Microservices from zero to hero

DAN NASTASA
FLORIN OLARIU

May 16, 2017

centric
connect.engage.succeed.

AGENDA

- About Centric
- “Italian” Architecture and Microservices ☺
- Onion and Spring
- What Is a Microservice?
- Building Monolithic Applications
- Marching Toward Monolithic Hell
- Microservices – Tackling the Complexity
- The Benefits of Microservices
- The Drawbacks of Microservices
- Why Microservices?
- Demo
- About Centric Internship
- Summary
- Bibliography

ABOUT CENTRIC

centric

ABOUT CENTRIC

- It is a Dutch company

ABOUT CENTRIC

- It is a Dutch company
- Portfolio

ABOUT CENTRIC

- It is a Dutch company
- Portfolio
  - Software solutions

ABOUT CENTRIC

- It is a Dutch company
- Portfolio
  - Software solutions
  - IT Outsourcing

ABOUT CENTRIC

- It is a Dutch company
- Portfolio
  - Software solutions
  - IT Outsourcing
  - Business process outsourcing

ABOUT CENTRIC

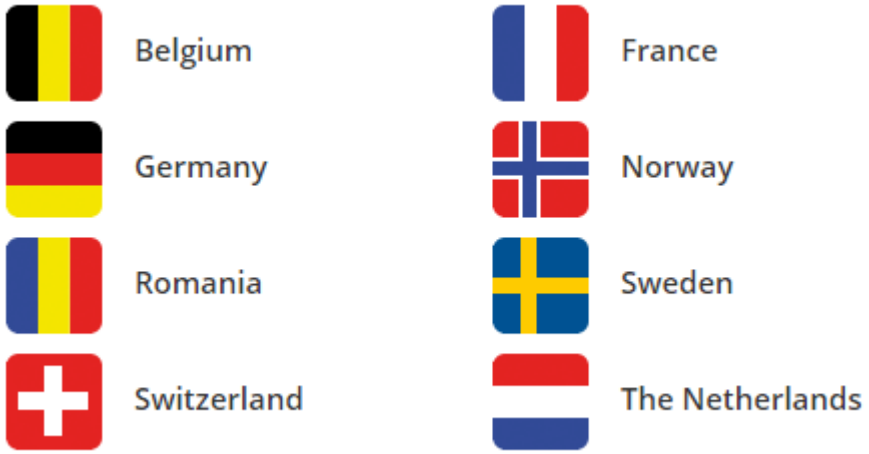
- It is a Dutch company
- Portfolio
  - Software solutions
  - IT Outsourcing
  - Business process outsourcing
  - Staffing services

ABOUT CENTRIC

- It is a Dutch company
- Portfolio
  - Software solutions
  - IT Outsourcing
  - Business process outsourcing
  - Staffing services
- Geographic Area

ABOUT CENTRIC

- It is a Dutch company
- Portfolio
  - Software solutions
  - IT Outsourcing
  - Business process outsourcing
  - Staffing services
- Geographic Area



centric

“ITALIAN” ARCHITECTURE AND MICROSERVICES ☺

“ITALIAN” ARCHITECTURE AND MICROSERVICES ☺

“ITALIAN” ARCHITECTURE AND MICROSERVICES ☺

“ITALIAN” ARCHITECTURE AND MICROSERVICES ☺

ONION AND SPRING

ONION AND SPRING

- **Domain Model layer**, where our entities and classes closely related to them e.g. value objects reside

ONION AND SPRING

- **Domain Model layer**, where our entities and classes closely related to them e.g. value objects reside
- **Domain Services layer**, where domain-defined processes reside

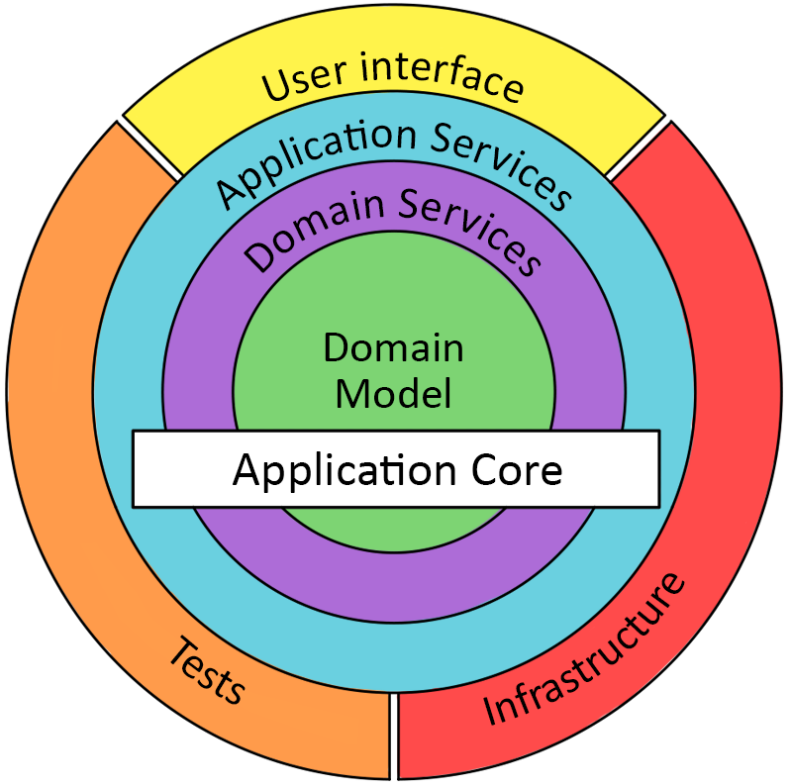
ONION AND SPRING

- **Domain Model layer**, where our entities and classes closely related to them e.g. value objects reside
- **Domain Services layer**, where domain-defined processes reside
- **Application Services layer**, where application-specific logic i.e. our use cases reside

