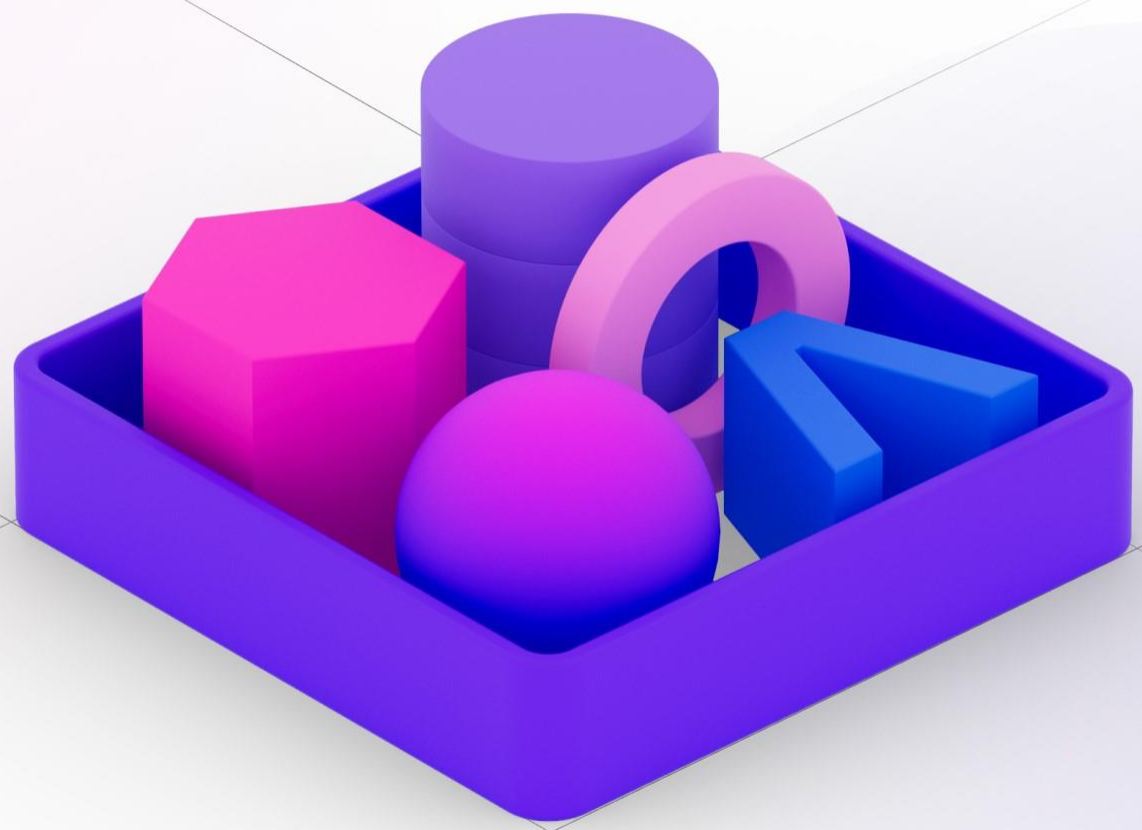
.NET Conf



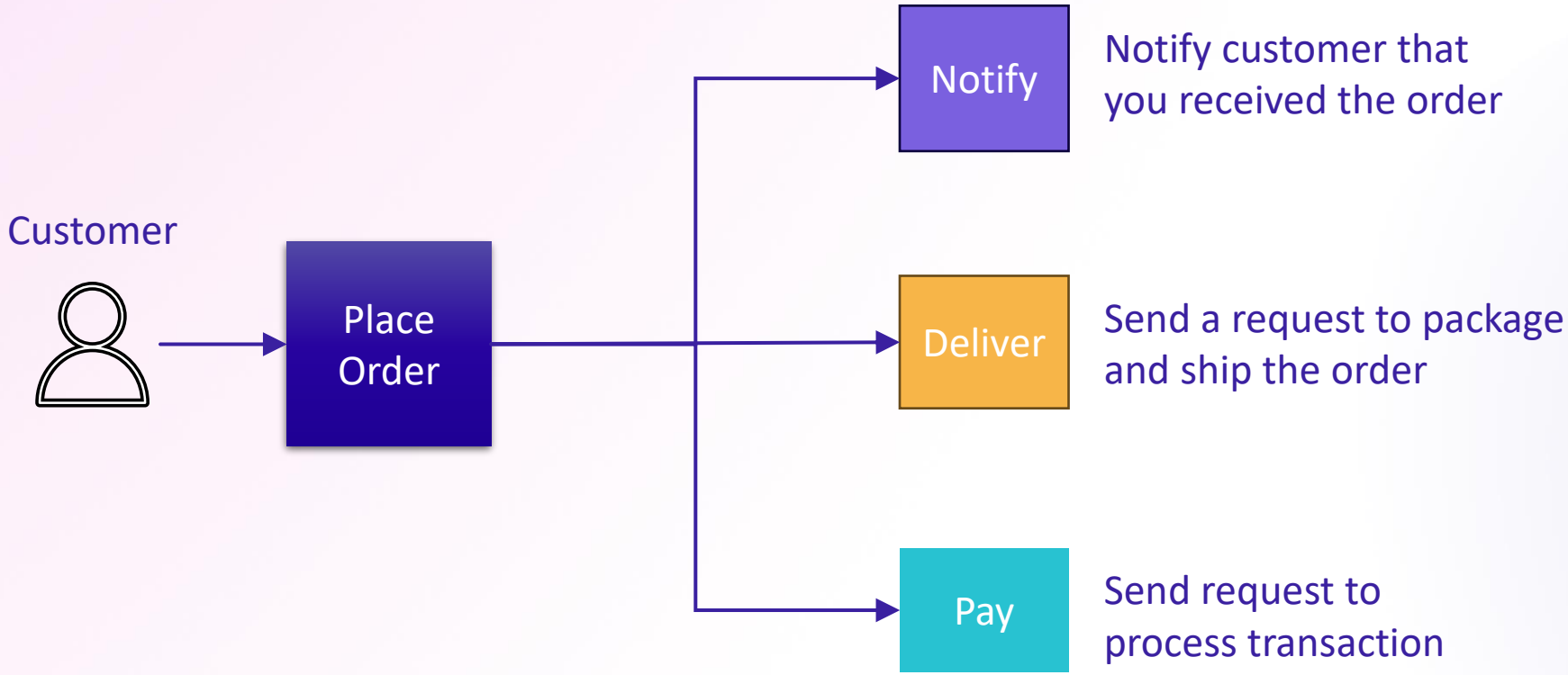
Building next-gen applications with Event-Driven Architectures

