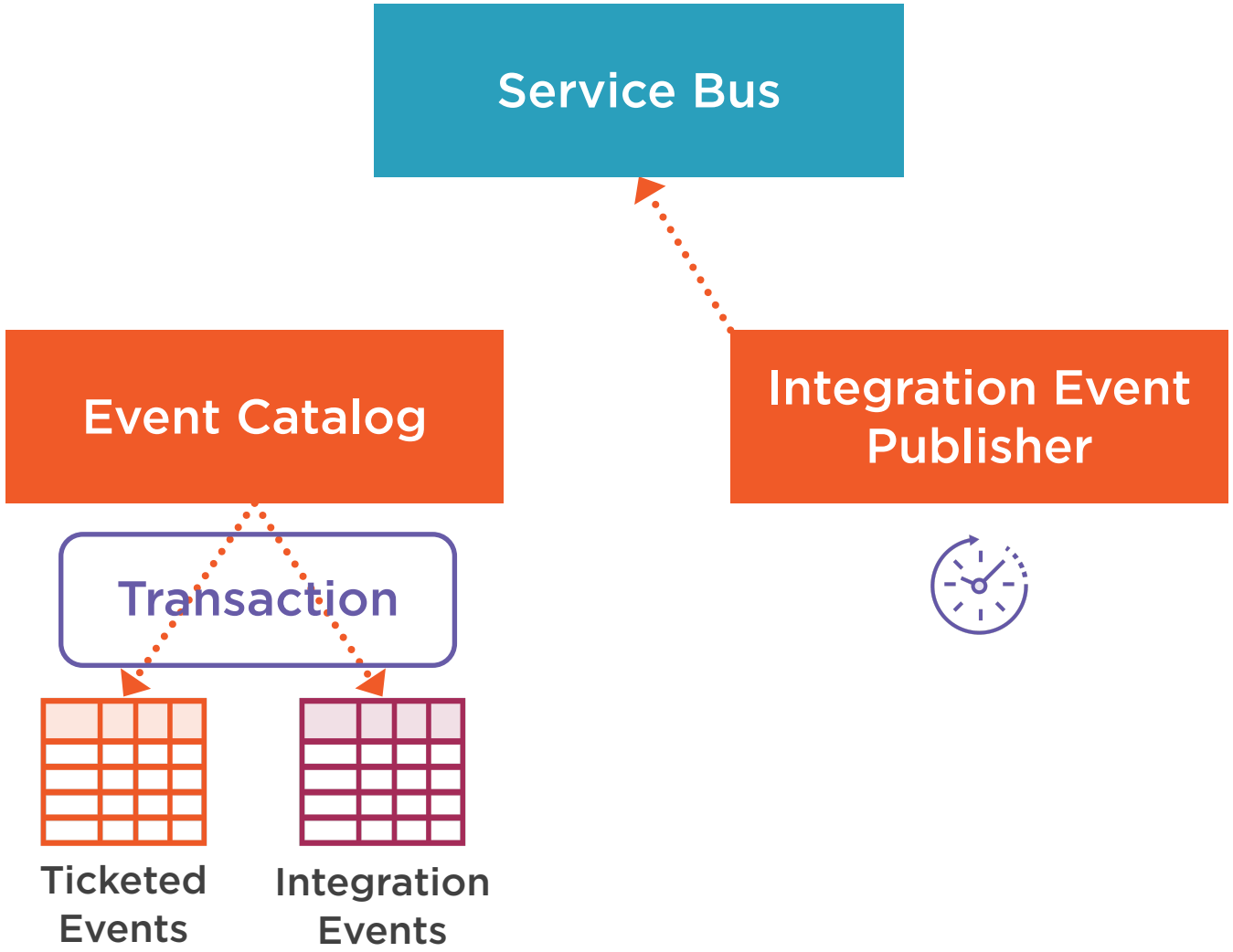
Event Sourcing Implementation



Event Sourcing Implementation

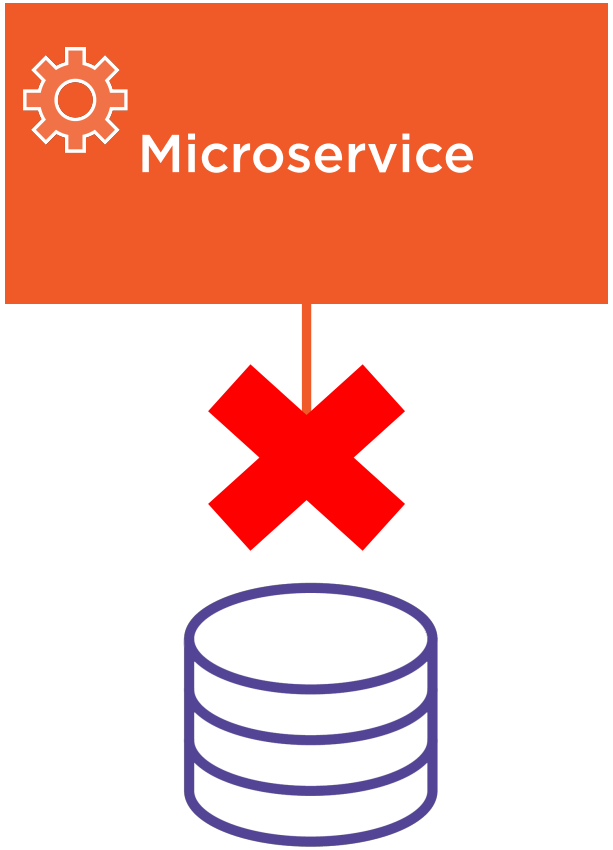


Event Sourcing Implementation

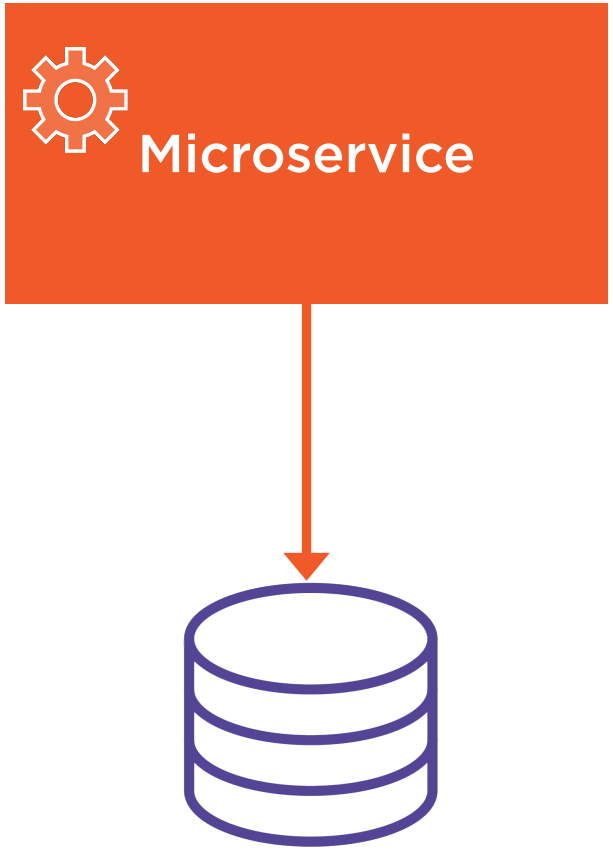


EF Core Connection Resiliency

Database Connection Failure



Database Connection Failure



EF Core connection resiliency

Automatically retries failed database commands

Supply an execution strategy

Built-in execution strategies for DB providers

EF Core Connection Resiliency

SQL Server Provider Execution Policy

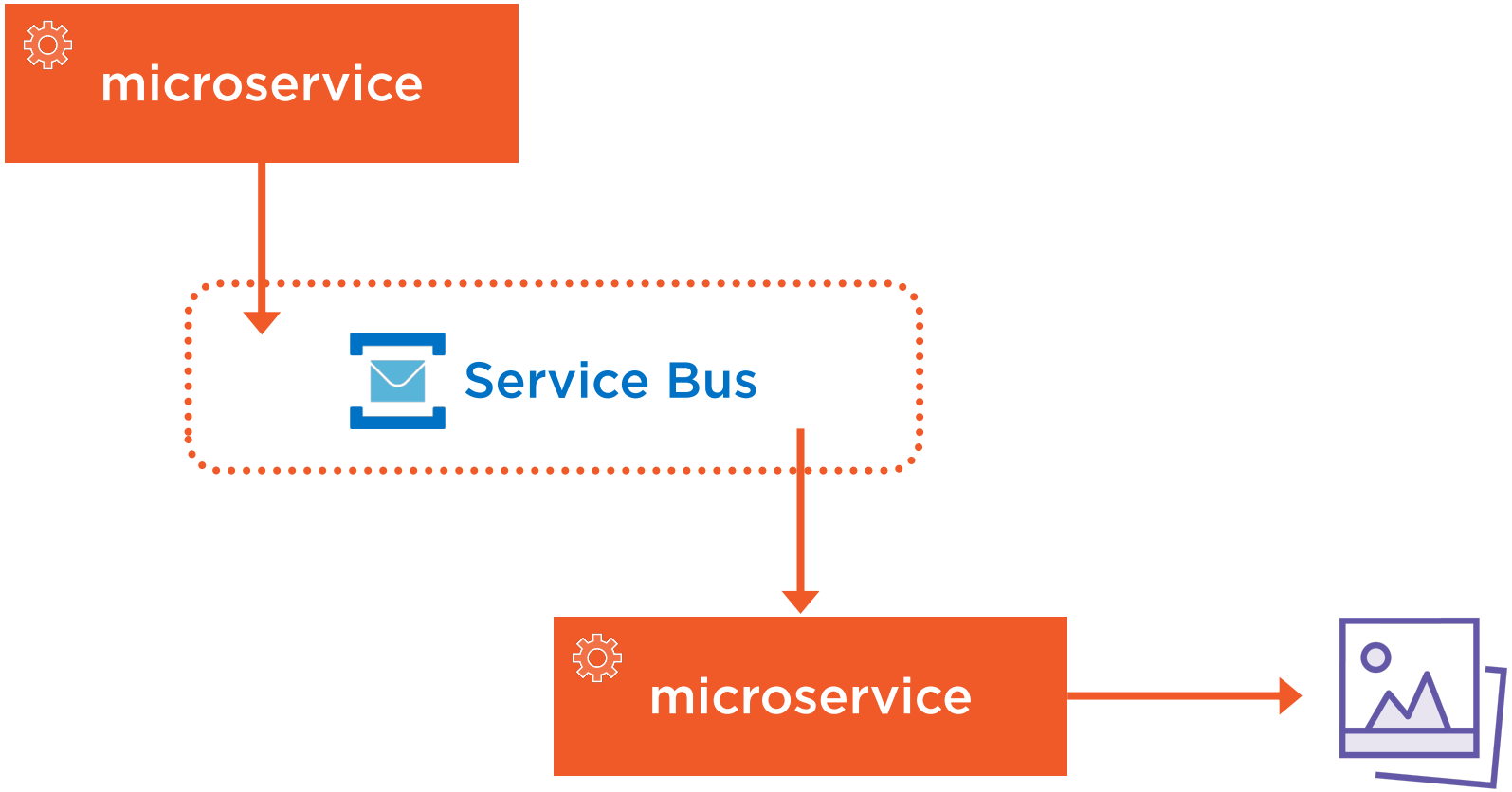
```
services.AddDbContext<EventCatalogDbContext>(
                options => options.UseSqlServer(
                    Configuration.GetConnectionString("DefaultConnection"),
                    providerOptions => providerOptions.EnableRetryOnFailure()));
```