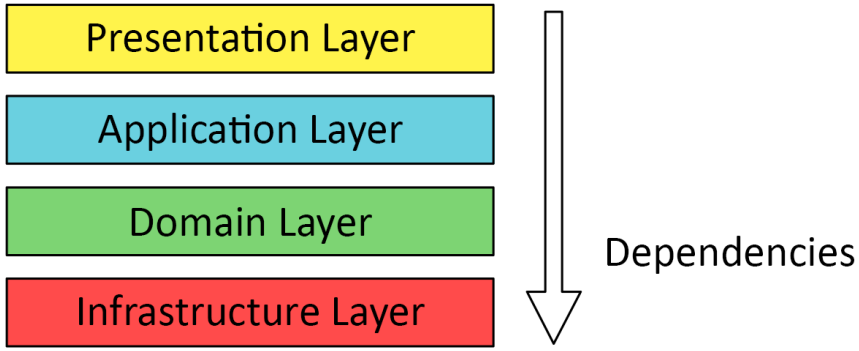
ONION AND SPRING

- **Domain Model layer**, where our entities and classes closely related to them e.g. value objects reside
- **Domain Services layer**, where domain-defined processes reside
- **Application Services layer**, where application-specific logic i.e. our use cases reside
- **Outer layer**, which keeps peripheral concerns like UI, databases or tests

