Introduction to Event-Driven Microservices

Nikos Orfanoudakis
Software Engineer / Architect @Enartia

Agenda

- The Whys & Hows of microservices
- The hardest part about microservices
- Event-Driven Design mindset & characteristics
- Overview of Event-Driven Design Patterns
- What to do next

Why microservices are so popular nowadays?

How the microservice architecture helps you scale?

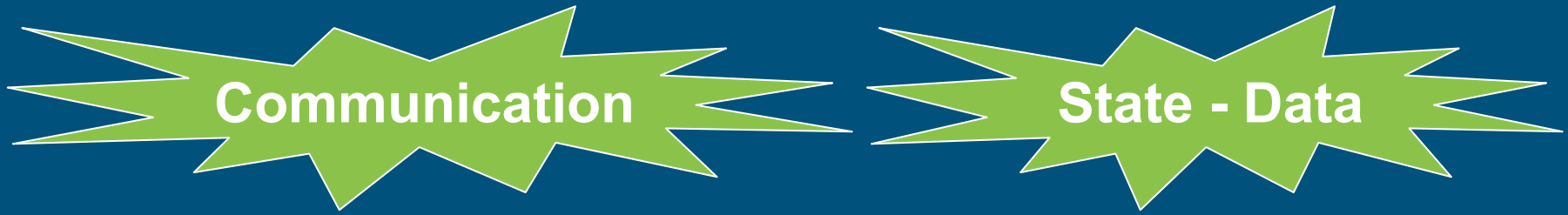
...but, if you manage to build such kind of autonomous teams, then why don’t you design your systems that way as well?

Microservices are distributed systems

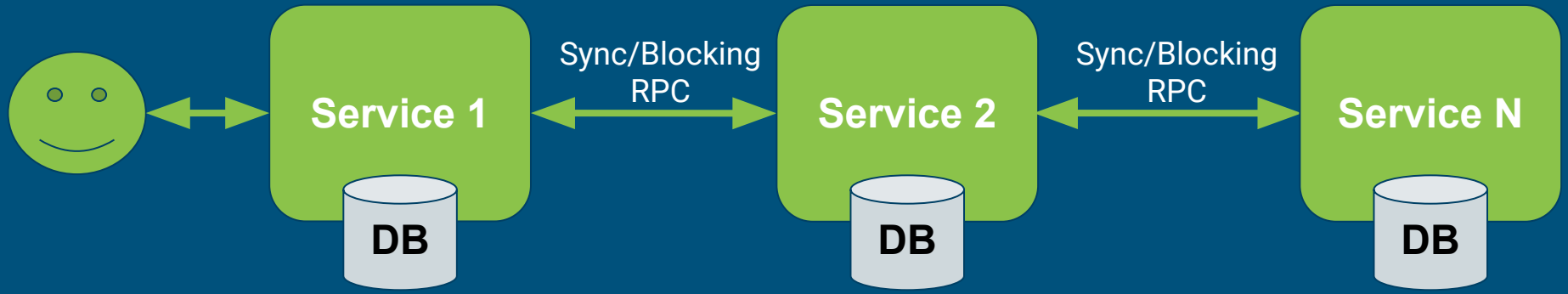
The hardest part about microservices

! MicroserviceImpl

...but...



Is synchronous (blocking) communication a good option for your microservices ecosystem?



KEEP CALM
&
GO ASYNC

SAY NO TO BLOCKING I/O
&
TRY NOT TO TALK TO ANYONE DIRECTLY

Event-Driven Design

Event-Driven Design

Don’t design your system based on its domain objects,

let the domain objects and bounded contexts derive from the behaviour of the system

