Improving Reliability, Scalability, and Extensibility with Event-Driven Architecture

Scott Carter, Principal Software Engineer

8/8/2022

Agenda

- What is Event-Driven Architecture (EDA)?
  - Events
  - Event Broker
  - Producers
  - Consumers

- Comparison with Request-Based Solutions

- EDA Challenges

- Getting started with EDA
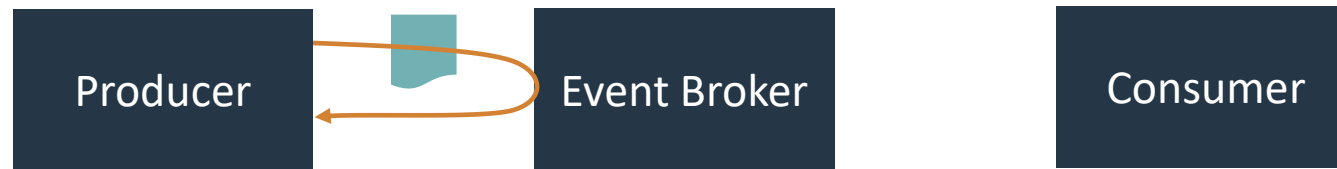
Definition and Components

What is Event-Driven Architecture?

System of loosely coupled microservices that exchange information through the production and consumption of events
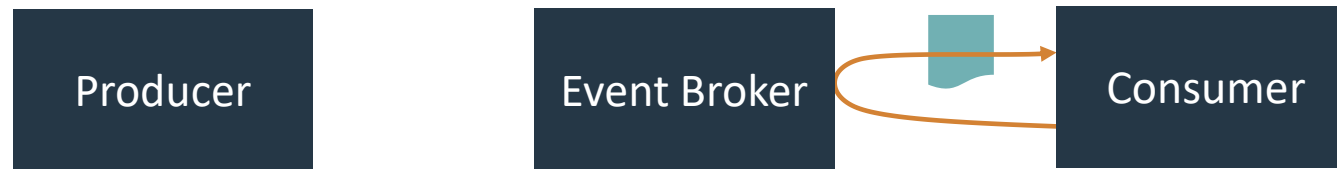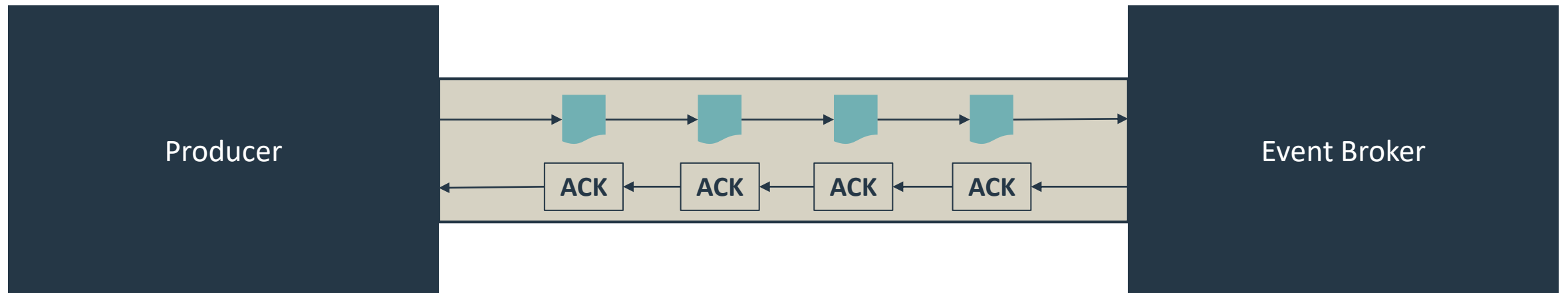
Basic Components

Producer → Event Broker → Consumer

Asynchrony

Producer

Event Broker
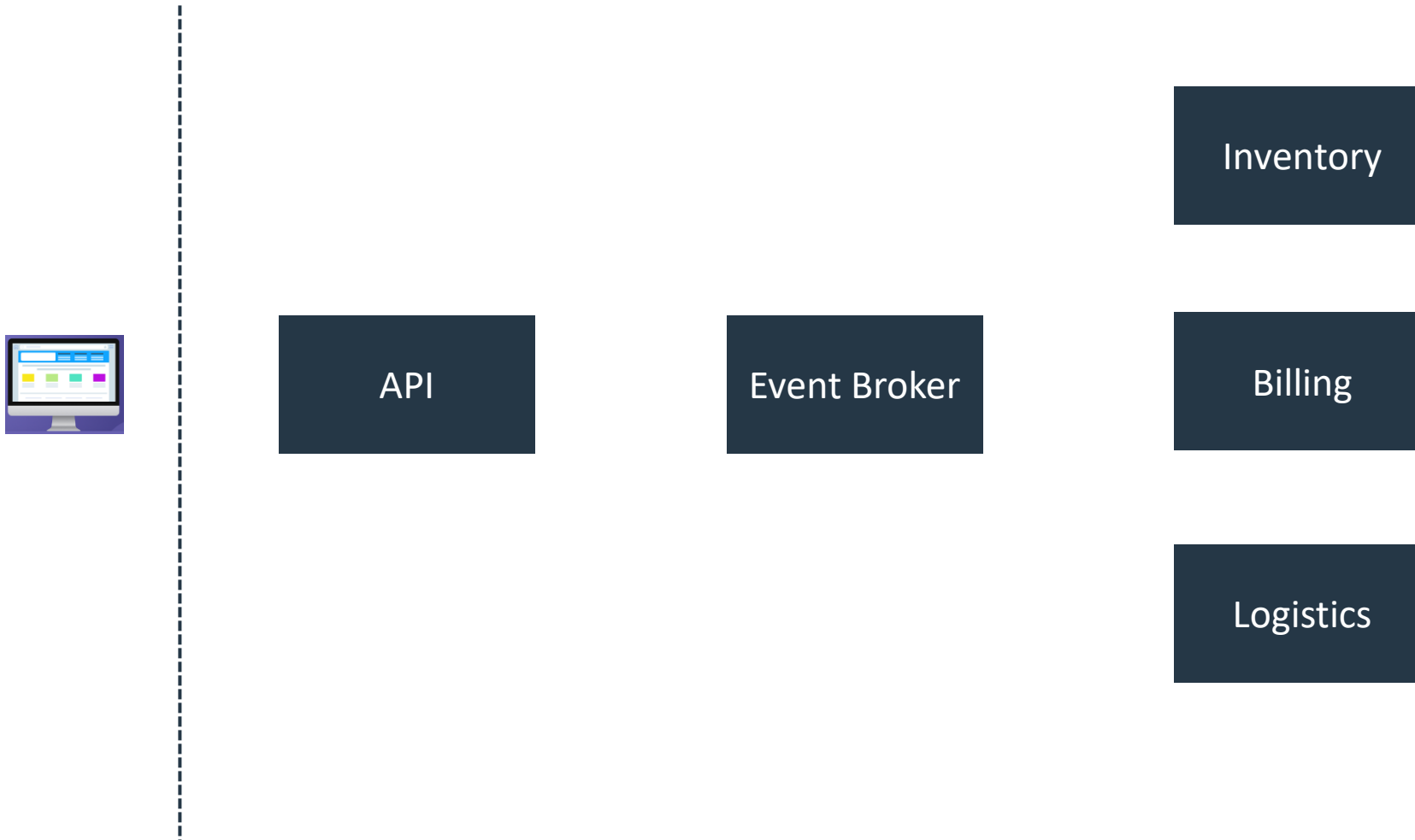
Consumer
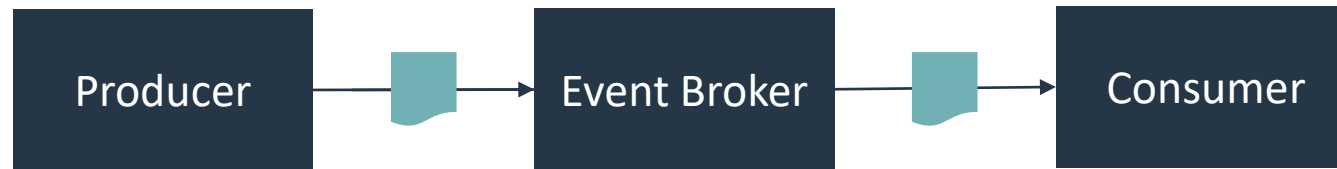
Asynchrony

