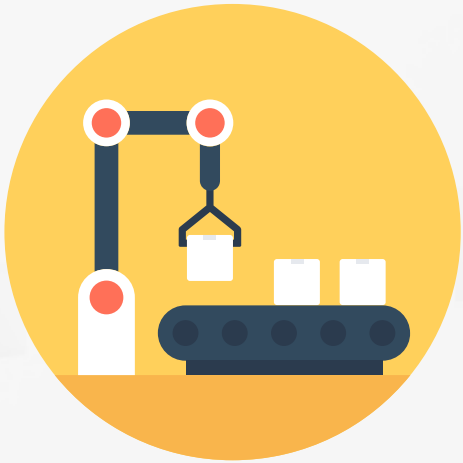
Microservices
– experiences from the front line

Stig Irming-Pedersen

copenhagen software


copenhagen
software

Device production



- 50 years old company
- Established globally with many subsidiaries/distributors/resellers
- Challenge: Large and closed system landscape in HQ for internal use
- Goal: Automate customer processes with self-service
- Strategy: Develop online ordering integrated with new ERP and other systems

Icons from www.flaticon.com

Content delivery

- 25 years old company
- Transforming business from shipping physical mediums to online streaming
- Challenge: The web is moving fast and the system is a large monolith
- Goal: API as a product
- Strategy: Split into services first, then move authoritative data afterwards

Parallel distribution

- 15 years old company
- Rapid growth throughout Europe leveraging custom software for many different areas
- Challenge: Better insights across growing organization, and expansions require further custom software
- Goal: Use shared system for standard functionality and produce custom software where necessary
- Strategy: Move to SaaS-provided ERP and leverage microservices

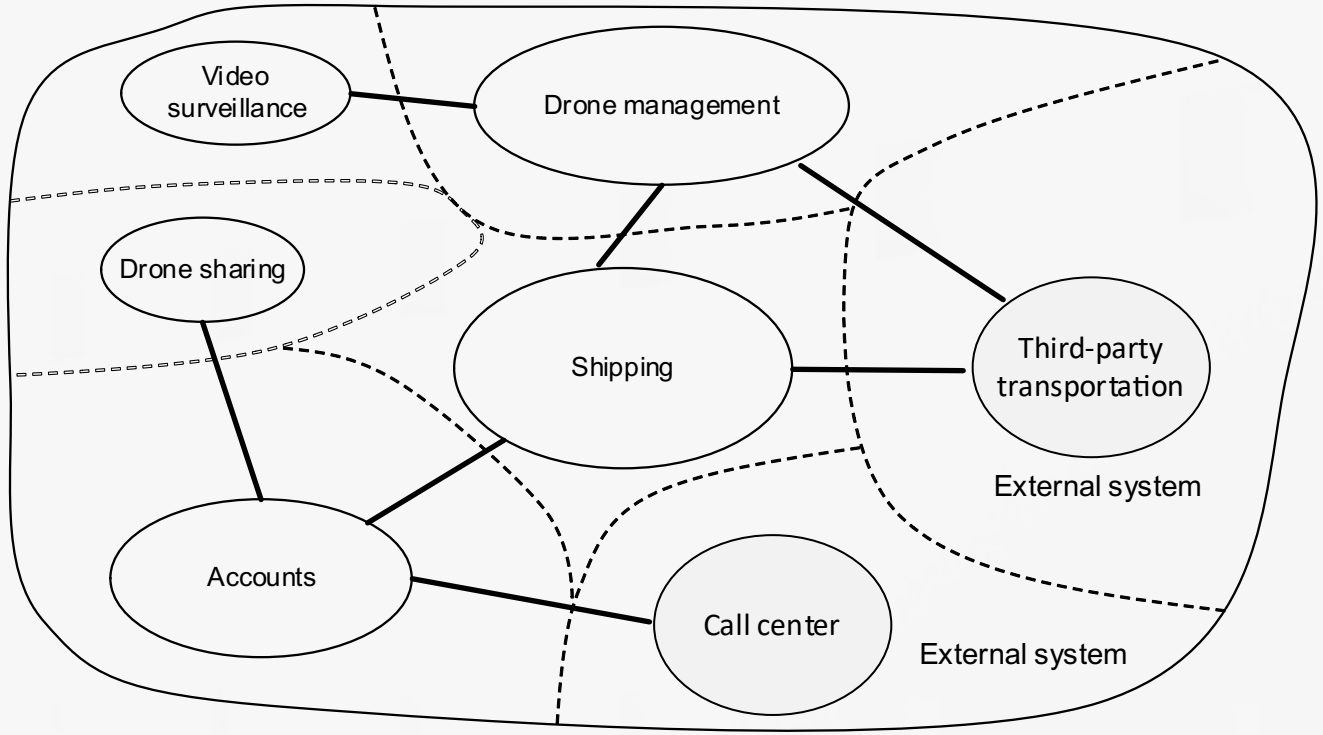
Microservice introduction

- Split software into smaller chunks
- … and leverage great principles for software development:
  - Agile methodologies
  - Domain driven design
  - Clean code
  - Test driven development
  - Cross-functional teams
  - Continuous integration
  - Continuous delivery
  - DevOps mindset
  - Cloud native