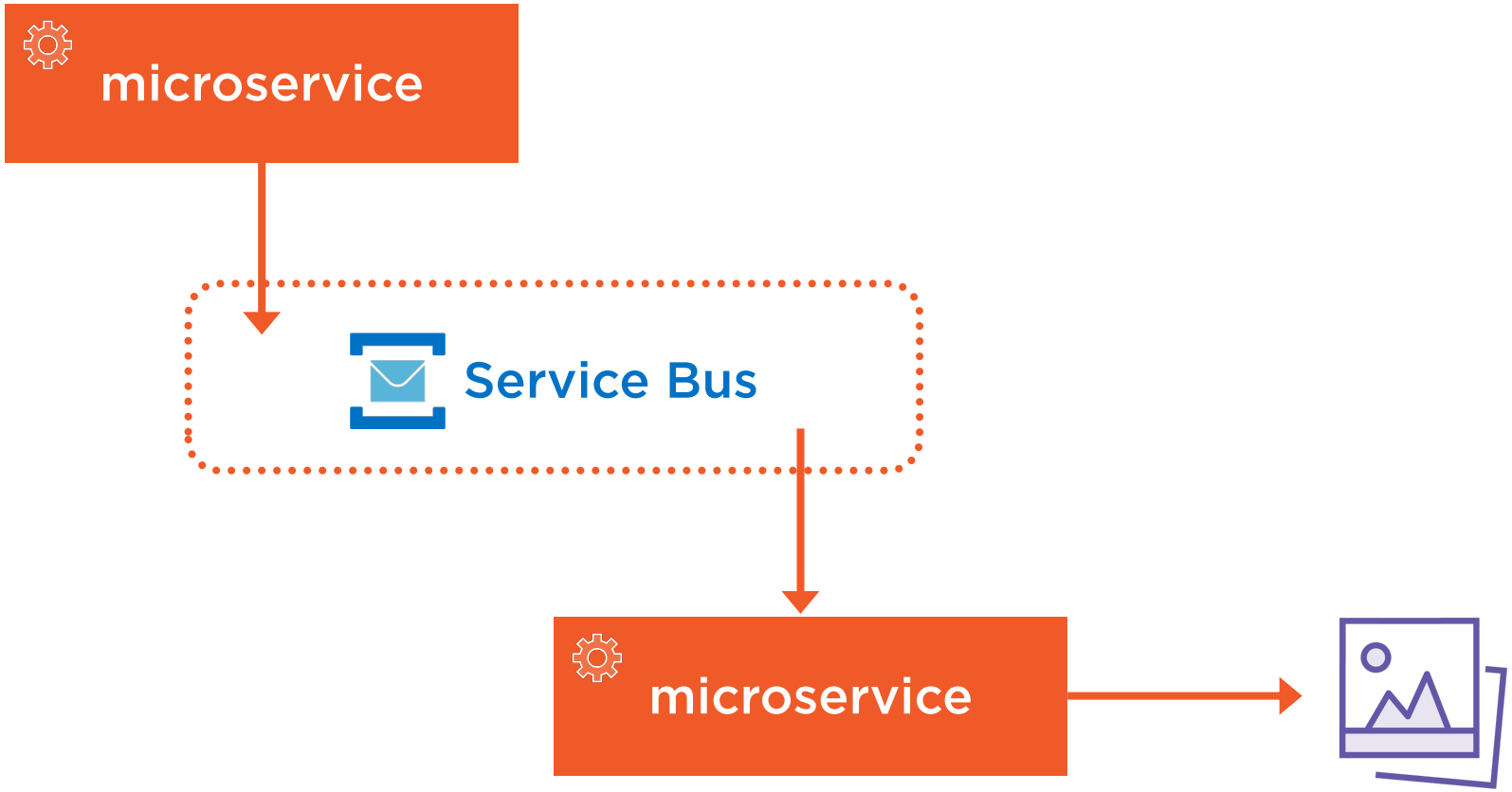
EF Core Connection Resiliency

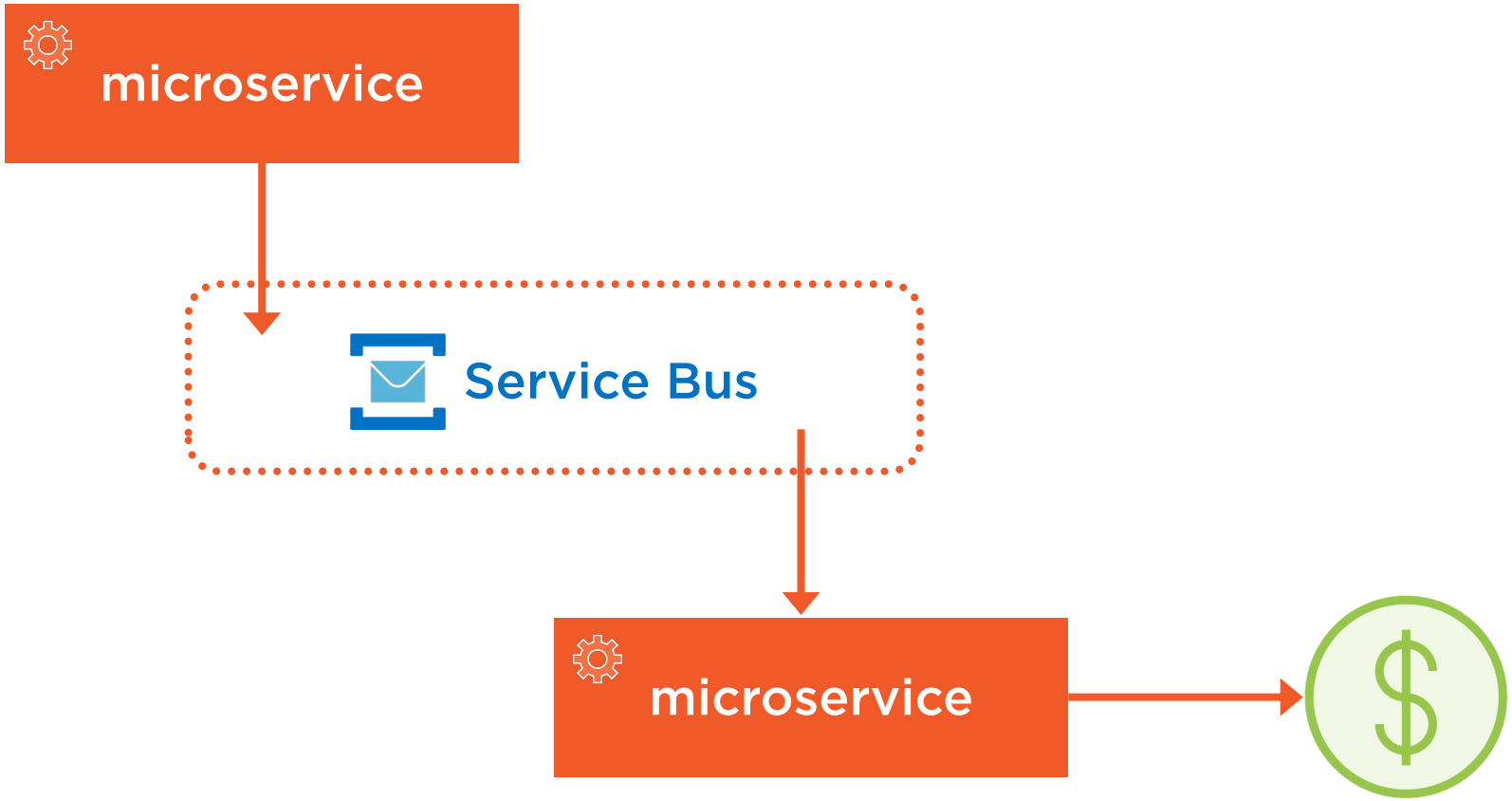
SQL Server Provider Execution Policy (with parameters)