Efficient Communications

Example

Sample Event-Based Point of Sale System

Inventory

API

Event Broker

Billing

Logistics

Sample Event-Based Point of Sale System

Sample Event-Based Point of Sale System

Sample Event-Based Point of Sale System

3) Order accepted

HTTP

API

Event Broker

Inventory

Billing

Logistics

Sample Event-Based Point of Sale System

Sample Event-Based Point of Sale System

API

Event Broker

Inventory

Billing

Logistics

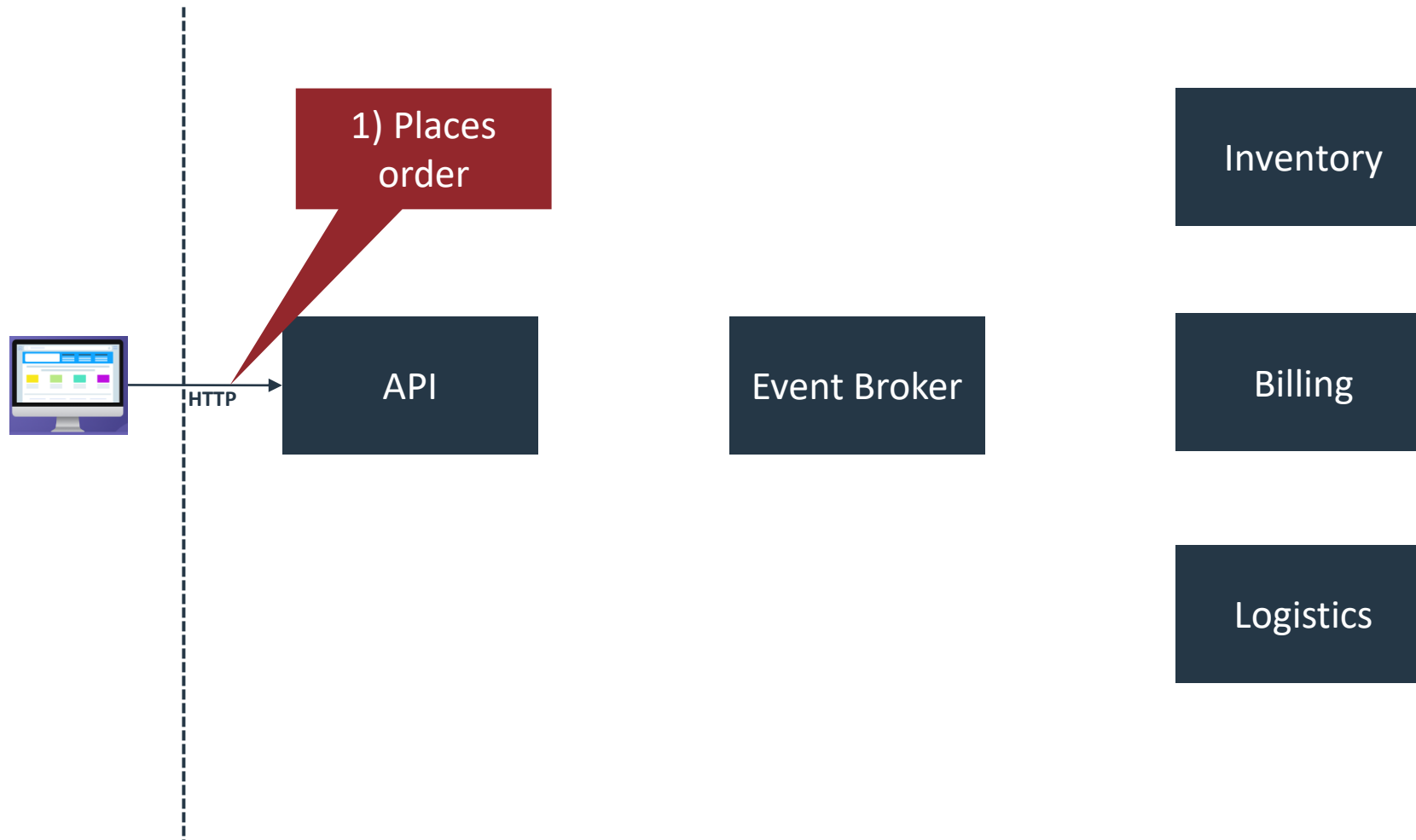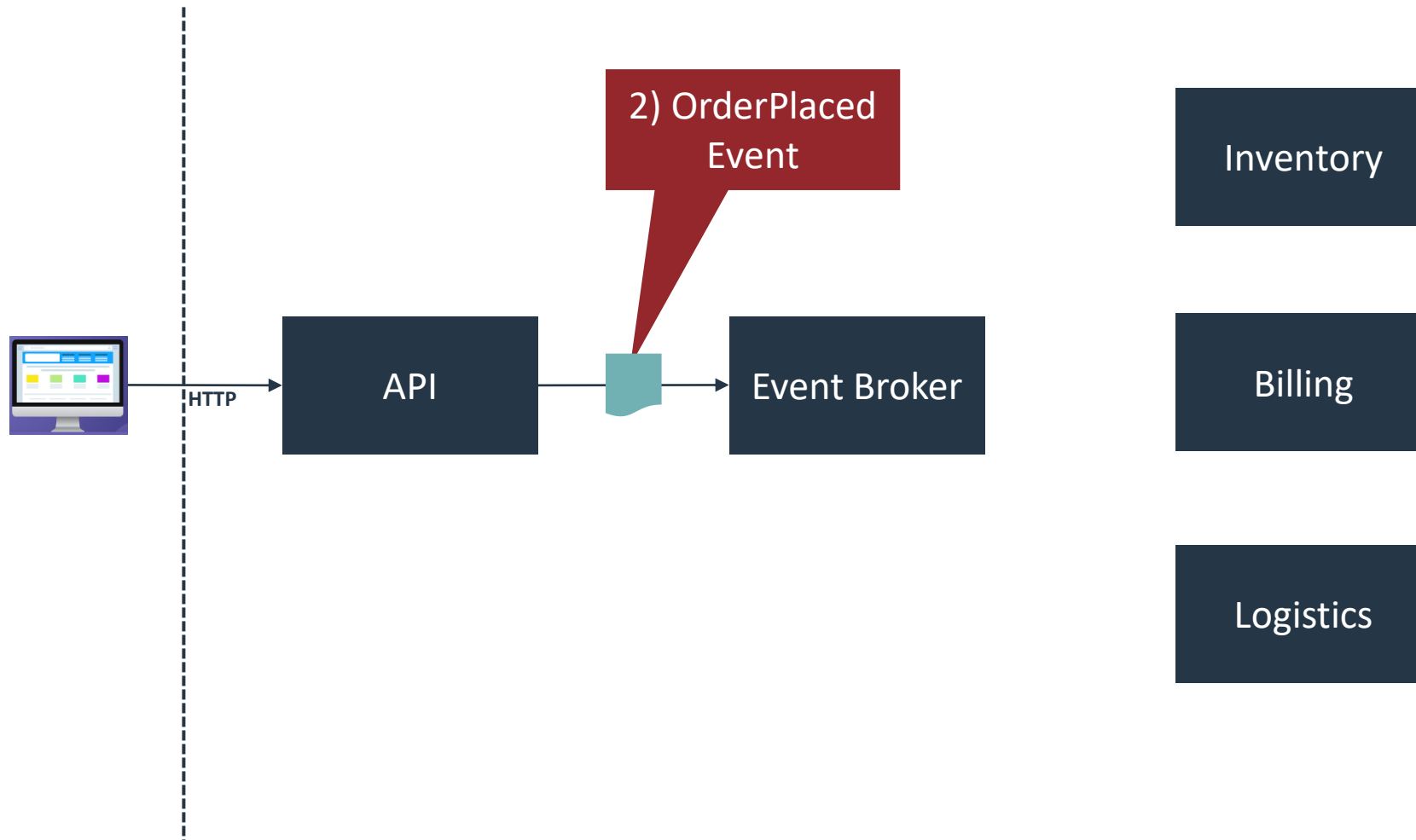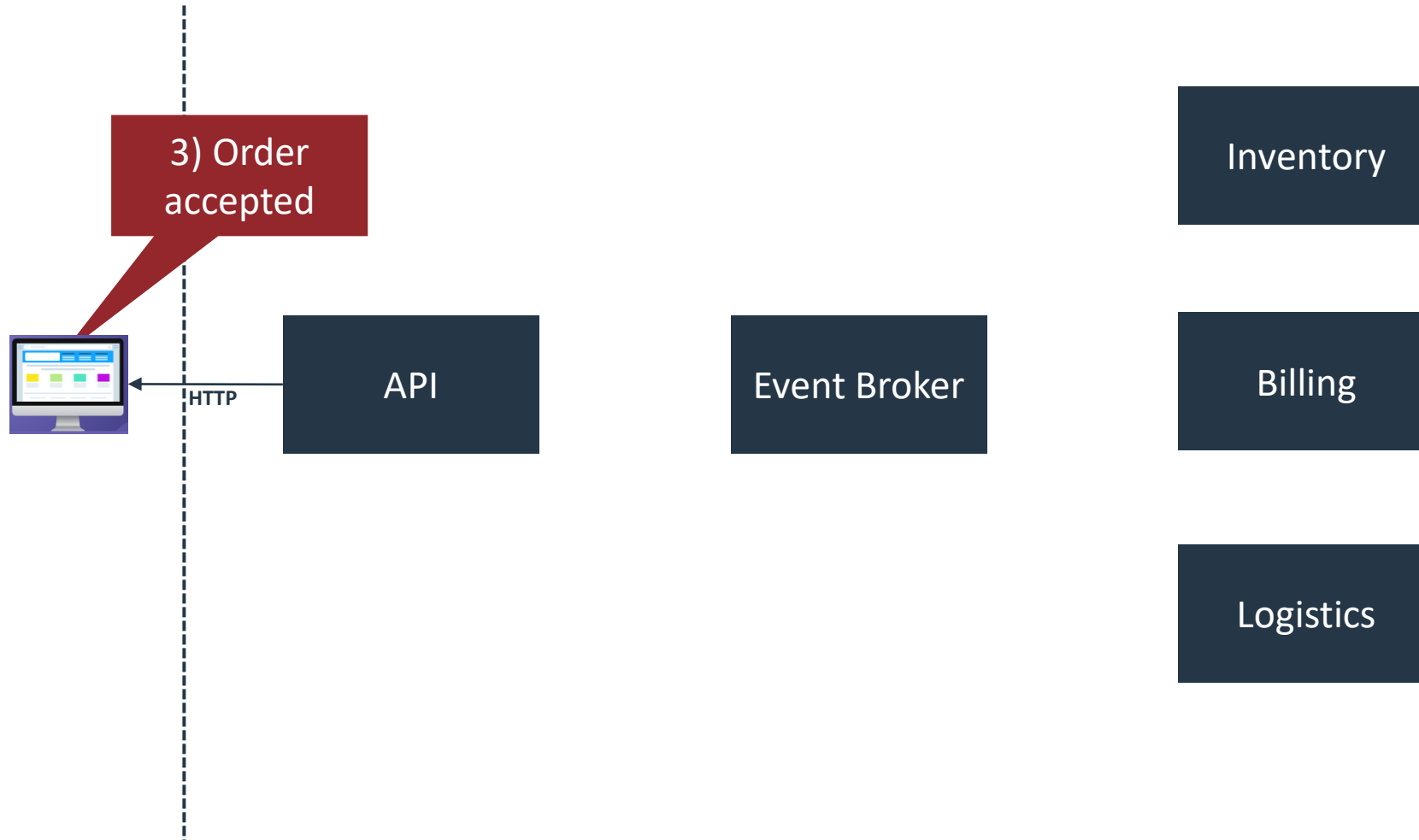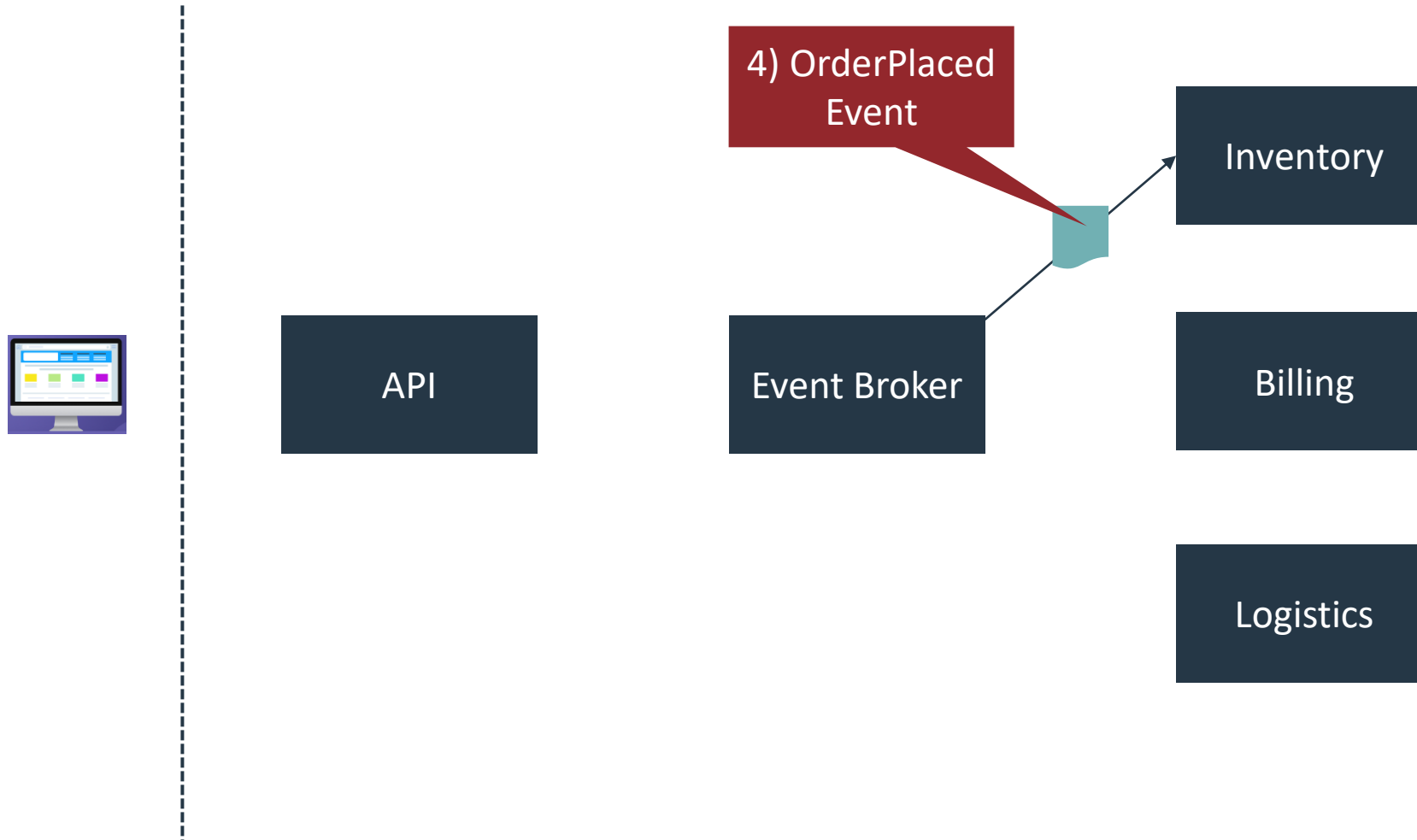5) OrderReserved Event
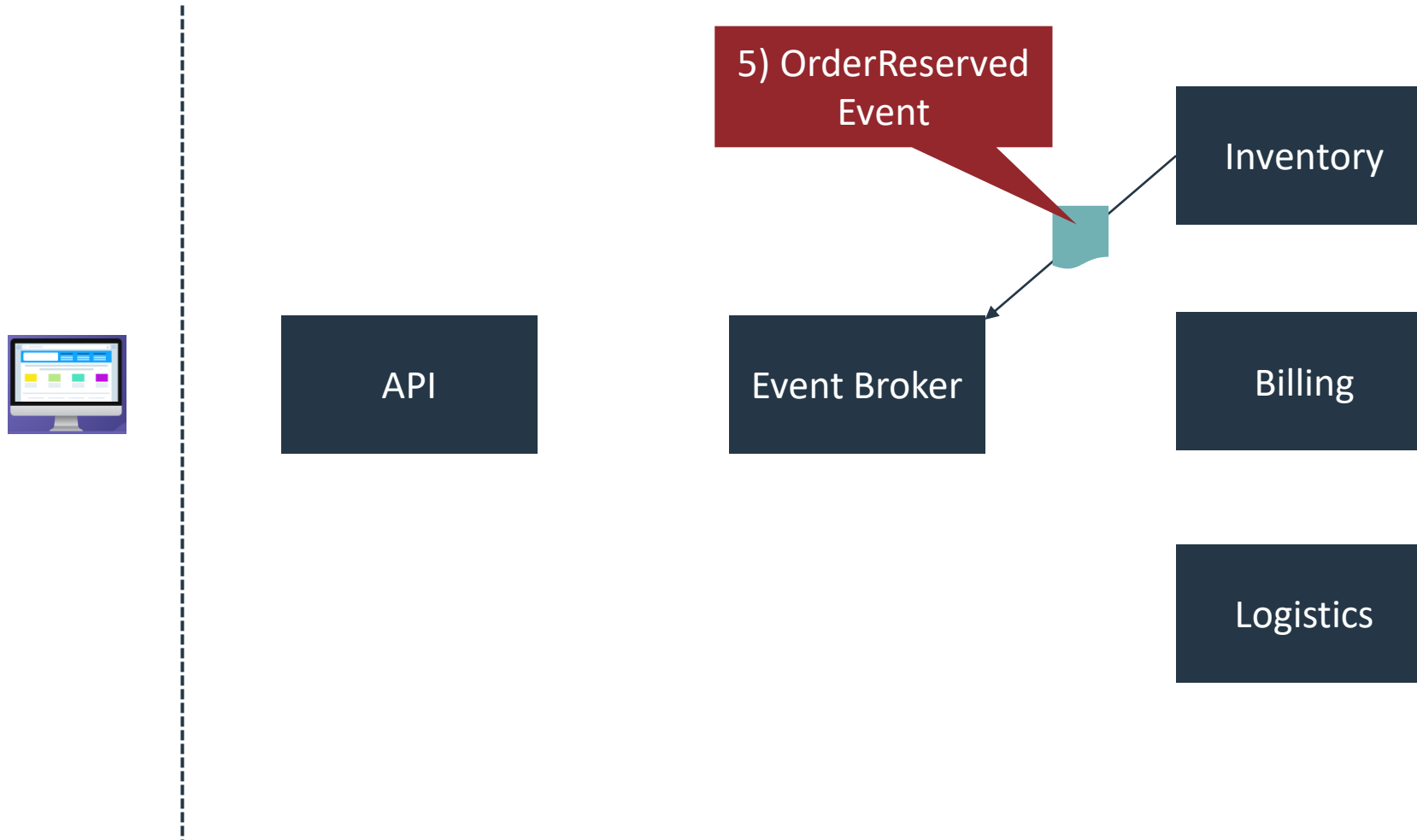
Sample Event-Based Point of Sale System
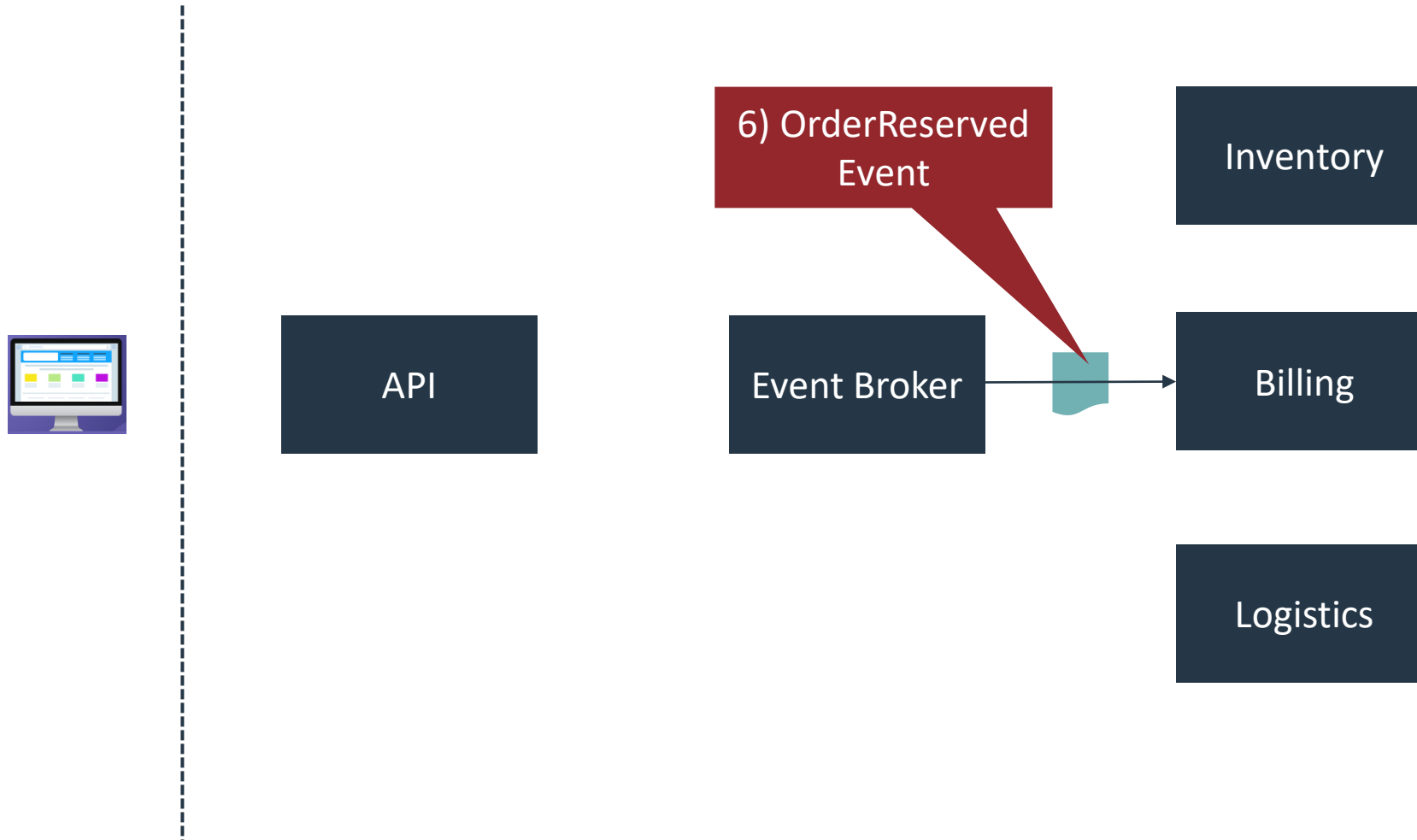
Sample Event-Based Point of Sale System
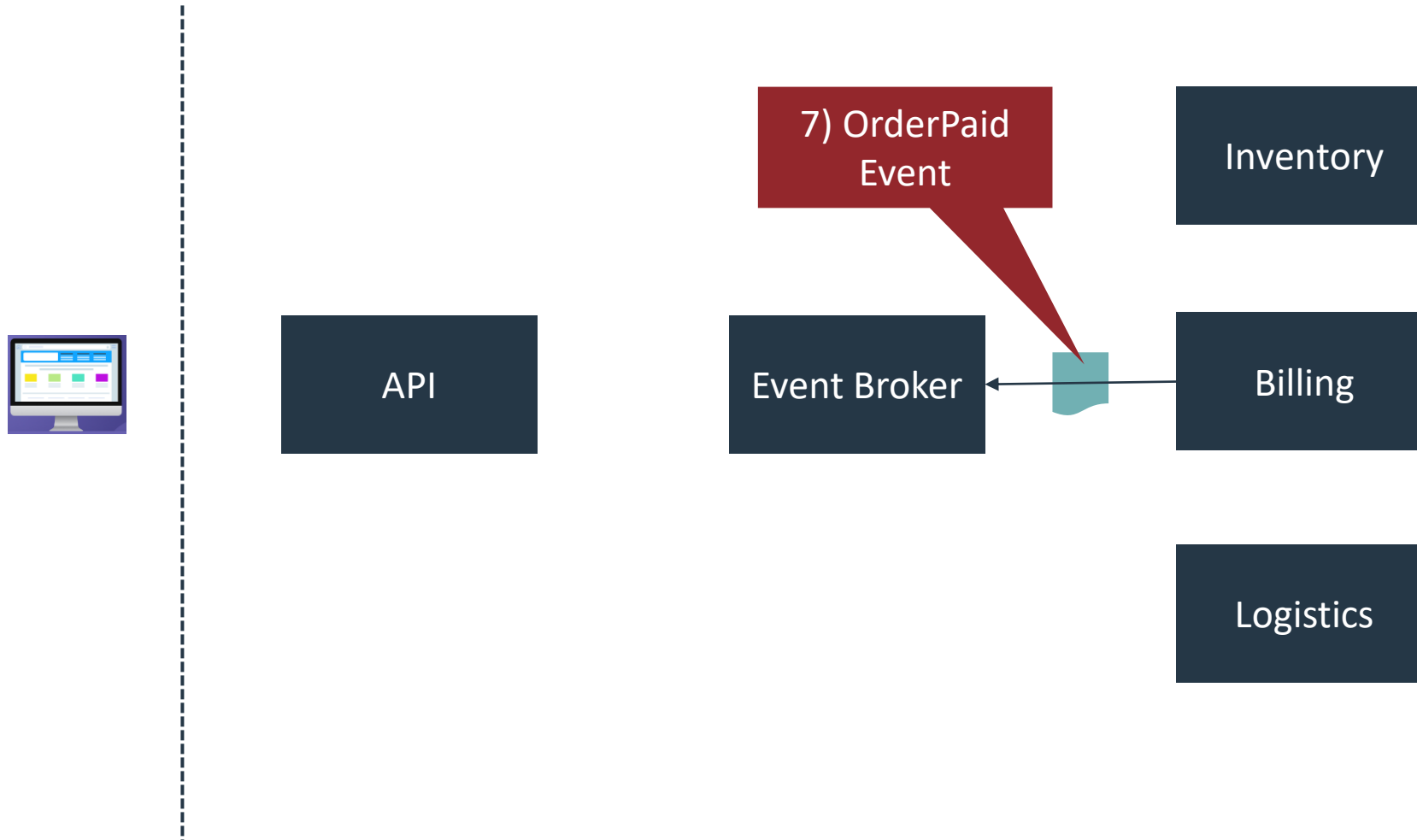
Sample Event-Based Point of Sale System
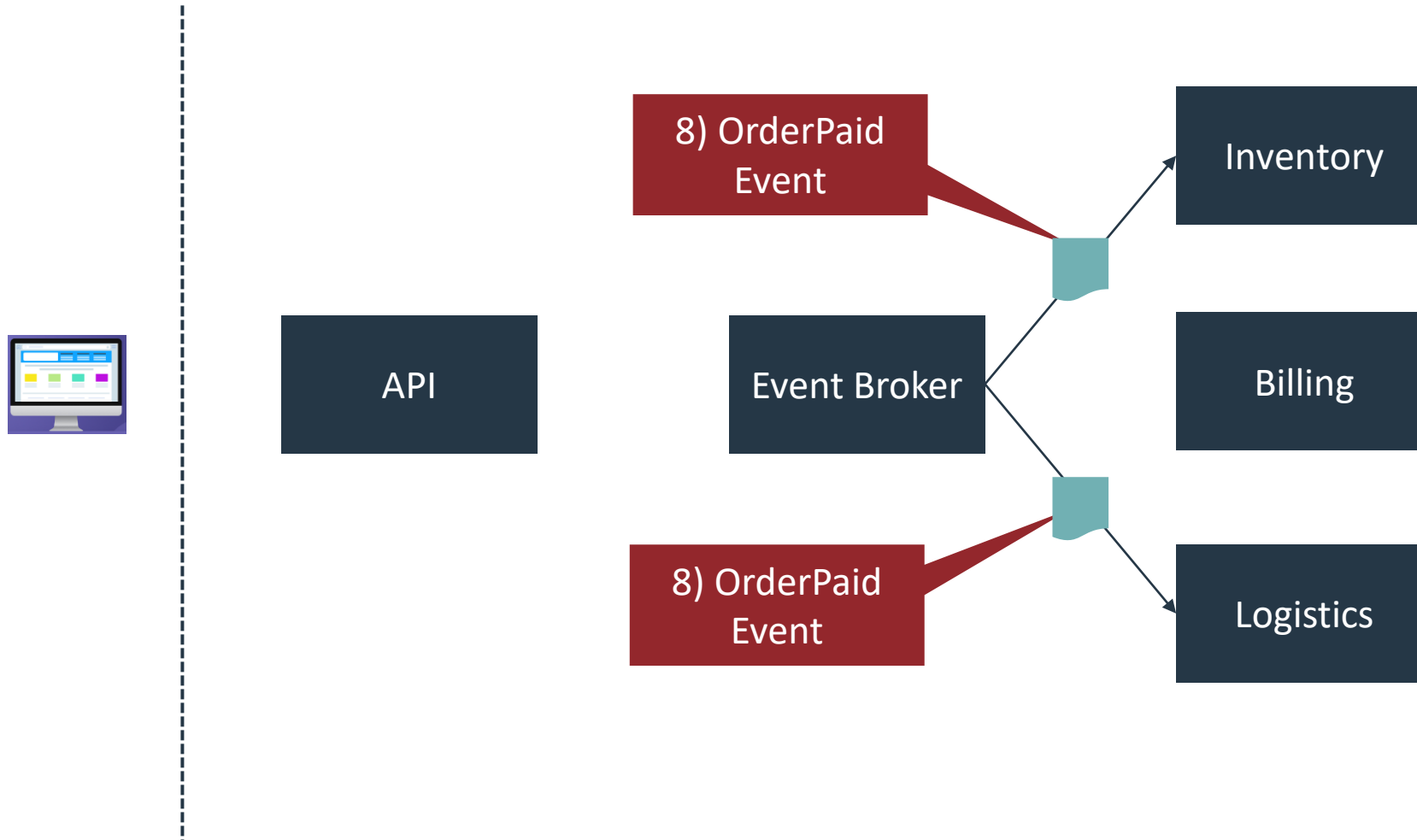
Component Closer Look

Events

```
{
    "eventName" : "orderPlaced",
    "orderId": "760b5301-295f-4fec-95f8-6b303a3b824a",
    "customerId": 28623823,
    "productId": 31334,
    "quantity": 1,
    "paymentMethodId": 2
    "timestamp": "2021-02-09T11:12:17+0000"
}
```