Microservice characteristics

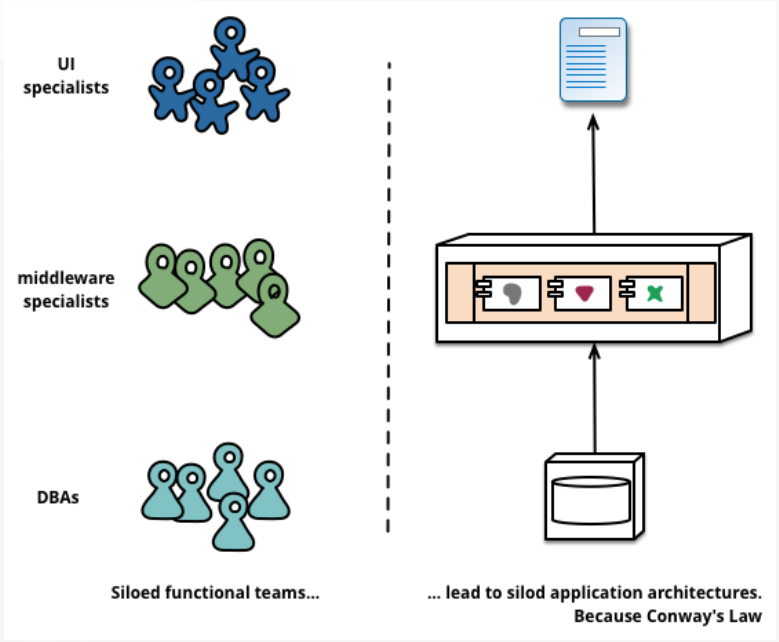
- Autonomous for own functionality
- Authoritative for own data (master data)
- Run in isolated processes
- Individual technology choices
- Deployed independently
- Separate unit of scalability
- Part of distributed system
- Build to last for a long time

Define boundaries



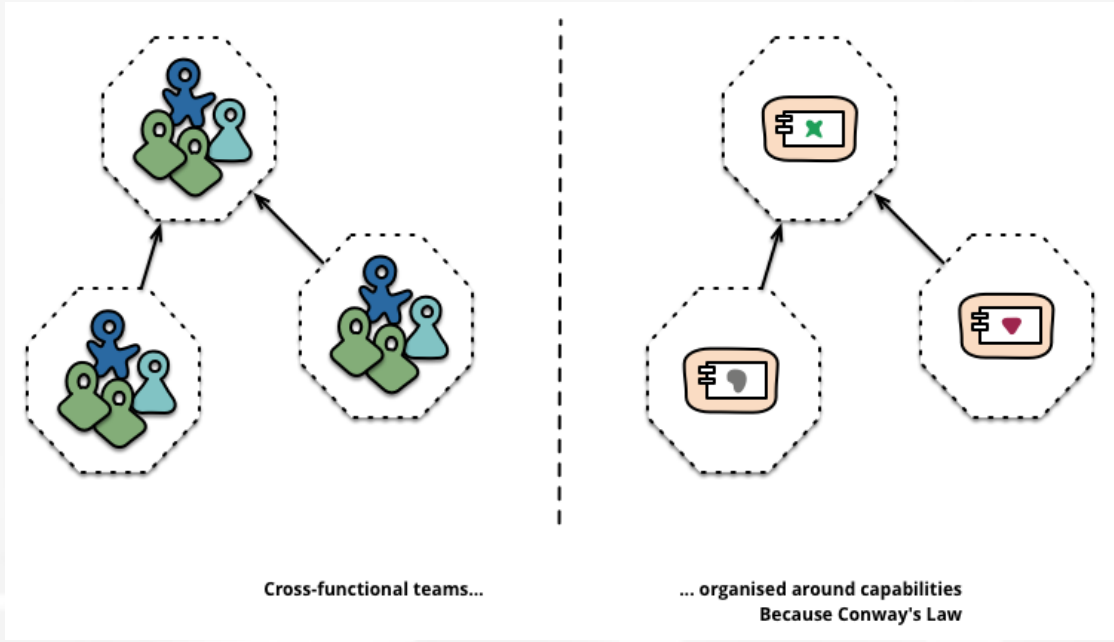
- Domain Driven Design by Eric Evans
- Bounded by business capabilities
- Stack fields that go together in piles
- High cohesion and invariants
- Core and supporting domains
- Strategic focus on differentiating software