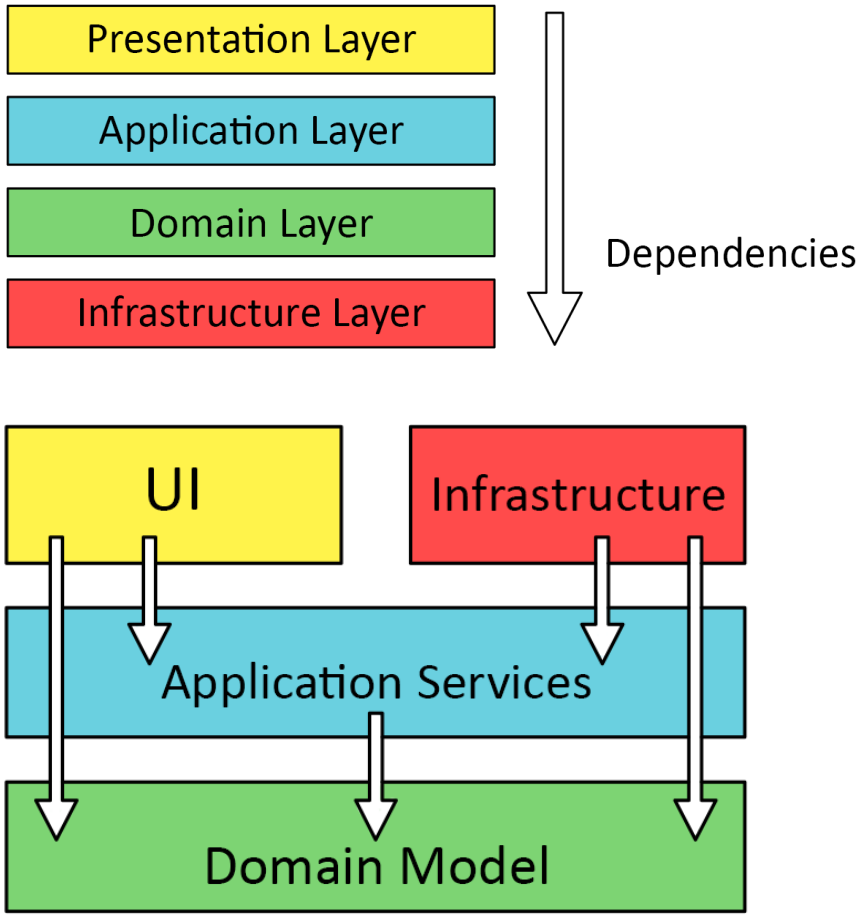
ONION AND SPRING



ONION AND SPRING

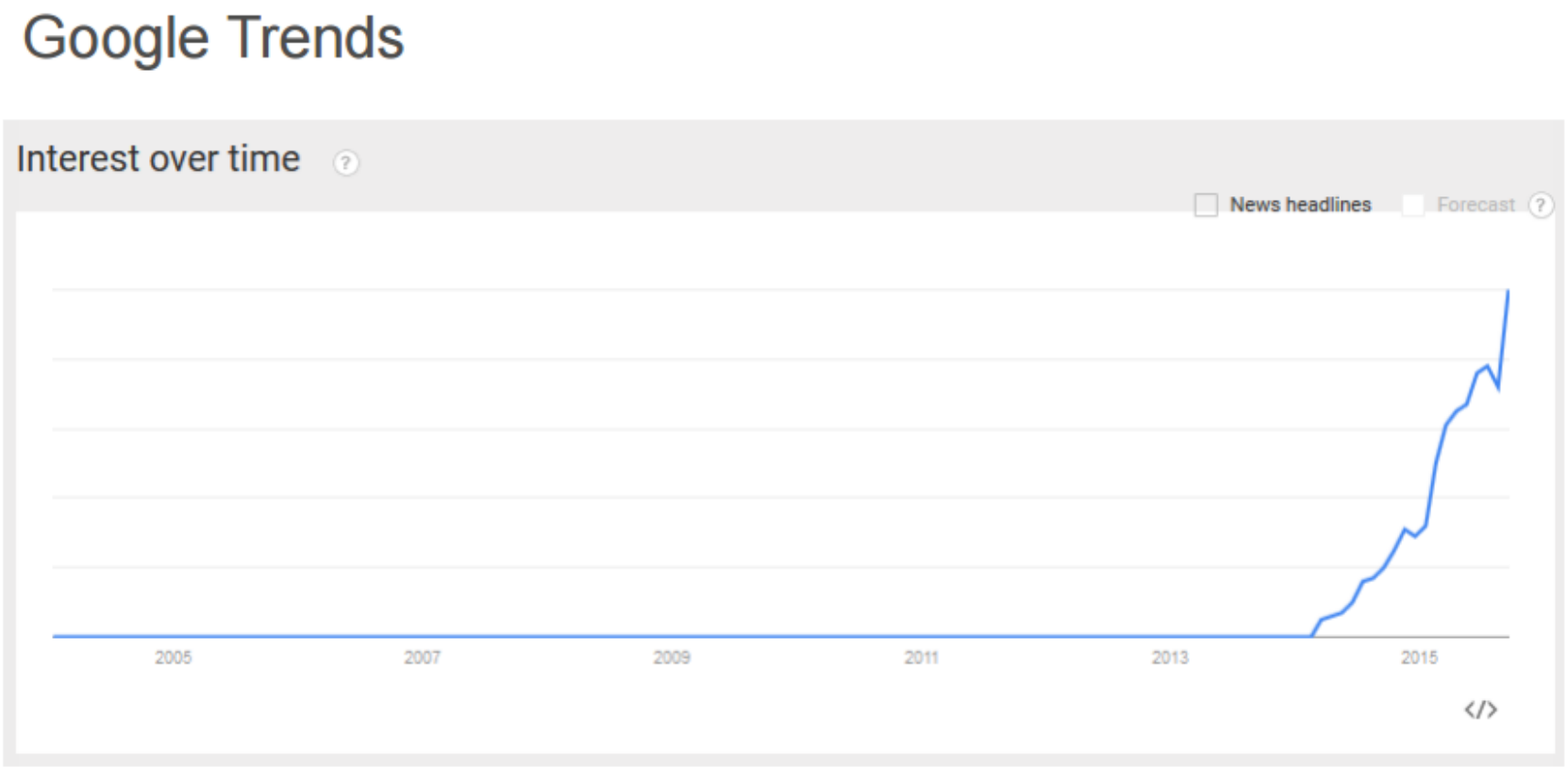


ONION AND SPRING



WHAT IS A MICROSERVICE?

WHAT IS A MICROSERVICE?



centric

WHAT IS A MICROSERVICE?

- “Microservices are a thing these days.”

  Phil Calçado, former Director of Engineering, SoundCloud

WHAT IS A MICROSERVICE?

- “Microservices are small, autonomous services that work together.”

  Sam Newman, Thoughtworks

WHAT IS A MICROSERVICE?

- “Loosely coupled service-oriented architecture with bounded contexts.”

  Adrian Cockcroft, Battery Ventures

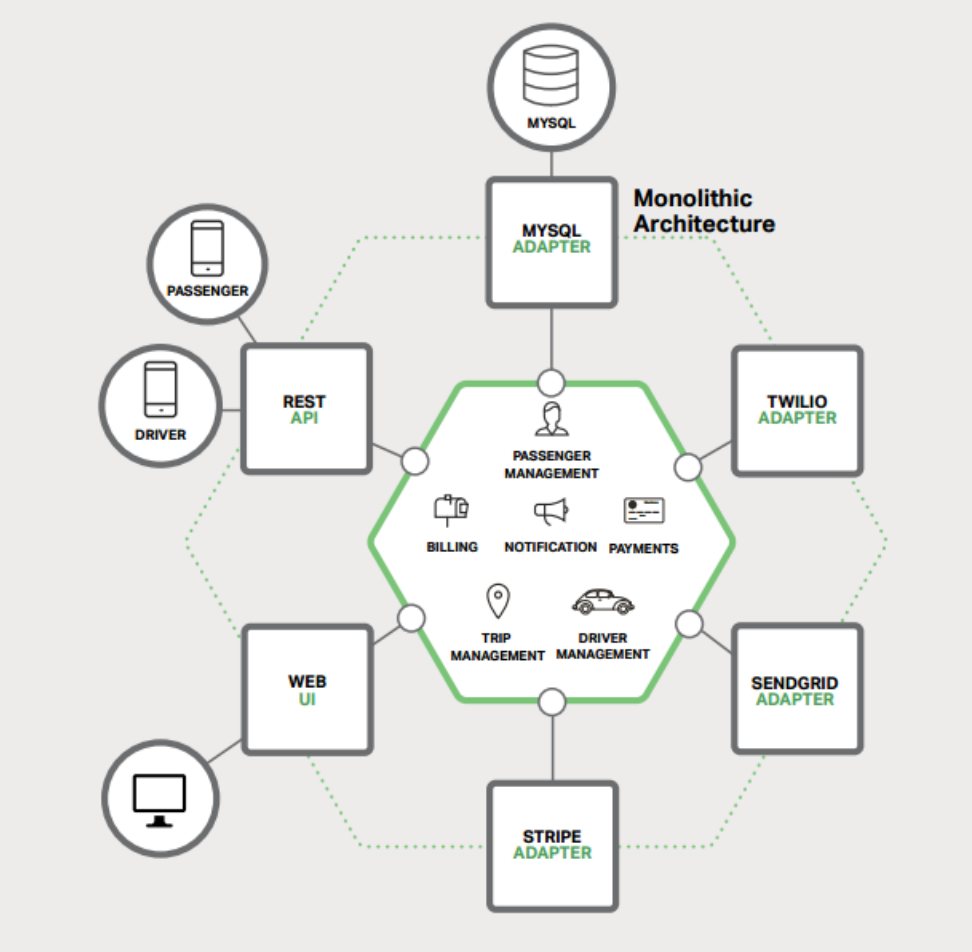
WHAT IS A MICROSERVICE?

- “A microservice is an independently deployable component of bounded scope that supports interoperability through message-based communication.”

  “Microservice Architecture-Aligning Principles, Practices, and Culture”

BUILDING MONOLITHIC APPLICATIONS

BUILDING MONOLITHIC APPLICATIONS



centric

BUILDING MONOLITHIC APPLICATIONS

- These applications are simple to test and debug.

BUILDING MONOLITHIC APPLICATIONS

- These applications are simple to test and debug.
- These applications are also simple to deploy.

BUILDING MONOLITHIC APPLICATIONS

- These applications are simple to test and debug.
- These applications are also simple to deploy.
- These applications are scalable.

MARCHING TOWARD MONOLITHIC HELL

MARCHING TOWARD MONOLITHIC HELL

- In time the application become too complex.

MARCHING TOWARD MONOLITHIC HELL

- In time the application become too complex.
- Being too large is very difficult for any developer to fully understand.