Event Broker
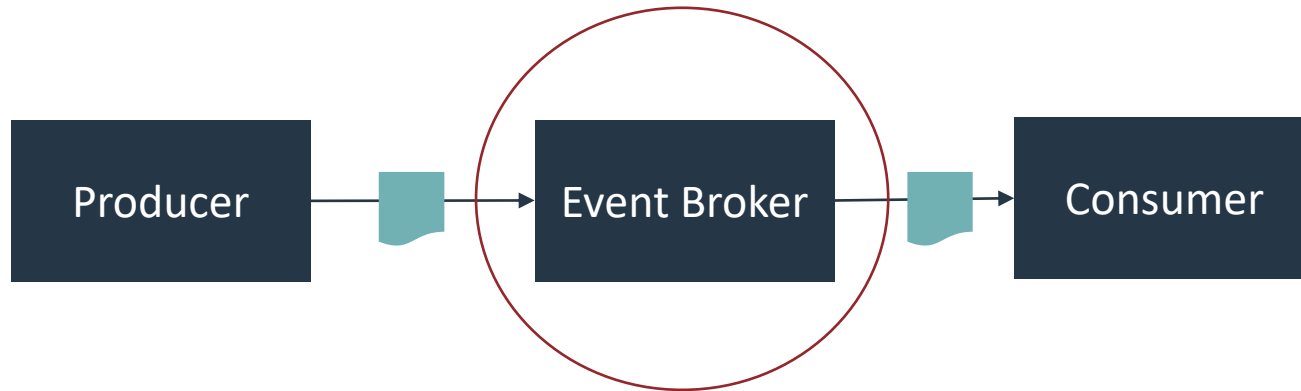
Event Broker Organization

Event Broker Organization

Bank Account Event Stream

Account Topic / Partition 1

Most Recent ◀─────────────────────────────────▶ Oldest

Bank Account Event Stream

Account Topic / Partition 1

Jan 6, 1:10pm
AccountCreated
AccountId: 4354

Most Recent ← → Oldest

Bank Account Event Stream

Account Topic / Partition 1

Jan 6, 1:30pm
CashDeposited
AccountId: 4354
Amount: $500

Jan 6, 1:10pm
AccountCreated
AccountId: 4354

Most Recent   Oldest

Bank Account Event Stream

Account Topic / Partition 1

Jan 12, 2:33am
CheckProcessed
AccountId: 4354
Amount: $75

Jan 6, 1:30pm
CashDeposited
AccountId: 4354
Amount: $500

Jan 6, 1:10pm
AccountCreated
AccountId: 4354

Most Recent

Oldest

Bank Account Event Stream

Account Topic / Partition 1

Jan 17, 10:45am	Jan 12, 2:33am	Jan 6, 1:30pm	Jan 6, 1:10pm
AtmWithdrawl	CheckProcessed	CashDeposited	AccountCreated
AccountId: 4354	AccountId: 4354	AccountId: 4354	AccountId: 4354
Amount: $40	Amount: $75	Amount: $500	

Most Recent ← Oldest

Bank Account Event Stream

Account Topic / Partition 1

Jan 19, 2:33am	Jan 17, 10:45am	Jan 12, 2:33am	Jan 6, 1:30pm	Jan 6, 1:10pm
DepositReceived	AtmWithdrawl	CheckProcessed	CashDeposited	AccountCreated
AccountId: 4354	AccountId: 4354	AccountId: 4354	AccountId: 4354	AccountId: 4354
Amount: $650	Amount: $40	Amount: $75	Amount: $500	

Most Recent   Oldest

Bank Account Event Stream

Account Topic / Partition 1

Jan 30, 12:11am	Jan 19, 2:33am	Jan 17, 10:45am	Jan 12, 2:33am	Jan 6, 1:30pm	Jan 6, 1:10pm
CheckProcessed	DepositReceived	AtmWithdrawl	CheckProcessed	CashDeposited	AccountCreated
AccountId: 4354	AccountId: 4354	AccountId: 4354	AccountId: 4354	AccountId: 4354	AccountId: 4354
Amount: $475	Amount: $650	Amount: $40	Amount: $75	Amount: $500	

Most Recent ← → Oldest
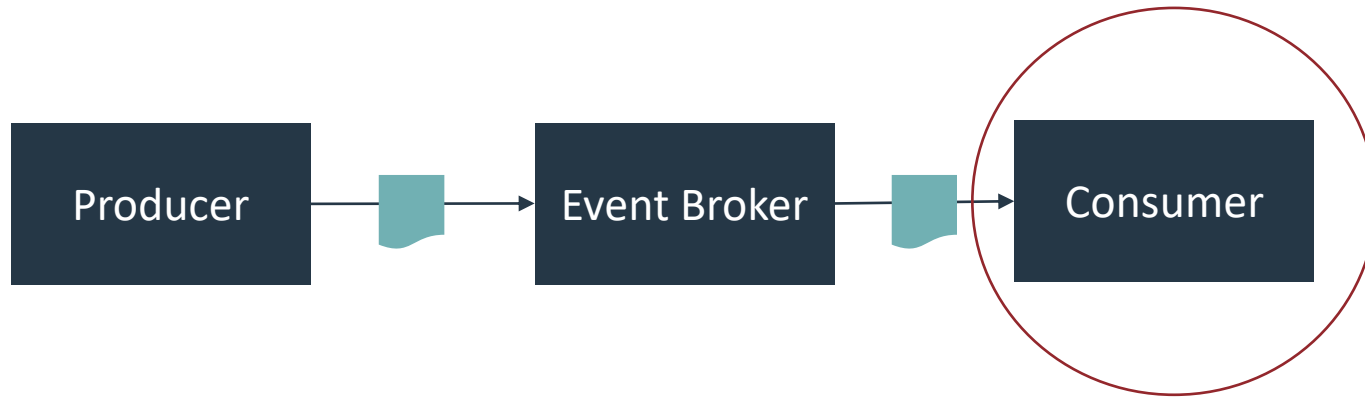
Producer Details

Publishing Various Event Types
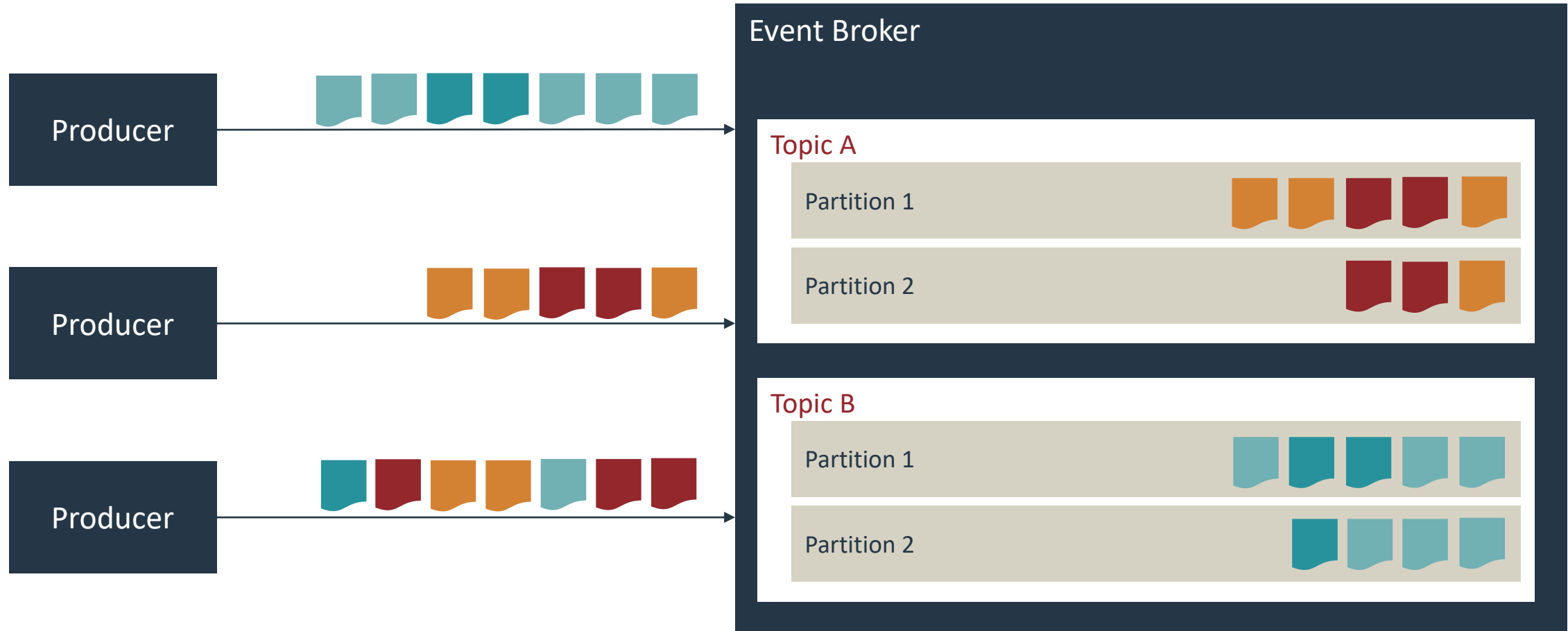
Consumer Details

Ordered Consumption

Account Topic / Partition 1

| Offset 5 | Offset 4 | Offset 3 | Offset 2 | Offset 1 | Offset 0 |

Jan 30, 12:11am
CheckProcessed
AccountId: 4354
Amount: $475

Jan 19, 2:33am
DepositReceived
AccountId: 4354
Amount: $650

Jan 17, 10:45am
AtmWithdrawl
AccountId: 4354
Amount: $40

Jan 12, 2:33am
CheckProcessed
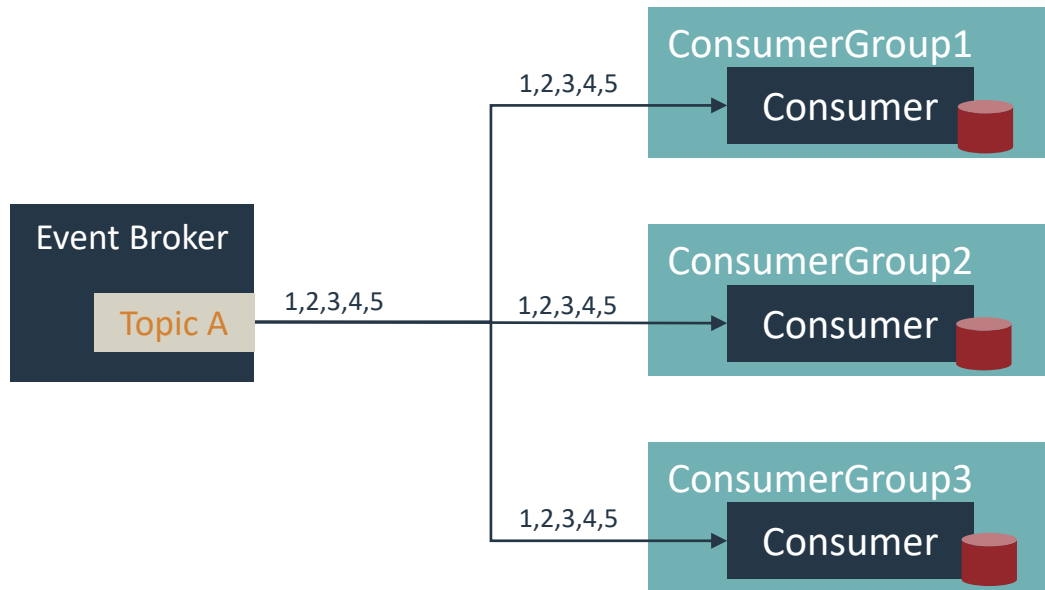AccountId: 4354
Amount: $75

Jan 6, 1:30pm
CashDeposited
AccountId: 4354
Amount: $500

Jan 6, 1:10pm
AccountCreated
AccountId: 4354

Most Recent — Oldest

CreditCheck
Service
LastRead: 4

Reporting
Service
LastRead: 1

Scaling with Consumer Groups

Multiple Consumer Groups

Event Broker — Topic A → 1,2,3,4,5

ConsumerGroup1 — 1,2,3,4,5 → Consumer

ConsumerGroup2 — 1,2,3,4,5 → Consumer

ConsumerGroup3 — 1,2,3,4,5 → Consumer

Scaling with Consumer Groups

Multiple Consumer Groups

Event Broker

Topic A

1,2,3,4,5

ConsumerGroup1
1,2,3,4,5
Consumer

ConsumerGroup2
1,2,3,4,5
Consumer

ConsumerGroup3
1,2,3,4,5
Consumer

Multiple Consumers In Single Group

Event Broker

Topic A

1,2,3,4,5

ConsumerGroup1

1,4
Consumer

2,5
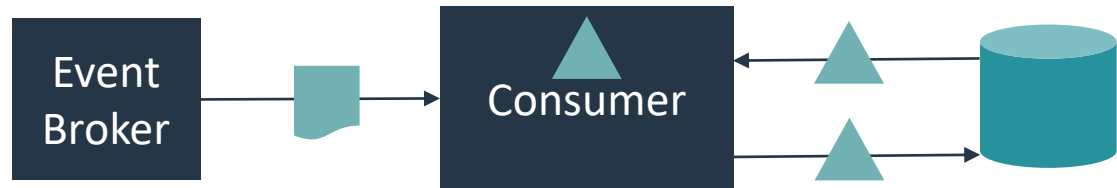Consumer

3
Consumer

Consumer Data Management

- Consumers may not act on received events immediately
  - May collect data needed to process later events

- Helps decouple microservices

Consumer Data Management

Projections / Material Views

Event Broker → Consumer ⇄ [database]

Consumer Data Management
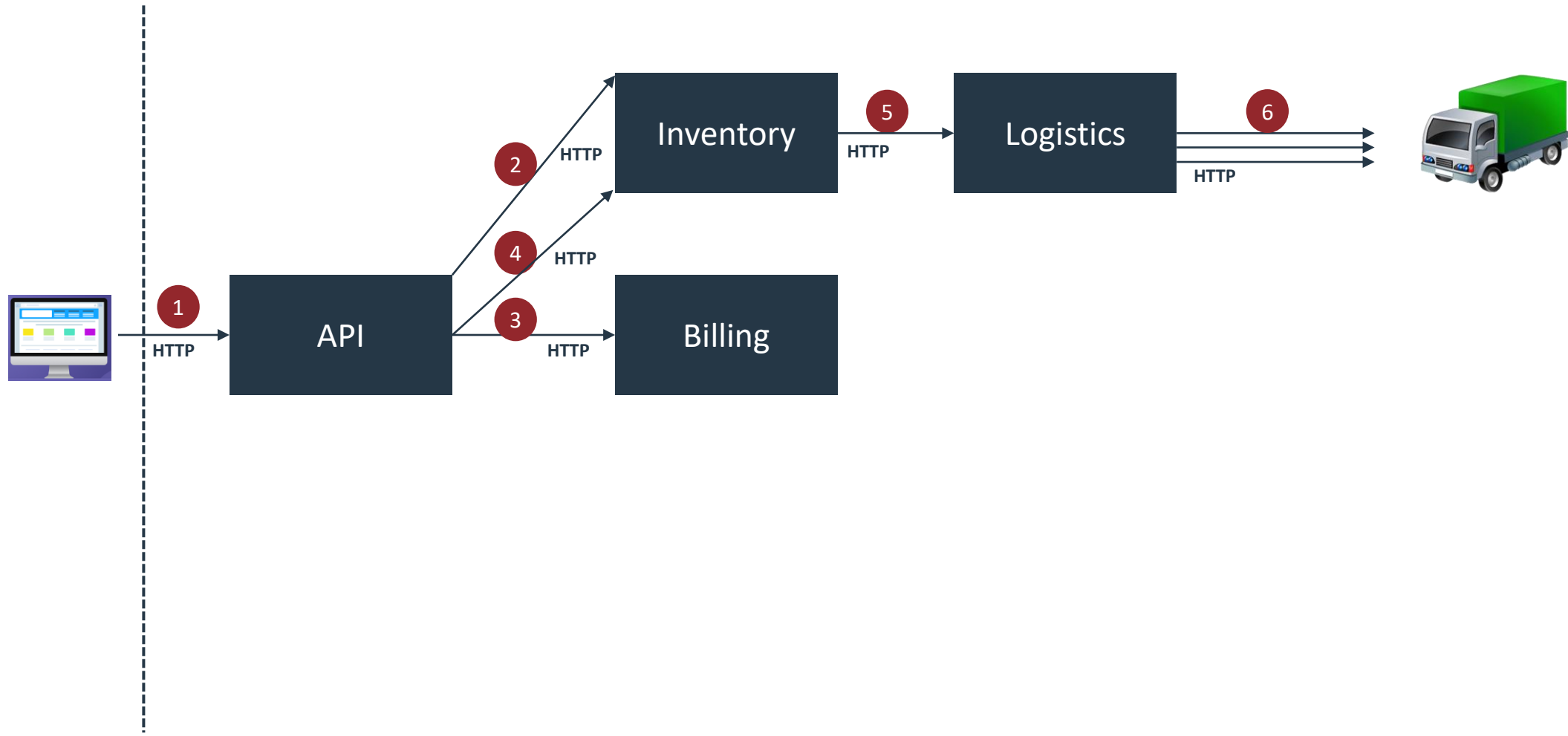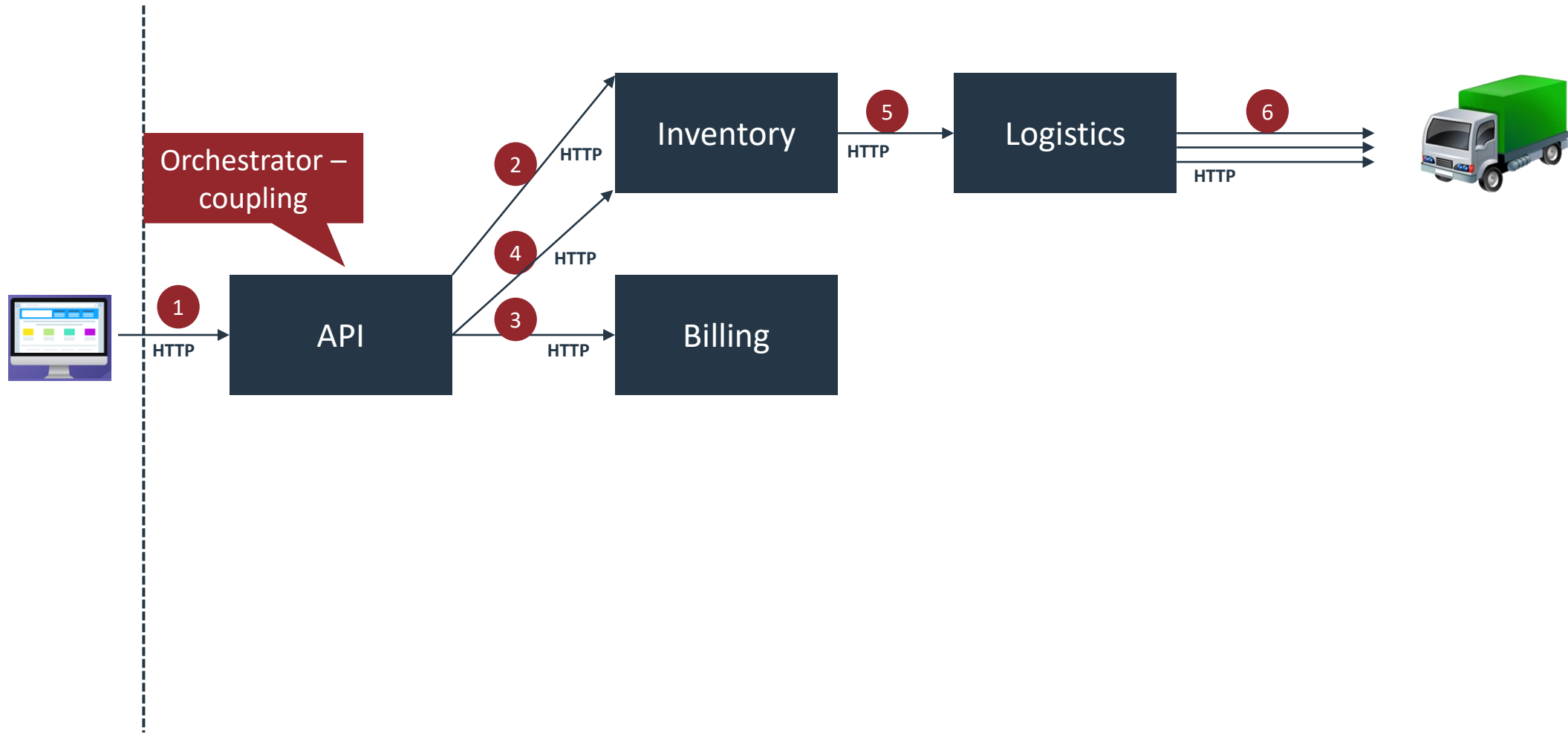
Request-Based vs Event-Based Solution Comparison