Teena Idnani

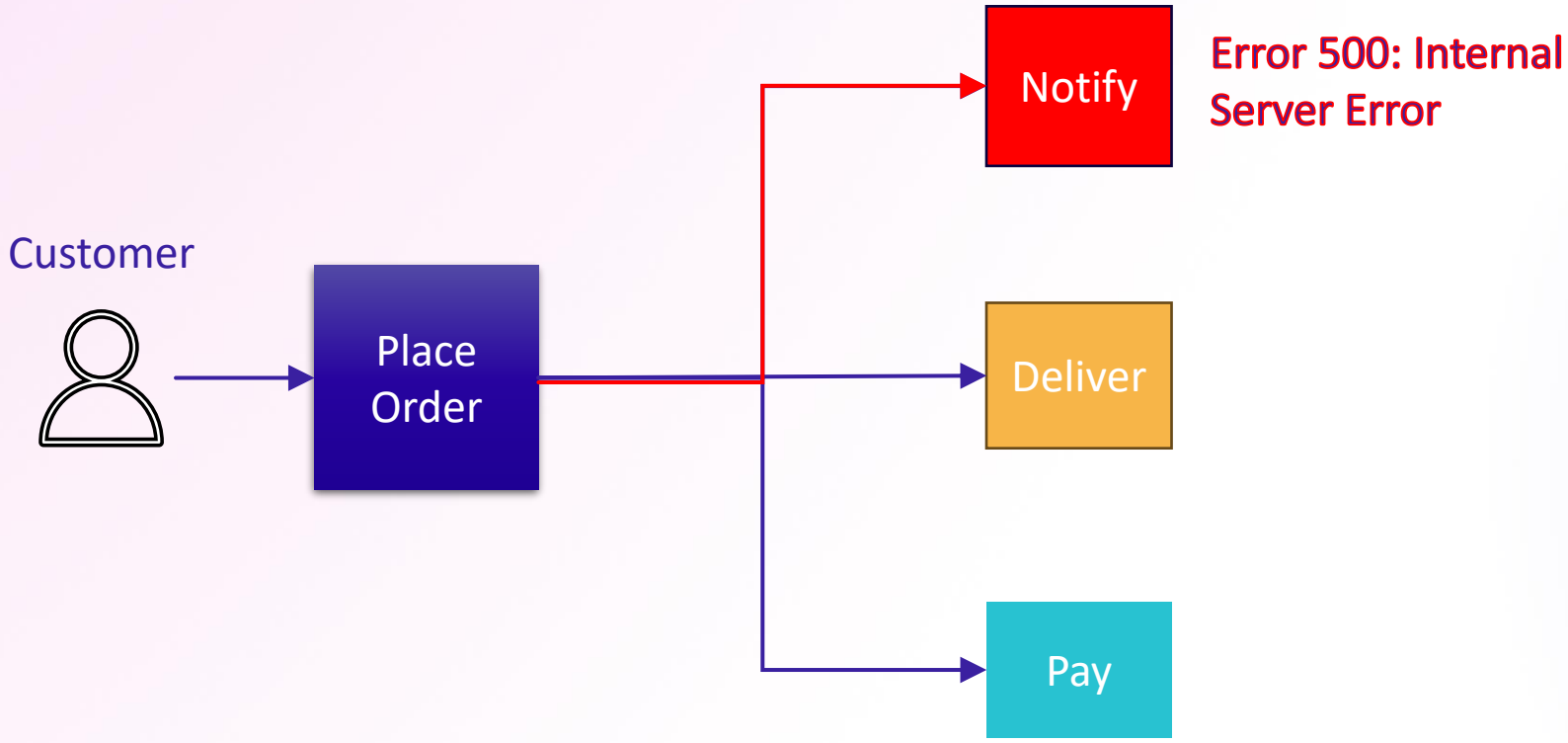
”By 2025, 50% of enterprise applications will transition from synchronous request/response-based interfaces to asynchronous event-driven messaging.

Source: IDC FutureScape, "Worldwide Future of Digital Infrastructure 2023 Predictions", October 2022