MARCHING TOWARD MONOLITHIC HELL

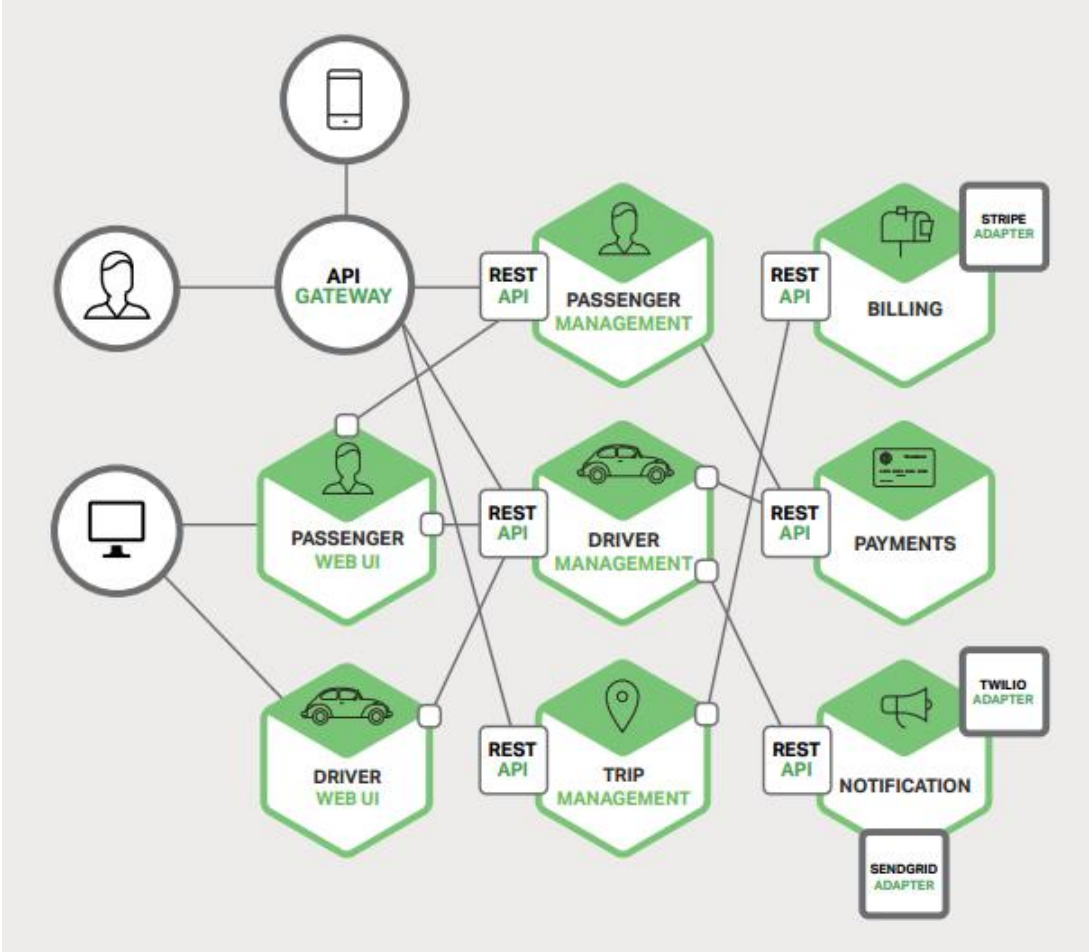
- In time the application become too complex.
- Being too large is very difficult for any developer to fully understand.
- A large application is an obstacle to continuous deployment.

MARCHING TOWARD MONOLITHIC HELL

- In time the application become too complex.
- Being too large is very difficult for any developer to fully understand.
- A large application is an obstacle to continuous deployment.
- Another problem with monolithic applications is reliability.