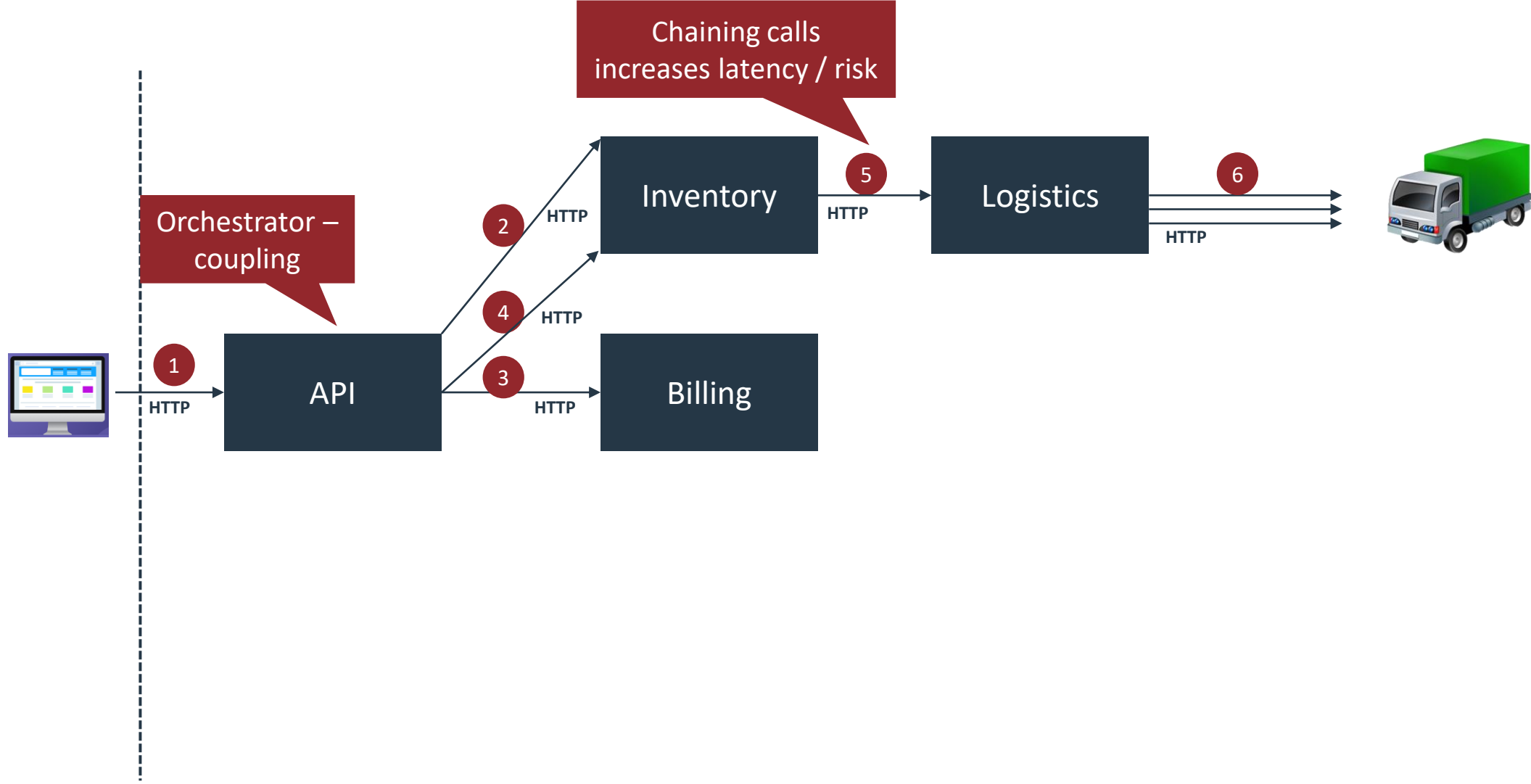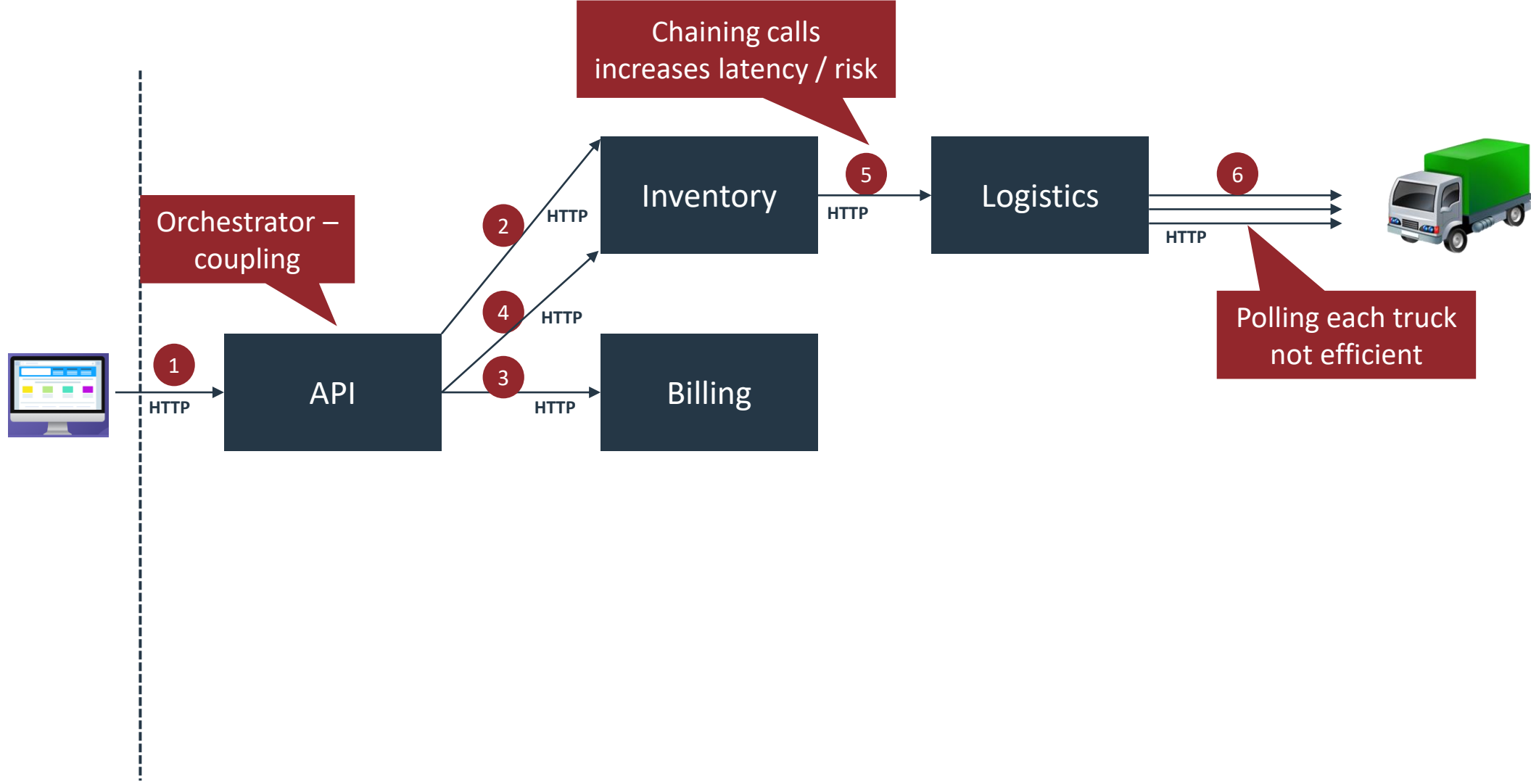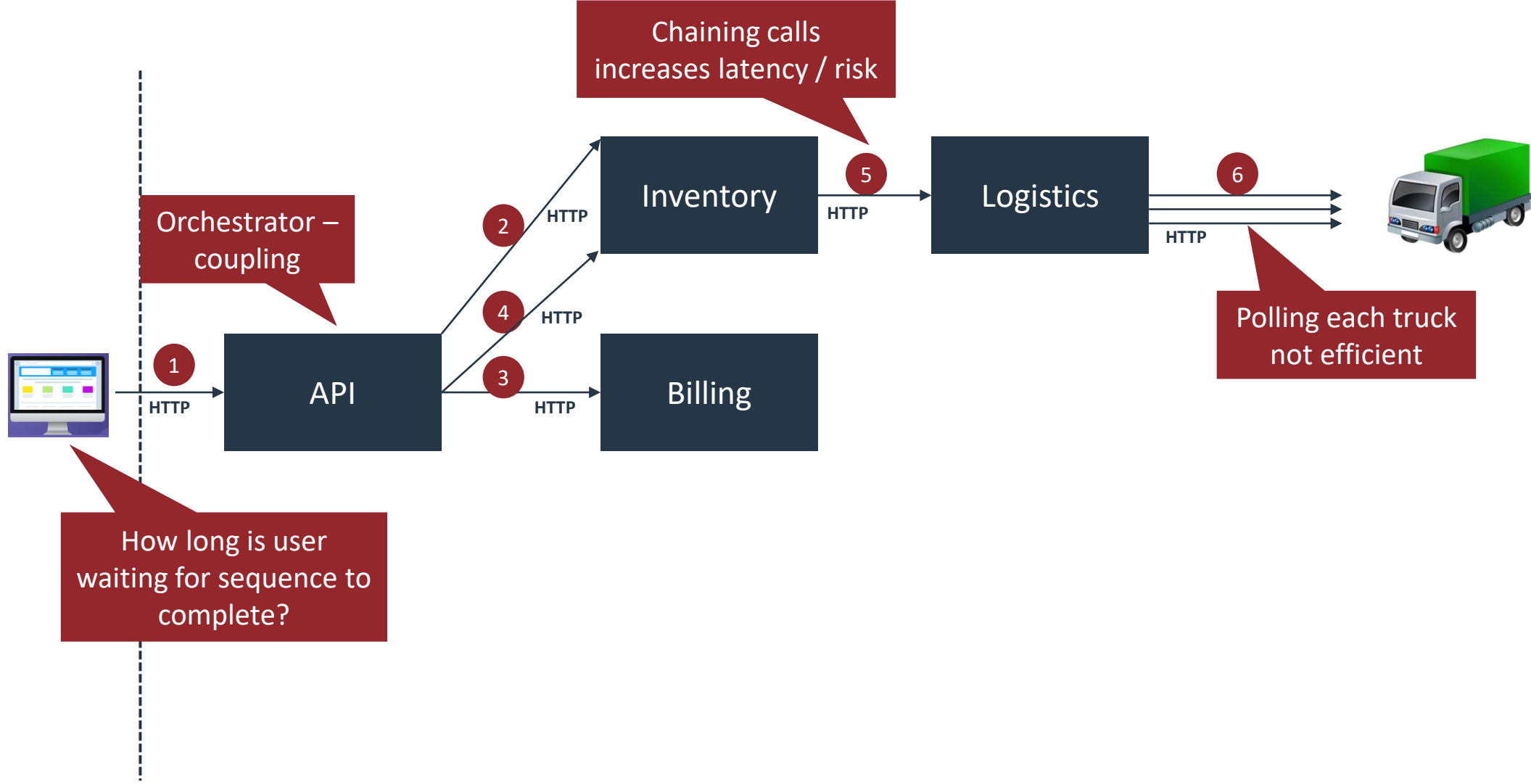# Request-Based Point of Sale

Request-Based Point of Sale

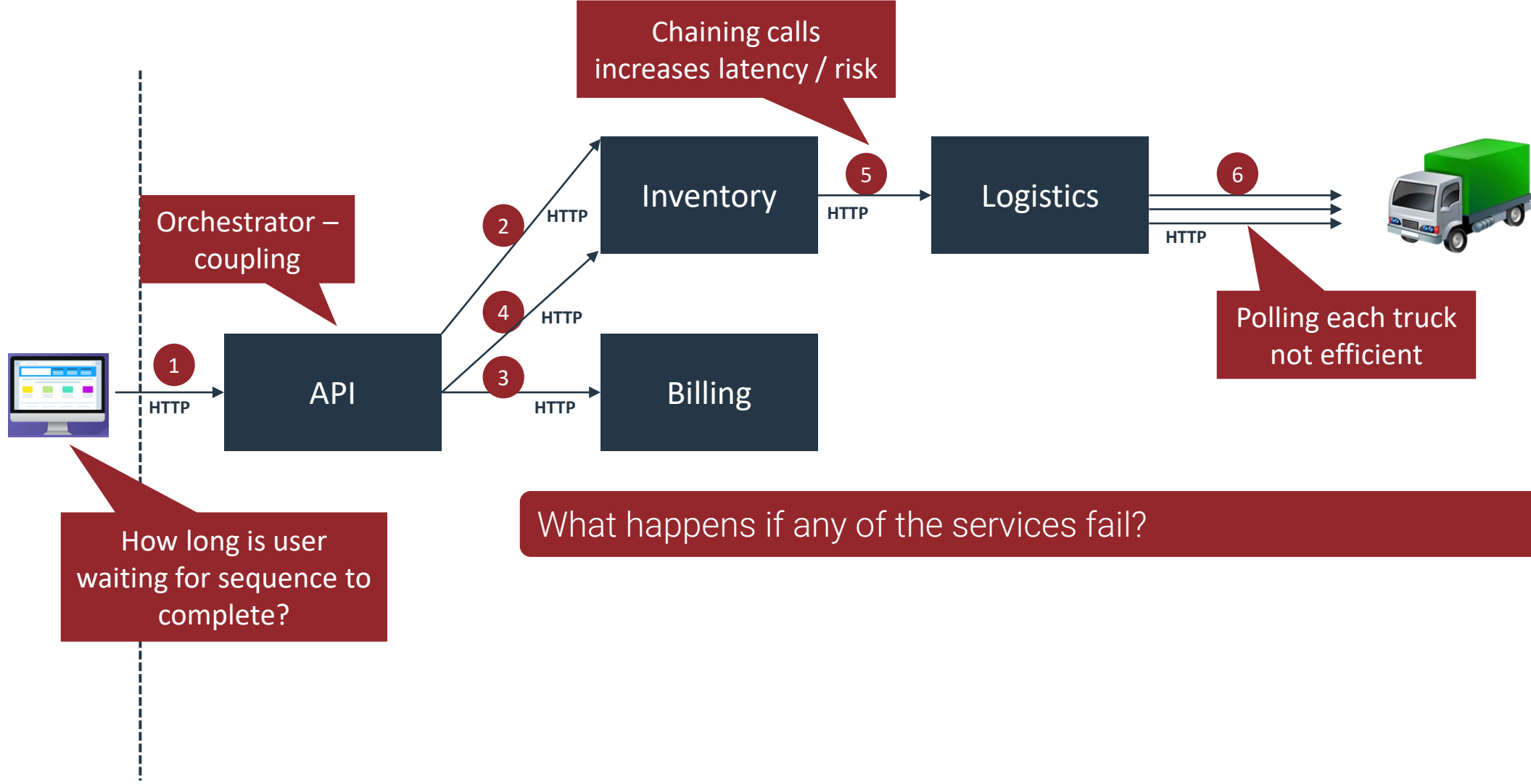# Request-Based Point of Sale

Request-Based Point of Sale
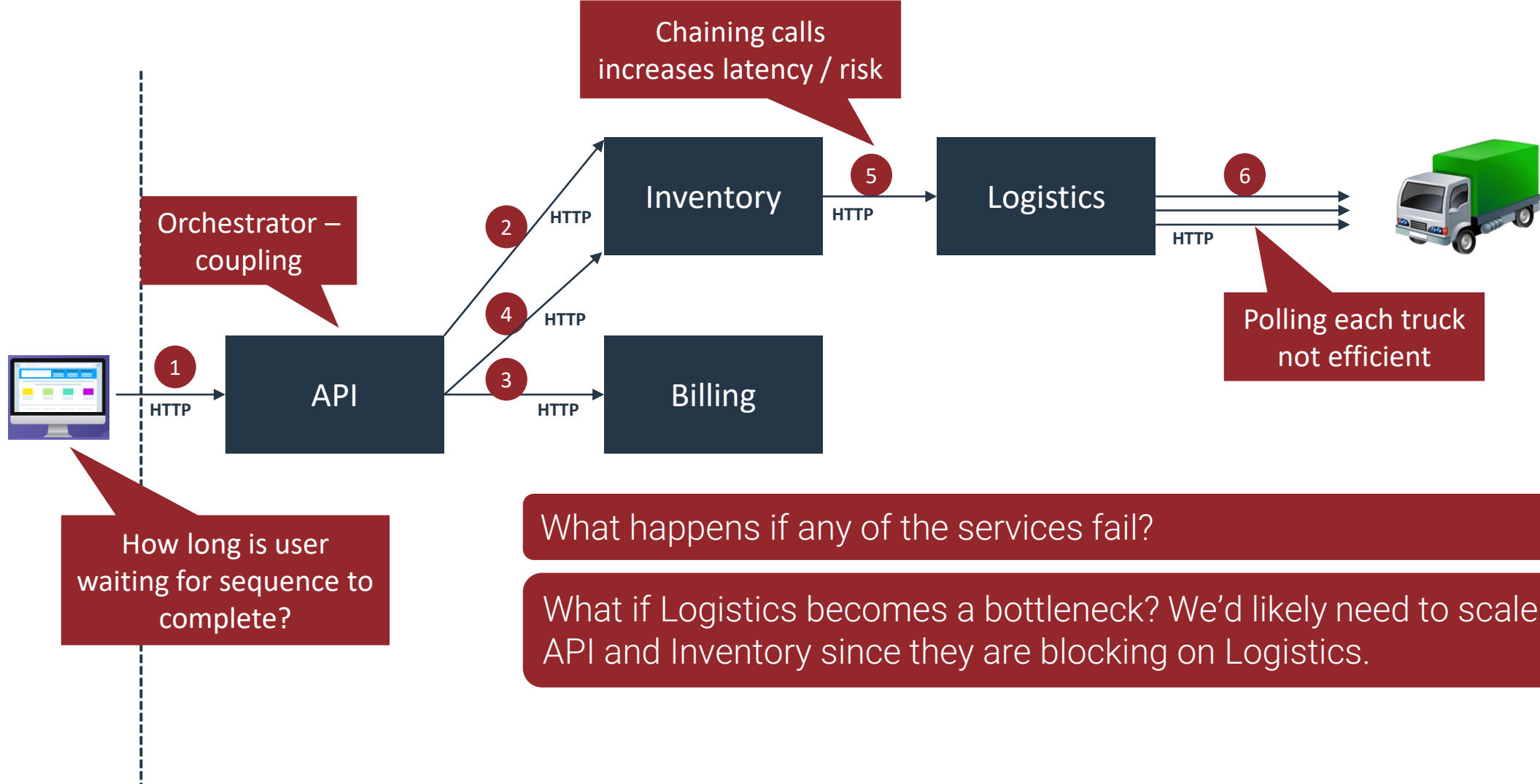
Request-Based Point of Sale

Chaining calls increases latency / risk

Orchestrator – coupling

Polling each truck not efficient

How long is user waiting for sequence to complete?

API

Inventory

Billing

Logistics

1 — HTTP

2 — HTTP

3 — HTTP

4 — HTTP

5 — HTTP

6 — HTTP

Request-Based Point of Sale

Chaining calls increases latency / risk

Orchestrator – coupling

Inventory

5 HTTP

Logistics

6 HTTP

2 HTTP

4 HTTP

1 HTTP

API

3 HTTP

Billing

Polling each truck not efficient

How long is user waiting for sequence to complete?

What happens if any of the services fail?

Request-Based Point of Sale

Chaining calls increases latency / risk

Orchestrator – coupling

Polling each truck not efficient

How long is user waiting for sequence to complete?