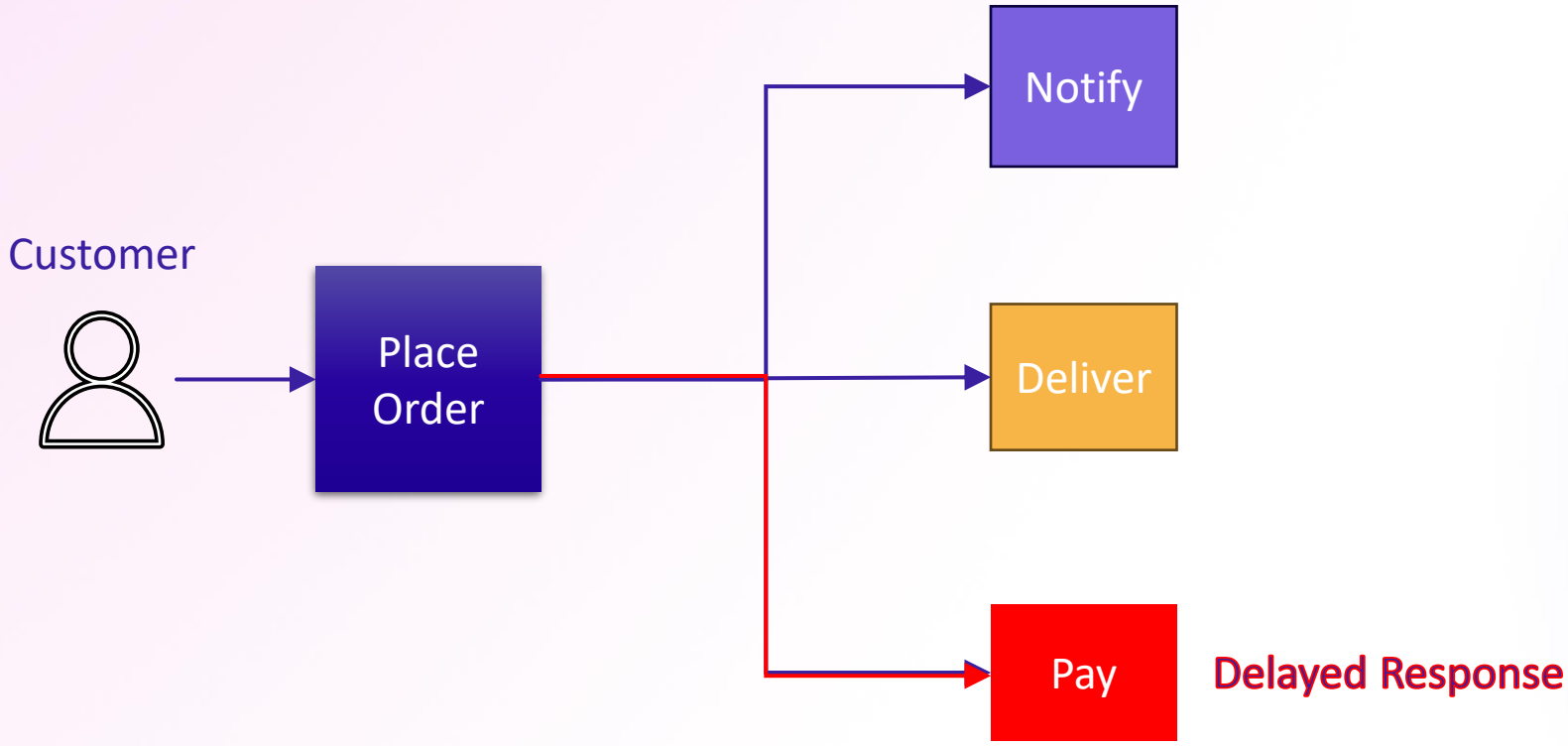
Synchronous Architectures: Scenario

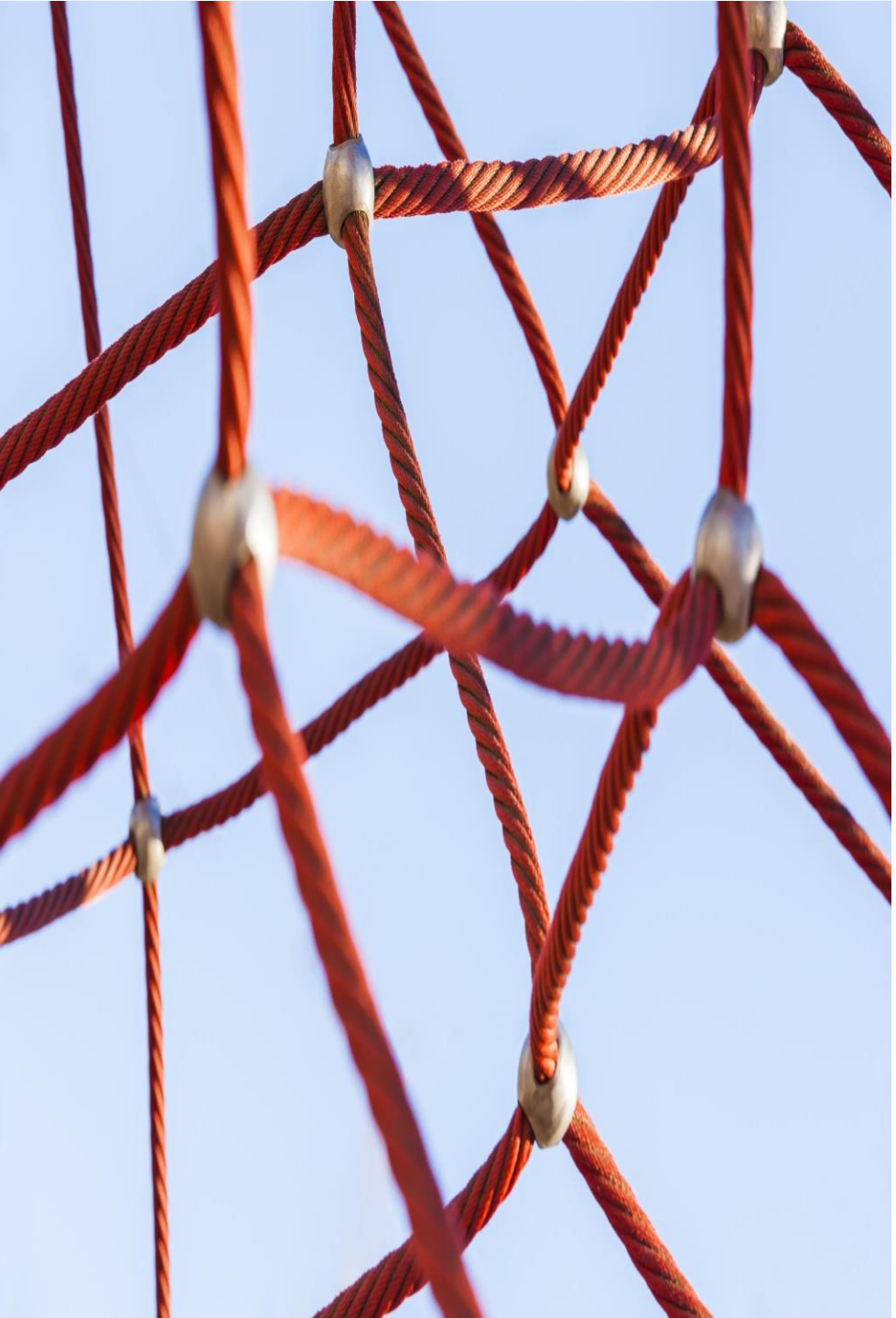


Synchronous Architectures: Challenges



Synchronous Architectures: Challenges

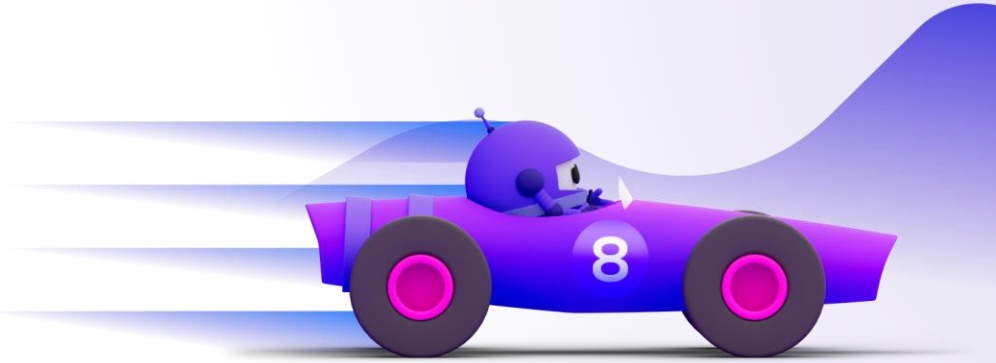




Challenges with Synchronous Architectures

- Latency and delay
- Tight coupling
- Cascading failures
- Complex coordination
- Error handling
- Scaling difficulty

Event Driven Integrations to the rescue



event

A lightweight notification of a condition or a state change that occurs in your environment