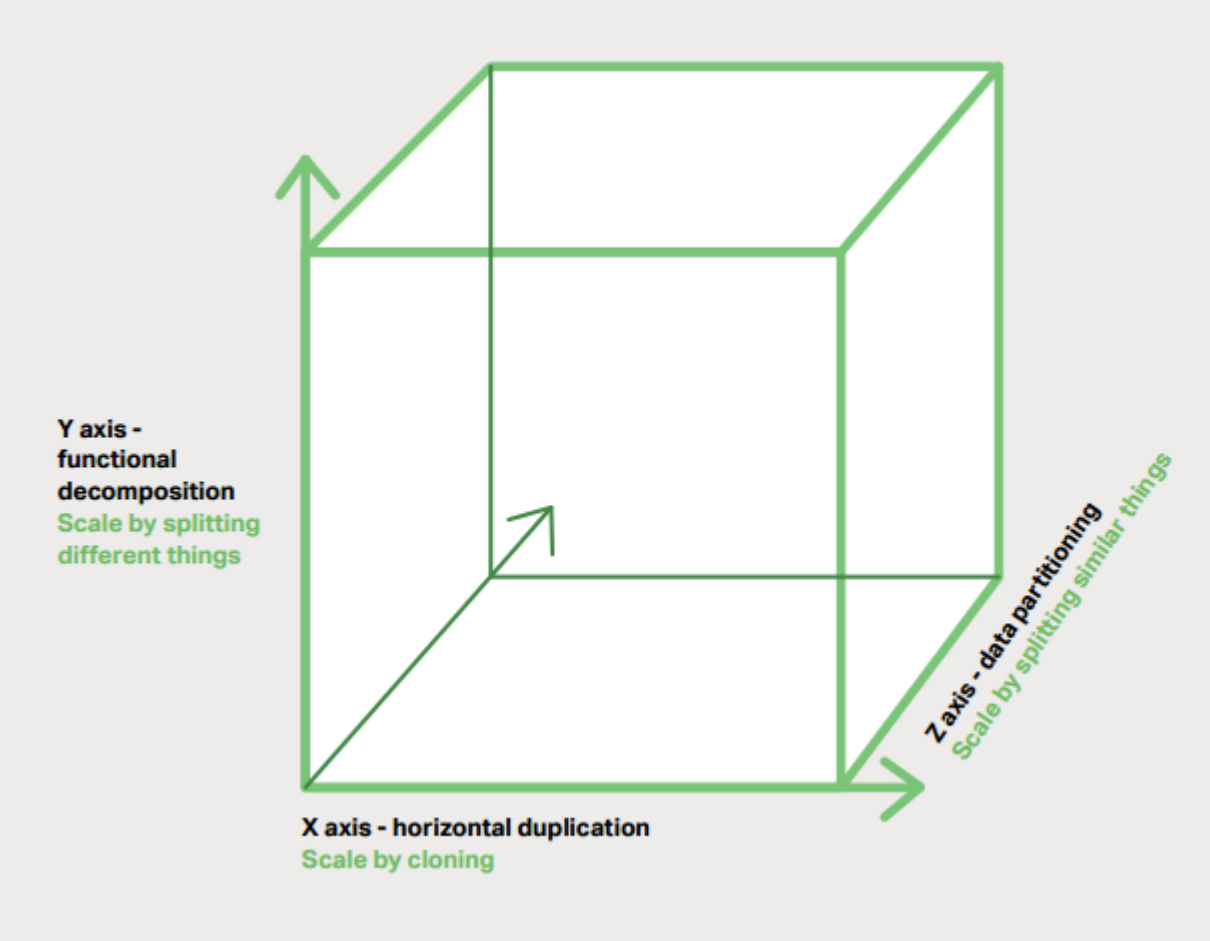
MICROSERVICES – TACKLING THE COMPLEXITY

MICROSERVICES – TACKLING THE COMPLEXITY



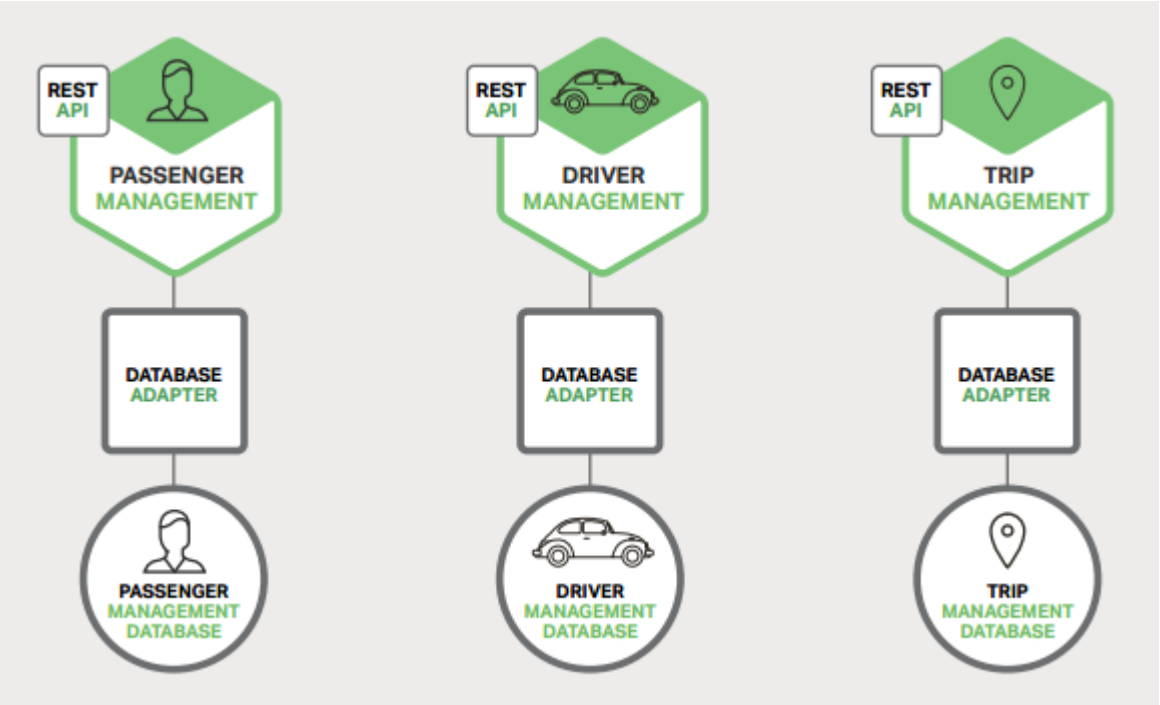
centric

MICROSERVICES – TACKLING THE COMPLEXITY



centric

MICROSERVICES – TACKLING THE COMPLEXITY



centric

THE BENEFITS OF MICROSERVICES

THE BENEFITS OF MICROSERVICES

- It tackles the problem of complexity.

THE BENEFITS OF MICROSERVICES

- It tackles the problem of complexity.
- Enforces modularity.

THE BENEFITS OF MICROSERVICES

- It tackles the problem of complexity.
- Enforces modularity.
- Enables each service to be developed independently by a team that is focused on that service.

THE BENEFITS OF MICROSERVICES

- It tackles the problem of complexity.
- Enforces modularity.
- Enables each service to be developed independently by a team that is focused on that service.
- Enables each microservice to be deployed independently.

THE BENEFITS OF MICROSERVICES

- It tackles the problem of complexity.
- Enforces modularity.
- Enables each service to be developed independently by a team that is focused on that service.
- Enables each microservice to be deployed independently.
- Enables each service to be scaled independently.

THE DRAWBACKS OF MICROSERVICES

THE DRAWBACKS OF MICROSERVICES

- The name itself.

THE DRAWBACKS OF MICROSERVICES

- The name itself.
- The complexity that arises from the fact that a microservices application is a distributed system.

THE DRAWBACKS OF MICROSERVICES

- The name itself.
- The complexity that arises from the fact that a microservices application is a distributed system.
- The partitioned database architecture.

THE DRAWBACKS OF MICROSERVICES

- The name itself.
- The complexity that arises from the fact that a microservices application is a distributed system.
- The partitioned database architecture.
- Testing is also much more complex.

THE DRAWBACKS OF MICROSERVICES

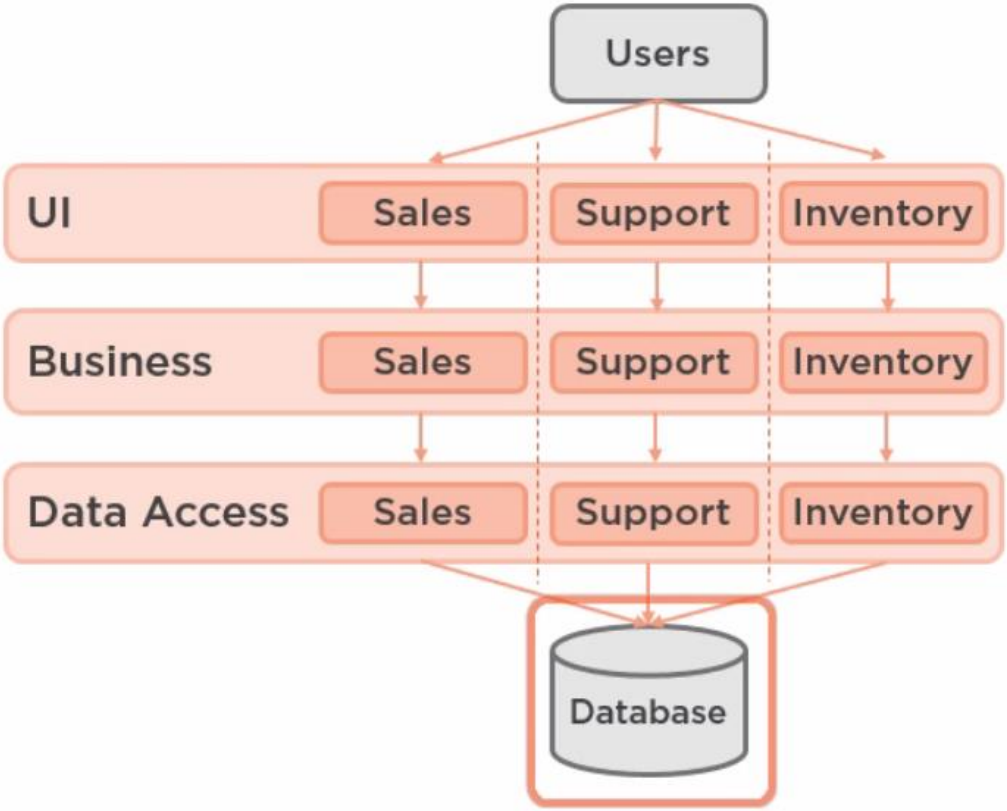
- The name itself.
- The complexity that arises from the fact that a microservices application is a distributed system.
- The partitioned database architecture.
- Testing is also much more complex.
- Deploying a microservices-based application is also much more complex.