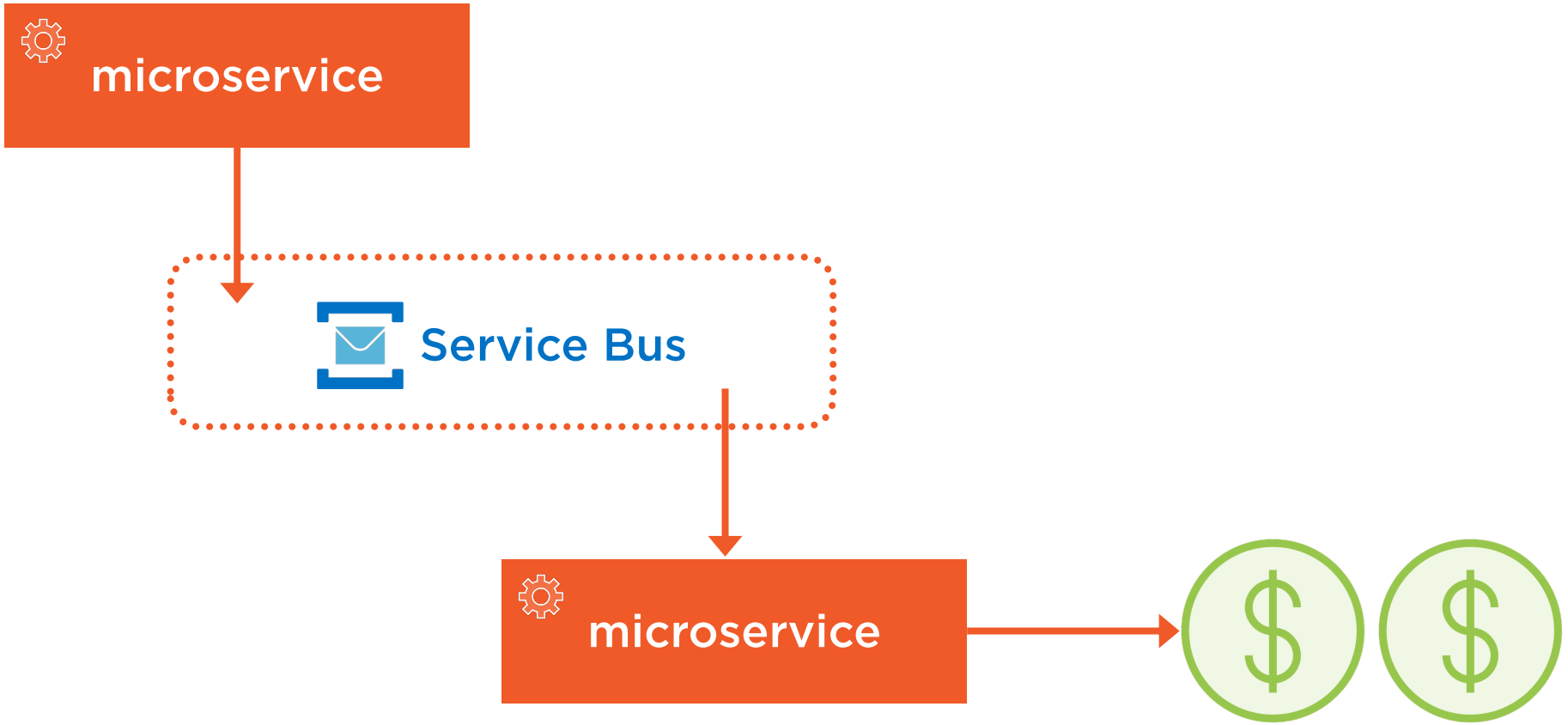
```
services.AddDbContext<EventCatalogDbContext>(
            options => options.UseSqlServer(
                Configuration.GetConnectionString("DefaultConnection"),
                providerOptions => providerOptions.EnableRetryOnFailure(
                    maxRetryCount: 10,
                    maxRetryDelay: TimeSpan.FromSeconds(30),
                    errorNumbersToAdd: null)
                )
            );
```

