Building Better Microservices API with Event Driven Architecture

Irwan Butar Butar
Python Conference Indonesia – 2023
pycon.id

Agenda

- Shift of **Communication**
- The Need of **Asynchronous** Integration
- Understanding **Event-Driven** Integration
- Event-Driven **Microservices**

About Me

Irwan Butar Butar
Solution Architect at solace. edasummit.com

2022 Sr. Pre-Sales Consultant at **Neo4j**
2021 Senior Data Engineer at BBP.sg
2019 Professional Service Consultant at Kinetica
2014 Senior Consultant at TIBCO

Find me:
irwan.butarbutar@gmail.com
https://www.linkedin.com/in/irwanbutarbutar

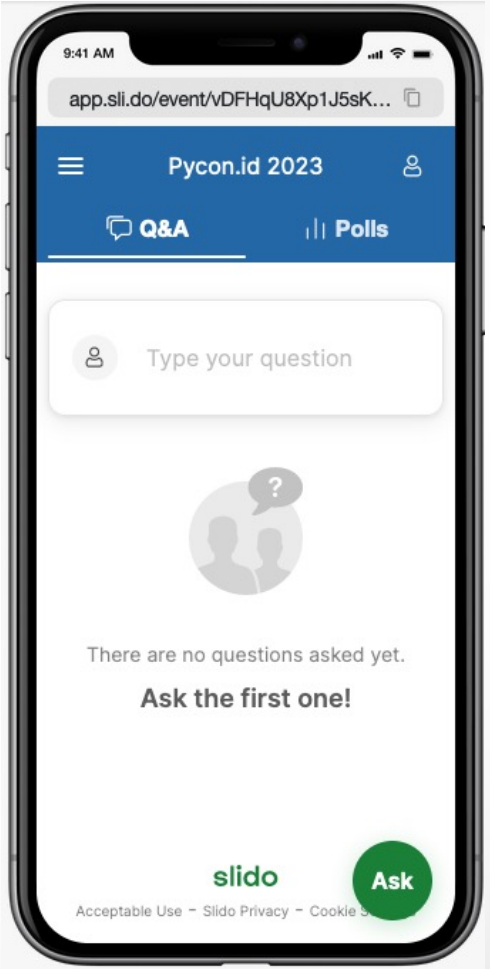
Open https://slido.com and Enter

#2653900

Slido

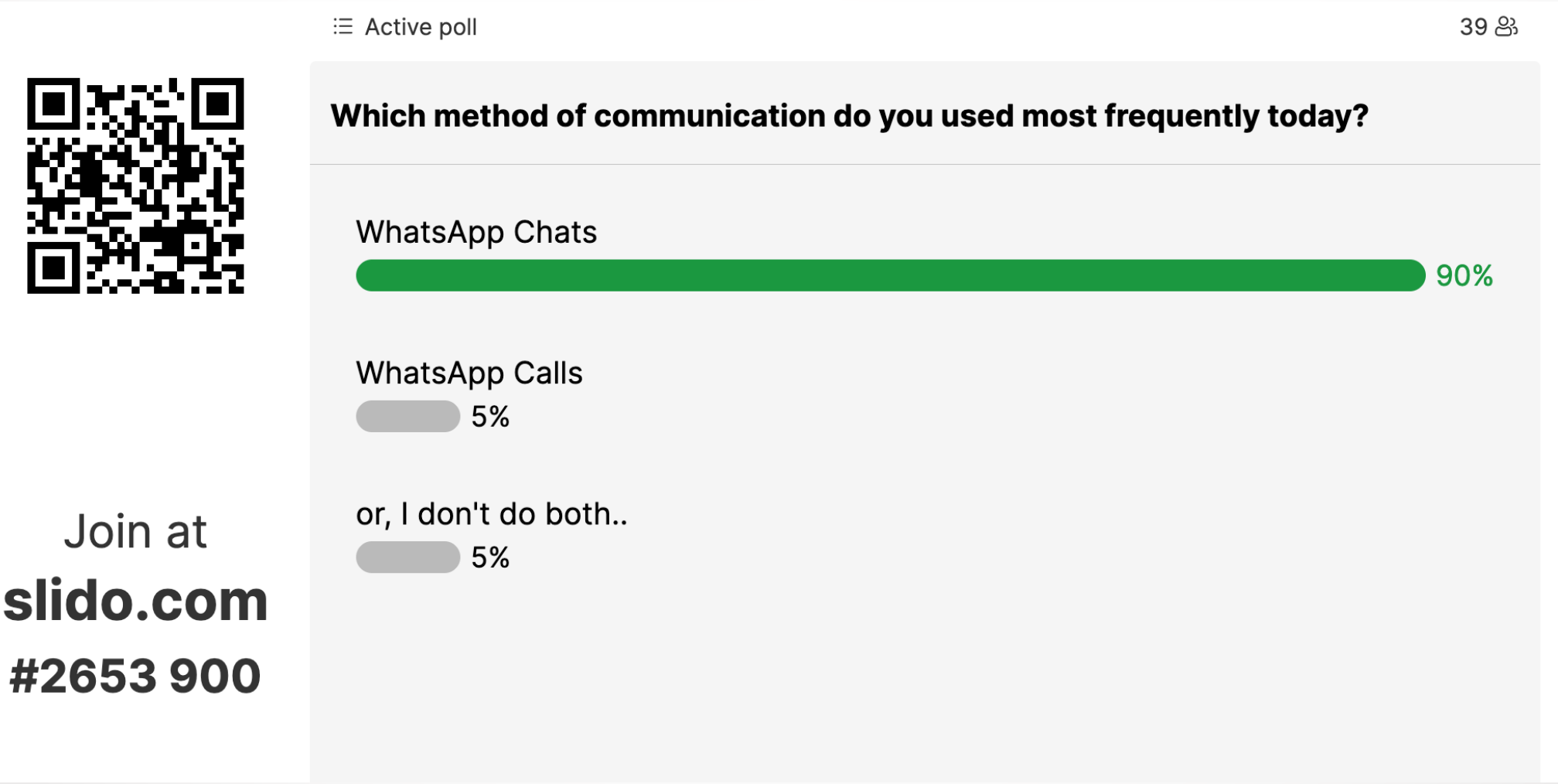
Open https://slido.com and Enter

#2653900



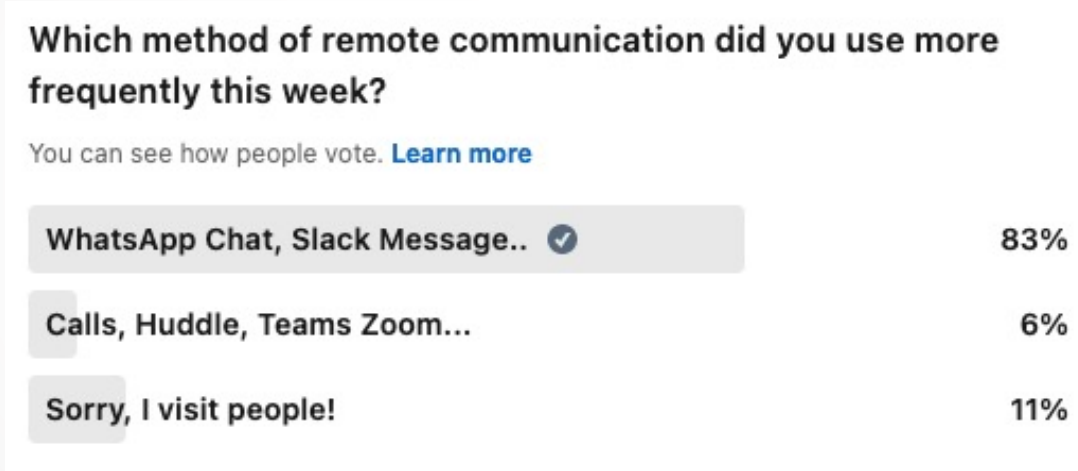
Polling

Join at
slido.com
#2653 900



Polling Result

I ran this poll in LinkedIn before and get this result:



Almost ALL vote for ***WhatsApp Chats***