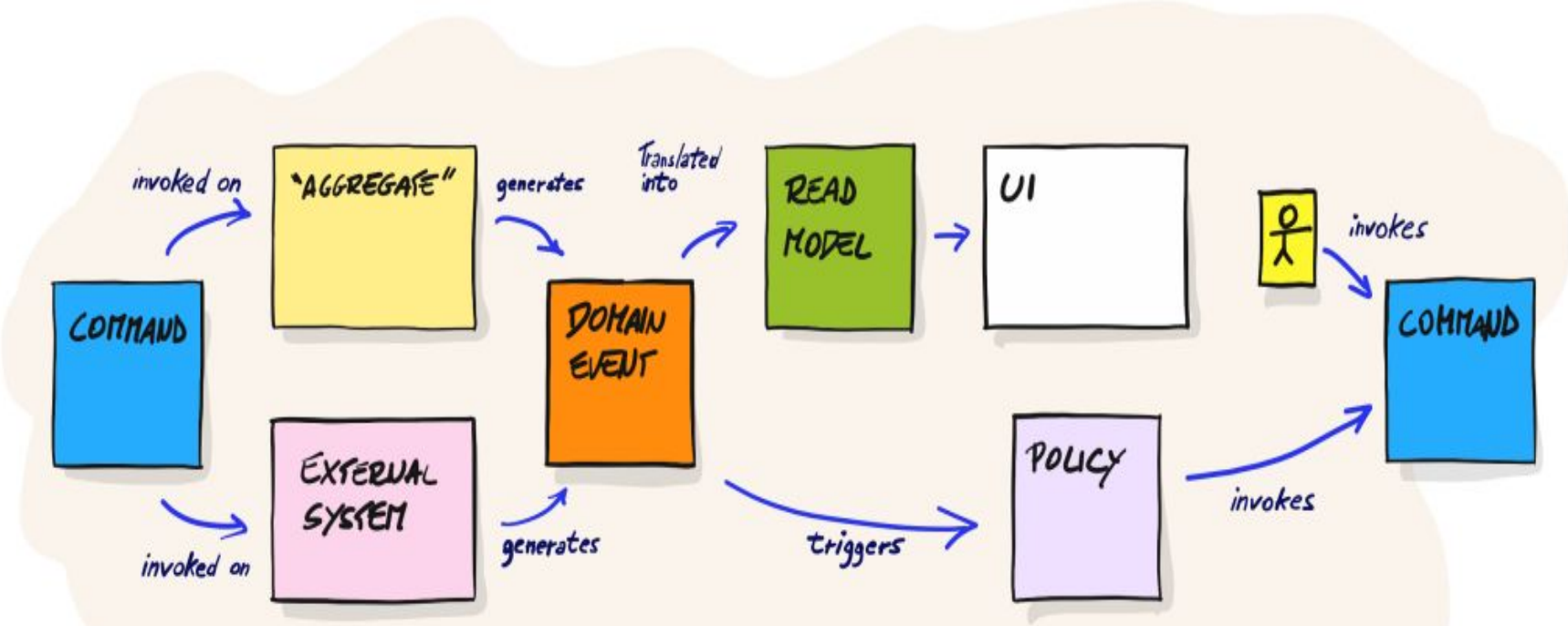
Behaviour is defined by...

- Commands

  (Action requests)
- Reactions

  (Side-effects)
- Events

  (Immutable Facts)



Event Storming

https://user-images.githubusercontent.com/1964655/41868241-627f4136-78ad-11e8-8912-9da26ec2db22.png

Commands

Focus on intents

- Directed
- Single addressable destination
- Distributed focus
- Command & Control

Events

Focus on facts

- Intentless
- Anonymous
- Make announcements regardless if there is anyone that cares about them
- Local focus
- Autonomy!

Event-Driven Microservices

Main characteristics:

- Receive events / commands
- Choose to which ones they react
- Publish new events to the rest of the microservices

  in an asynchronous fashion

...but wait...
where do all these events end up?

Event Store

It can be used for...

- Communication
- Integration
- Replication
- Persistence
- Consensus

…okay, let’s assume we are happy with the asynchronous communication…

What about state?

Strong consistency is an illusion on distributed systems

KEEP CALM
&
ACCEPT EVENTUAL
CONSISTENCY

Failure is inevitable...
Don’t try to avoid it, try to handle it!