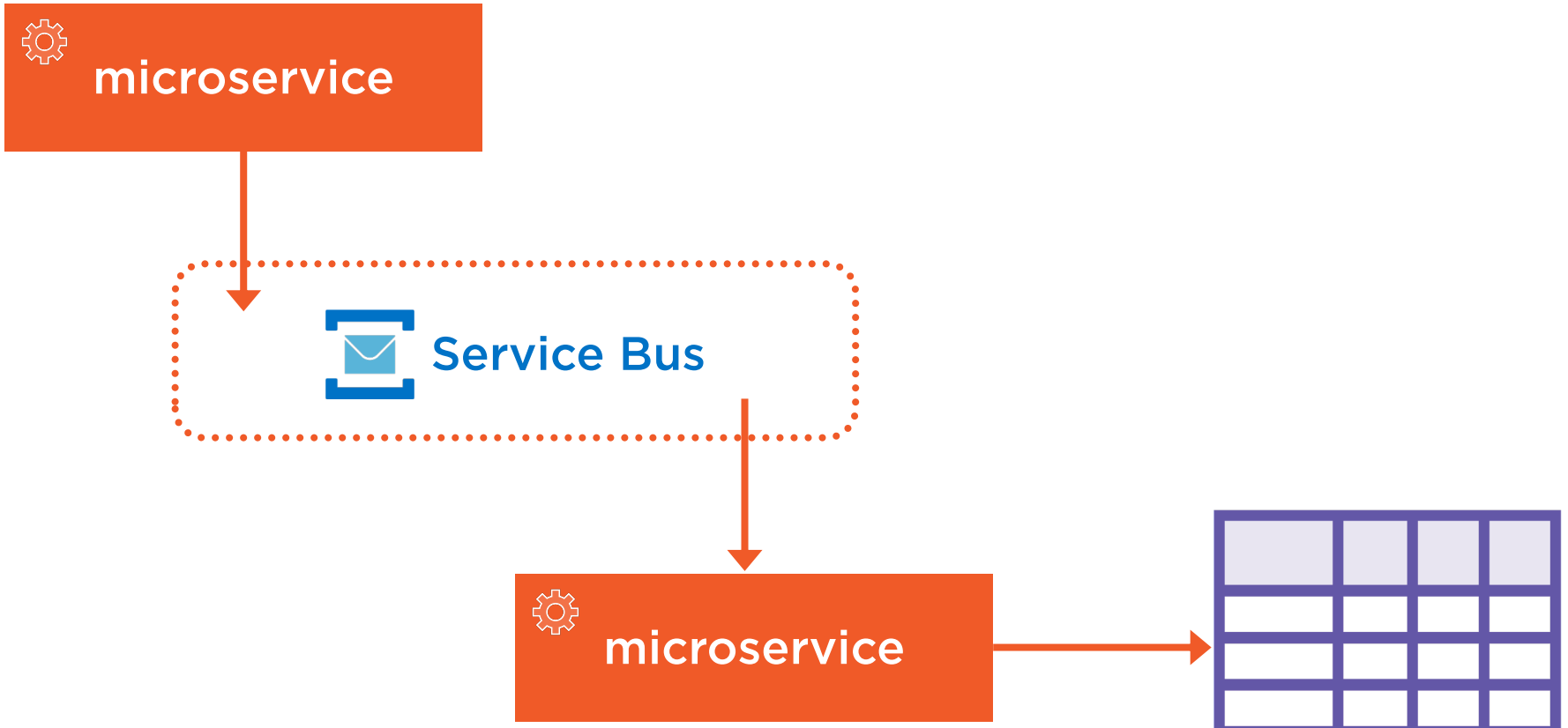
Handling Duplicate Messages

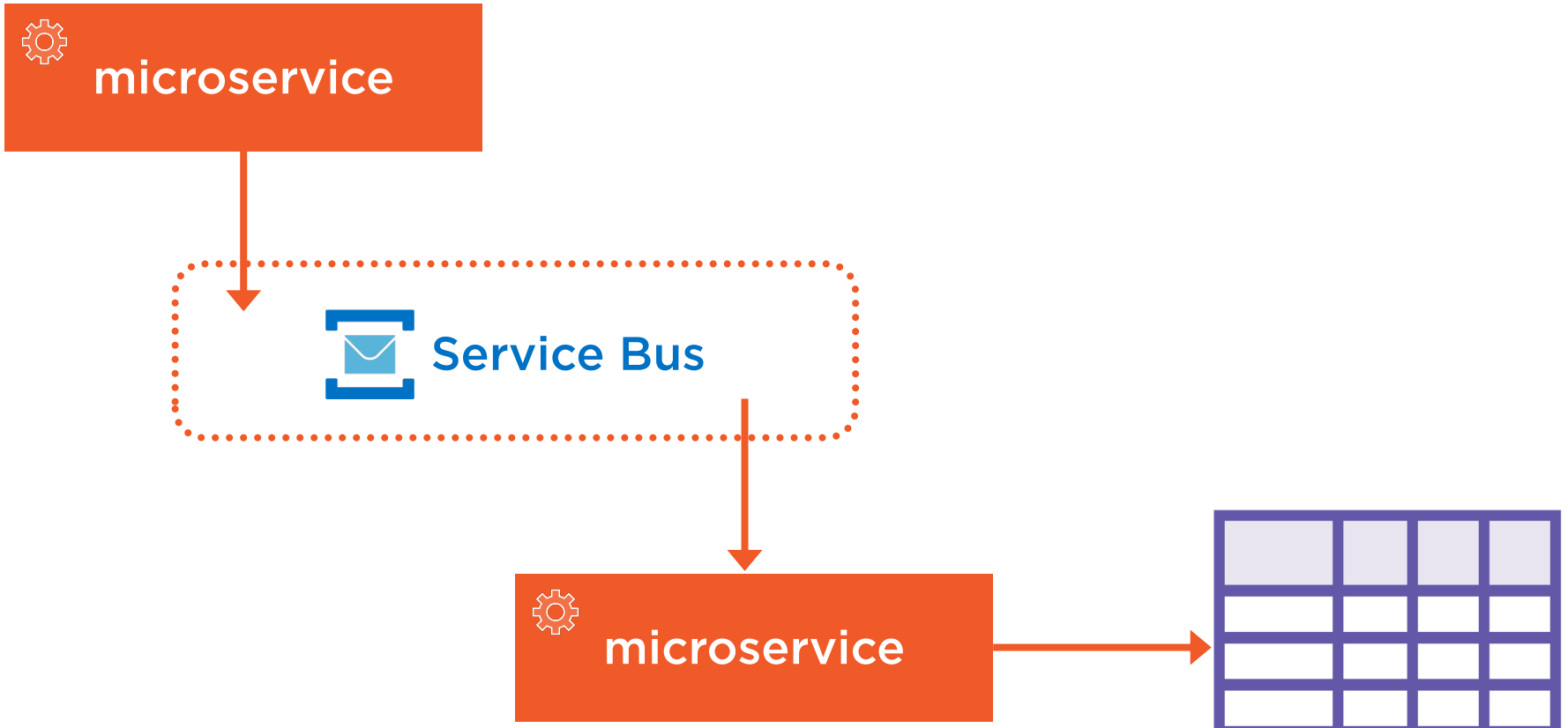

microservice
Service Bus
microservice

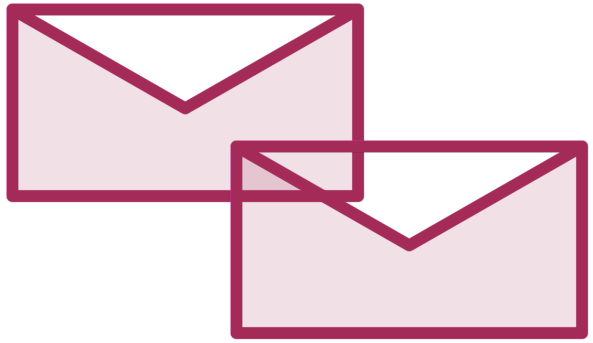

microservice
Service Bus
microservice


microservice
Service Bus
microservice


microservice
Service Bus
microservice


microservice
Service Bus
microservice


microservice
Service Bus
microservice



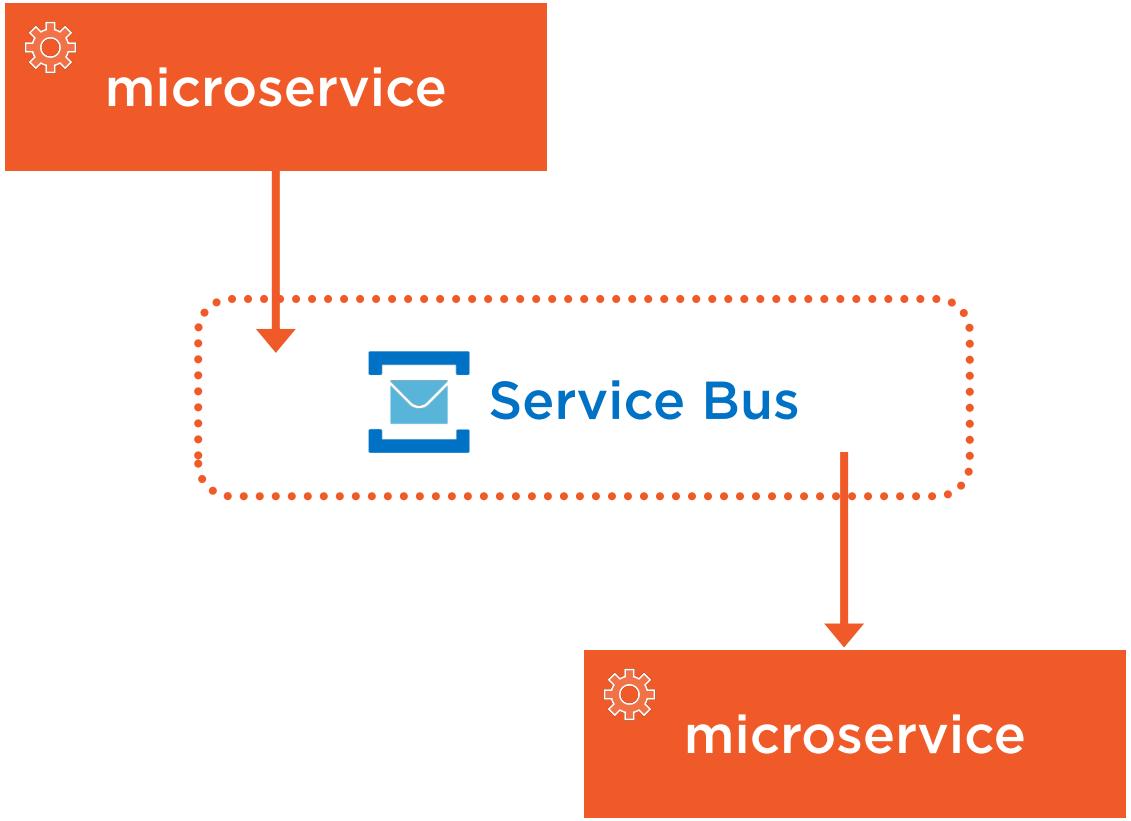