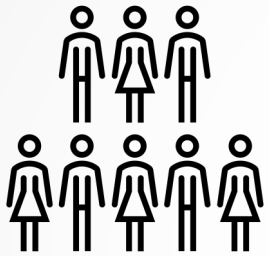
Only few of them *vote for* ***WhatsApp Calls***

And some people select the **3rd** option
basically, saying they are not using phone but prefer face-to-face meeting

Shift of Communication

Technology is evolving and the world is always changing

Communication Trends



indirect

Messaging
Letter/Pager

SMS

Internet Messaging
Email/IRC

Social Media
Chats/Streaming/Subscription

direct

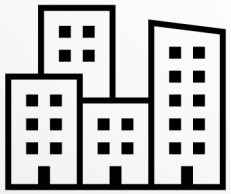
Fixed Line
Home/Public Phone

Mobile Phone
GSM/Radio

Internet Call
VoIP/Video Call

1990 — 2000 — 2010 — 2020 — Now...

Data Streaming
Apache Kafka



indirect

File Transfer
FTP

Messaging
MQ/JMS/MQTT

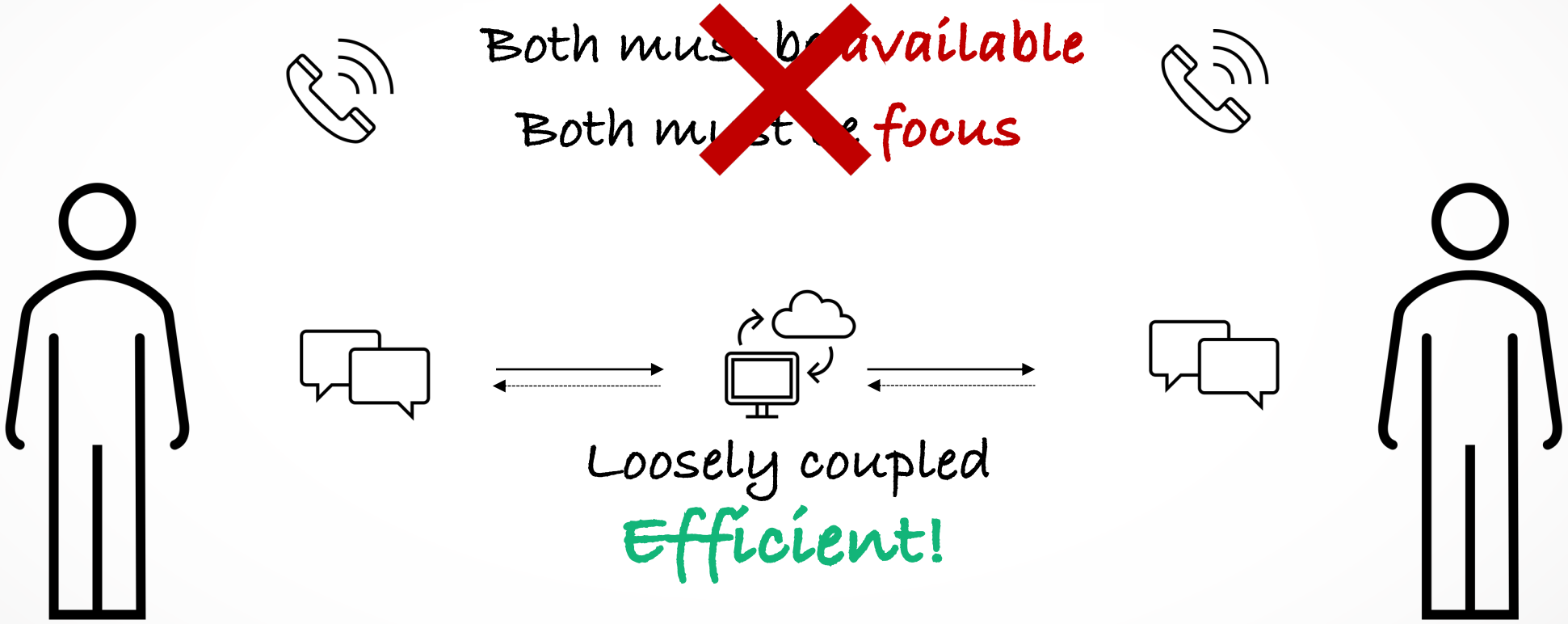
direct

RPC/Socket
CORBA, ISO8583

HTTP
Web/SOAP

HTTP/Websocket (API)
REST/gRPC/GraphQL

Calling vs Messaging



Older personal messaging technology unreliable
Only big enterprise can have good ones

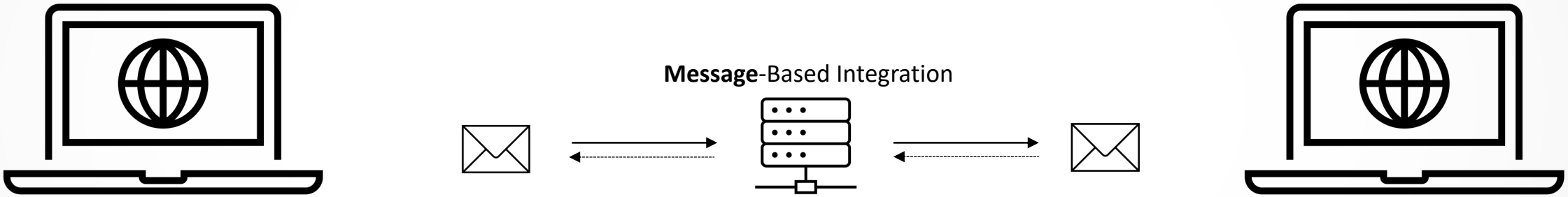
Nowadays...Technology is massive and cheap
And, Pervasive