Event-Driven architecture helps a lot in managing failure by designing recovery mechanisms as part of the business flow

Event-Driven Design

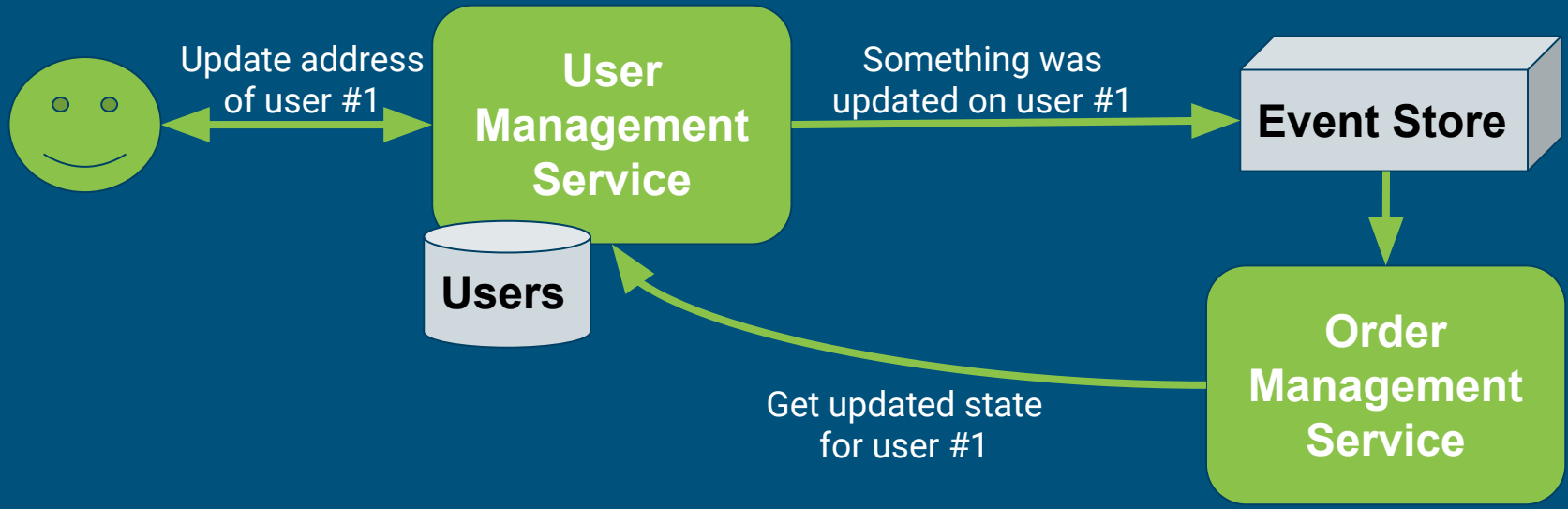
Overview of Patterns

Event-Driven Design: Patterns

1. Event notification
2. Event-based State Transfer
3. Event Sourcing
4. Command Query Responsibility Segregation (CQRS)