WHY MICROSERVICES?

WHY MICROSERVICES?

- **Gilt**: “From Monolith Ruby App to Distributed Scala Micro-Services” (NYC Tech Talks) [Link]
- **Nike**: “Nike’s Journey to Microservices” (AWS Re:Invent 2014) [Link]
- **SoundCloud**: ”Building Products at SoundCloud - Part III: Microservices in Scala and Finagle” [Link]
- **Capital One**: “Lack Of Legacy Lets Capital One Build Nimble Infrastructure” [Link]
- **Hailo**: “A Journey into Microservices” [Link]
- **Autoscout24**: “Why Autoscout24 changes its technology” [Link]
- **Zalando**: “From Monolith to Microservices” [Link]

WHY MICROSERVICES?



centric

WHY MICROSERVICES?

Sales

Sales opportunity

Contact

Sales person

Product

Sales territory

Support

Support ticket

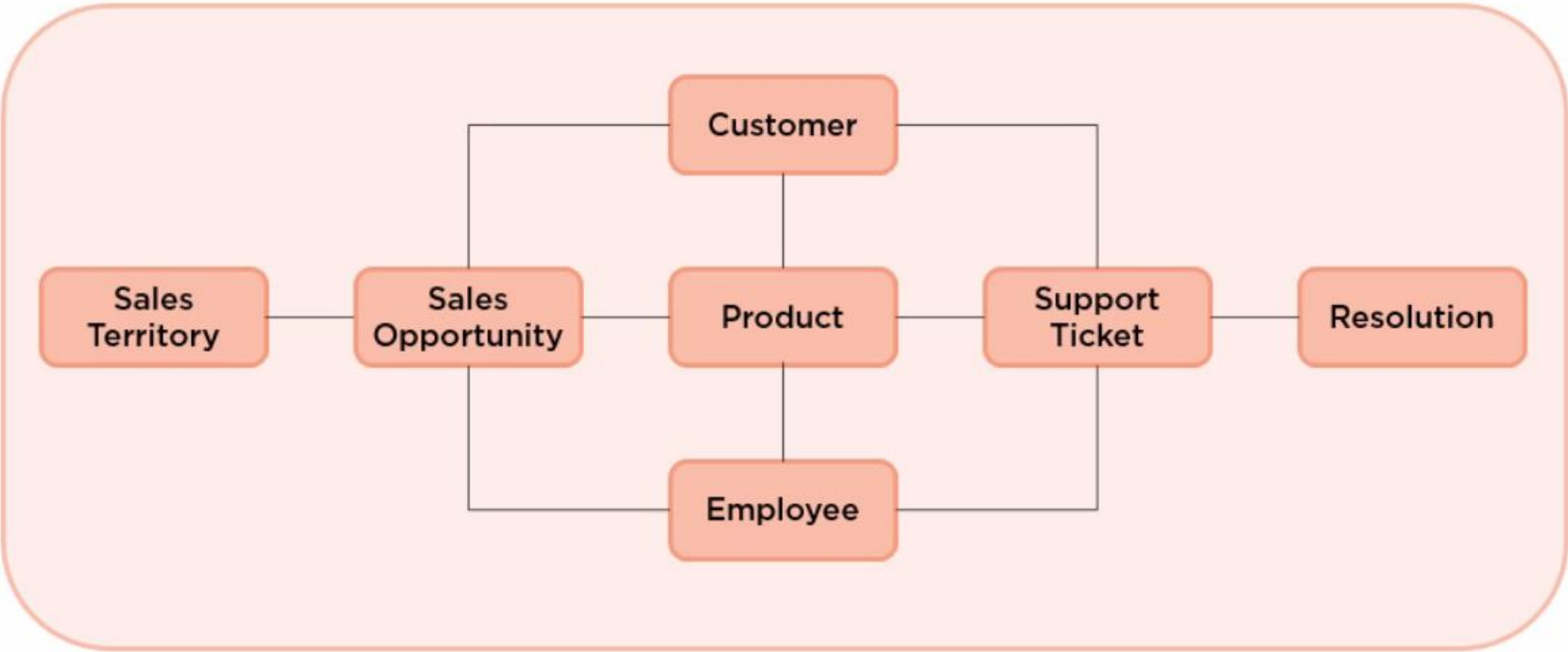
Customer

Support person

Product

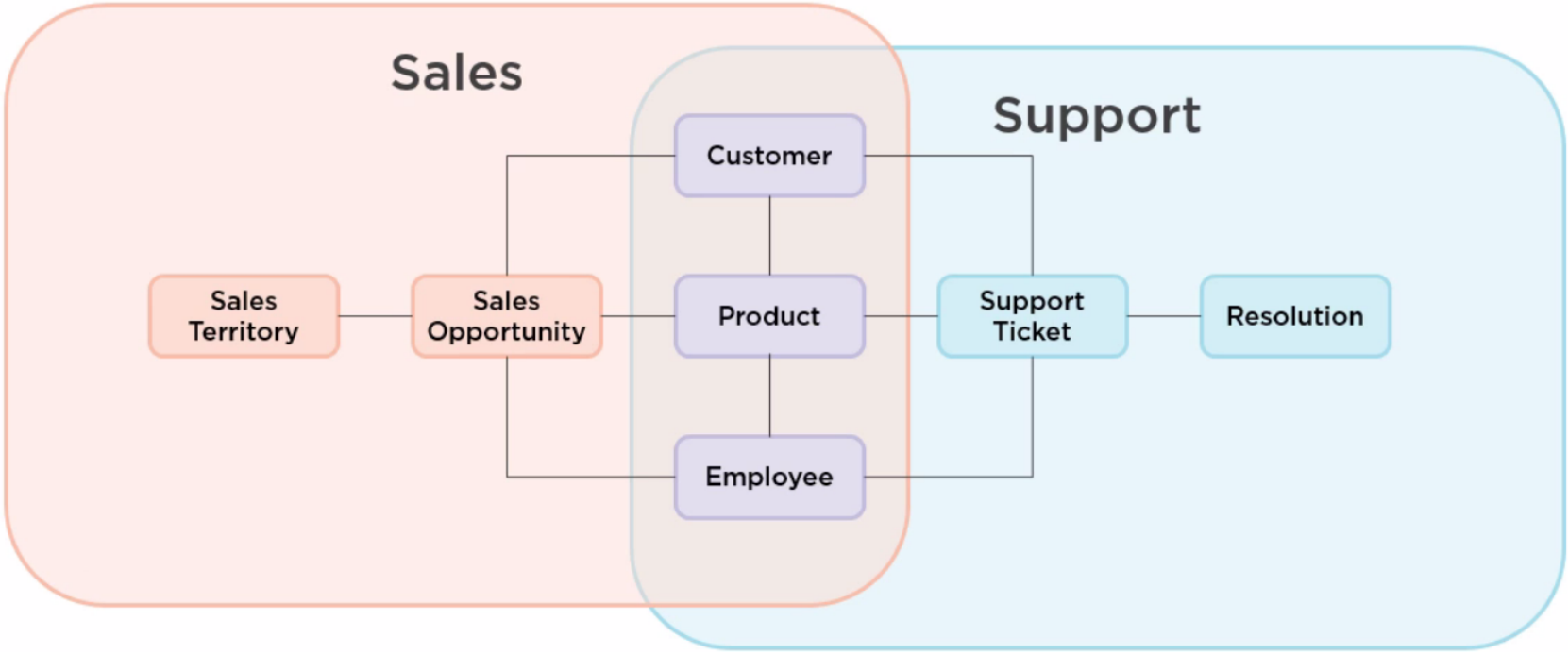
Resolution

WHY MICROSERVICES?

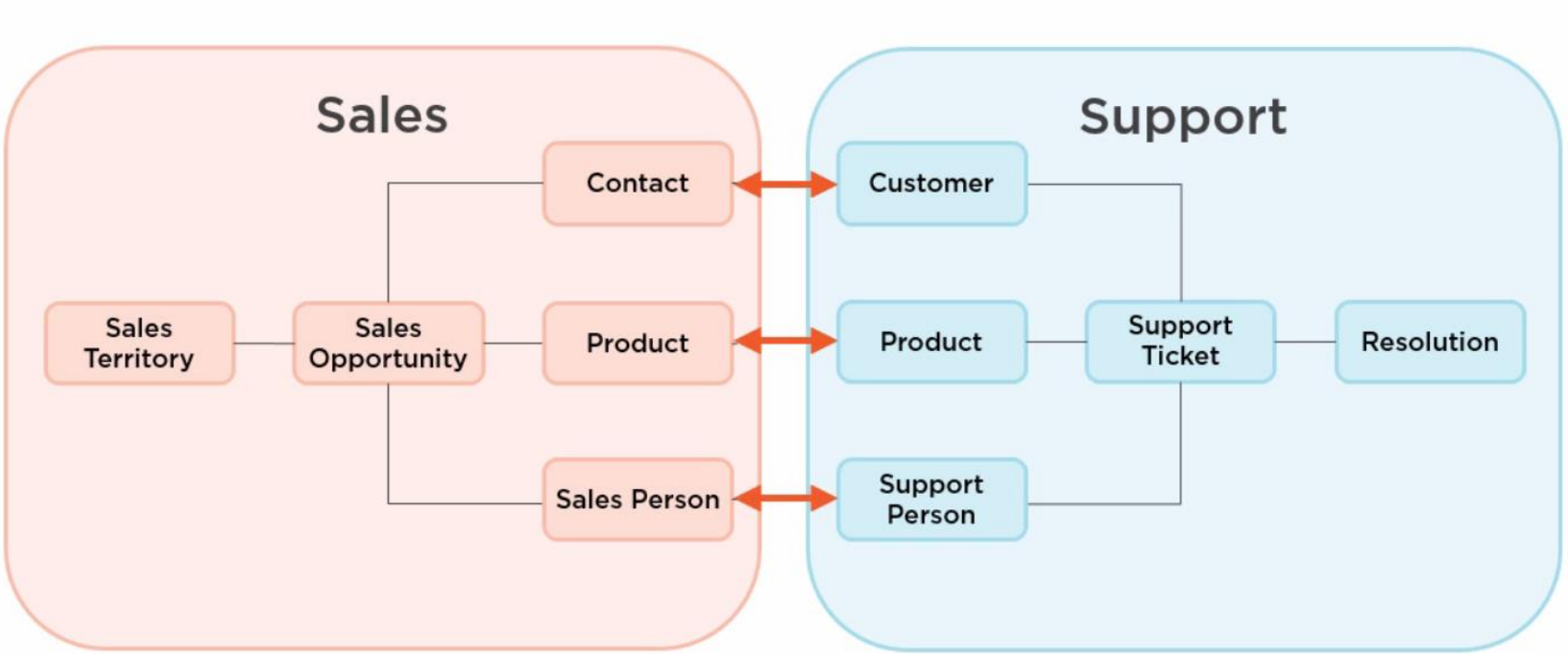


centric

WHY MICROSERVICES?



WHY MICROSERVICES?



centric

WHY MICROSERVICES?