Handling Duplicate Messages

App fails after sending message

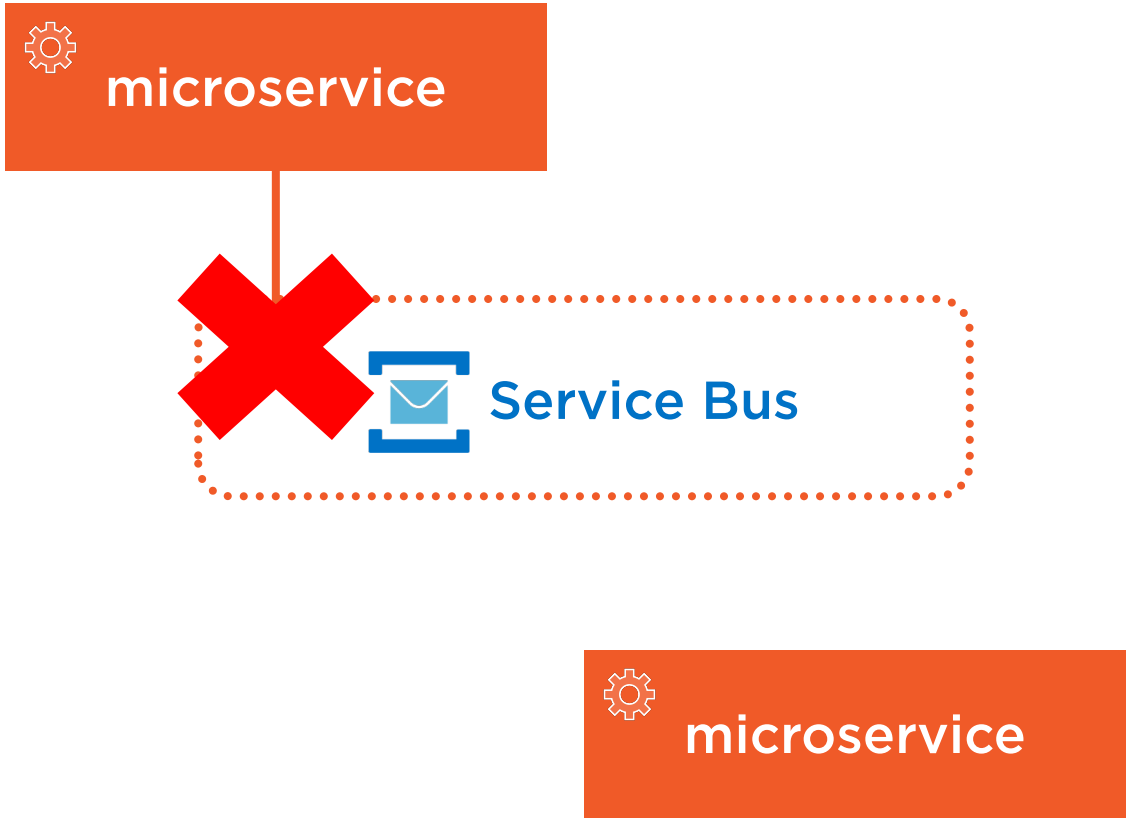
- Restarts and resends message

Acknowledgement not received

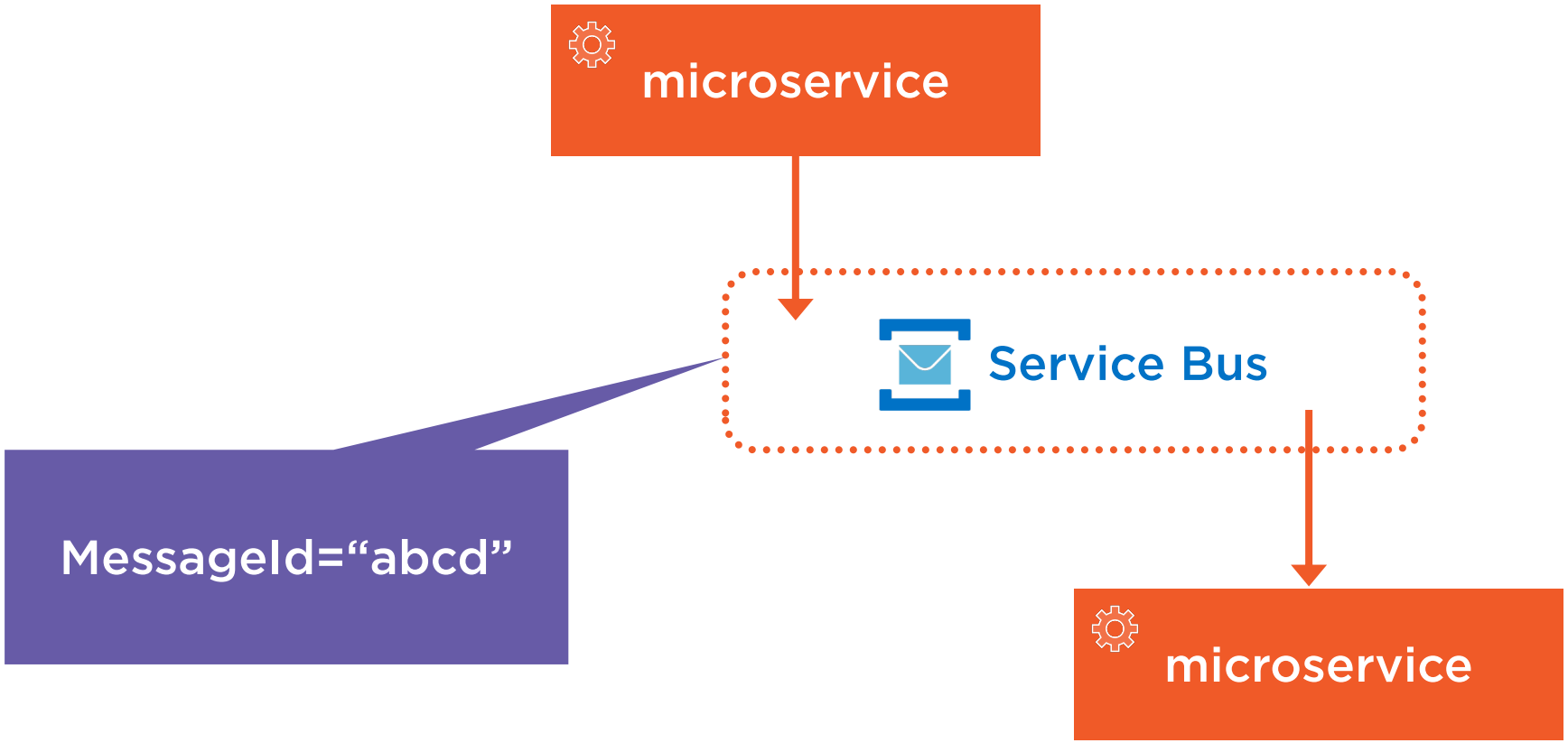
Detect Duplicates at Service Bus



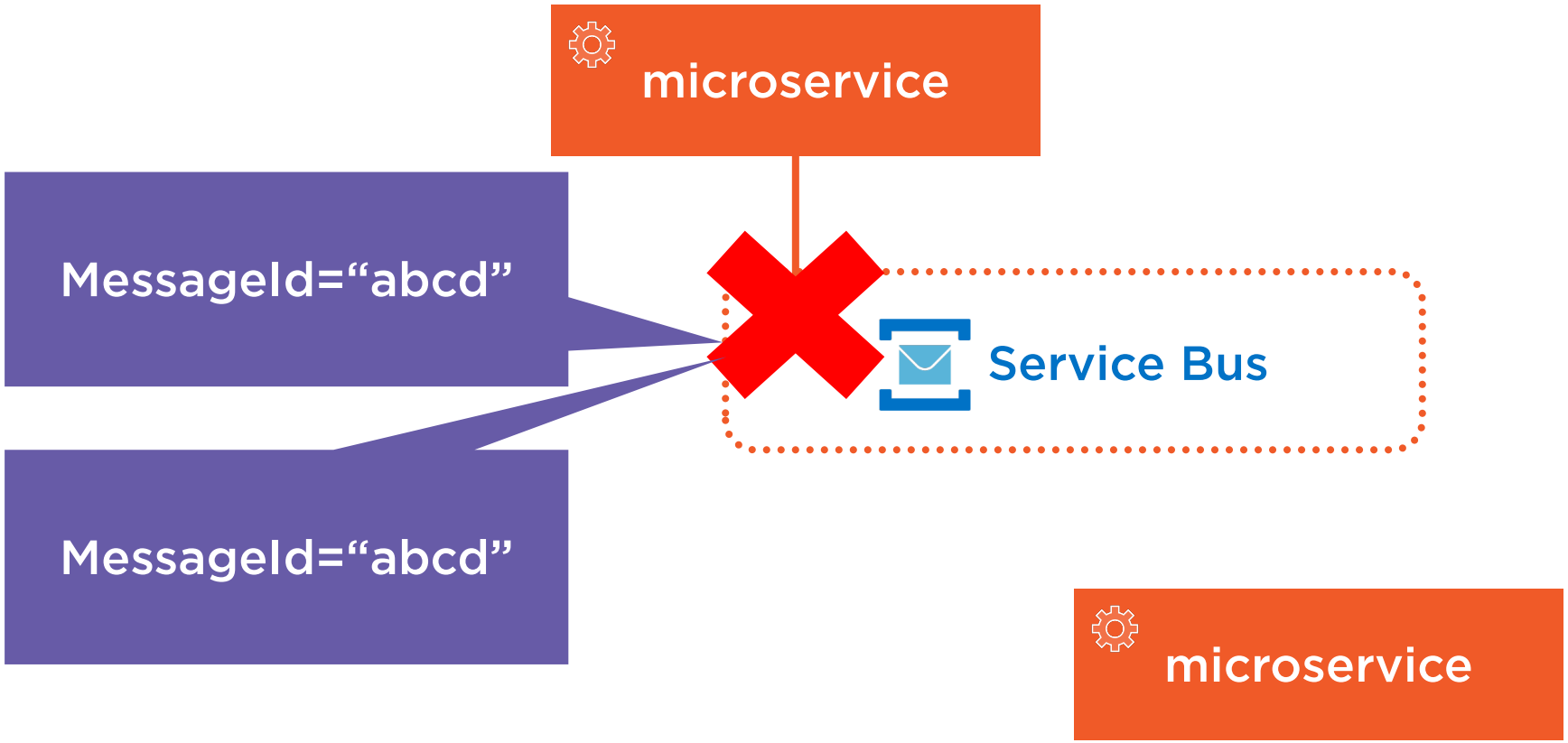
Detect Duplicates at Service Bus



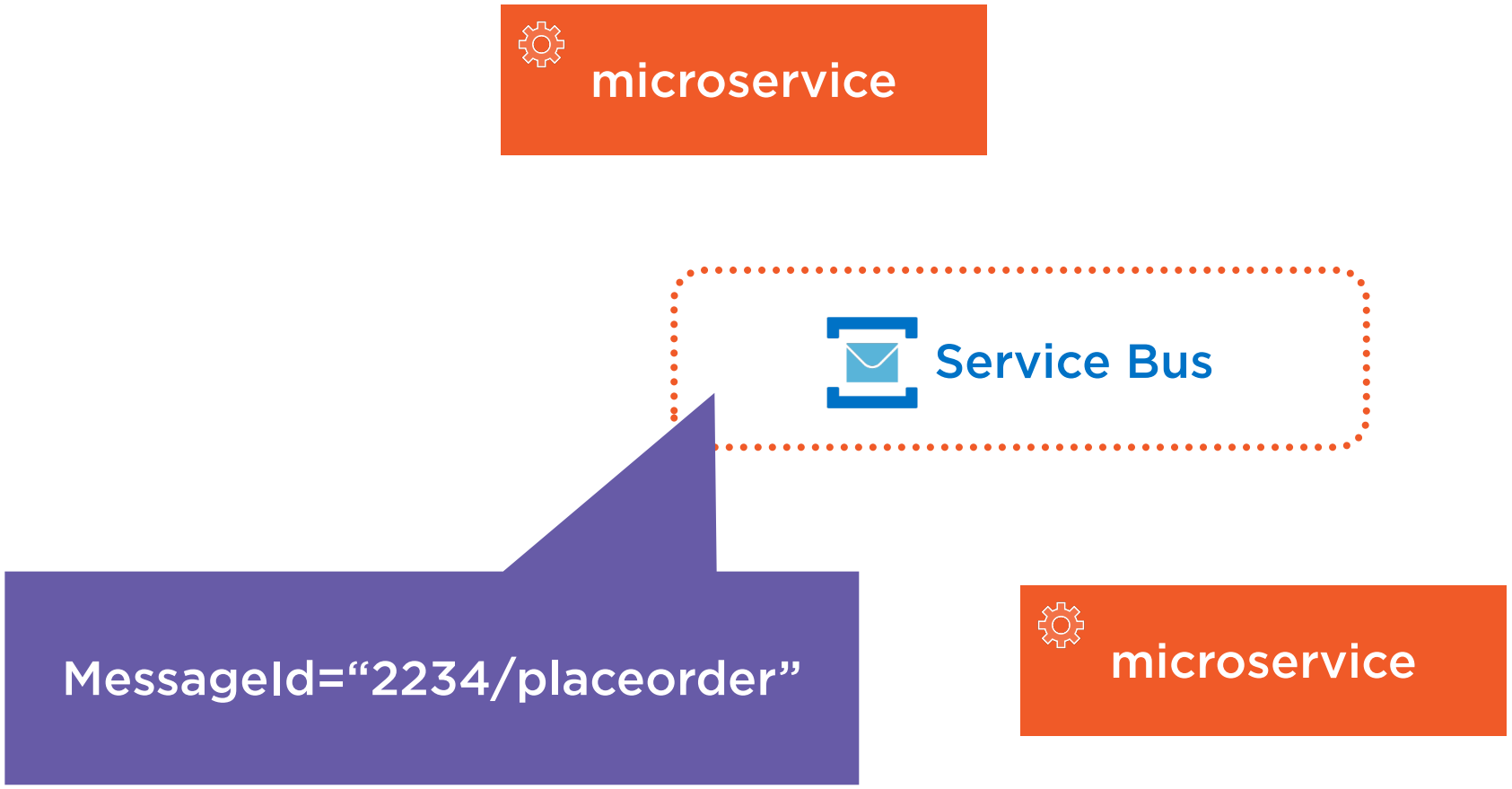
Detect Duplicates at Service Bus