Inventory

Logistics

API

Billing
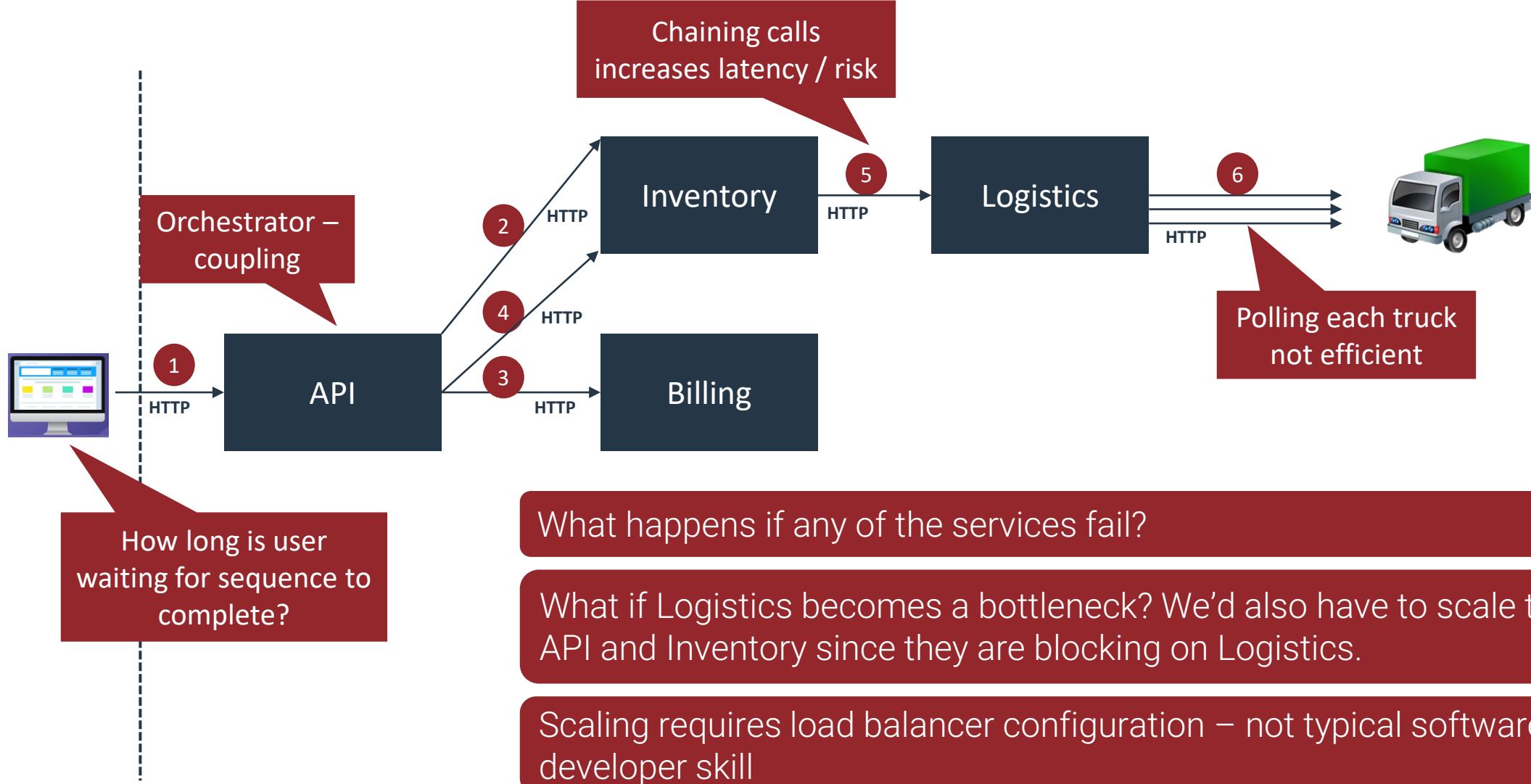
What happens if any of the services fail?

What if Logistics becomes a bottleneck? We'd likely need to scale the API and Inventory since they are blocking on Logistics.

Request-Based Point of Sale

Chaining calls increases latency / risk

Orchestrator – coupling

How long is user waiting for sequence to complete?

Polling each truck not efficient
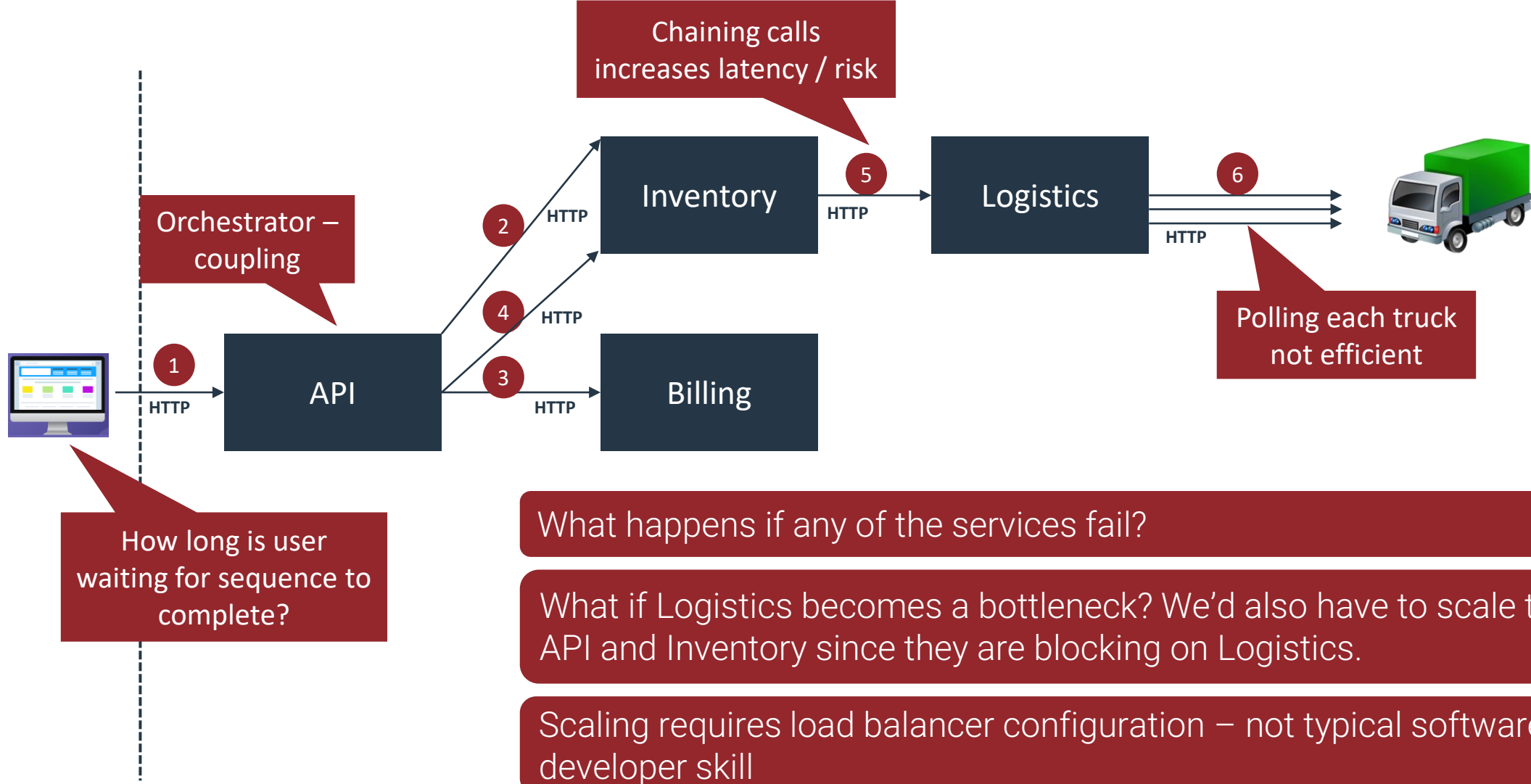
Inventory

Logistics

API

Billing

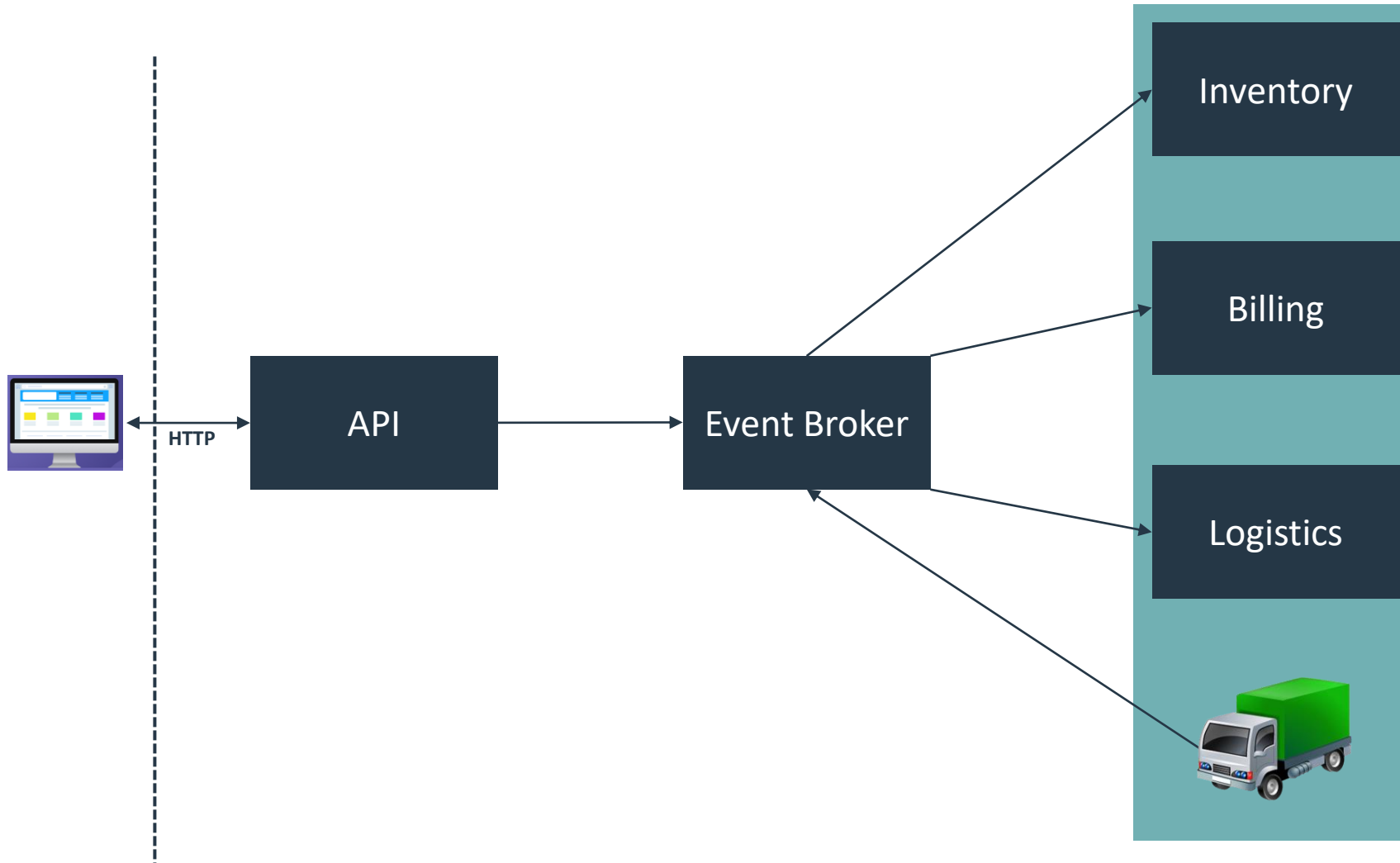What happens if any of the services fail?

What if Logistics becomes a bottleneck? We'd also have to scale the API and Inventory since they are blocking on Logistics.

Scaling requires load balancer configuration – not typical software developer skill
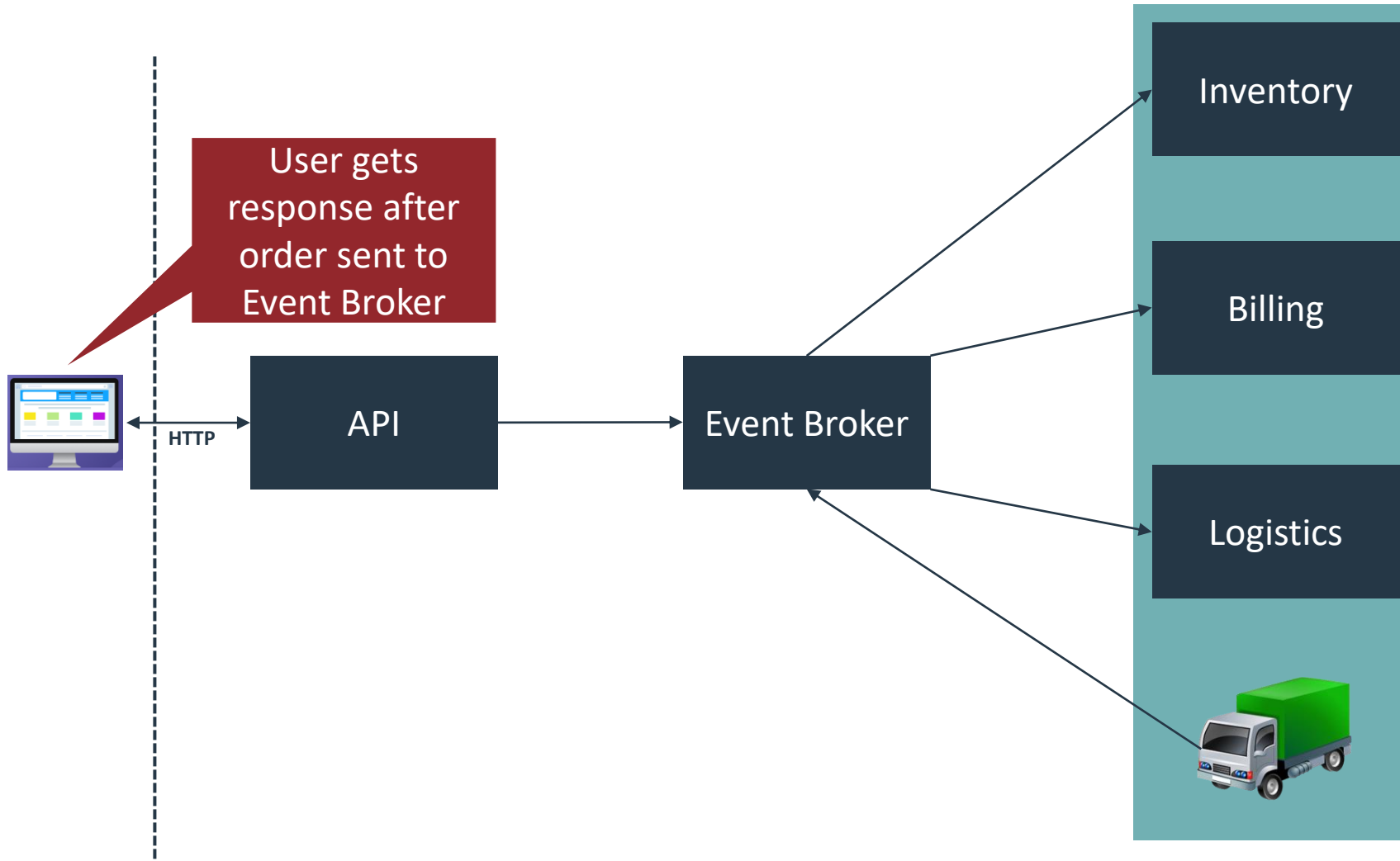
48

Request-Based Point of Sale

Chaining calls increases latency / risk

Orchestrator – coupling

Inventory

Logistics

API

Billing

1 — HTTP

2 — HTTP

3 — HTTP

4 — HTTP

5 — HTTP

6 — HTTP

Polling each truck not efficient

How long is user waiting for sequence to complete?

What happens if any of the services fail?

What if Logistics becomes a bottleneck? We'd also have to scale the API and Inventory since they are blocking on Logistics.

Scaling requires load balancer configuration – not typical software developer skill