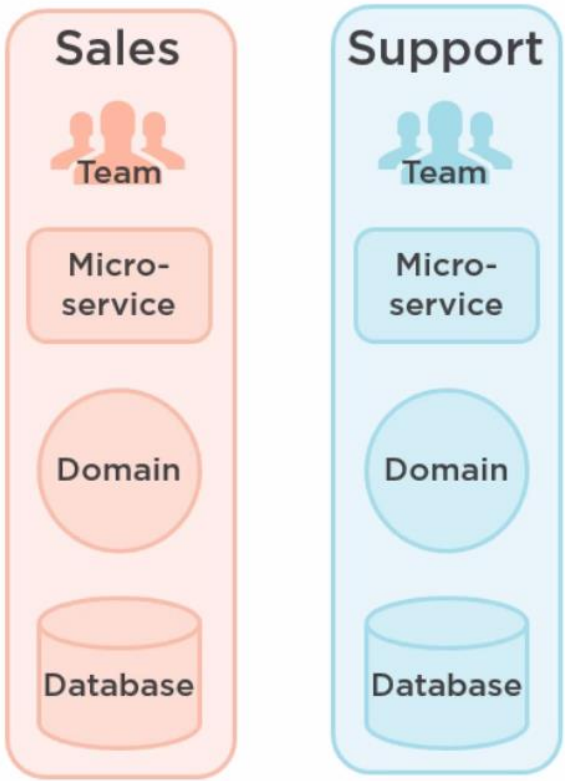
Bounded context

Cohesion/coupling

Single domain of knowledge

Consistent data model

Independence



centric

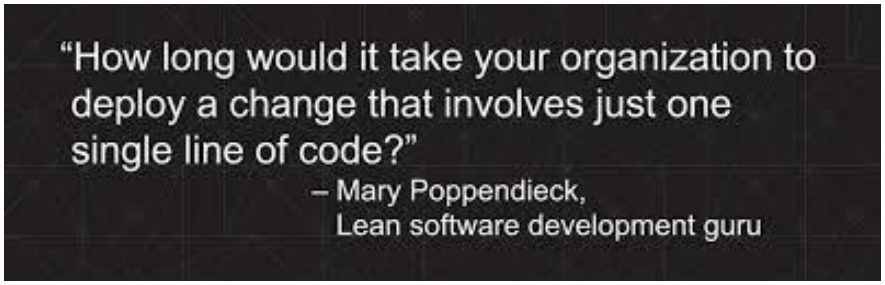
DEMO

ABOUT CENTRIC INTERNSHIP

centric

ONE MORE THING …

ONE MORE THING …



SUMMARY

SUMMARY

- Building complex applications is not an easy task.

SUMMARY

- Building complex applications is not an easy task.
- The Monolithic Architecture pattern only makes sense for simple, lightweight applications.

SUMMARY

- Building complex applications is not an easy task.
- The Monolithic Architecture pattern only makes sense for simple, lightweight applications.
- The Microservices Architecture pattern is the better choice for complex, evolving applications, despite the drawbacks and implementation challenges.

SUMMARY

- Building complex applications is not an easy task.
- The Monolithic Architecture pattern only makes sense for simple, lightweight applications.
- The Microservices Architecture pattern is the better choice for complex, evolving applications, despite the drawbacks and implementation challenges.
- Probably the best way to define boundaries for microservices is by using Bounded Context from DDD (Domain Driven Design)

BIBLIOGRAPHY

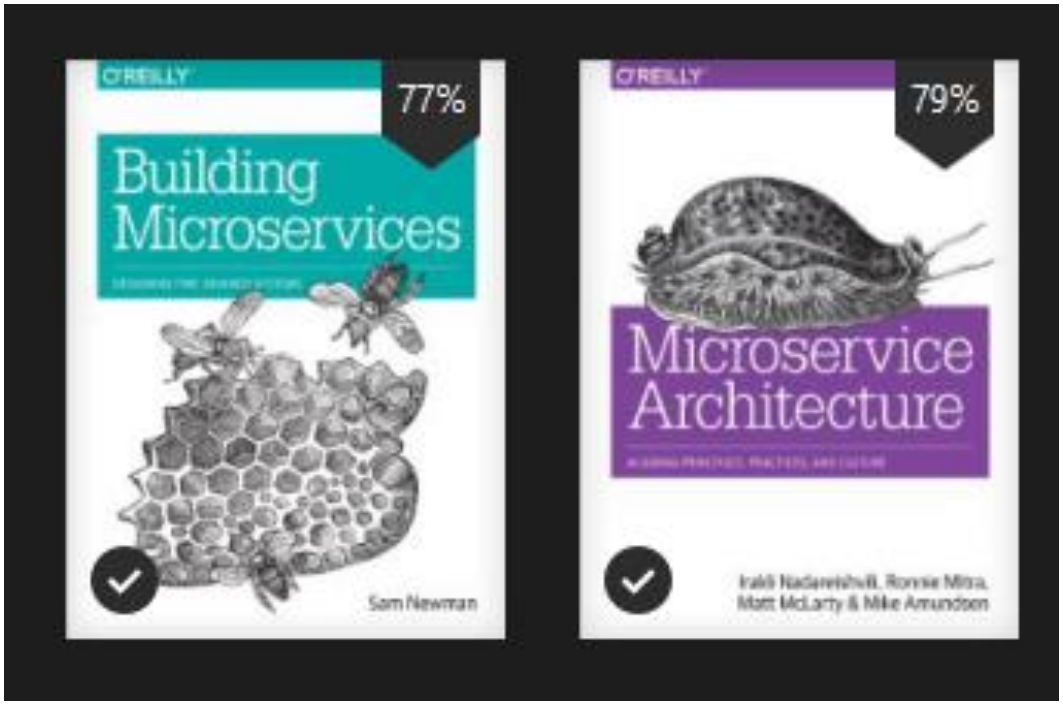
centric

BIBLIOGRAPHY

- Alagarasan, Vijay. “Seven Microservices Anti-patterns”, August 24, 2015.
- Cockcroft, Adrian. “State of the Art in Microservices”, December 4, 2014.
- Fowler, Martin. “Microservice Prerequisites”, August 28, 2014.
- Fowler, Martin. “Microservice Tradeoffs”, July 1, 2015.
- Humble, Jez. “Four Principles of Low-Risk Software Release”, February 16, 2012.
- Humble, Jez, Chris Read, and Dan North. “The Deployment Production Line”. In Proceedings of the conference on AGILE 2006, 113– 118.
- IEEE Computer Society. Kniberg, Henrik, and Anders Ivarsson. “Scaling Agile at Spotify”, October 2012.
- http://microservices.io/patterns/microservices.html

centric

BIBLIOGRAPHY



BIBLIOGRAPHY

centric

QUESTIONS?

centric
connect.engage.succeed.

THANK YOU!

dan.nastasa@centric.eu
florin.olariu@centric.eu

DAN NASTASA
FLORIN OLARIU

May 16, 2017

centric
connect.engage.succeed.