Focused around organization



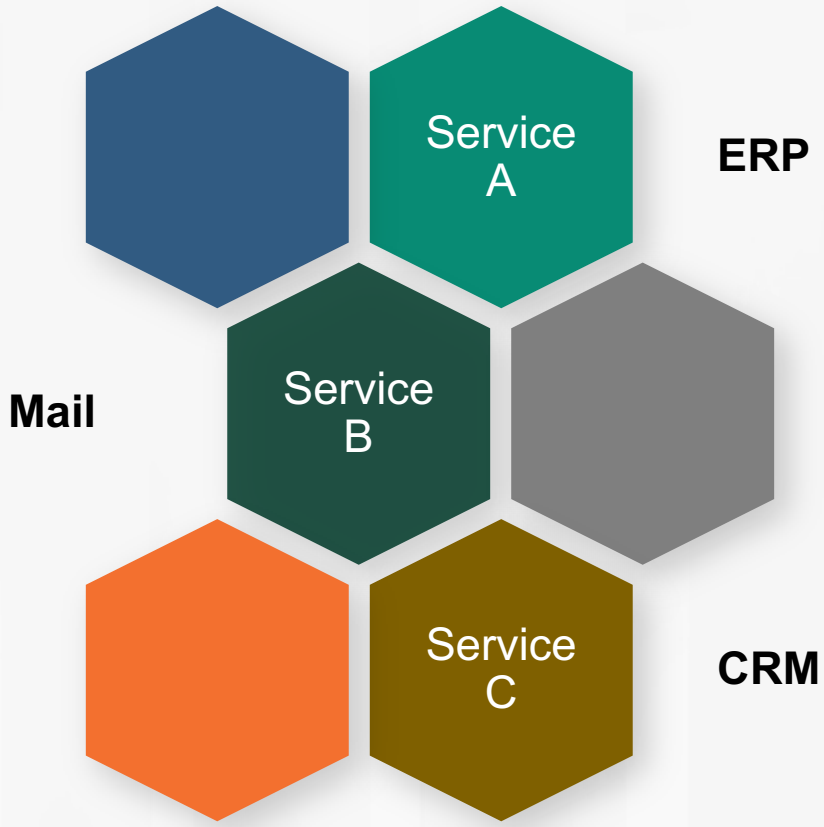
Conway's law:

"organizations which design systems ... are constrained to produce designs which are copies of the communication structures of these organizations."

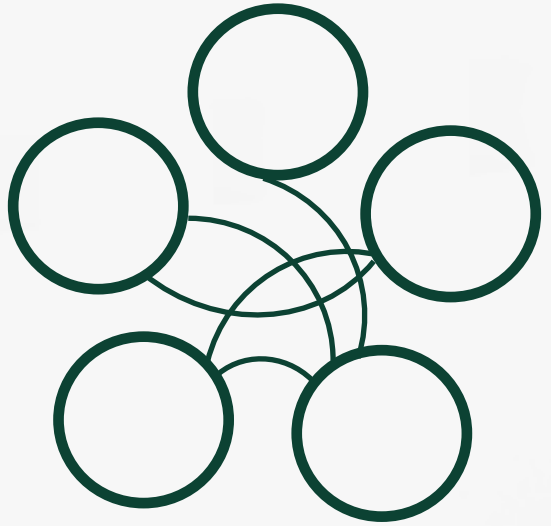


https://martinfowler.com/articles/microservices.html

Service landscape

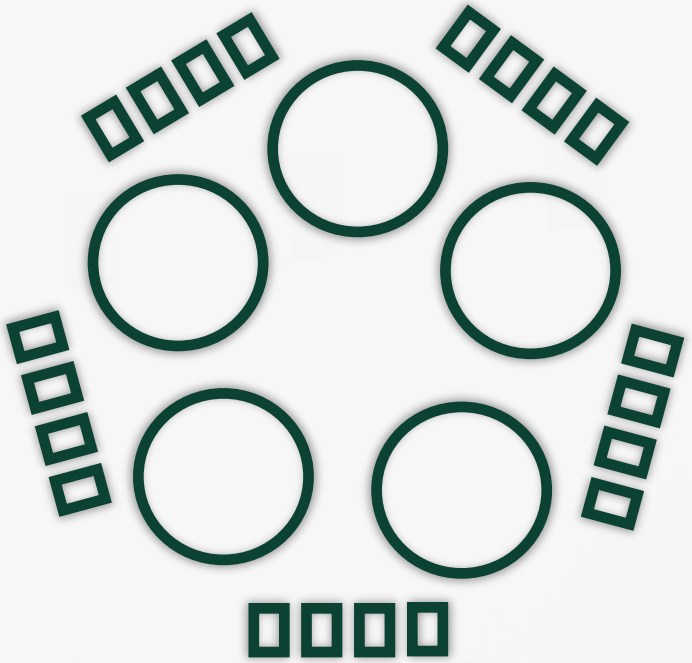


APIs



- REST: Richardsons maturity model
  - 0. RPC
  - 1. Resources
  - 2. HTTP verbs
  - 3. Hypermedia