Application Integration

API

Both must be **available**
Both must be **able to process**

API



Loosely coupled

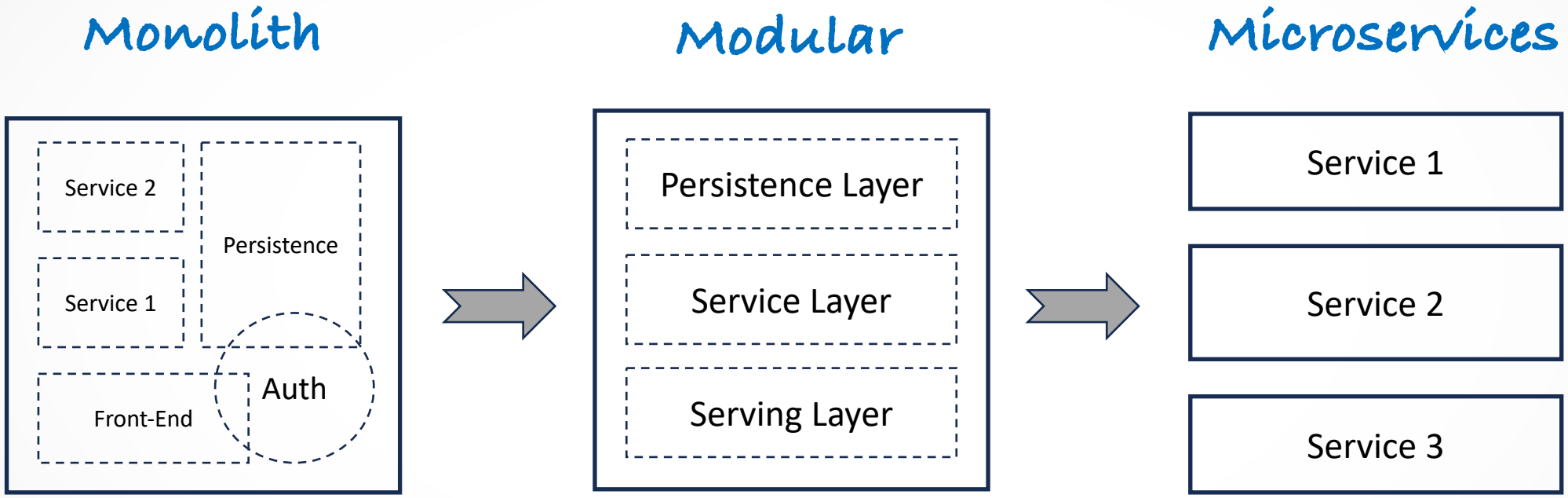
Efficient!

We understand the benefits of using asynchronous communication
Why did we always choose ~~API~~ as our first choice of integration?

Asynchronous Integration

The need for asynchronous integration for loosely-coupled microservices

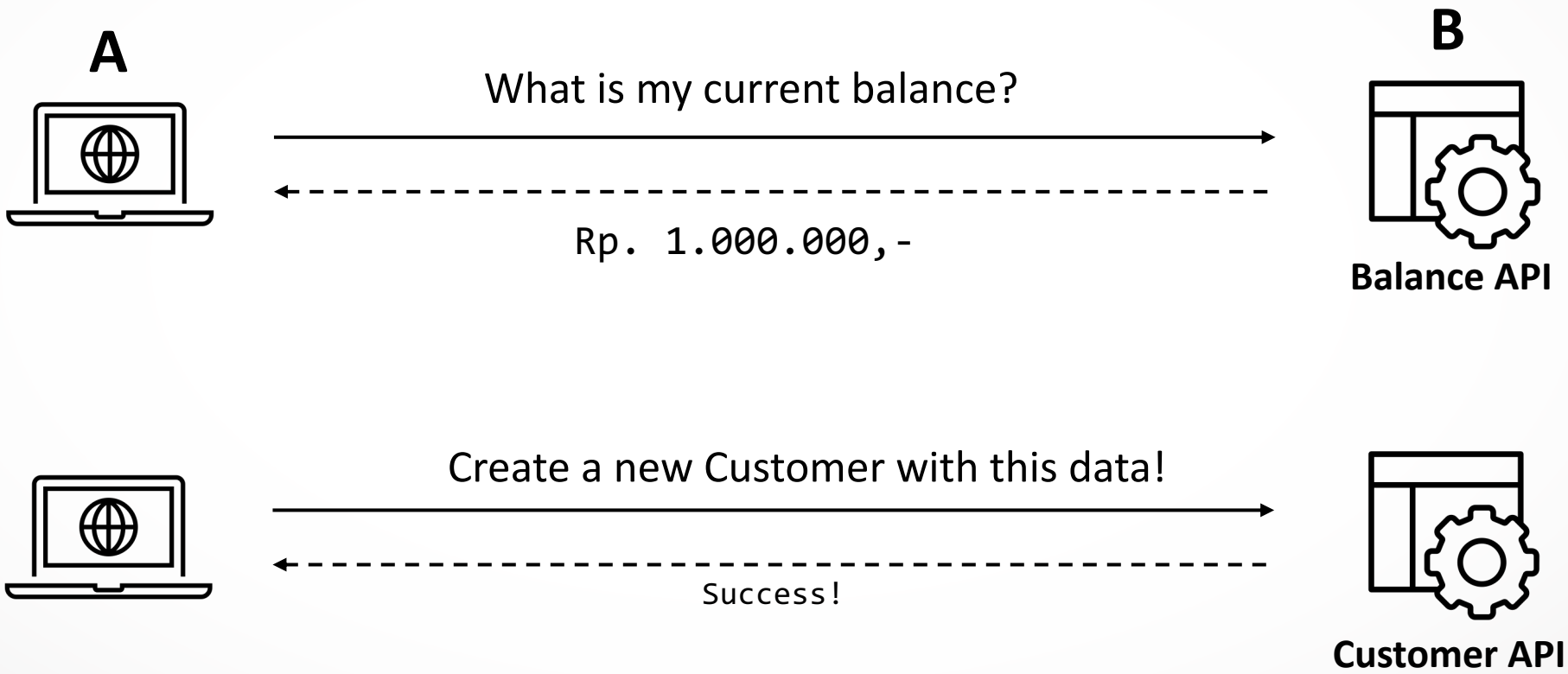
What is Microservices?