What if we want to add a Fraud or Recommendation service?
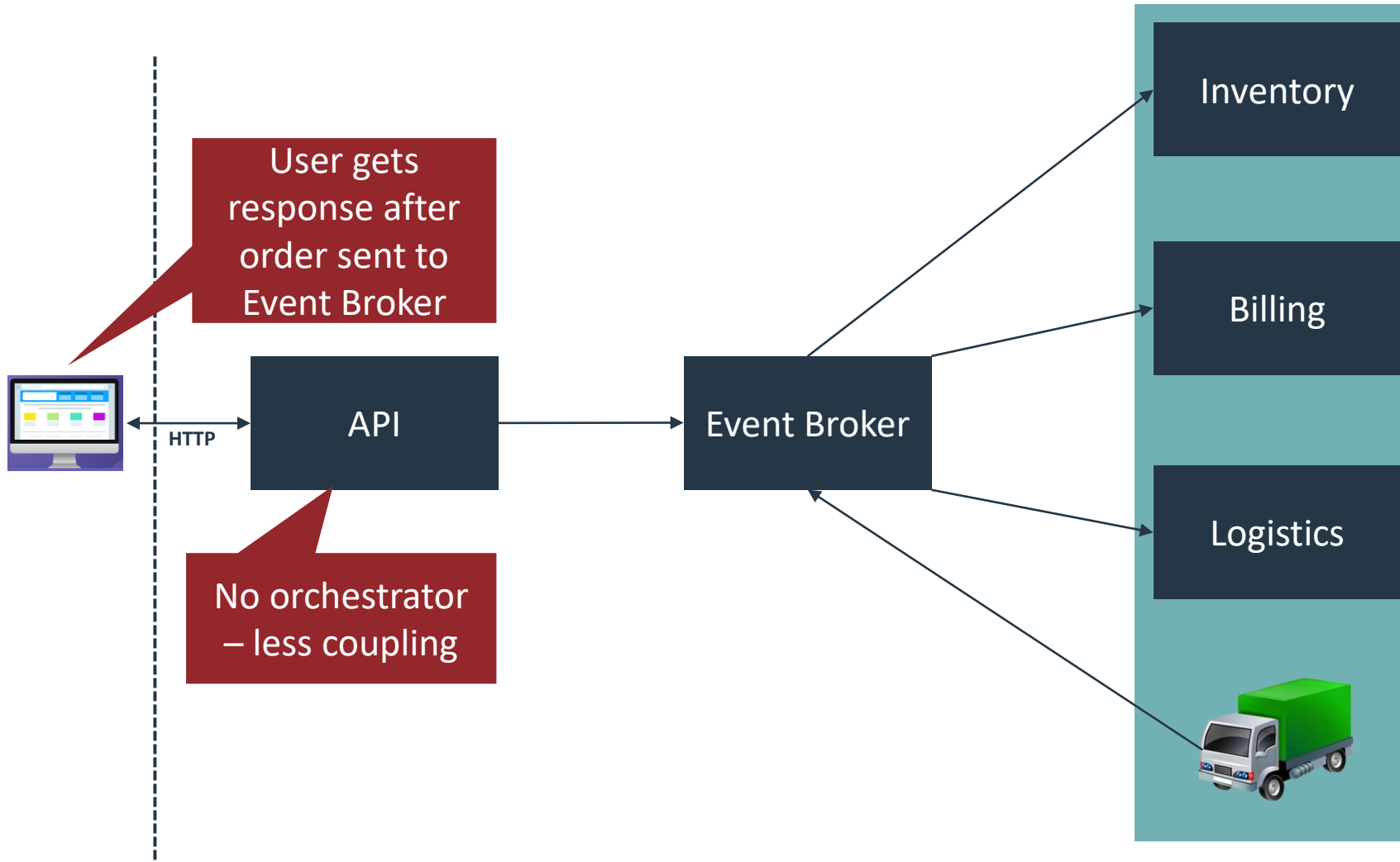
Event-Based Point of Sale
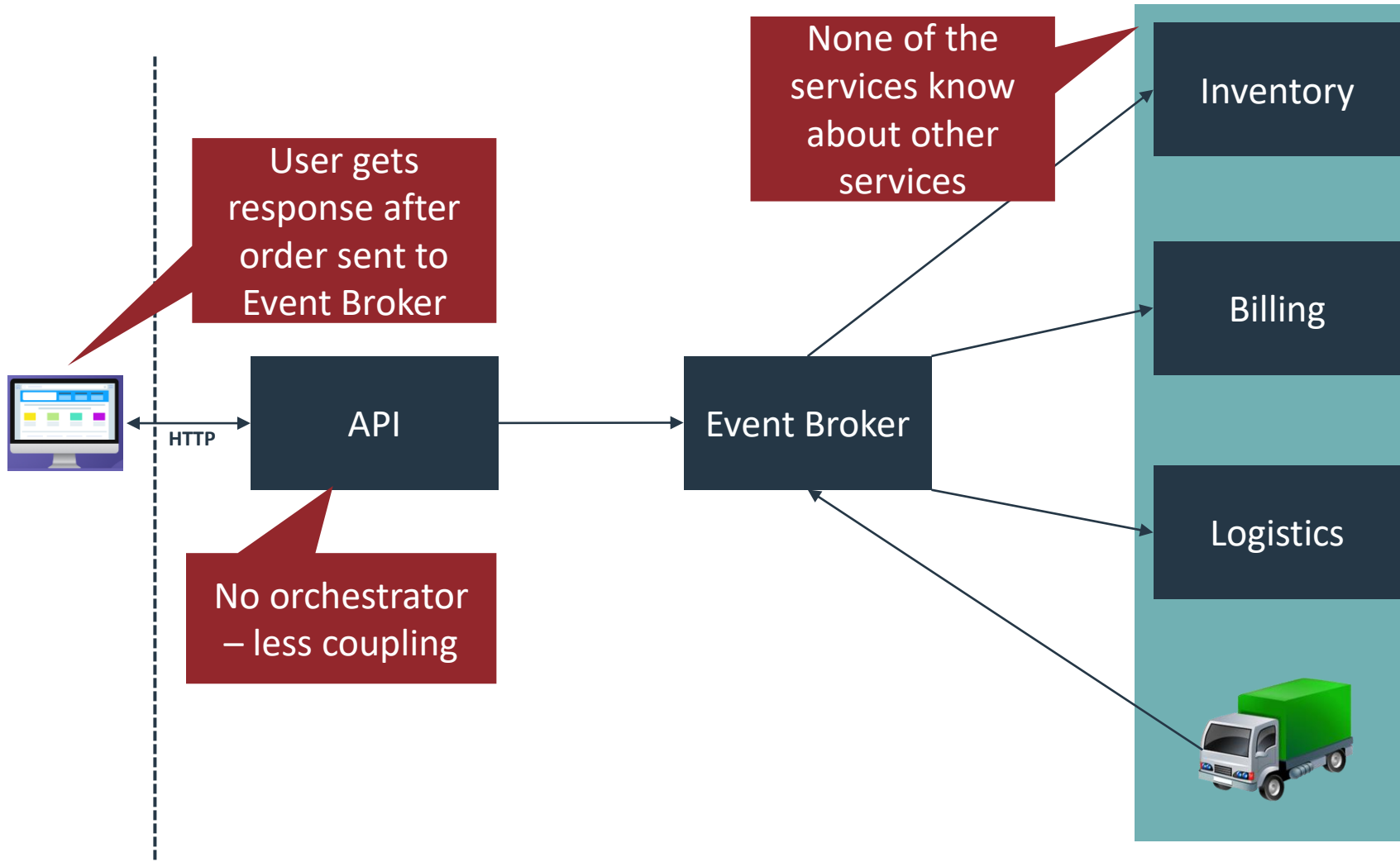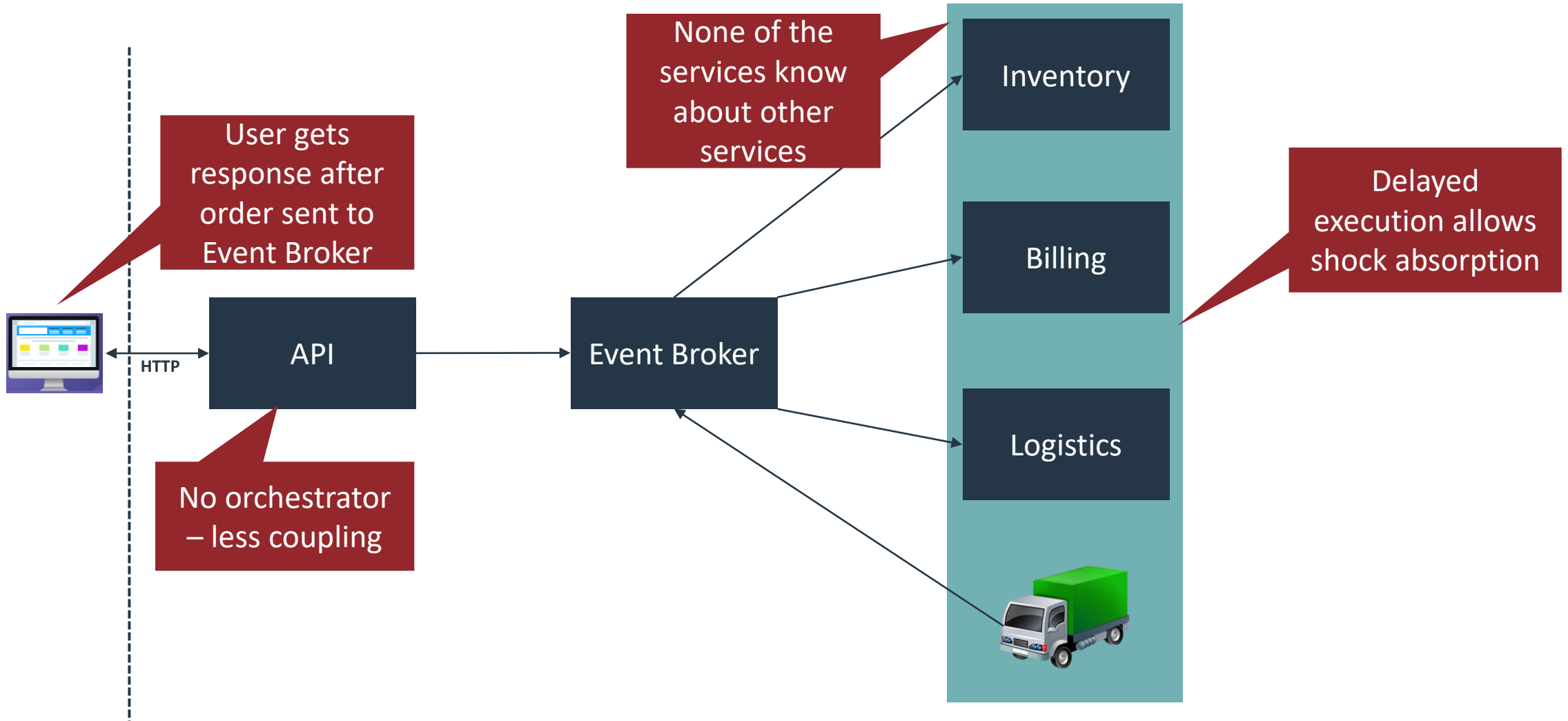
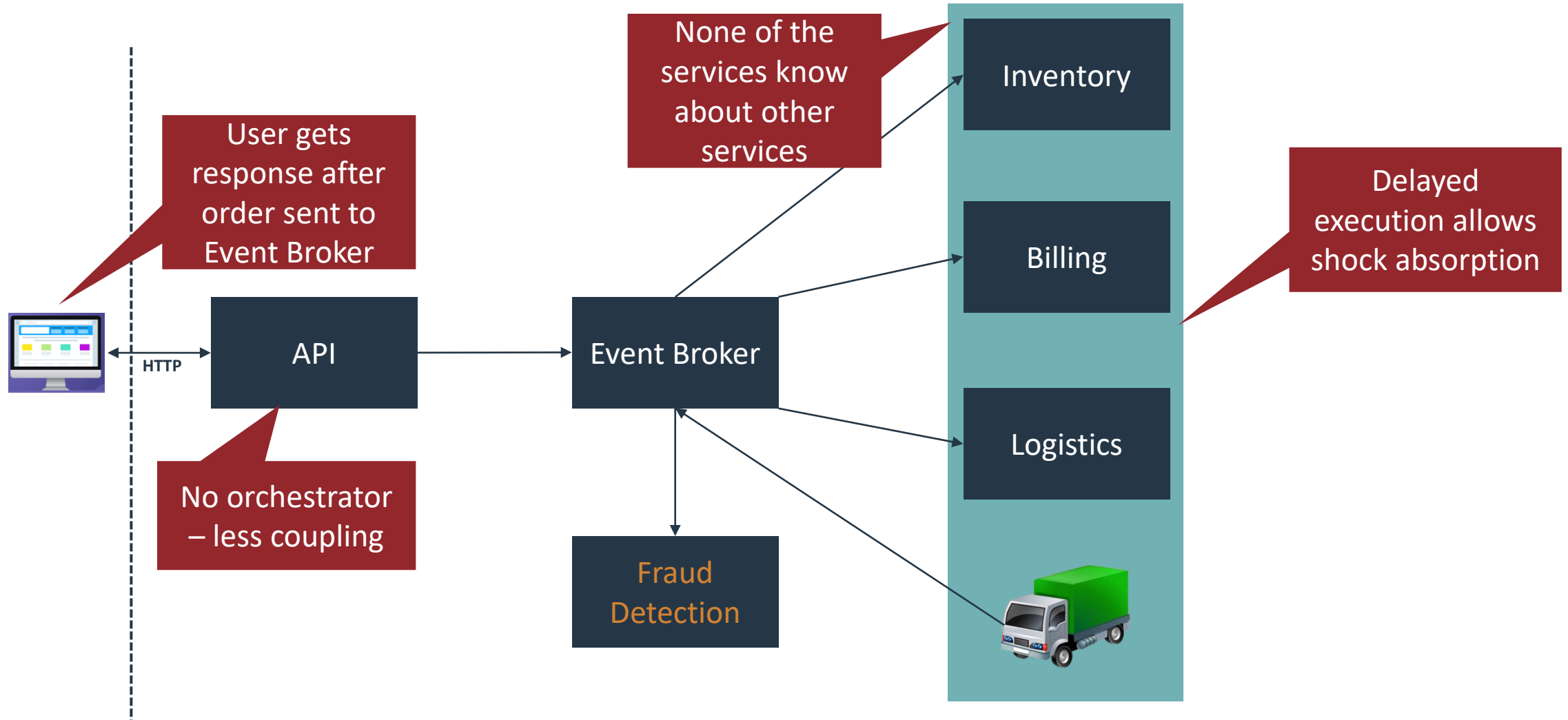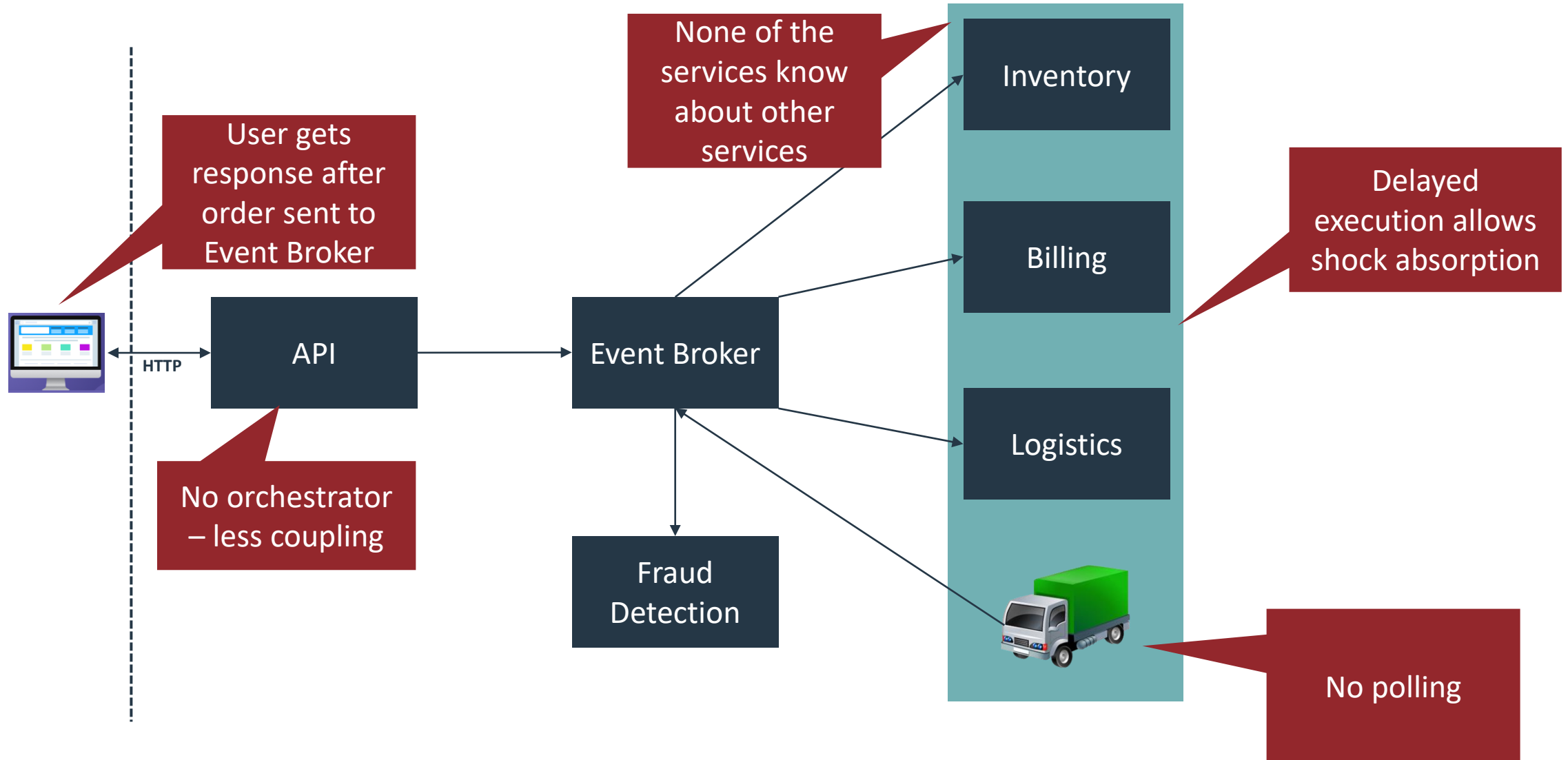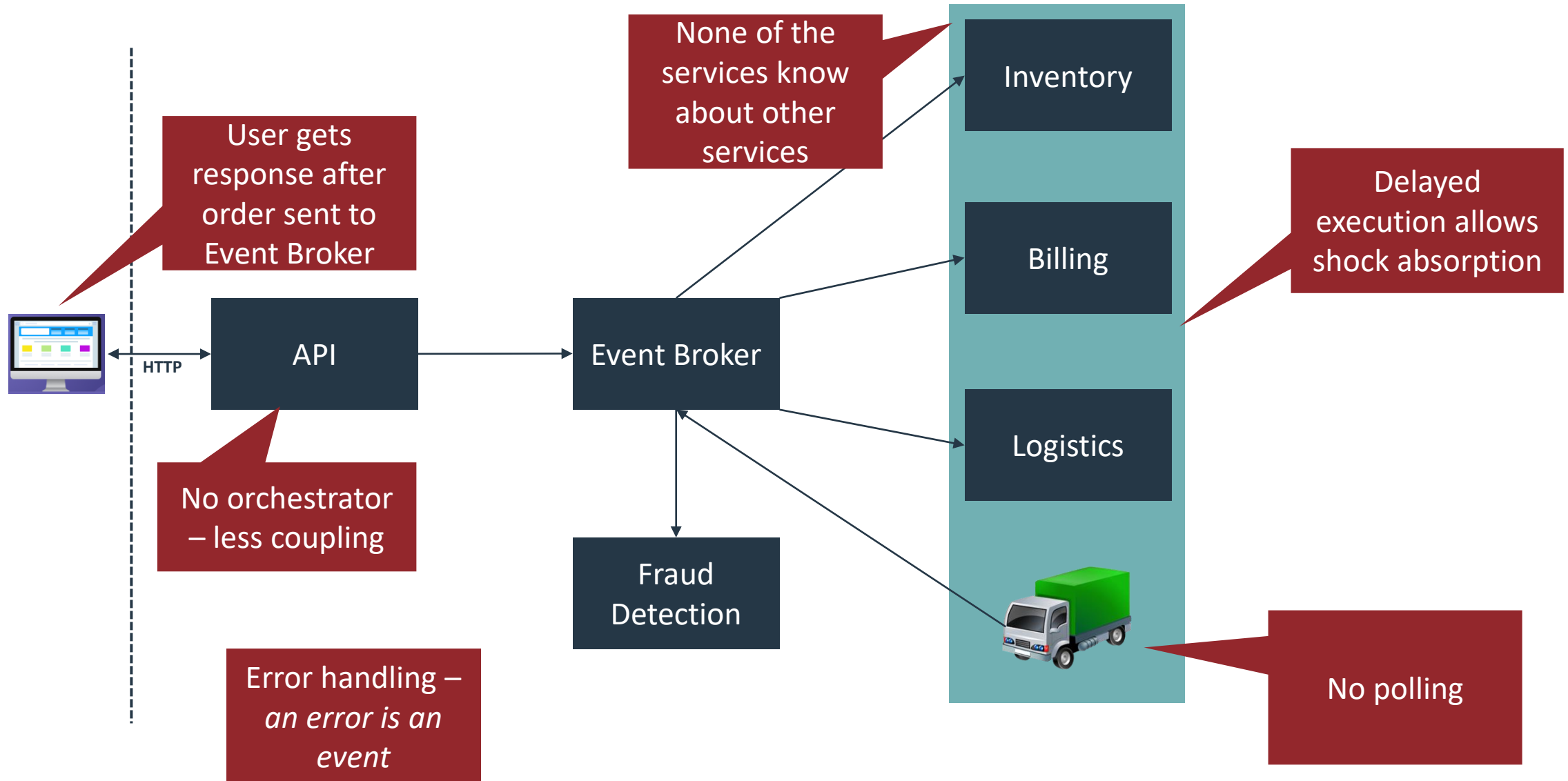# Event-Based Point of Sale