It all starts with business events

OrderCreated

PaymentInitiated

PaymentCompleted

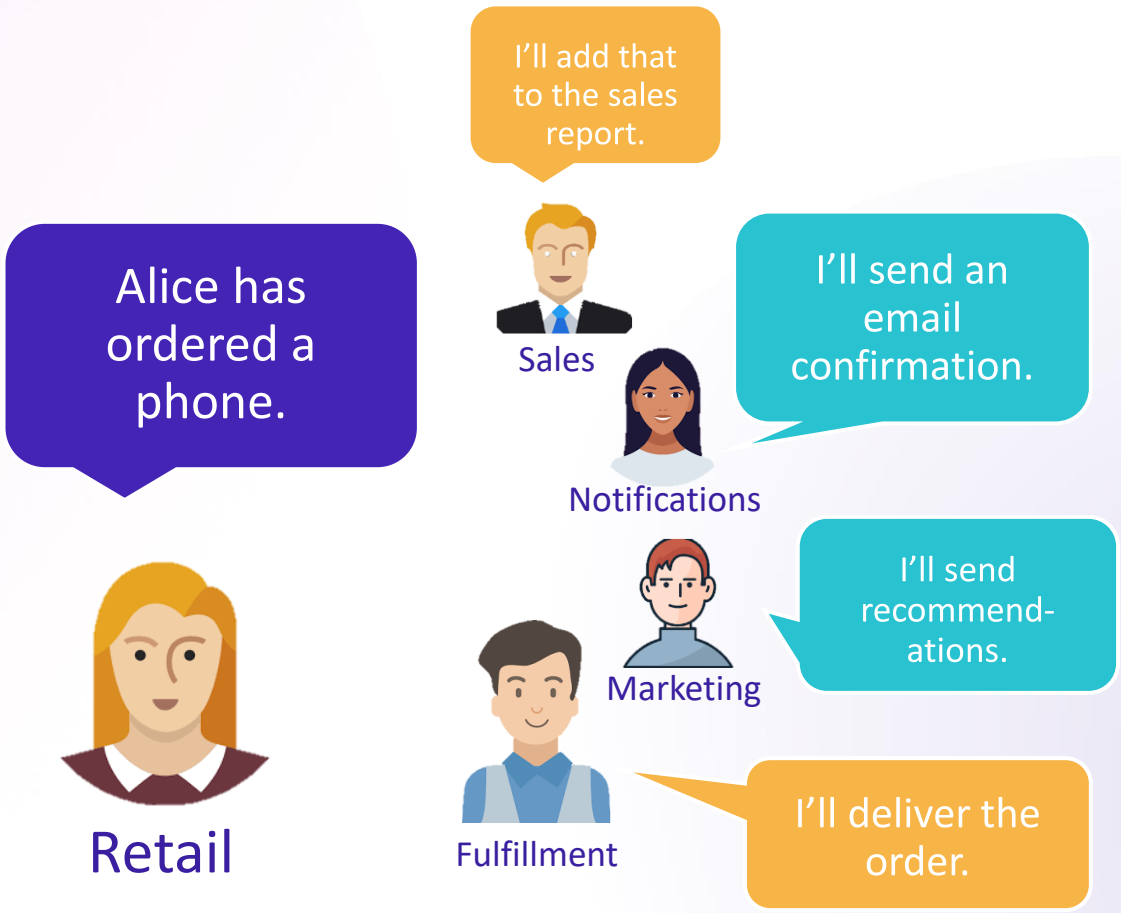
OrderPicked

OrderShipped

Events vs Commands

E.g. Alice has ordered a phone

Directed commands

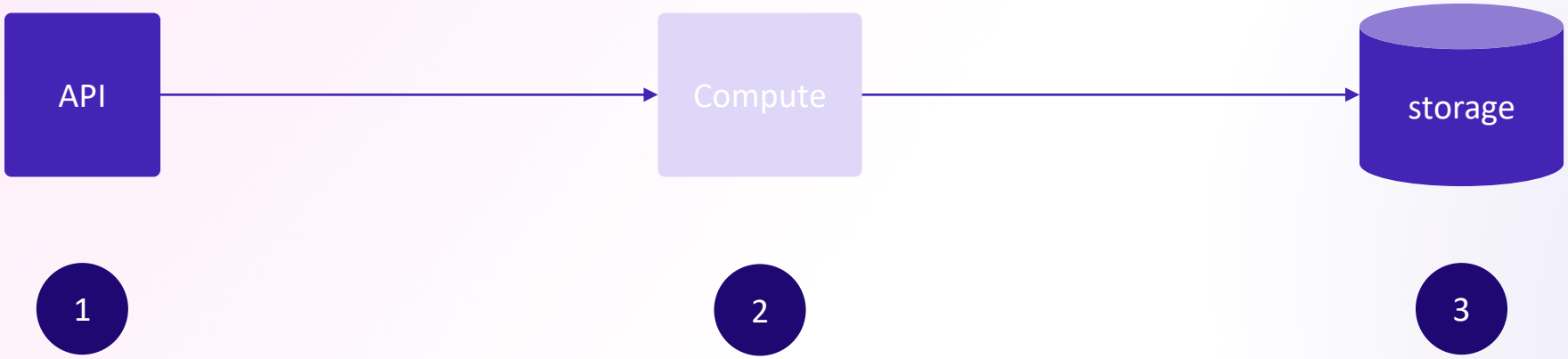


Observable events

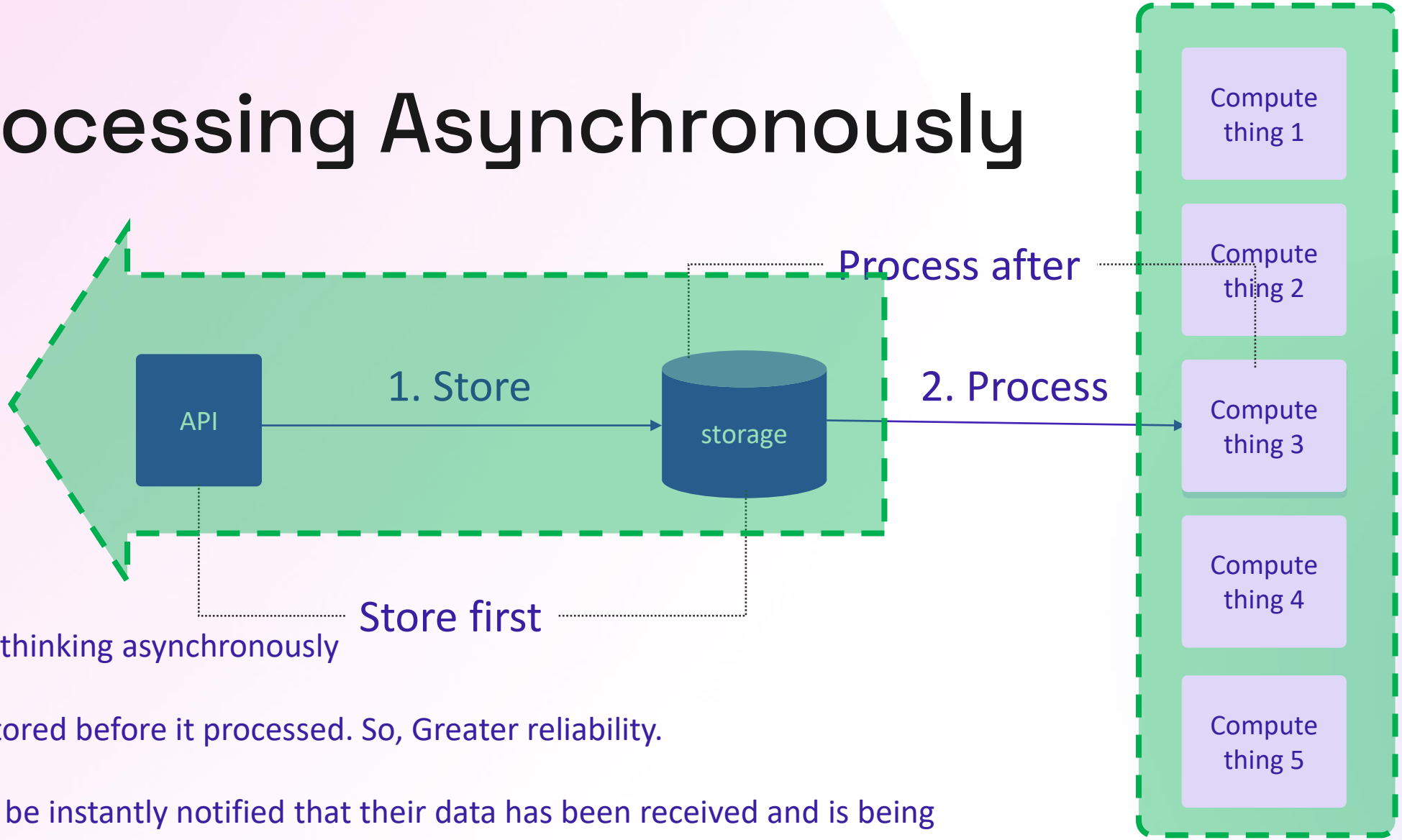
REMEMBER

How you design your events impacts your architecture!!

Synchronous Method Asynchronous Thinking



Processing Asynchronously



Benefits of thinking asynchronously

1. Data is stored before it processed. So, Greater reliability.

2. User can be instantly notified that their data has been received and is being processed. They can later poll the results whenever they like.