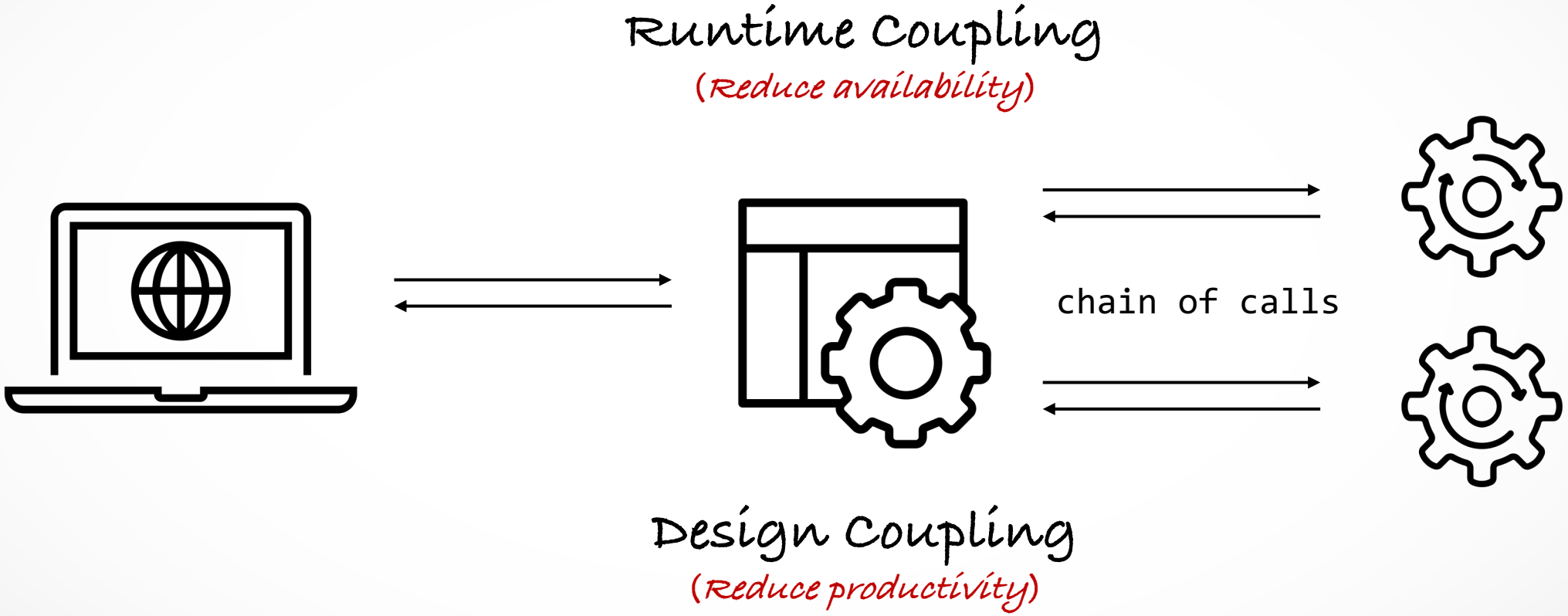
Independently Deployable
Loosely Coupled
Organized around Business Capabilities

https://***microservices***.io

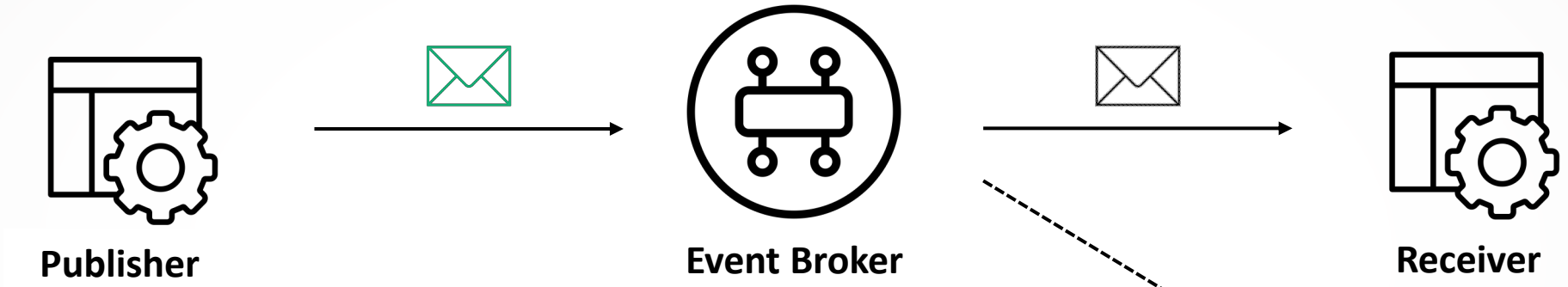
Conventional API Example



Issue with Conventional API



Asynchronous API

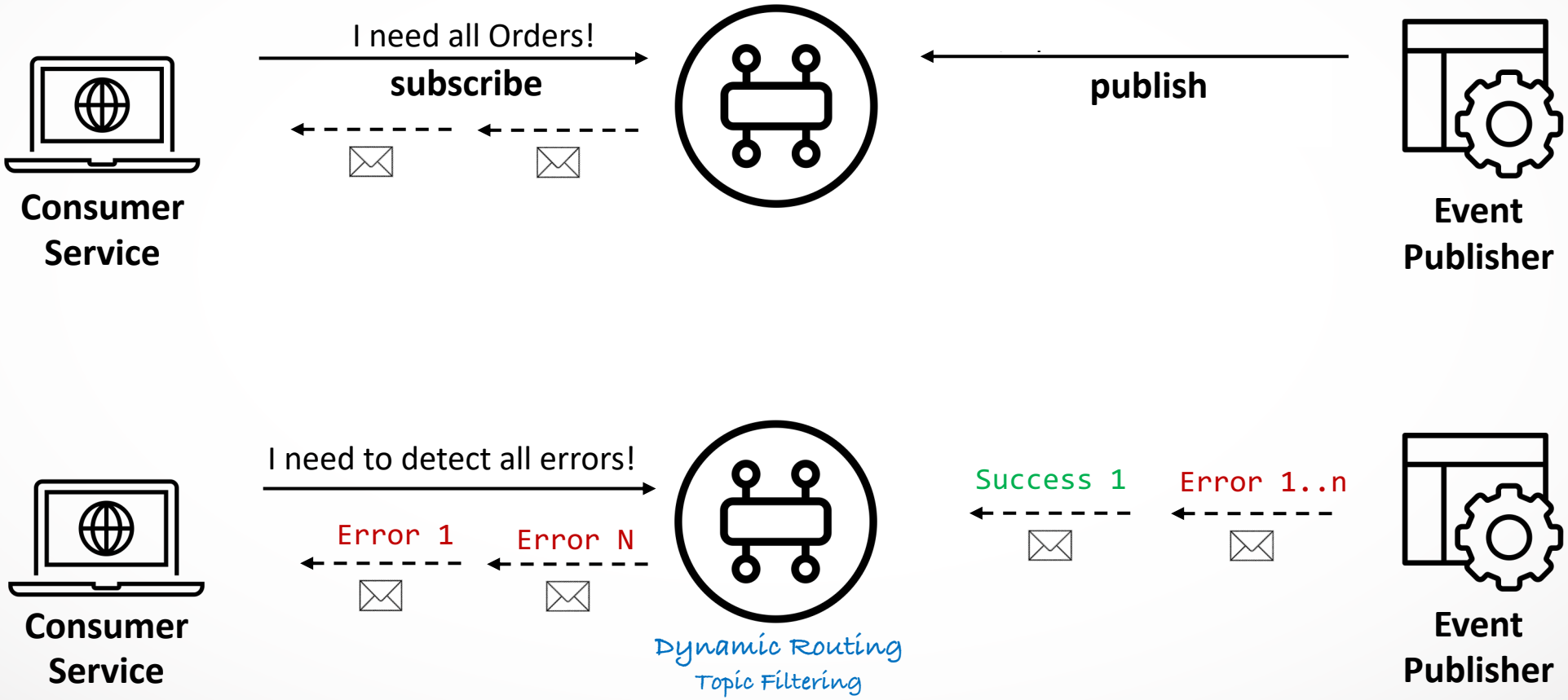


Messages:

- Data Stream
- Event
- Command
- Reply

Event-Driven Architecture (EDA) Concept

Asynchronous API Example



Async Language Support?