Event-Based Point of Sale

User gets response after order sent to Event Broker

API

HTTP

Event Broker

Inventory

Billing

Logistics

Event-Based Point of Sale

User gets response after order sent to Event Broker

No orchestrator – less coupling

API

HTTP

Event Broker

Inventory

Billing

Logistics

53

Event-Based Point of Sale

User gets response after order sent to Event Broker

None of the services know about other services

Inventory

No orchestrator – less coupling

HTTP

API

Event Broker

Billing

Logistics

Event-Based Point of Sale

User gets response after order sent to Event Broker

None of the services know about other services

Delayed execution allows shock absorption

No orchestrator – less coupling

HTTP

API

Event Broker

Inventory

Billing

Logistics

Event-Based Point of Sale

User gets response after order sent to Event Broker

None of the services know about other services

Delayed execution allows shock absorption

No orchestrator – less coupling

HTTP

API

Event Broker

Inventory

Billing

Logistics

Fraud Detection

Event-Based Point of Sale

User gets response after order sent to Event Broker

None of the services know about other services

Delayed execution allows shock absorption

No orchestrator – less coupling

No polling

HTTP

API

Event Broker

Inventory

Billing

Logistics

Fraud Detection

Event-Based Point of Sale

User gets response after order sent to Event Broker

No orchestrator – less coupling

Error handling – *an error is an event*

None of the services know about other services

Delayed execution allows shock absorption

No polling

HTTP

API

Event Broker

Fraud Detection

Inventory

Billing

Logistics

58

EDA Challenges

Event Broker Cluster
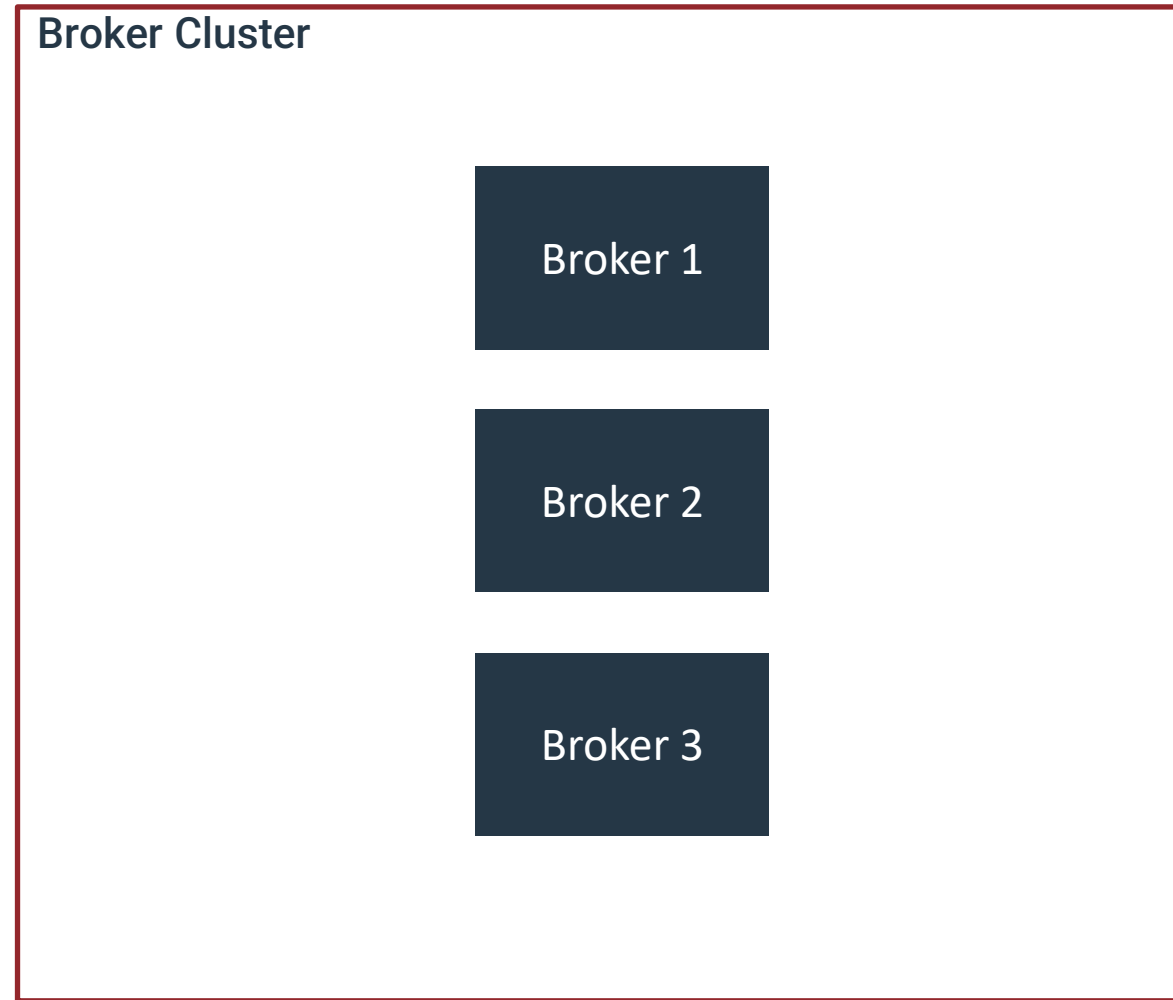
Notification System
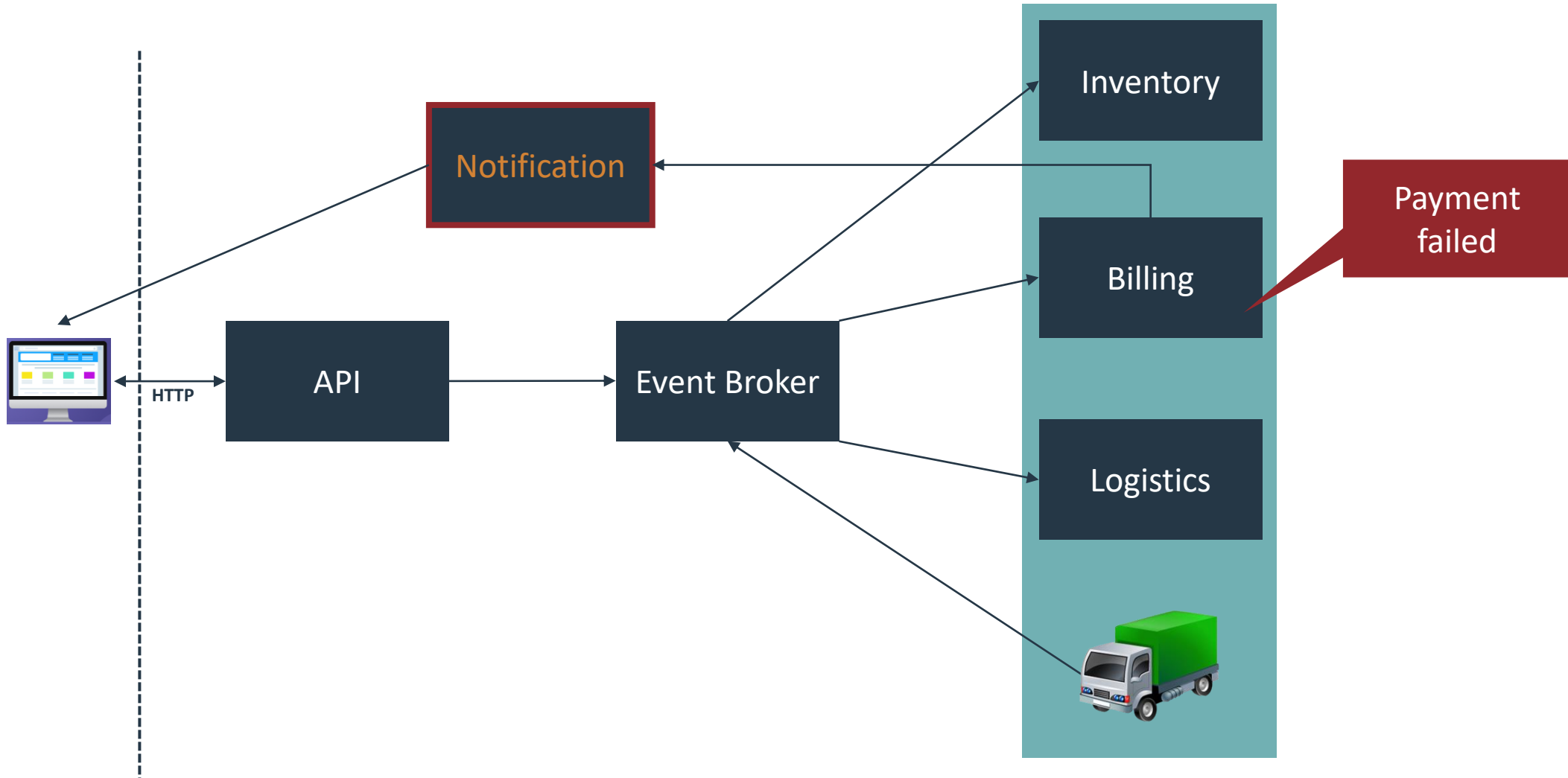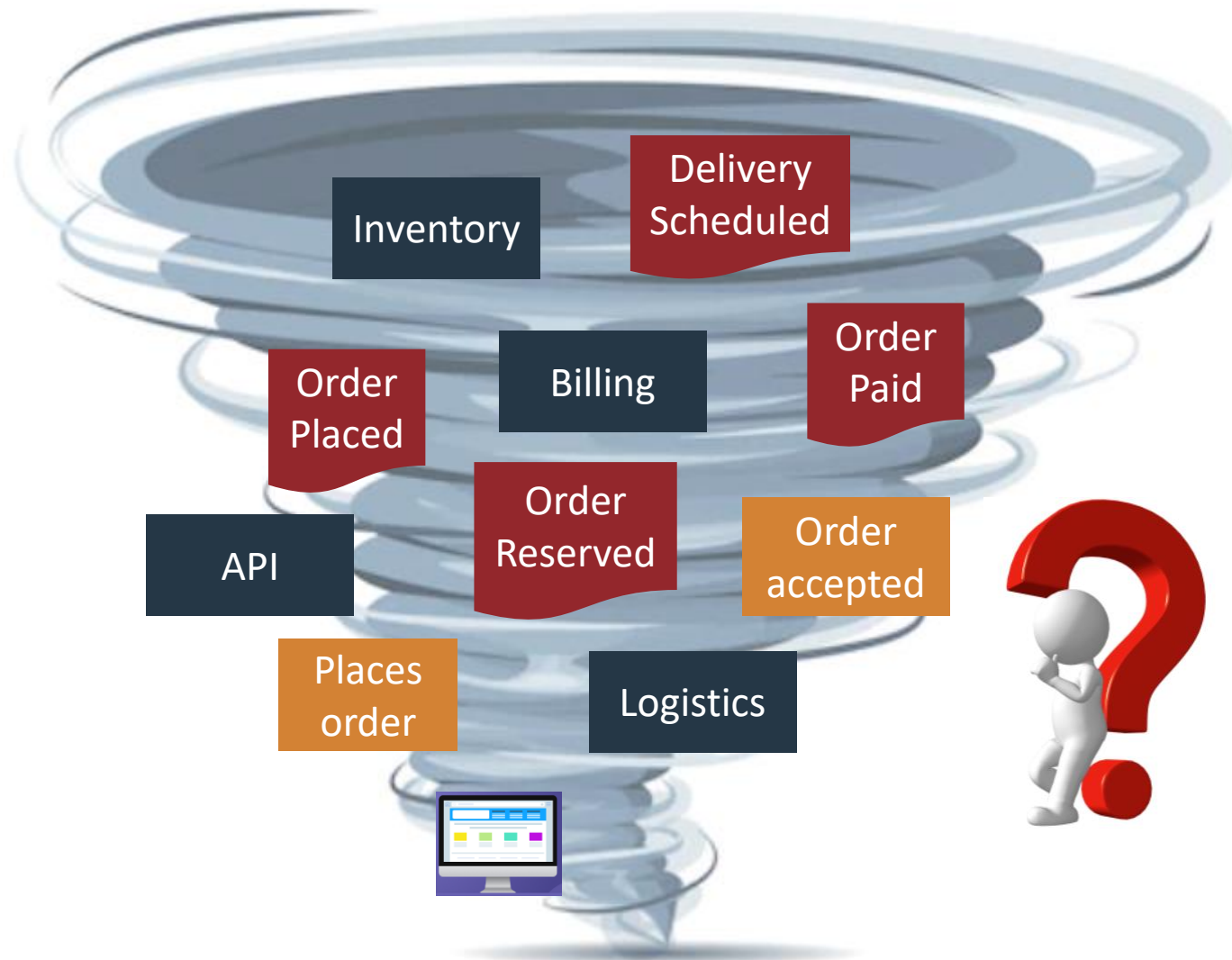
Topic / Event Design is Hard

Topic / Event Design

Topic Per Aggregate

Event Broker

Account Topic

Can hurt performance

Only needs new accounts

Consumer

Topic Per Event

Event Broker

Ordering difficult

AccountAdded Topic

MoneyDeposited Topic

MoneyWithdrawn Topic

FeeCharged Topic
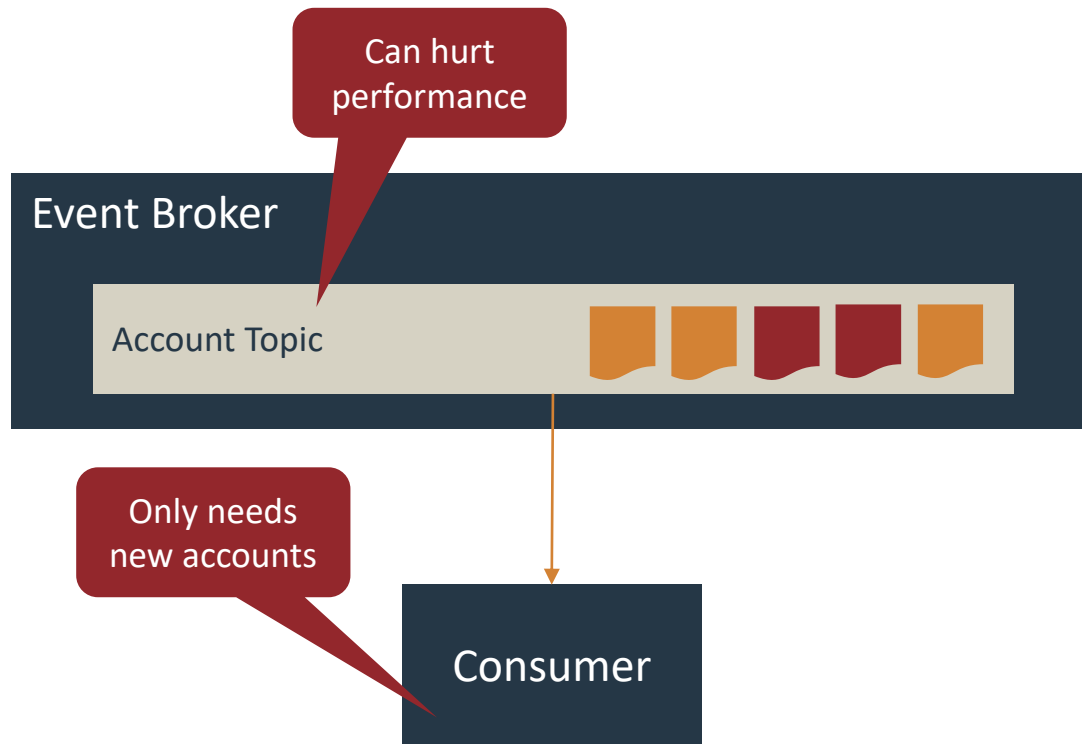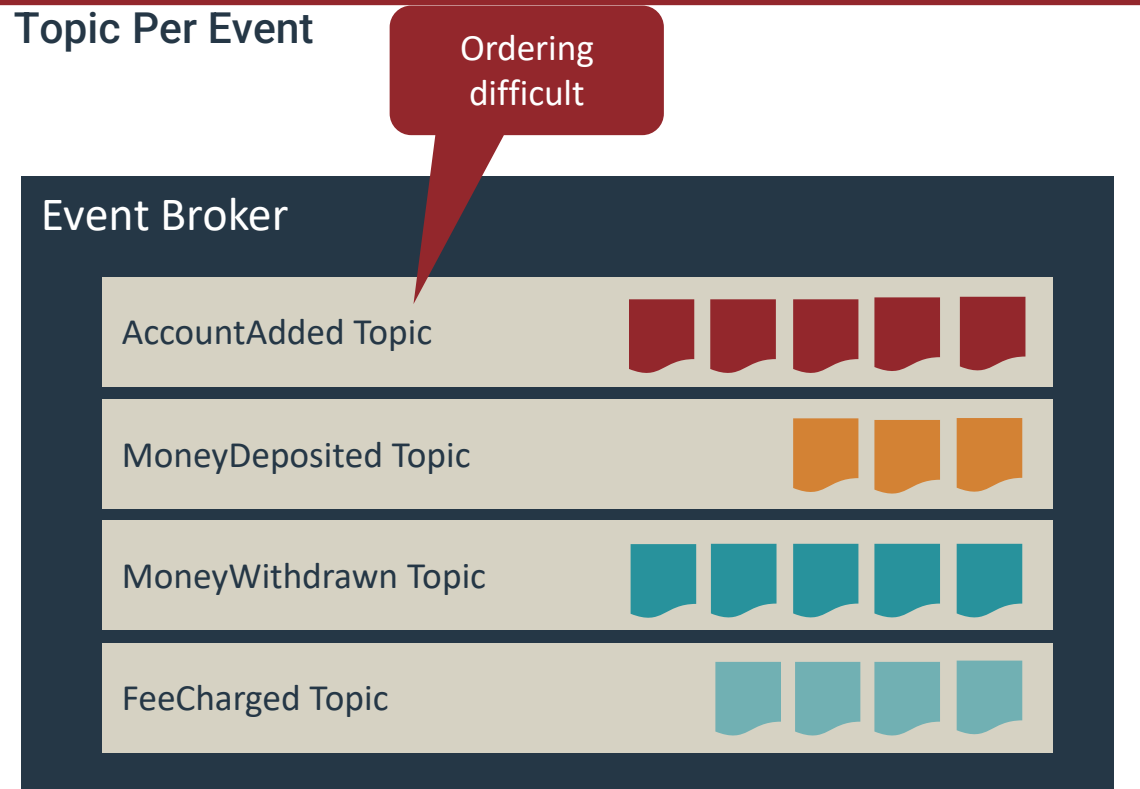
Complicated Troubleshooting

New Error Handling Concerns
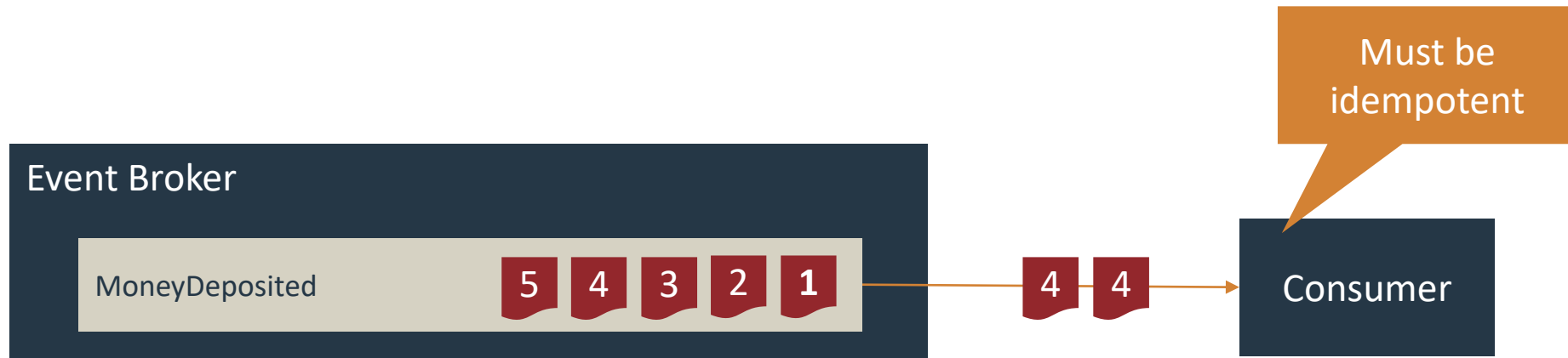
Getting Started with EDA

