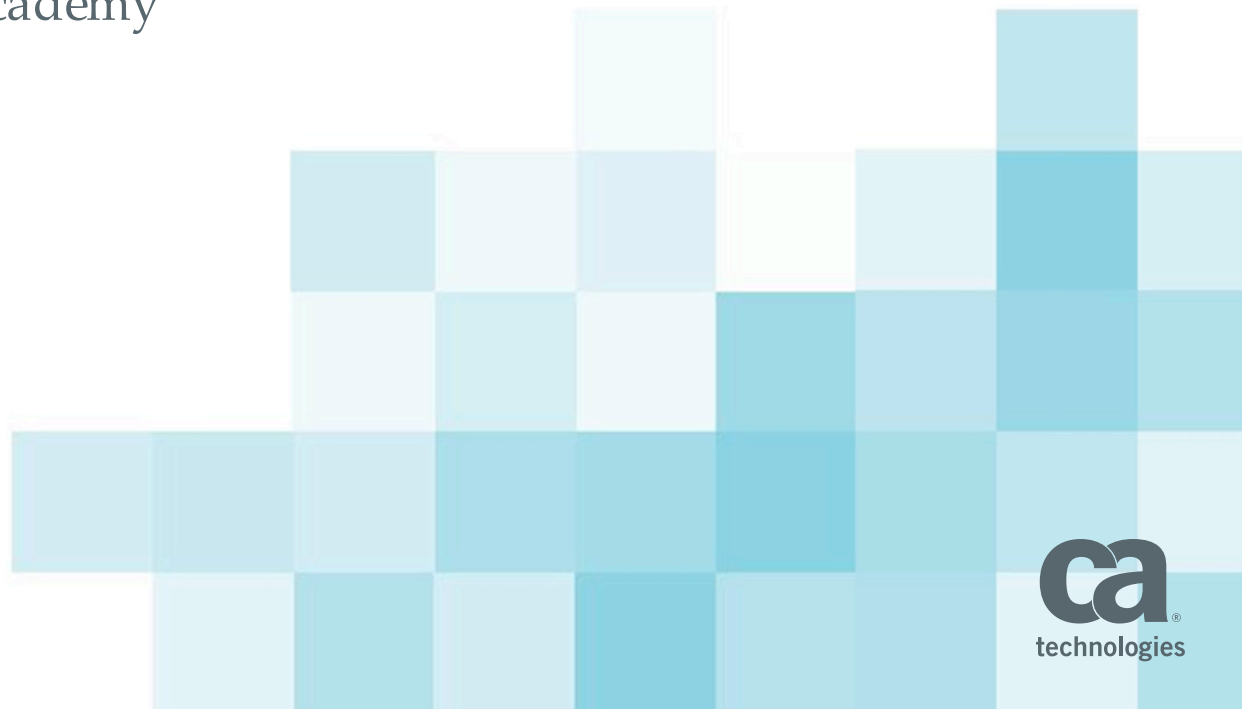


# Microservices In Practice

Irakli Nadareishvili,

Director of Strategy, API Academy

CA Technologies



REDFAR.TUMBLR.COM

**The Next Big Thing!**

# But, why?



# Microservices can greatly improve your capabilities in:

1 CONTINUOUS DELIVERY

2 PRODUCING EVOLVABLE AND SCALABLE ARCHITECTURE

3 ENABLING AGILE SOFTWARE DEVELOPMENT LIFECYCLE





OURGANGAPPRECIATION

Microservices is an architectural style, in which a complex software application is decomposed into smaller components ("micro services") that:

**1 COMMUNICATE USING LANGUAGE-AGNOSTIC APIS**

**2 ARE INDEPENDENTLY DEPLOYABLE VIA UNIVERSAL CONTAINERS**

**3 PRACTICE DECENTRALIZED DATA MANAGEMENT**

# Challenge: De-centralized Data Management

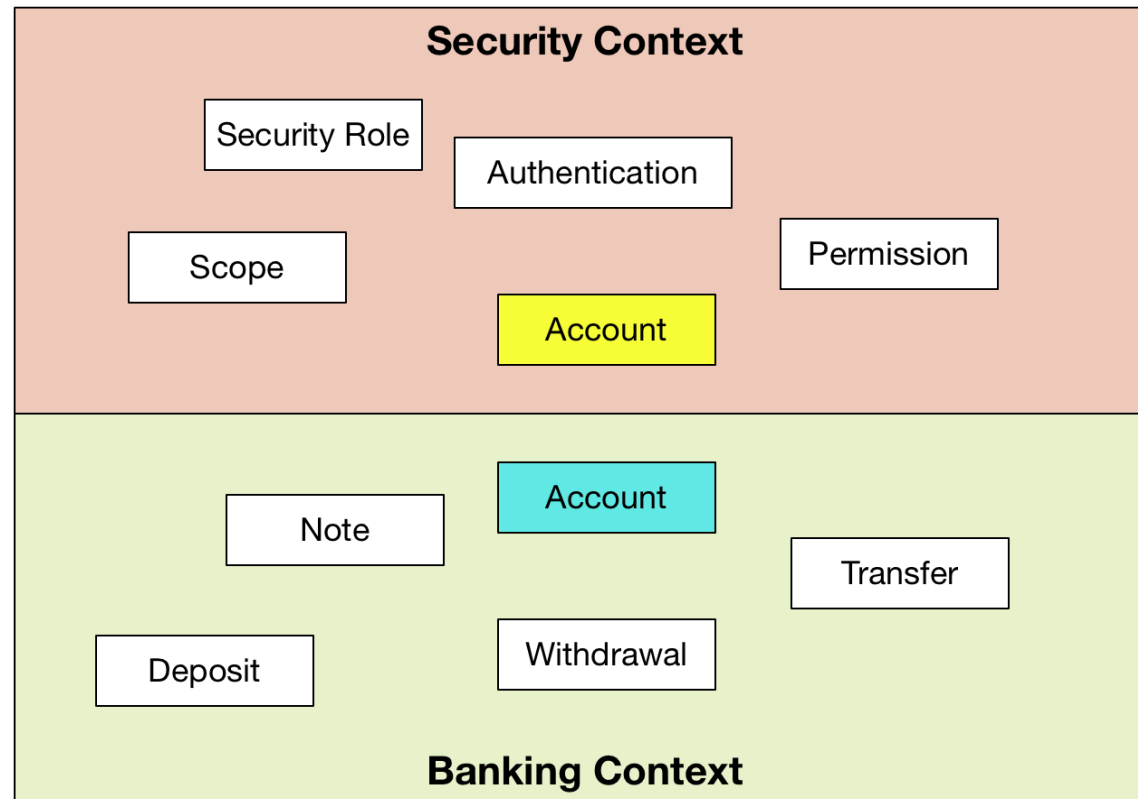
**Tool 1:**  
**Think of Capabilities,  
not: Data Models**



# Bounded Contexts

# Eric Evans: Bounded Contexts

An application domain consists of multiple bounded contexts. Residing within each BC are models that do not need to be shared outside, as well as other models that are shared externally.



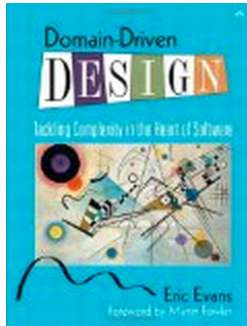
# Bounded Context = Capabilities.

“In your organization, you should be thinking not in terms of data that is shared, but about the capabilities those contexts provide the rest of the domain.”

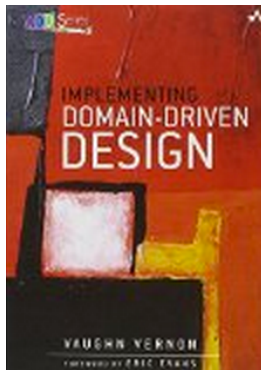
– Sam Newman, Building Microservices

**Caution:**  
**DDD and Bounded Context Identification  
Are Far From Trivial or Easy Tasks**

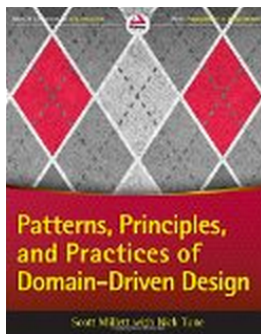
# Domain-Driven Design – More Information



Eric Evans: "Domain-Driven Design: Tackling Complexity in the Heart of Software" - 2003



Vaughn Vernon: "Implementing Domain-Driven Design" - 2013



Scott Millett: "Patterns, Principles, and Practices of Domain-Driven Design" - 2015

# Tool 2: Event Sourcing & CQRS

# Event Sourcing

Use an append-only store to record the full series of events that describe actions taken on data in a domain, rather than storing just the current state, so that the store can be used to materialize the domain objects.

<https://msdn.microsoft.com/en-gb/library/dn589792.aspx>



# Command and Query Responsibility Segregation (CQRS)

Segregate operations that read data from operations that update data by using separate interfaces.

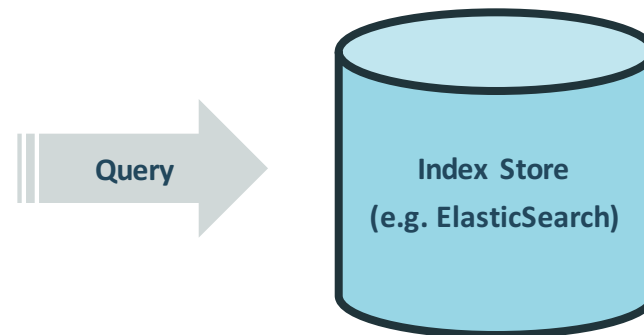
This pattern can maximize performance, scalability, and security; support evolution of the system over time through higher flexibility; and prevent update commands from causing merge conflicts at the domain level.