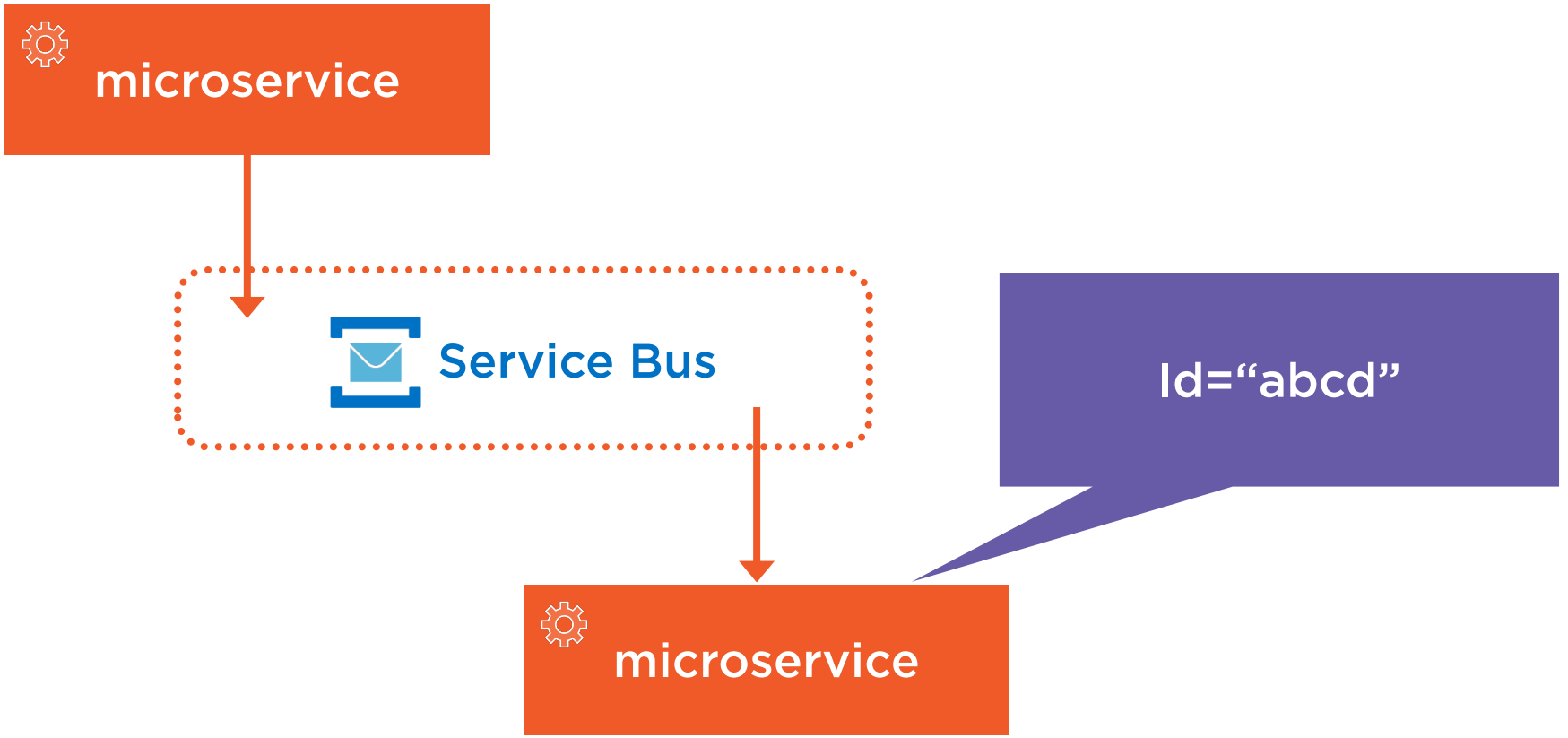
Detect Duplicates at Service Bus



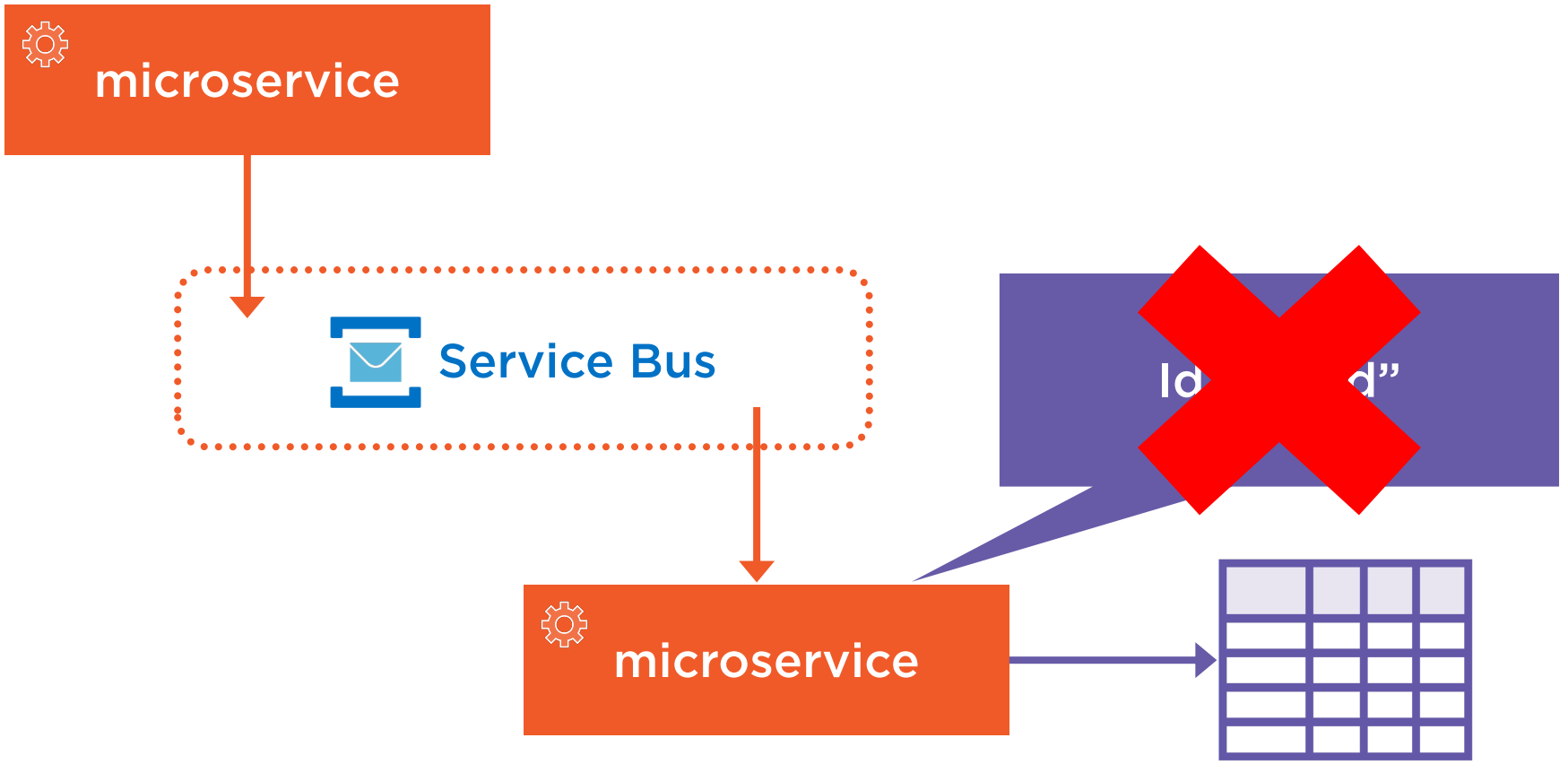
Detect Duplicates at Service Bus



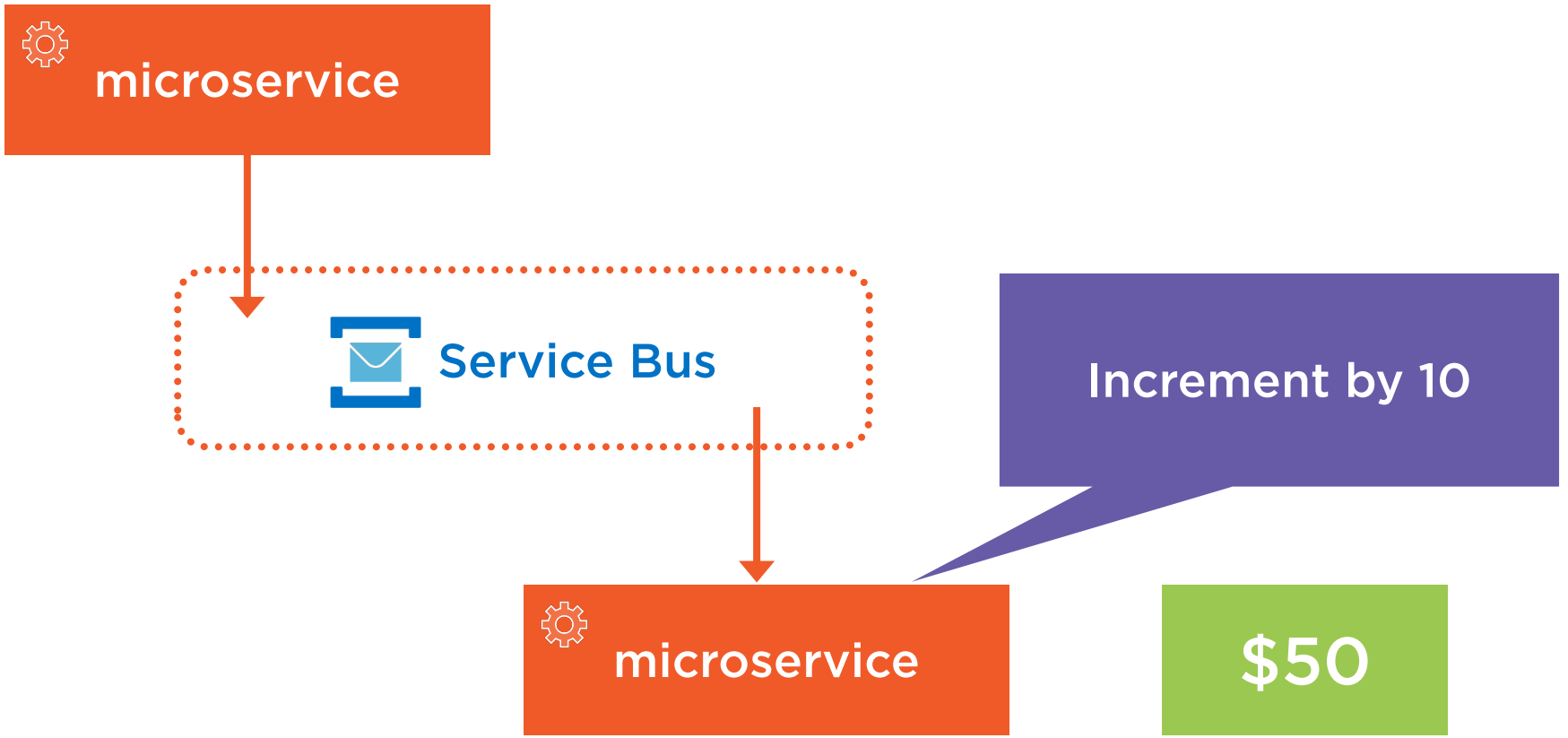
Detect Duplicates at Receiver



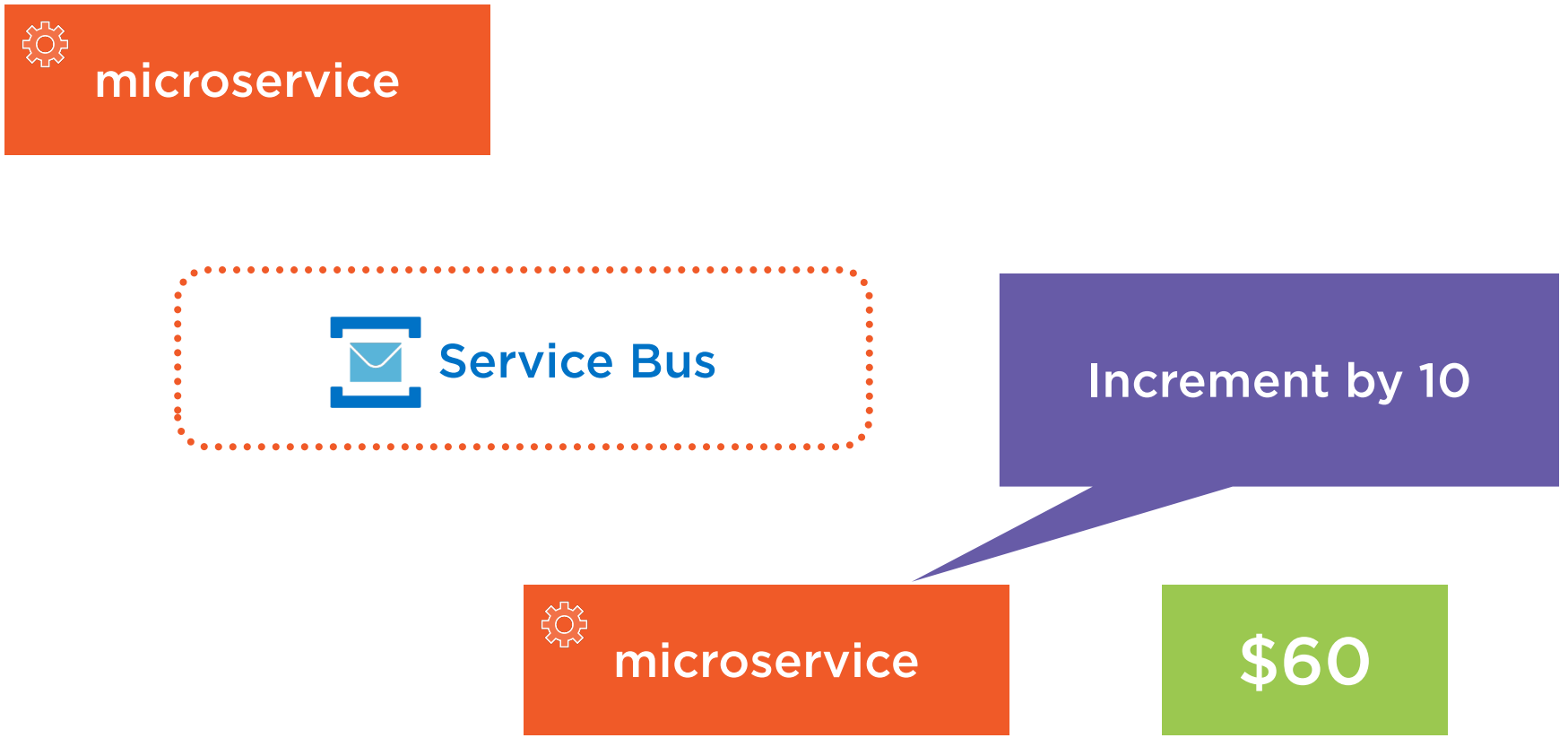
Detect Duplicates at Receiver



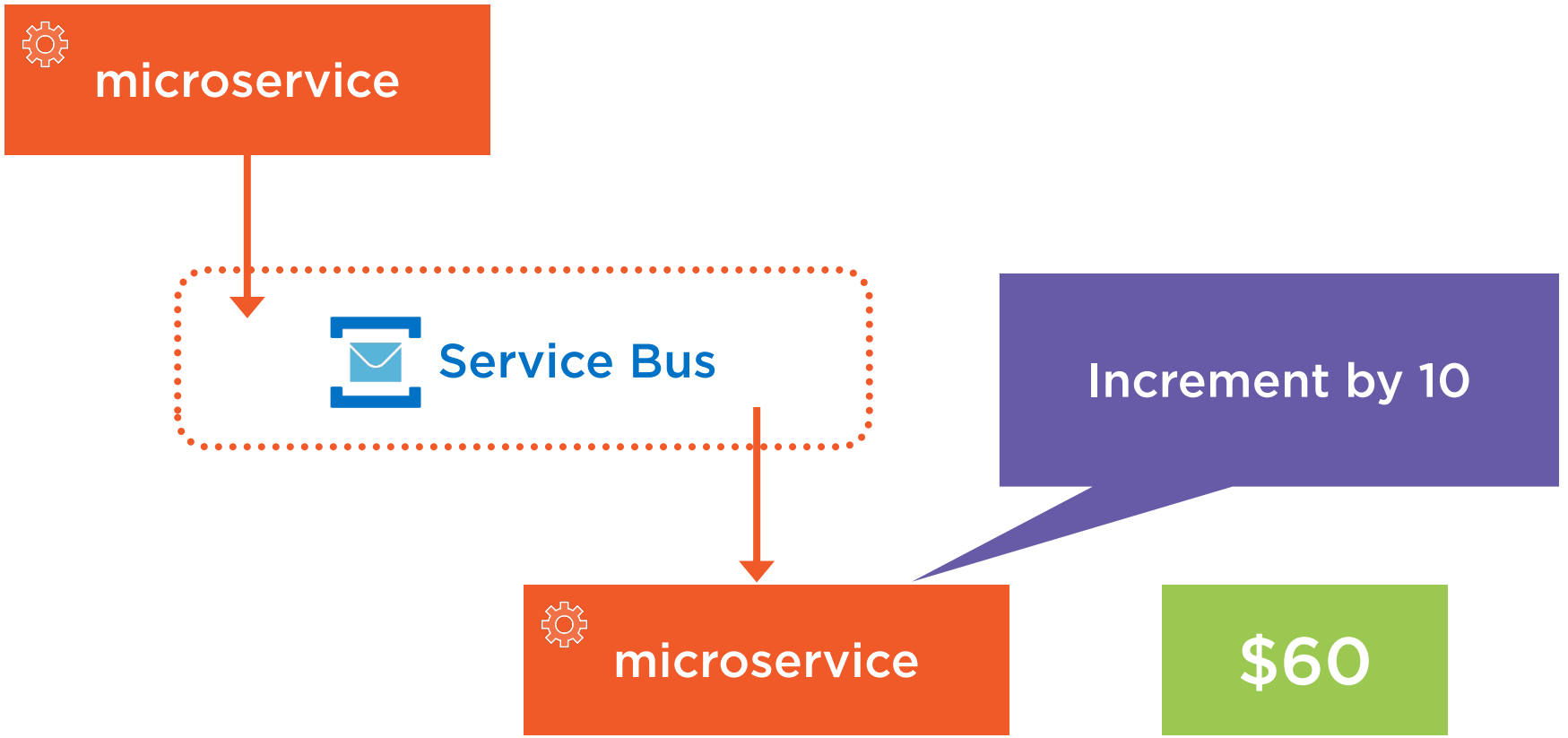
Detect Duplicates at Receiver



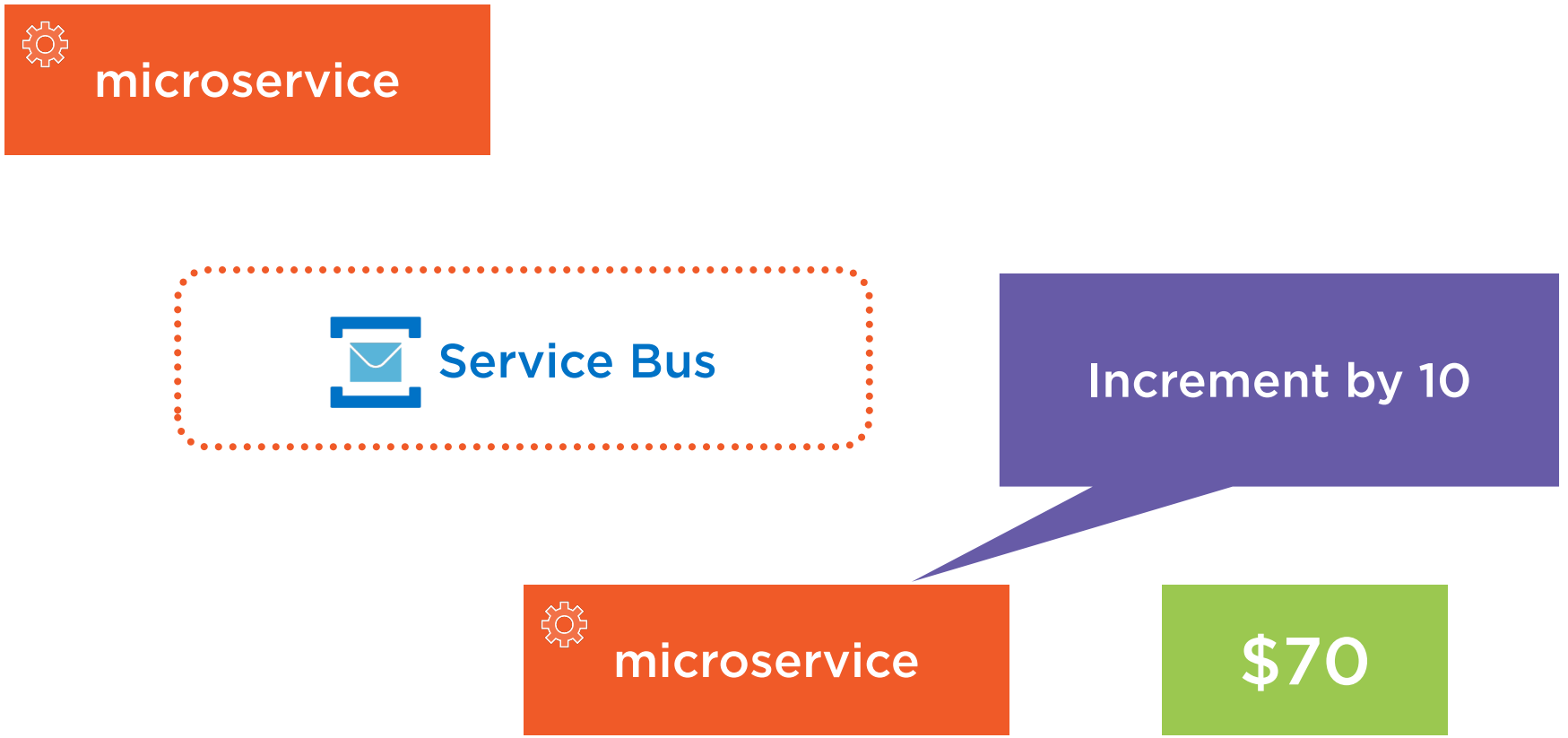
Detect Duplicates at Receiver