Create an Event Mindset
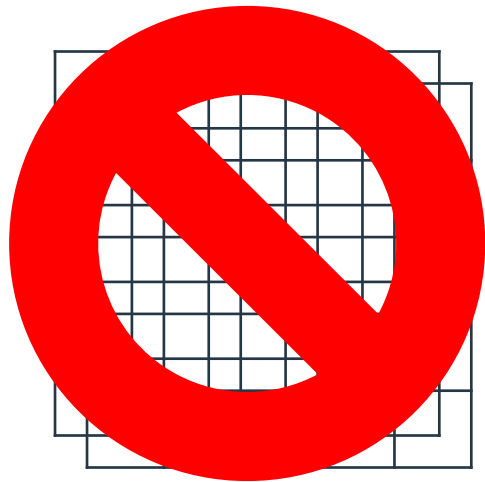
Practice Event Storming
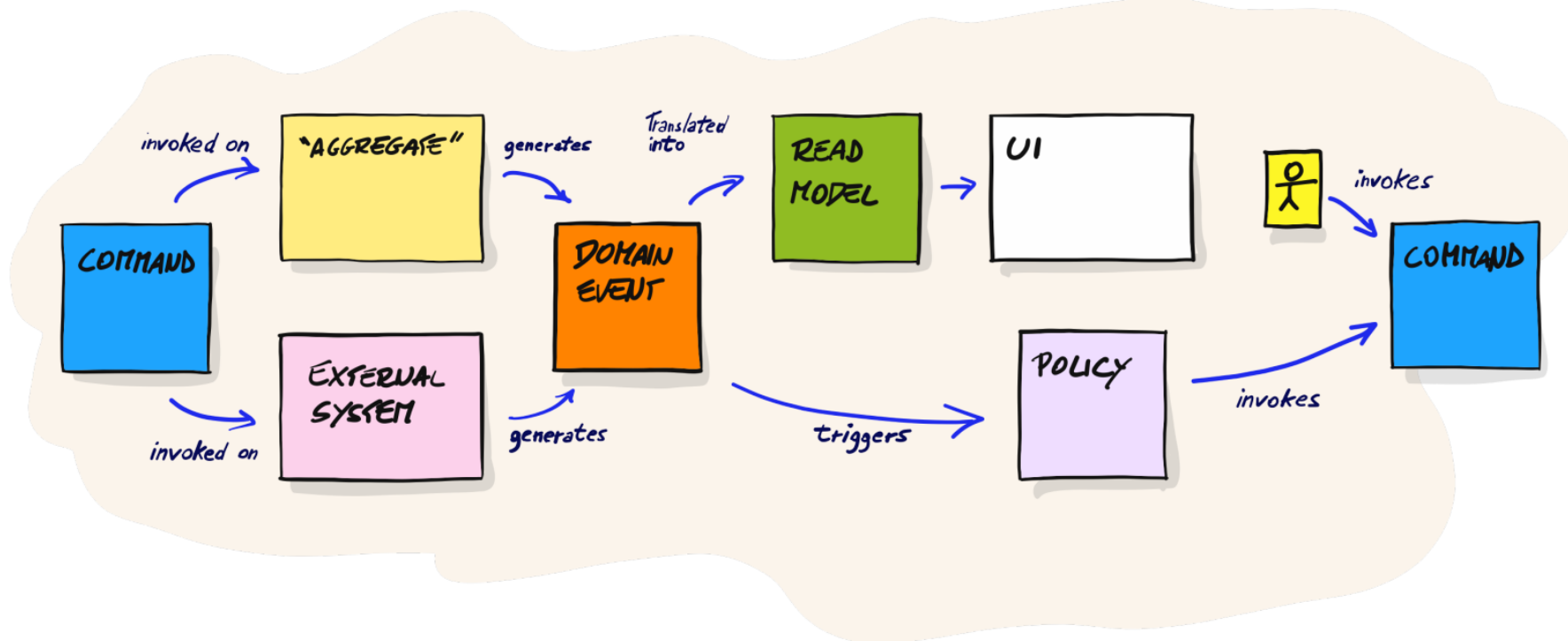
Learn About Messaging Options
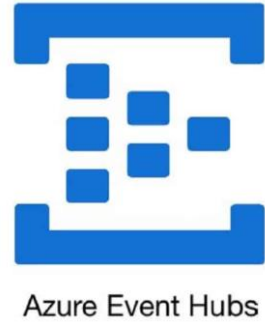
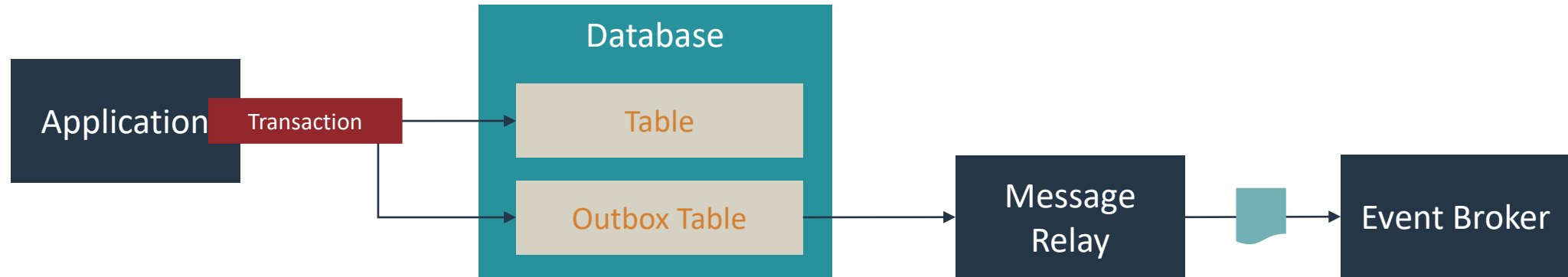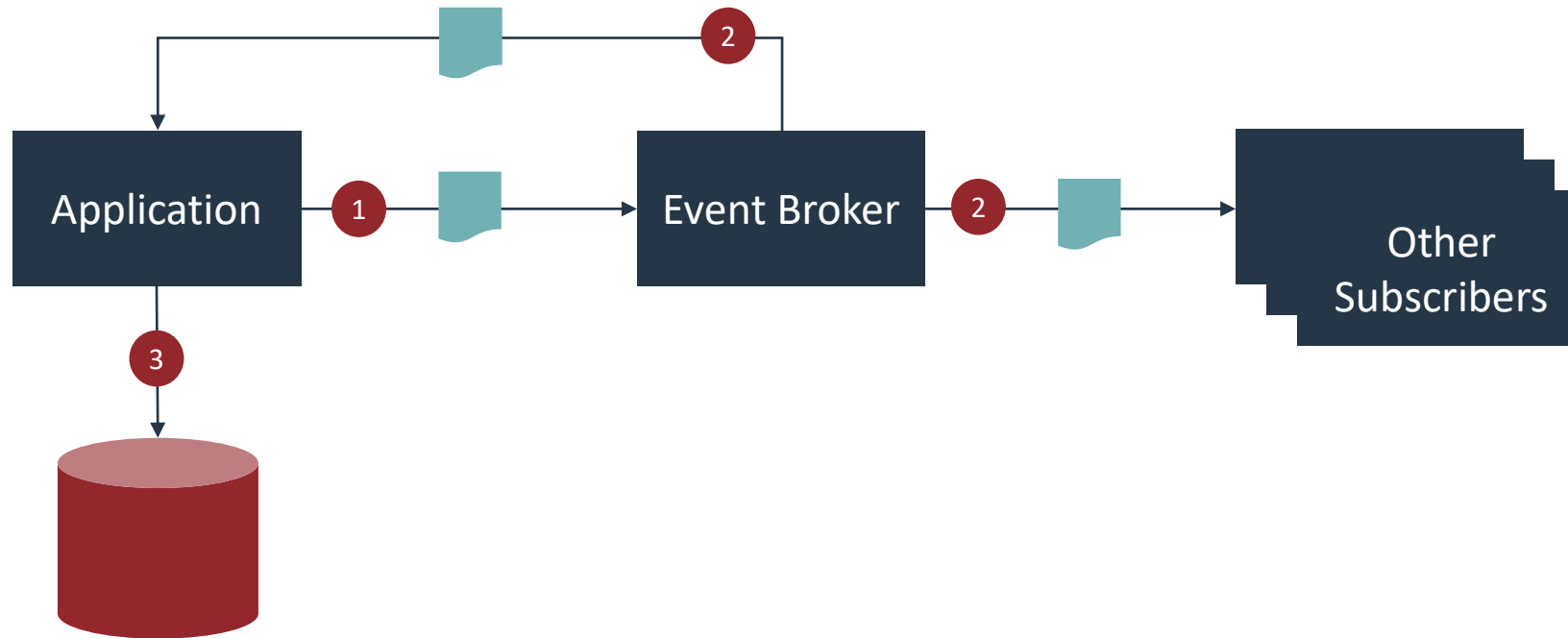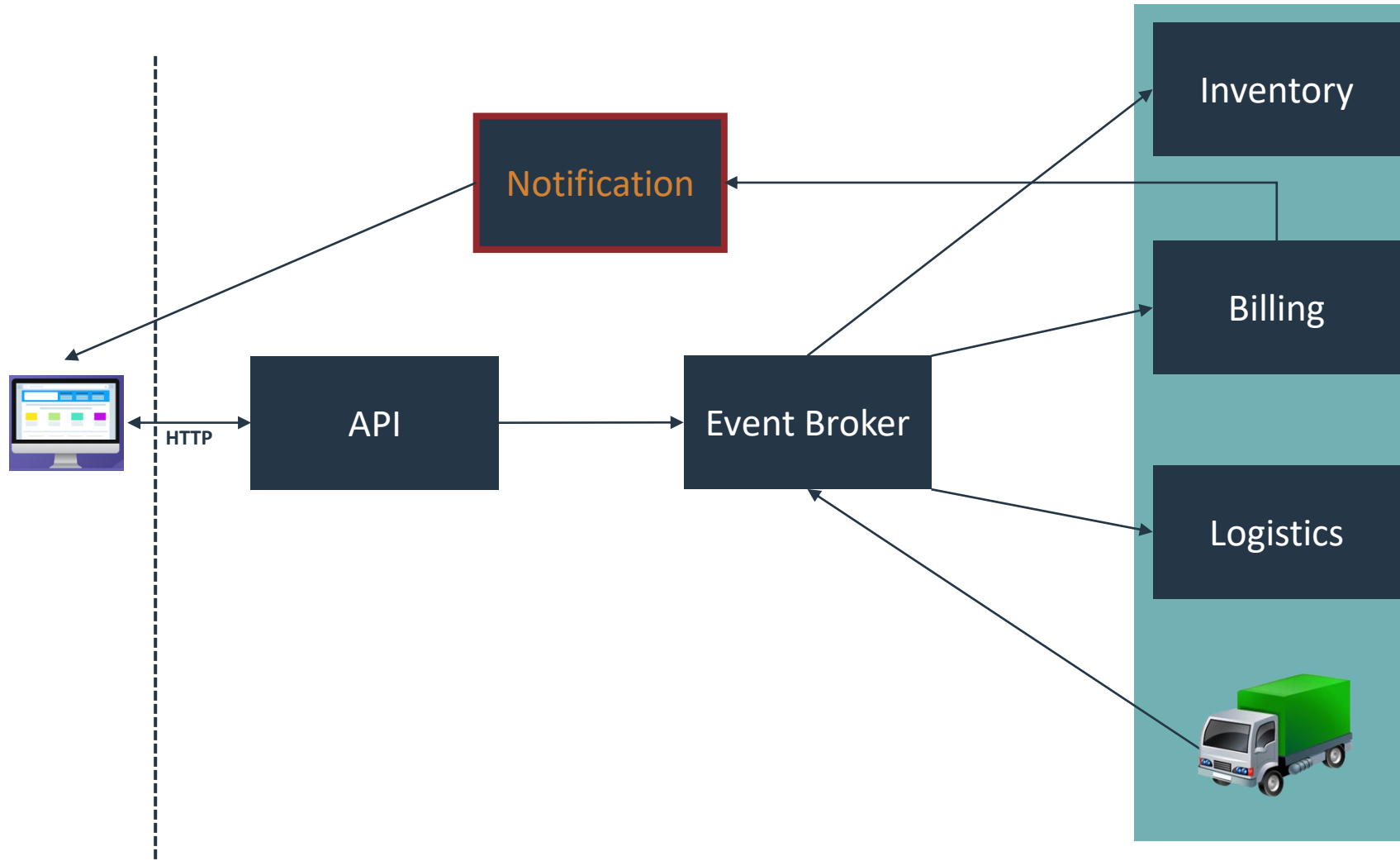# Consider Outbox Pattern

Prefer Listen to Yourself Pattern

Learn About Push Notification Options
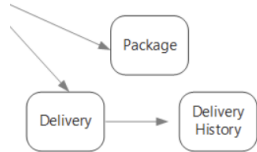
Understand Eventual Consistency

Each service can have different entity state

Use Event Storming to identify potential inconsistencies

Each potential inconsistency concerning?

Adjust service boundaries to resolve inconsistency?

Replace Read After Write with push notification?

Wrap Up

Summary

- Improve Reliability
  - **Asynchronous communications** minimize coupling
  - **Capturing event data** can reduce coupling
  - **Replayability** makes it easier to recover from some issues

- Improve Scalability
  - **Consumer groups** simplify scaling
  - **Parallel processing** across microservices is easier

- Improve Flexibility
  - Event ecosystem makes it **easier to add functionality** without the risks inherent to changing existing functionality
  - Event sourcing makes **time travel simple**

WellSky
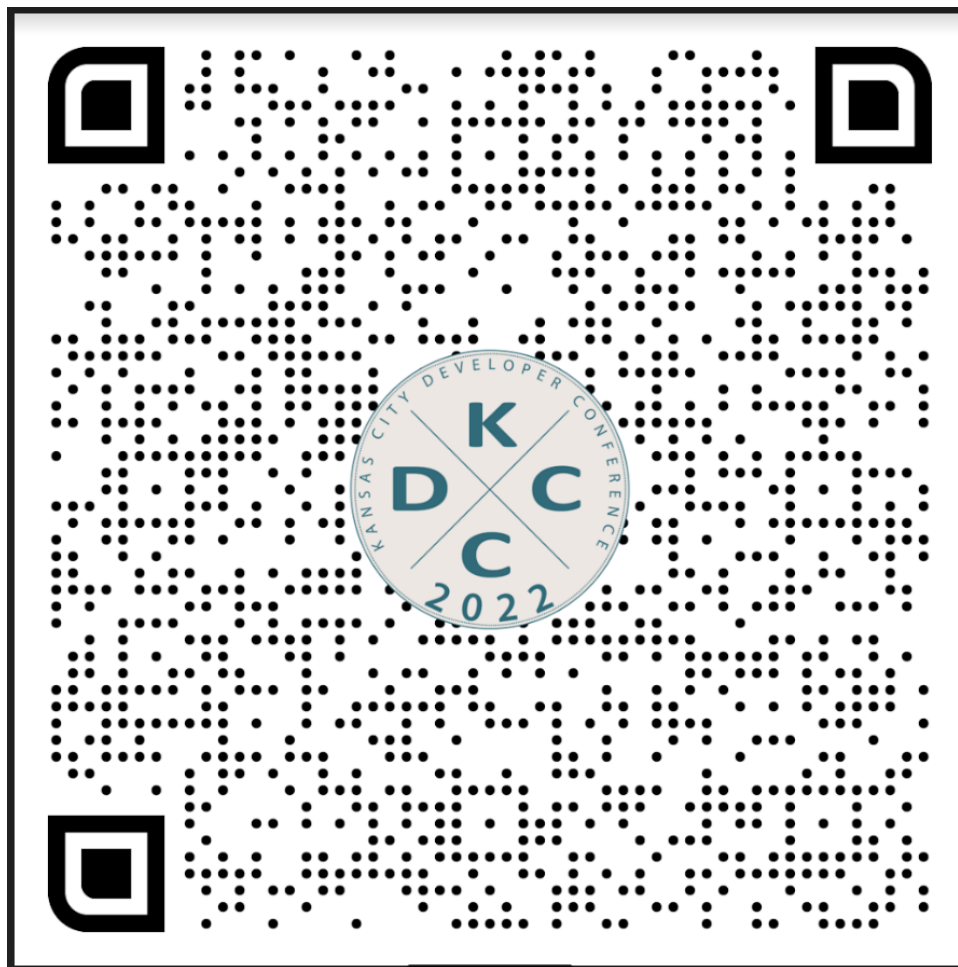
Thank you
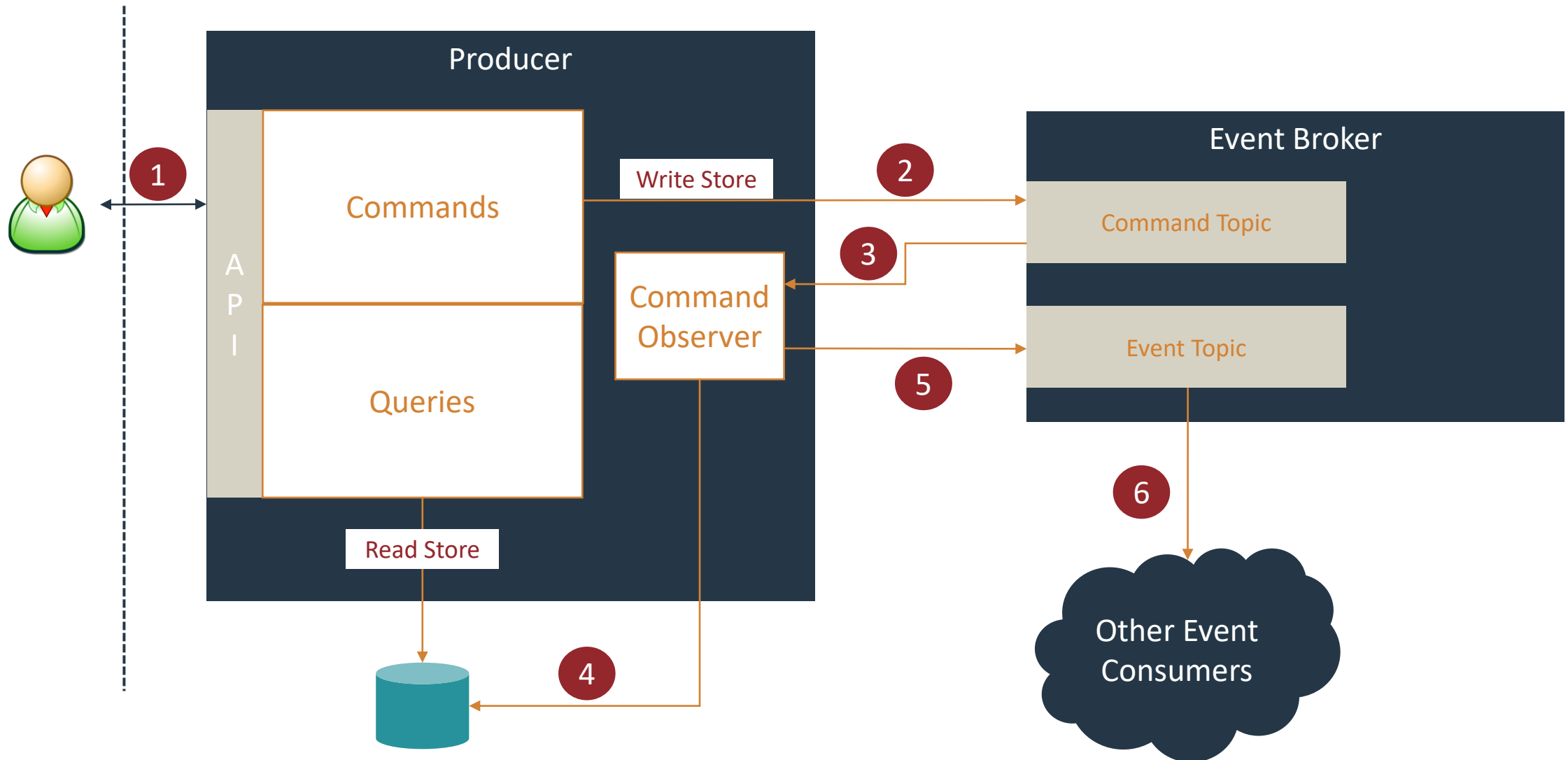
Contact us:
Scott Carter
Principal Software Engineer
scott.carter@wellsky.com
LinkedIn: https://www.linkedin.com/in/scott-carter/

Appendix

Producers and Command / Query Responsibility Separation (CQRS)

Warning

"Synchronous communication is the crystal meth of distributed software"

Mark Thomson

Understand Eventual Consistency