3. Data can be processed in parallel asynchronously

Event-driven applications benefits

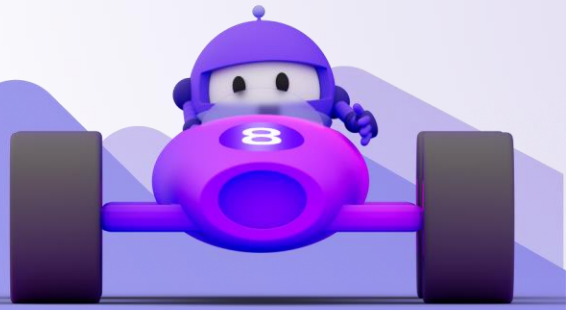
- Loose coupling
- Architectural agility
- High throughput
- Decoupled failure handling
- Dynamic scaling
- Higher availability

When you think “Event First”....

- How do we identify and design events?
- How do we build the event driven architecture?
- How can we scale event-driven architectures?

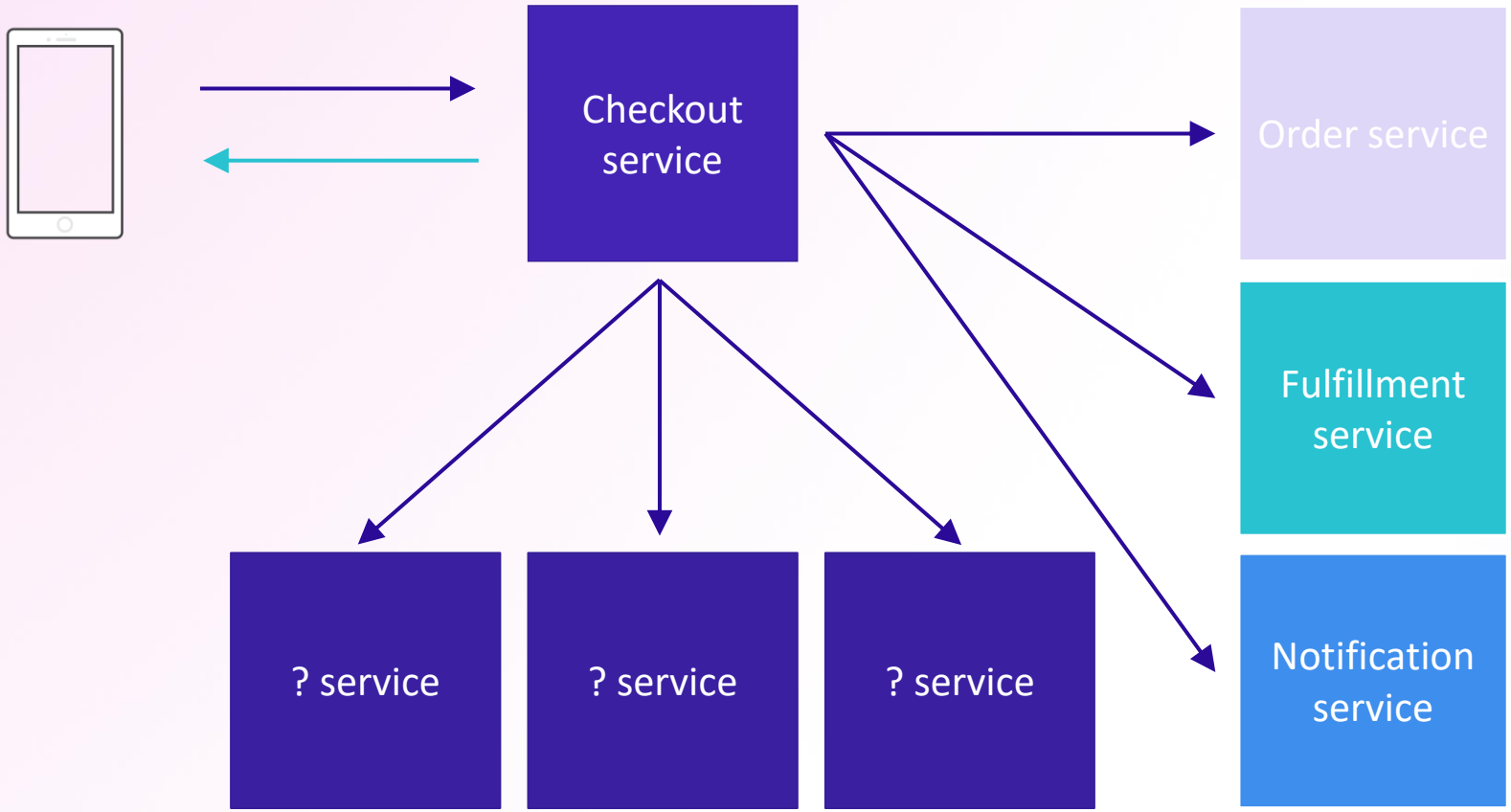
....Common questions start to form

Let’s look at the real example

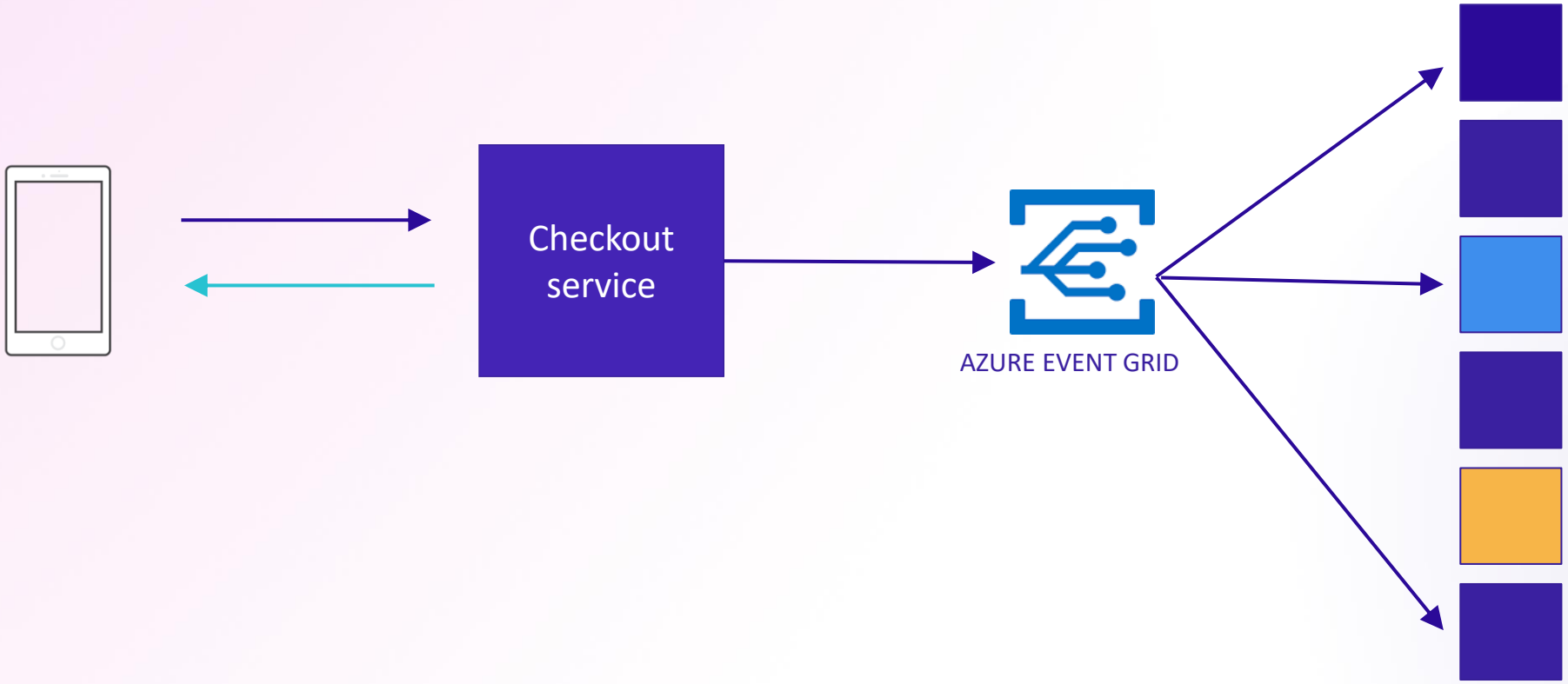


Launch a simple eCommerce site . . .

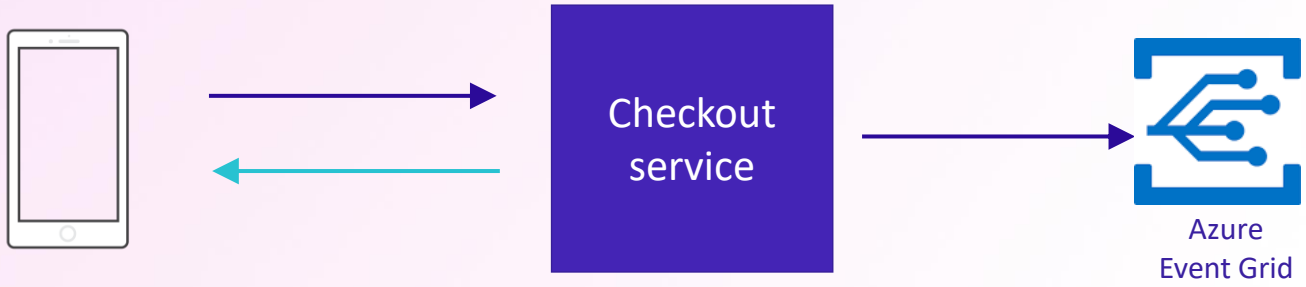
Thinking about scalability...