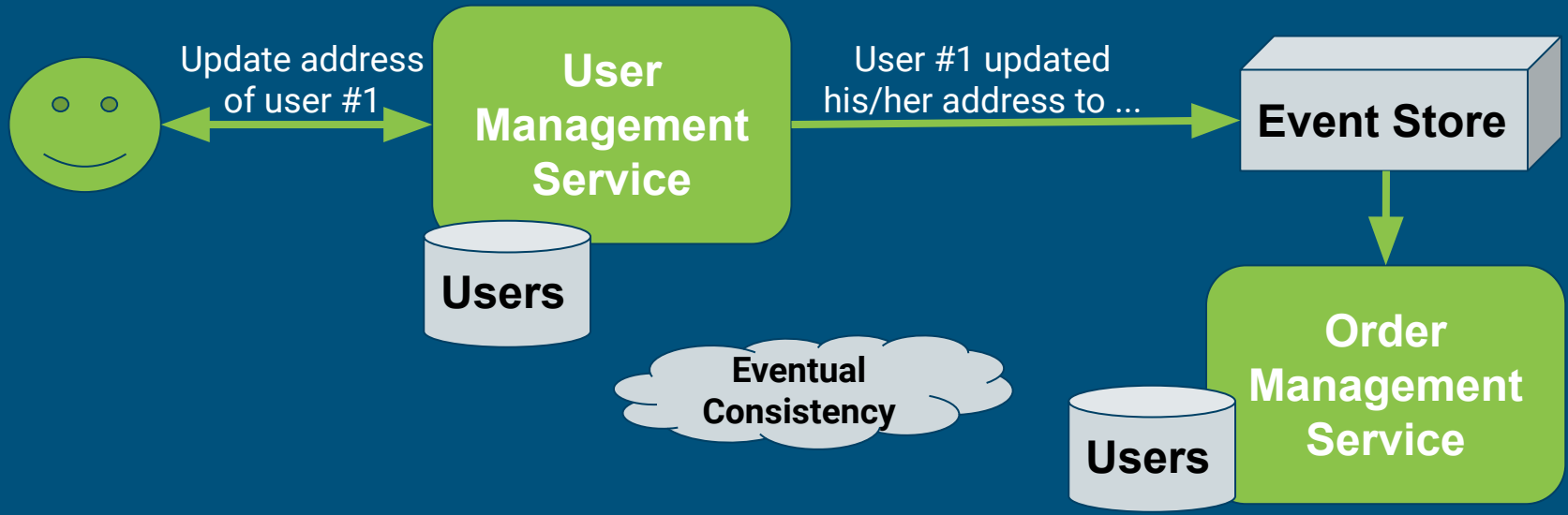
Pattern 1: Event notification

Microservices communicate via events & remote queries



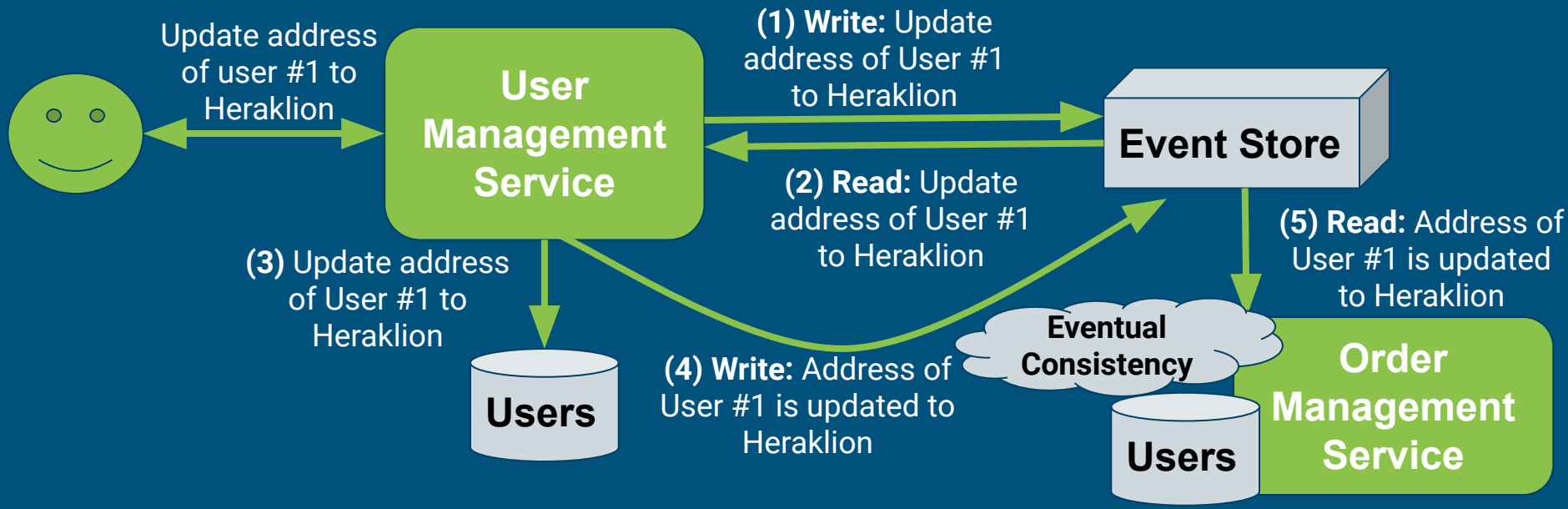
Pattern 2: Event-based State Transfer

Microservices can access data without calling the source (via data-replication)



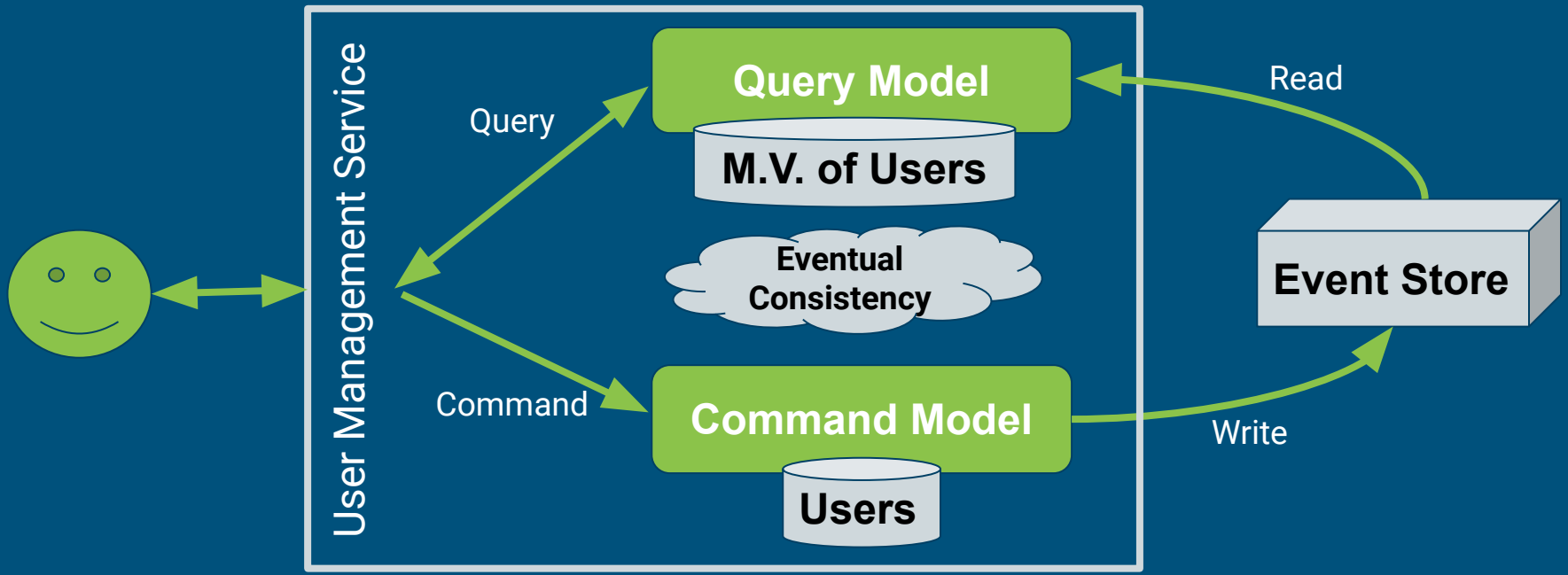
Pattern 3: Event Sourcing

Microservices use an event log as the source of truth for their state



Pattern 4: CQRS

Separate microservices for writing to and reading from the store



What to watch & read next...

- *Designing Events-First Microservices* by Jonas Bonér
- *The Many Meanings of Event-Driven Architecture* by Martin Fowler
- *The Hardest Part About Microservices: Your Data* by Christian Posta

Q & A

Thank you!