Callback?

Message-Based Point-to-Point?

Publish Subscribe?

Event-Driven Integration

Being truly asynchronous

Async Language Support

asyncio — Asynchronous I/O

asyncio is a library to write **concurrent** code using the **async/await** syntax.

asyncio is used as a foundation for multiple Python asynchronous frameworks that provide high-performance network and web-servers, database connection libraries, distributed task queues, etc.

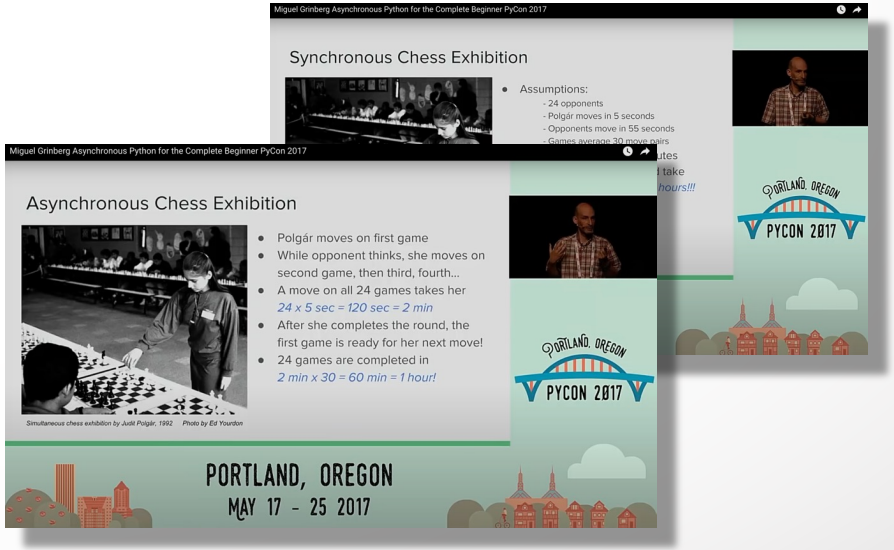
asyncio is often a perfect fit for IO-bound and high-level **structured** network code.

Hello World!

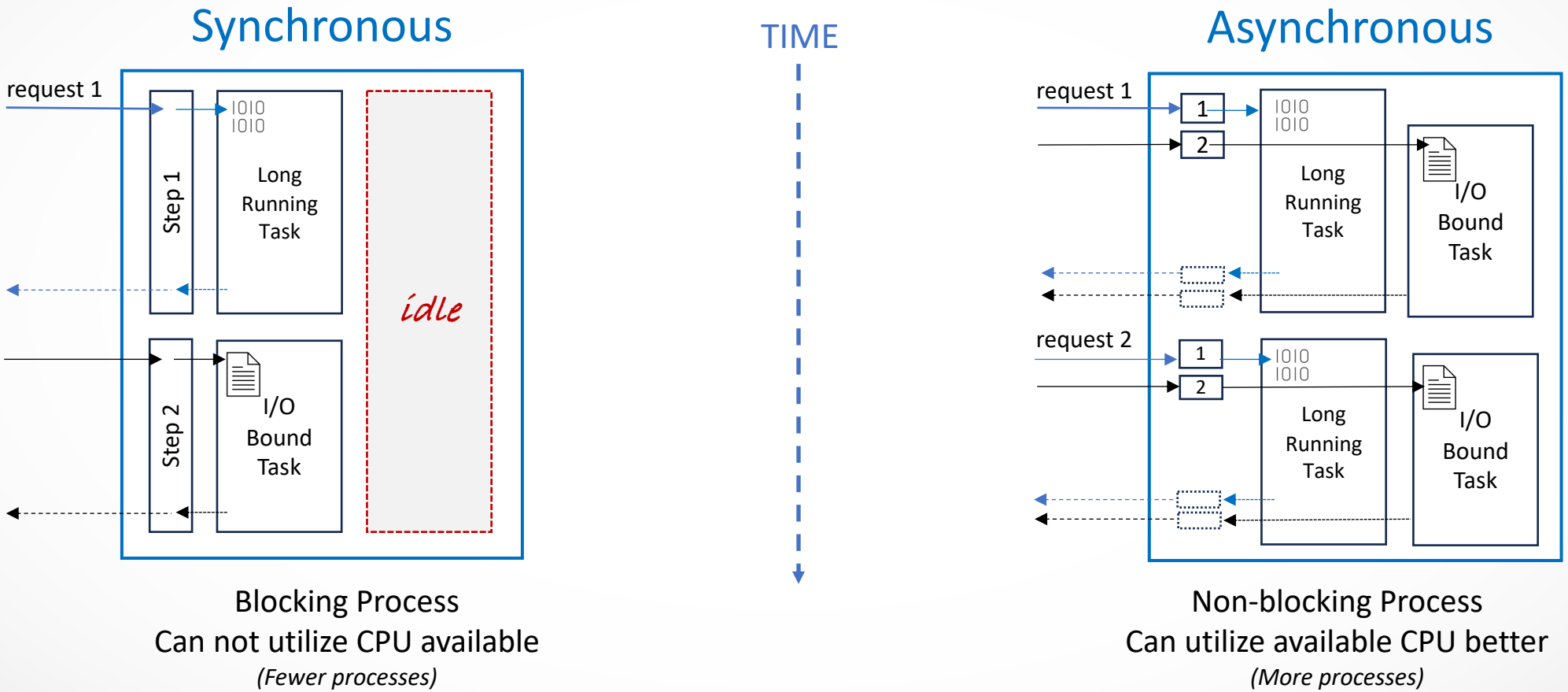
```
import asyncio

async def main():
    print('Hello ...')
    await asyncio.sleep(1)
    print('... World!')

asyncio.run(main())
```