Lets build this on Azure

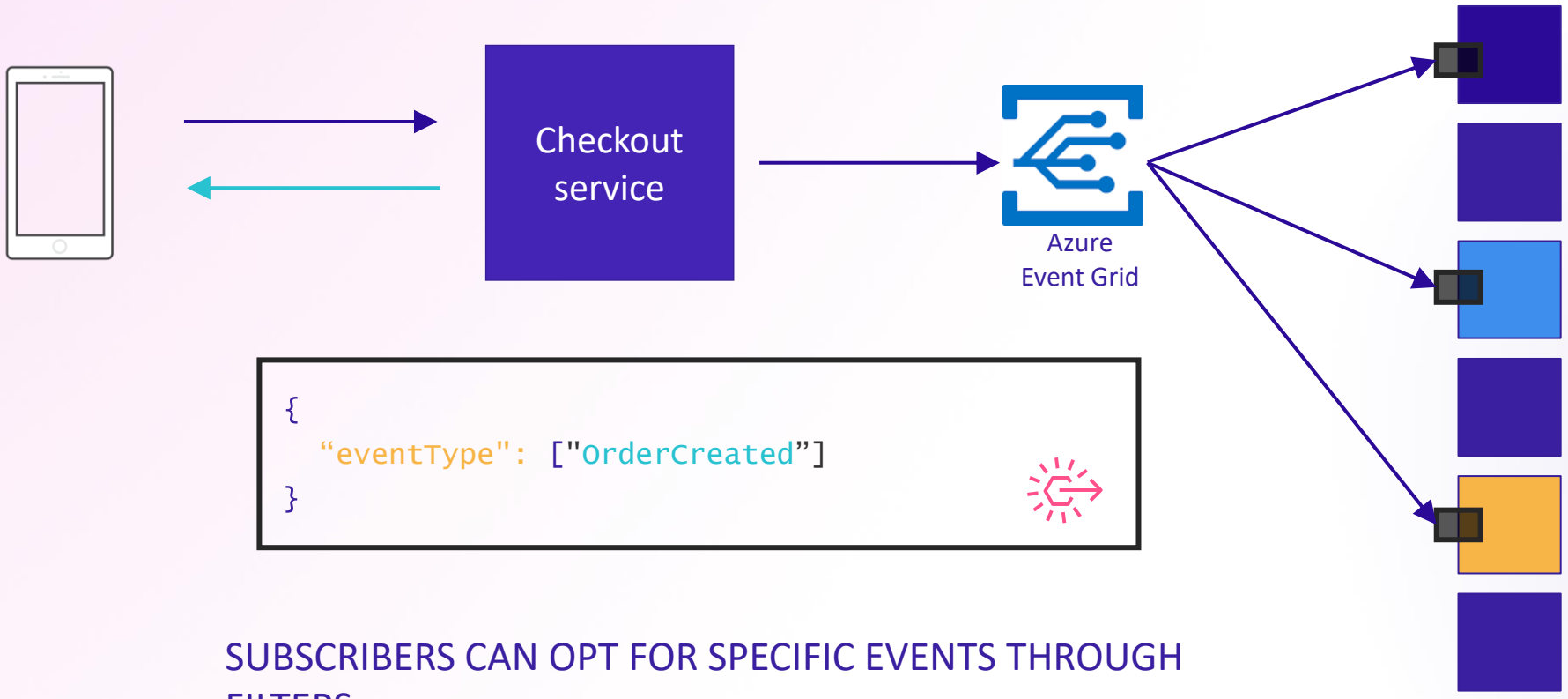


On an “OrderCreated” event



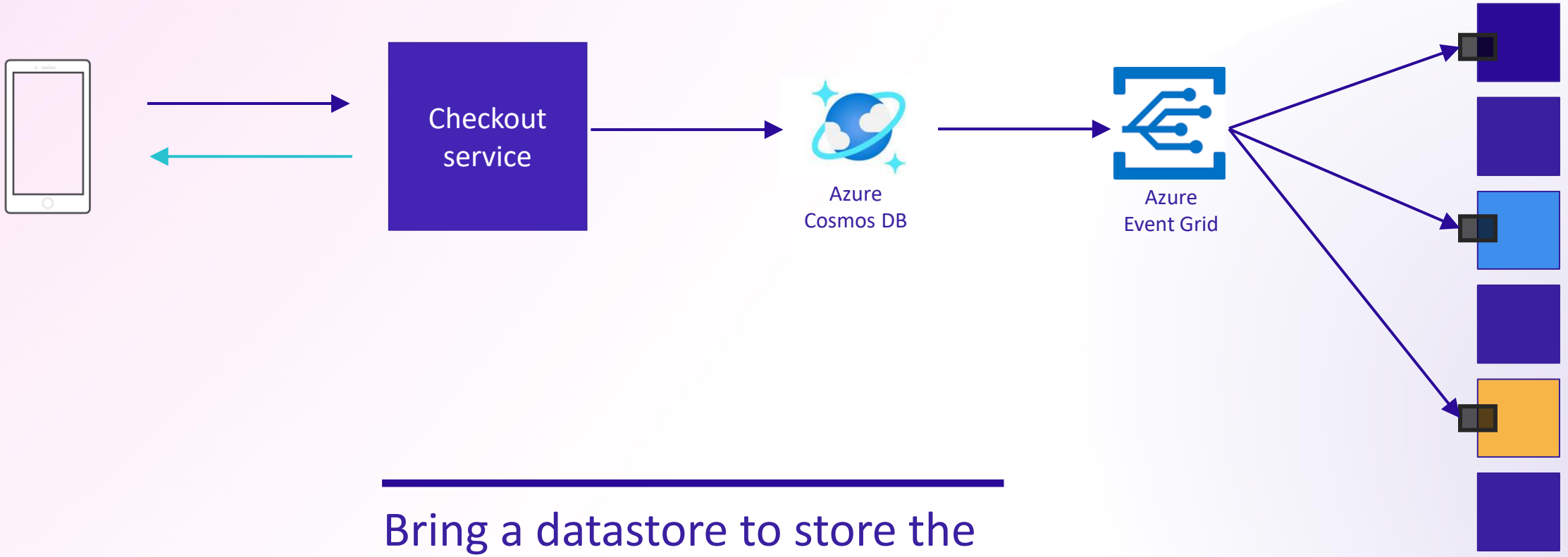

```
{
  "id": "30934",
  "eventType": “orderCreated",
  "subject": "department/widget",
  "eventTime": "2023-11-16T10:10:20+00:00",
  "data":{
    “customerId": "14",
    “customerName": "Nigel Tufnel",
    “customerEmail": "nigel@contoso.com
  }
}
```

EVENT GRID ENSURES DELIVERY TO THE SUBSCRIBERS.



SUBSCRIBERS CAN OPT FOR SPECIFIC EVENTS THROUGH FILTERS

Improve resiliency and scalability with event stores



Bring a datastore to store the messages before they are processed

The overall pattern

1. API stores data in CosmosDB
2. Use CosmosDB Trigger Azure Function to push the data as an event to Event Grid Topic
3. Once the event is published to the Event grid topic,

   different subscribers can handle the event the way

   they want.

Lets see it through a Demo...

Different PaaS services on azure

MANAGED SERVICES PROVIDE ROUTING, STORAGE, AND DISTRIBUTION OF EVENTS

Azure Service Bus

Messaging

Robust enterprise message brokering service. Supports queues, topics, sessions

Azure Event Grid

Eventing

Fully-managed event routing service

Azure Queue Storage

Message Queuing

Simple message queuing service. Provides asynchronous decoupling and resilience

Azure Event Hubs

Data Ingesting

High throughput data ingestion service

Wrap up

- ✓ Embrace ”Event First” thinking - model events upfront to enable decoupled components
- ✓ Store first, then process - leverage event stores for resilience and independent scalability
- ✓ Build scalable, resilient applications using event driven architectures - events unlock responsive and adaptable systems

Resources

Demo – Source Code

Microsoft Learn – Event Grid

Microsoft Learn – Event Grid Architecture

QUESTIONS?

THANK YOU