<https://msdn.microsoft.com/en-us/library/dn568103.aspx>

# Deposit Money Microservice



# Transactions List/Query Microservice



# What About Them Transactions?

# **Tool 3: Sagas**



## (Long-Lived Distributed Transactions)



# Booking Travel

Flight or  Train  Hotel  Car

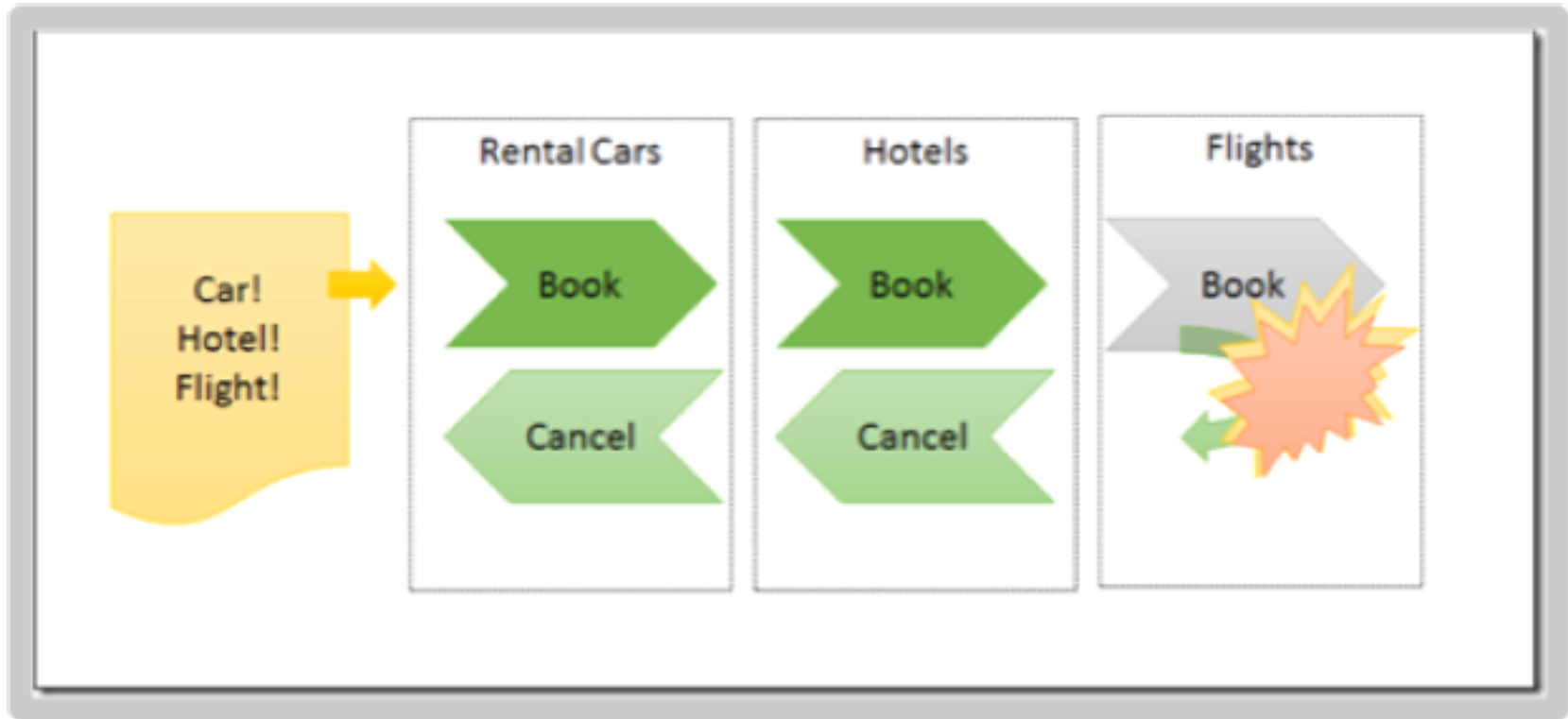
Round-trip  One-way  Multi-destination

Shop by Schedule  Shop by Price

From:  Depart:   Leaves  

To:  Return:   Leaves  

# No Shared State = No Distr. Transactions



Source: <http://vastars.com/clemensv/2012/09/01/Sagas.aspx>

## We ♥ Sagas!

Designed by: Hector Garcia-Molina & Kenneth Salem, Princeton, 1987

# One More Thing...



An elephant is standing on a wooden platform in a zoo enclosure. The elephant is facing right, and its trunk is visible. In the background, there is a wooden fence and a large umbrella. The text "How small should a Microservice be?" is overlaid on the image in a large, white, bold font with a black outline.

**How small should  
a Microservice be?**

**“Bounded context should be as big as it needs to be in order to fully express its complete Ubiquitous Language”**

**– Vaughn Vernon, *Implementing Domain–Driven Design*.**

**Many start coarse-grained and become more granular over time.**

*That said,* companies that have been building Microservices for long, report that they end with **hundreds** of Microservices.

**Where do we get  
hundreds of servers?**

# How do we afford so many servers?





## Irakli Nadareishvili

Director of Strategy, API Academy



@inadarei

@apiacademy

@cainc