https://docs.python.org/3/library/asyncio.html

https://www.youtube.com/watch?v=iG6fr81xHKA&t=269s



Native Asynchronous Processing



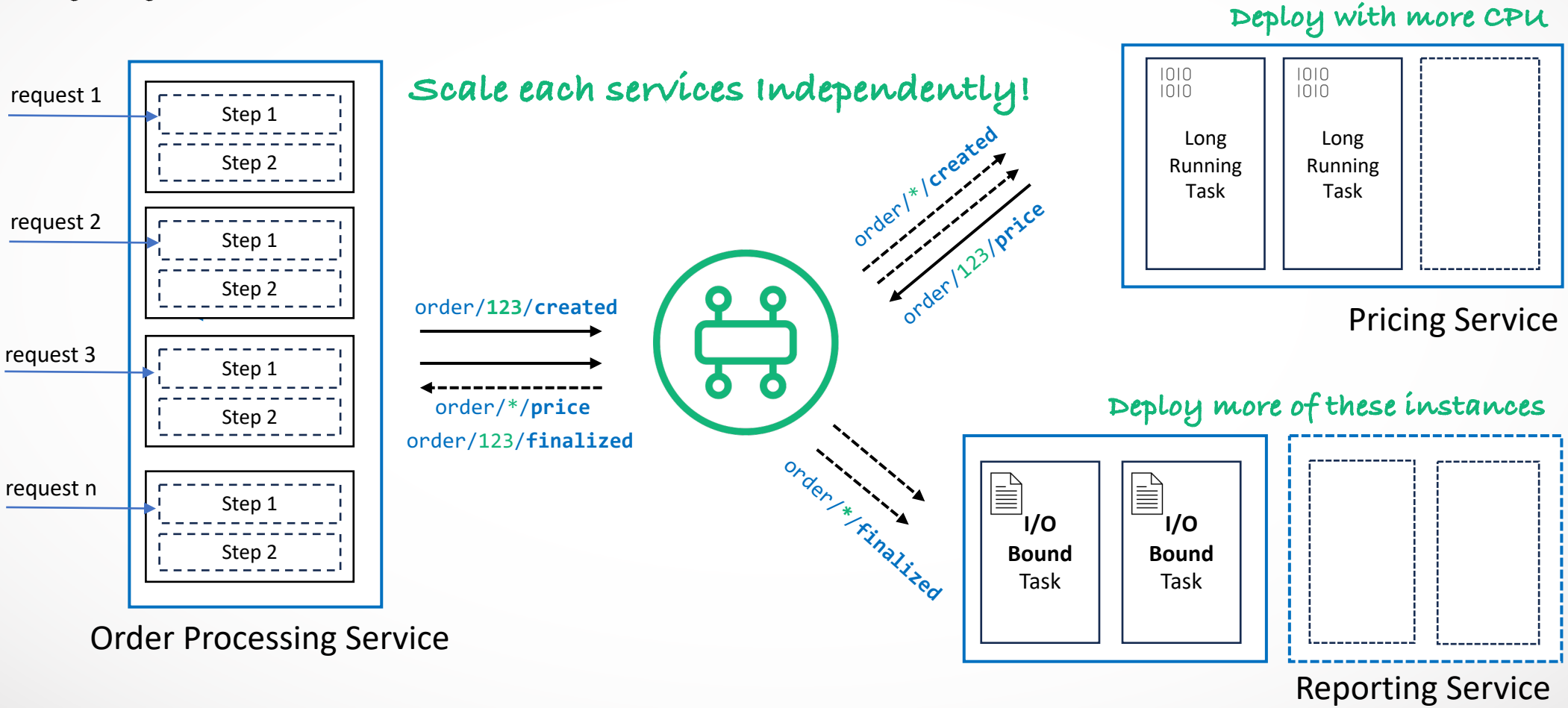
Blocking Process
Can not utilize CPU available
(Fewer processes)

Non-blocking Process
Can utilize available CPU better
(More processes)

Capacity to handle requests depends on number of available resources in a single instance