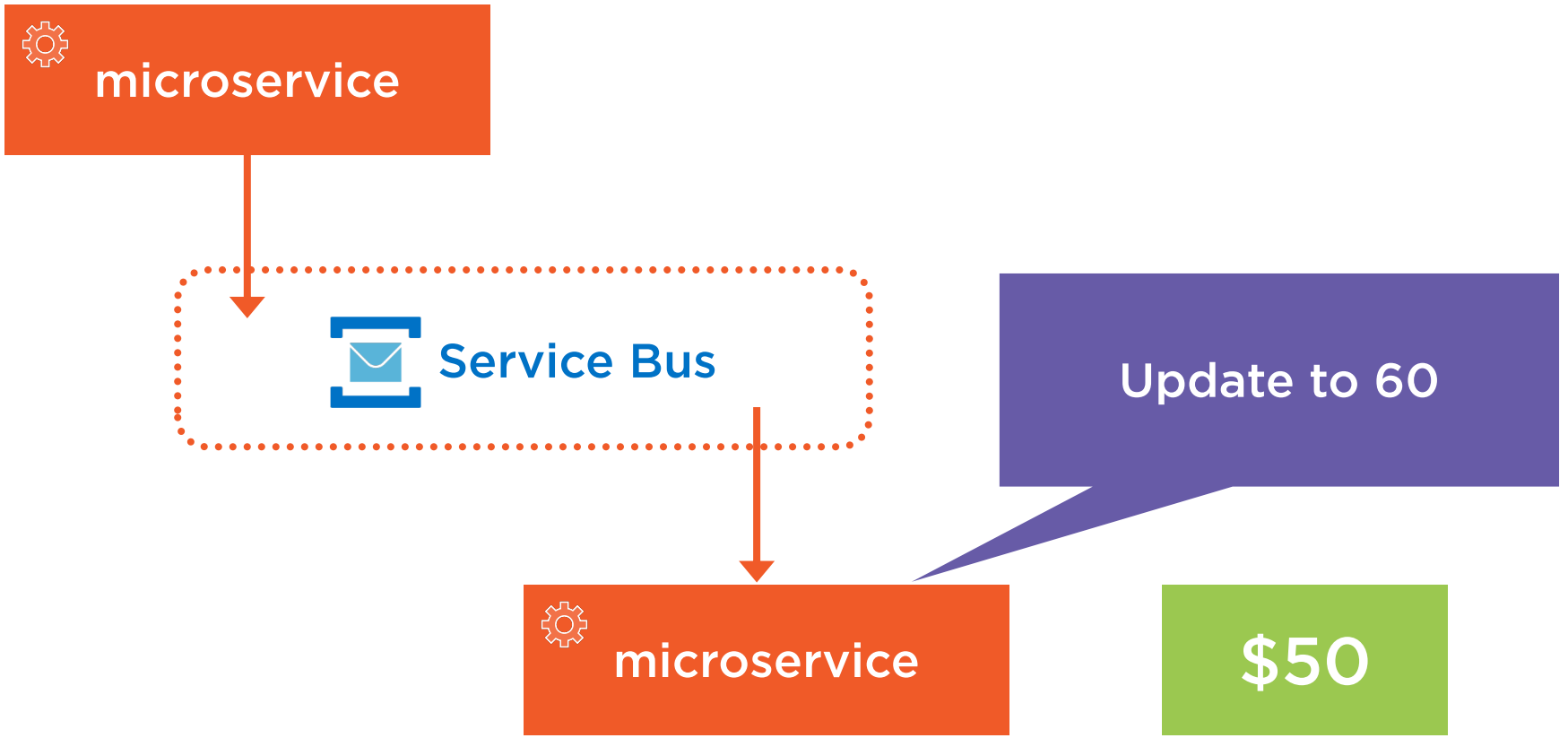
Detect Duplicates at Receiver



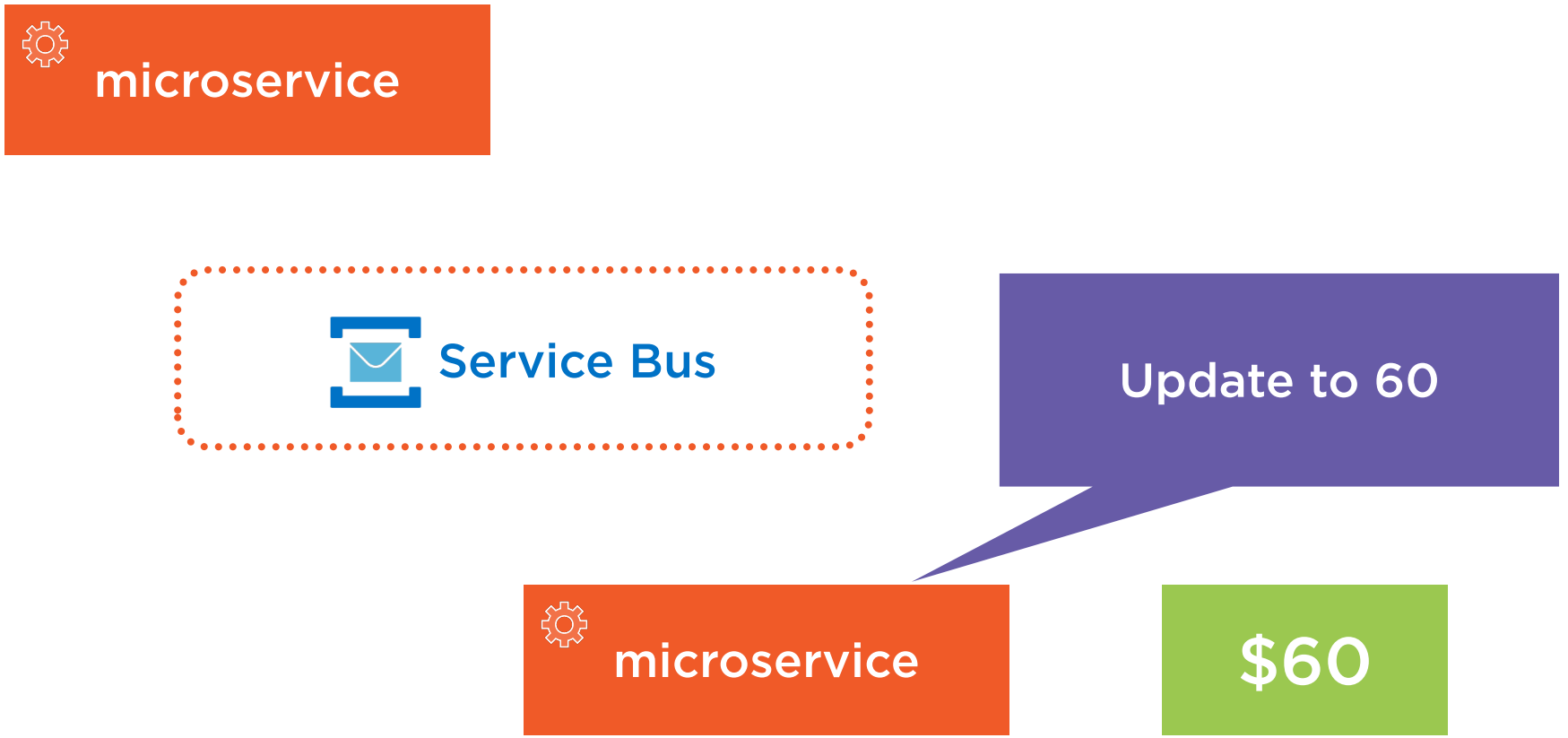
Detect Duplicates at Receiver



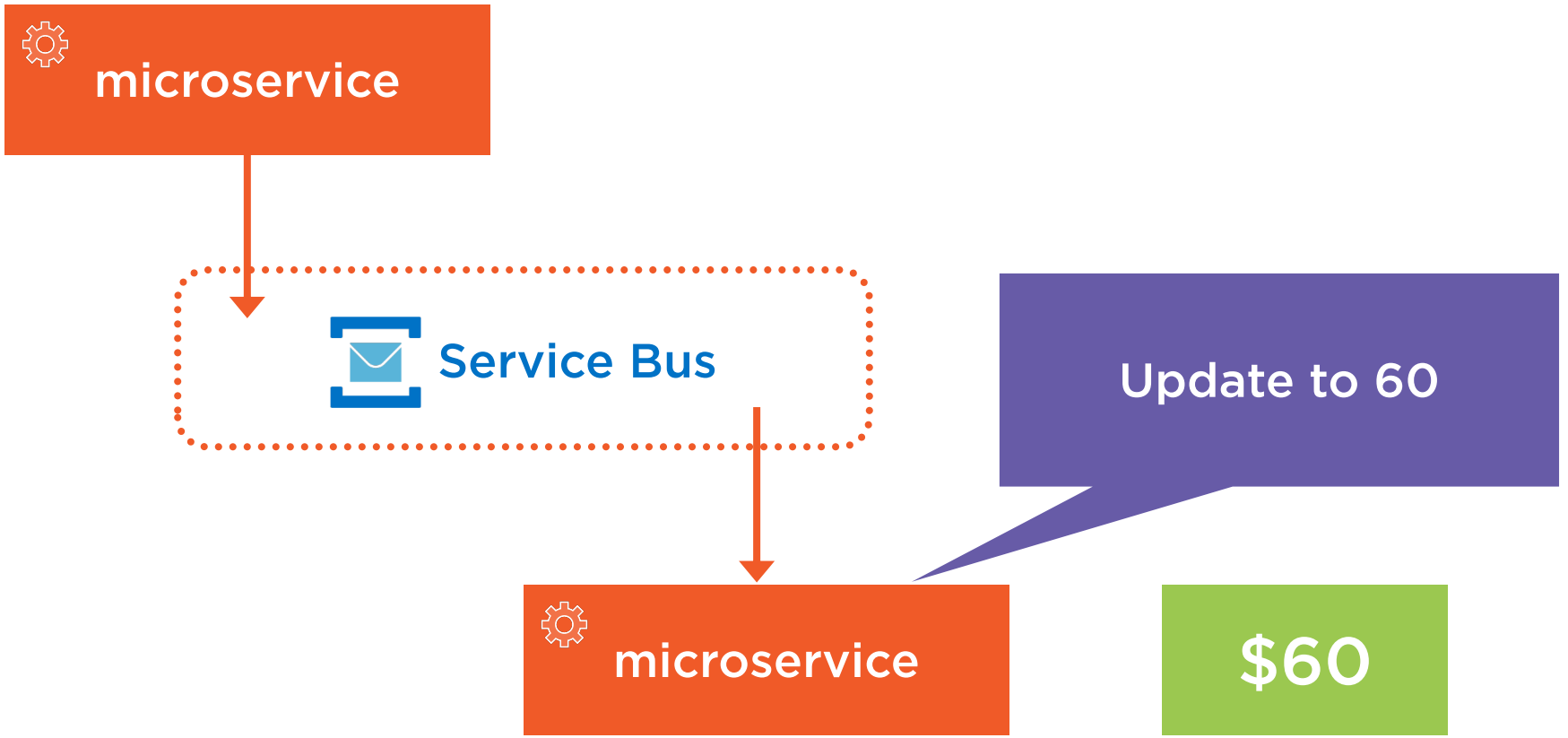
Detect Duplicates at Receiver

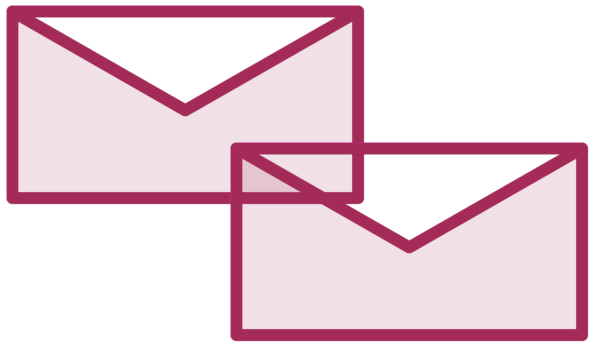


Detect Duplicates at Receiver



Detect Duplicates at Receiver





Handling Duplicate Messages

UPDATE table1 SET price=60 WHERE id=5 and version=1234

UPDATE table1 SET price=60 WHERE id=5 and price=50

GET * FROM Customer WHERE lastname="name" AND email="abc@g.com"

Course Summary

- Data consideration when designing microservices
- Globoticket sample project
- Data store options in microservices
- Integration events
- Event sourcing
- Handling duplicate messages

Thank You!

Neil Morrissey

@morrisseycode www.neilmorrissey.net

www.linkedin.com/in/neilmorrissey/