Event-Driven Architecture

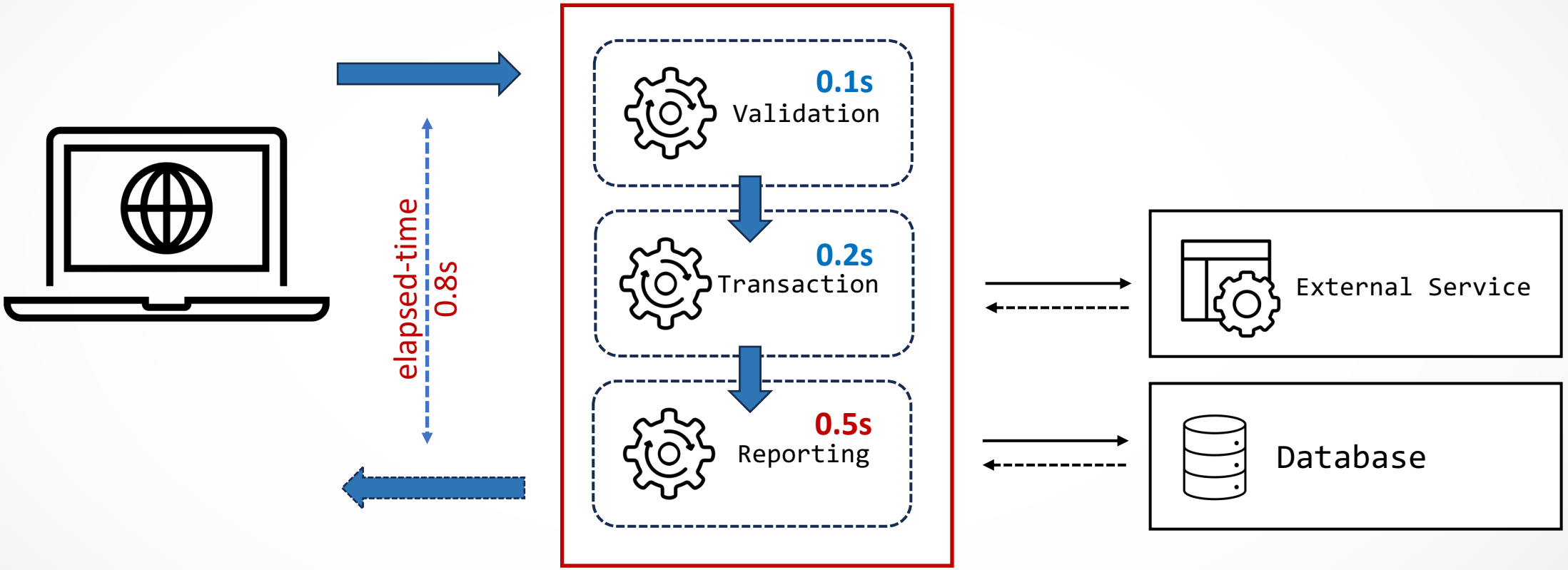
Truly Asynchronous



Event Driven Microservices

Applying Event-Driven Architecture on Microservices

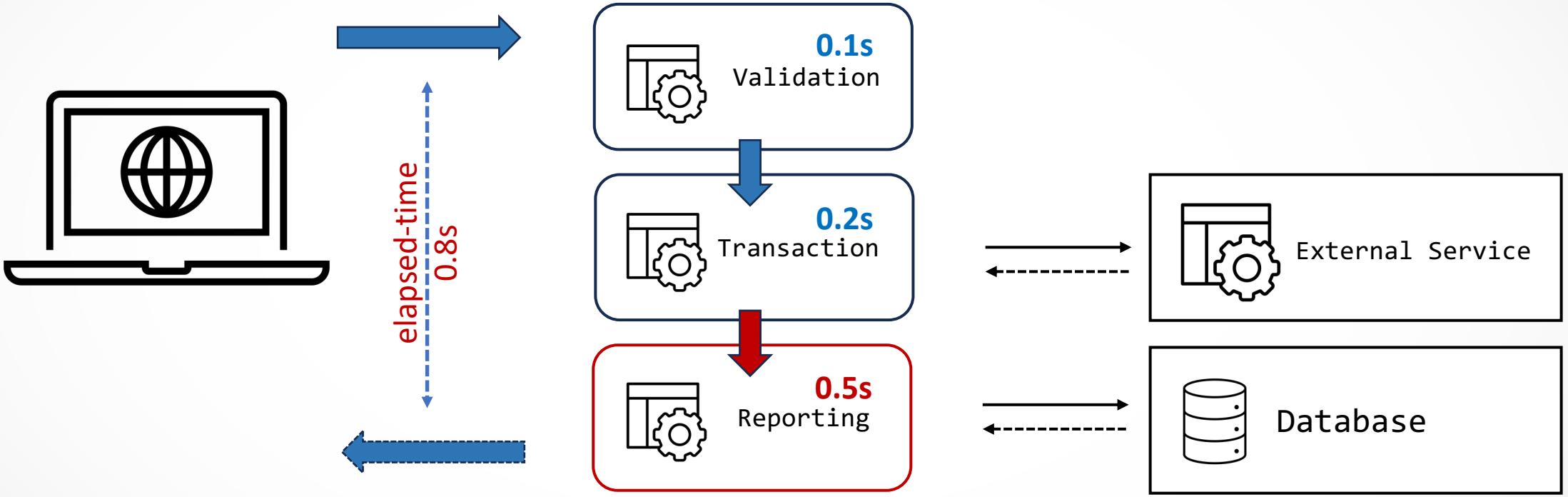
Macro-services API



Turn to microservices

*Identify process in your service that can be **refactored-out**.*

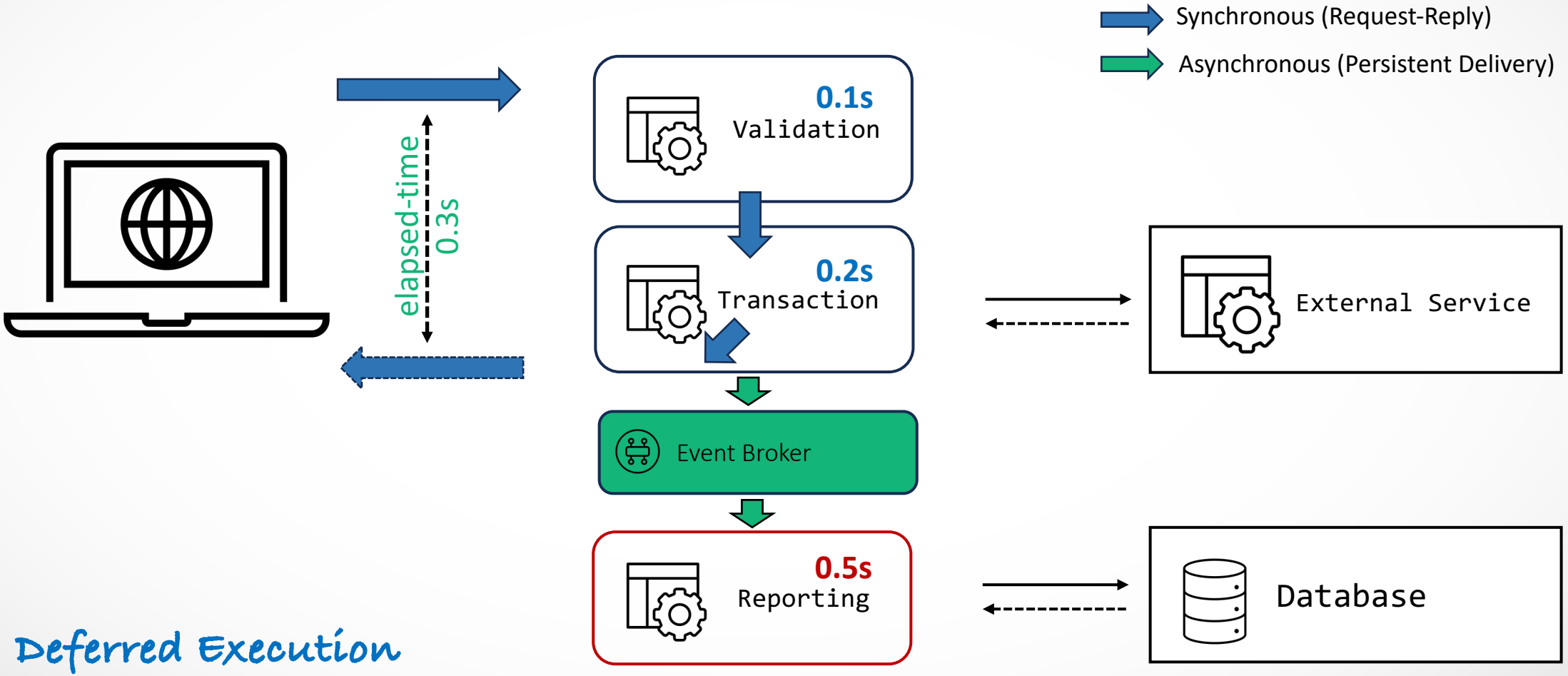
Synchronous Microservices API



Identify Supporting Logic

*Identify logic separate from core business logic to be **deferred**.*

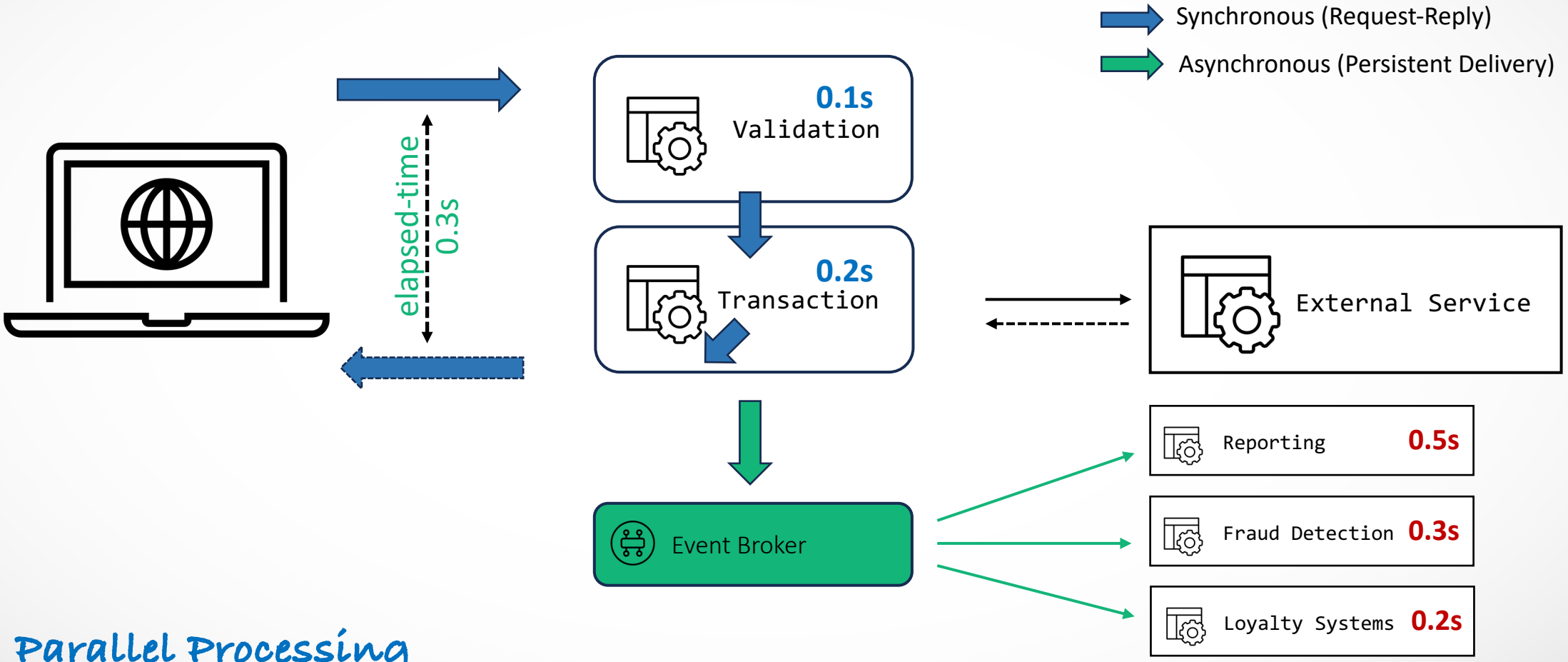
Asynchronous Microservices API



Deferred Execution

Execute supporting business logic separately with event-driven message.

Parallel Processing Microservices API



Parallel Processing

Add additional business logic without impacting elapsed-time.

Event-Driven Challenges

Increase Complexity

Monitoring and Tracing

Event-Driven Microservices Patterns

PyCon ID 2023

Decision Making

Orchestration vs **Choreography**

Centralized vs Event-Driven

Query Patterns

Command Query Responsibility Segregation (**C Q R S**)

Transactions Patterns

S A G A

Data Consistency across microservices

Summary

The World Event-Driven

The Need of Asynchronous Event-Driven Integration

Event-Driven Architecture

...from **API-Led** to **Event-Driven Integration**

Loosely coupled!
Truly Asynchronous!

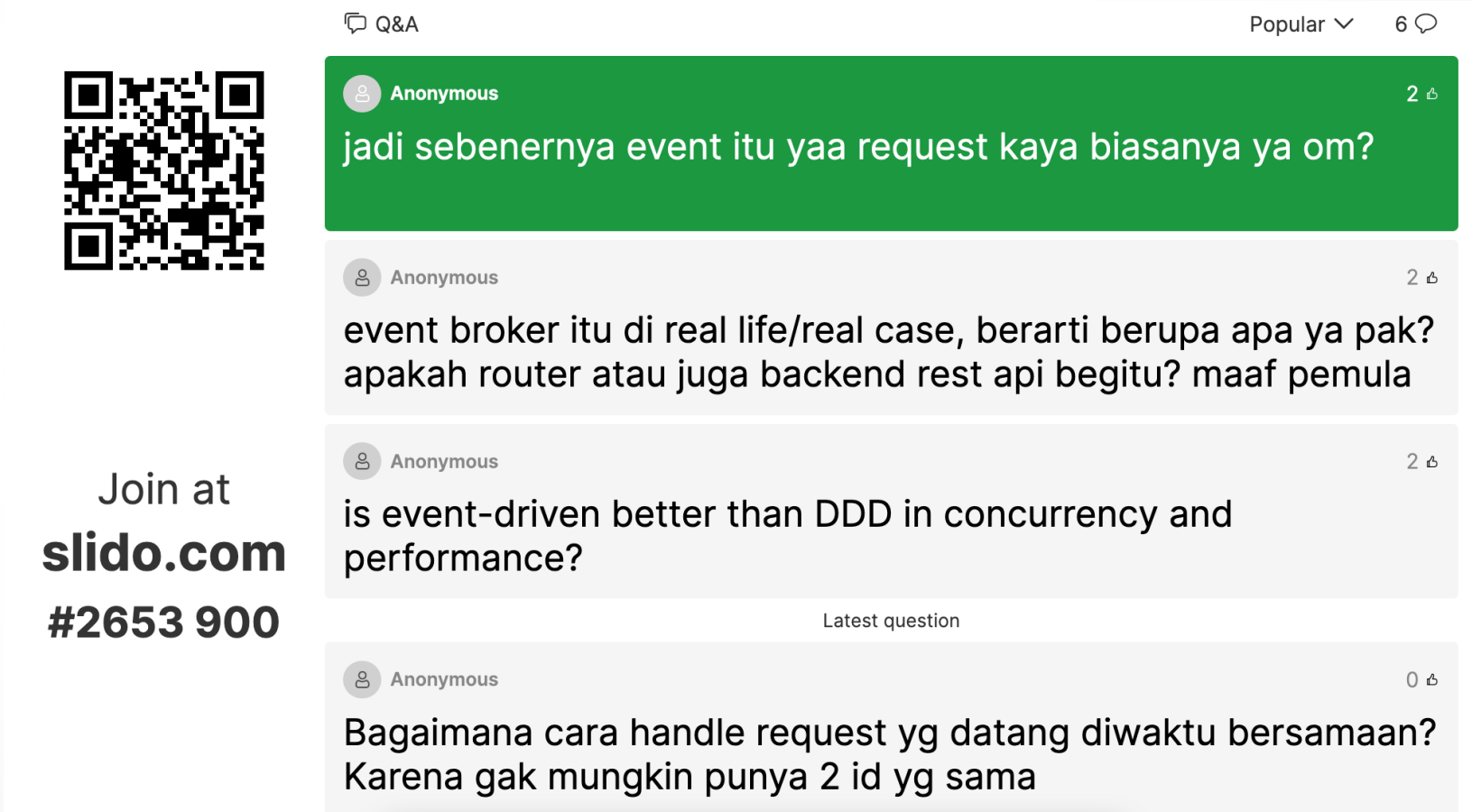
Any questions?

PyCon ID 2023

Q&A

*You can also post questions in **Slido.***

Join at
slido.com
#2653 900



...Be Event-Driven!

Irwan Butar Butar
Solution Architect at solace.

irwan.butarbutar@gmail.com
https://www.linkedin.com/in/irwanbutarbutar

solace.dev