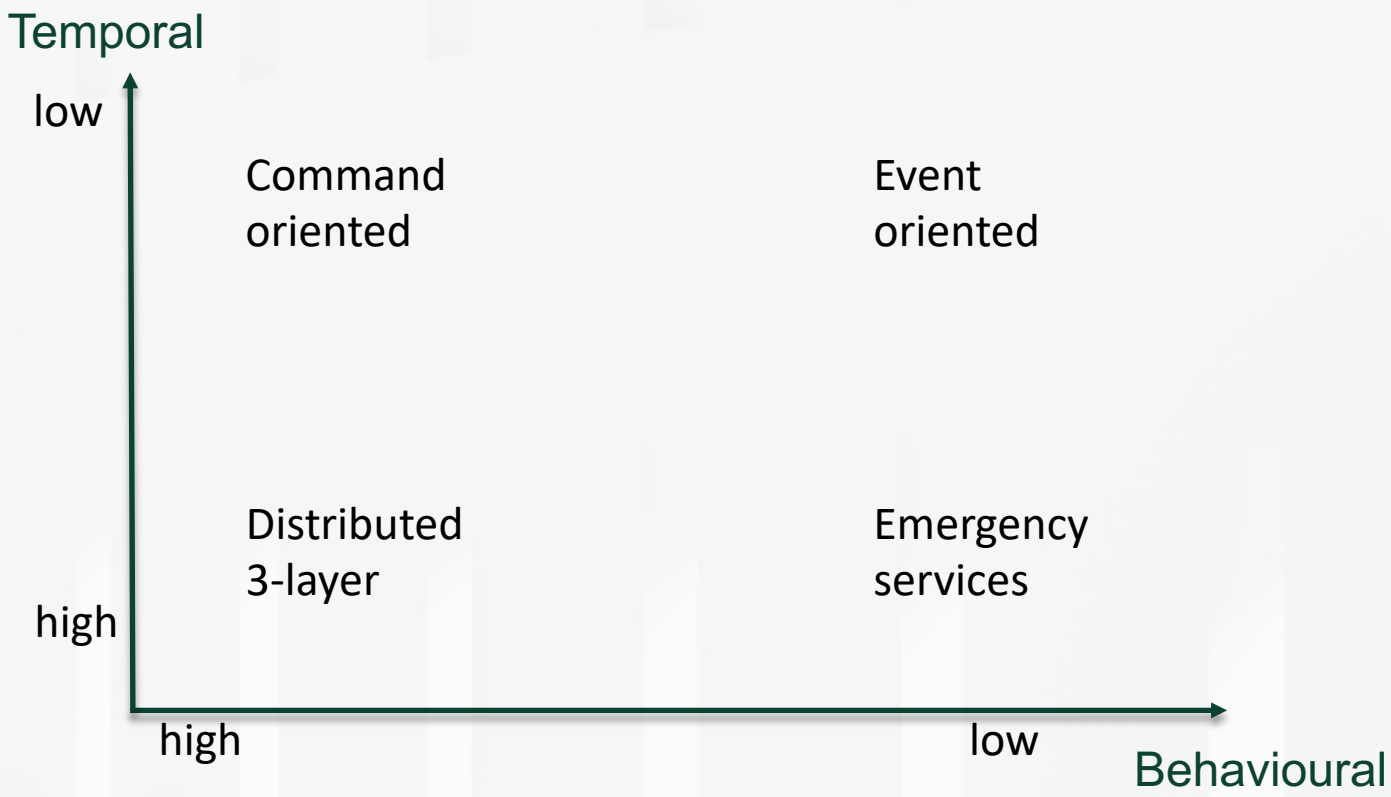
Messaging



Communication patterns:

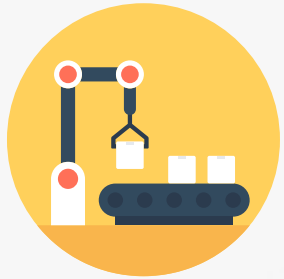
- Orchestrated (commands)
- Choreographed (events)

Coupling



http://iansrobinson.com/2009/04/27/temporal-and-behavioural-coupling/

How we chose com-munication

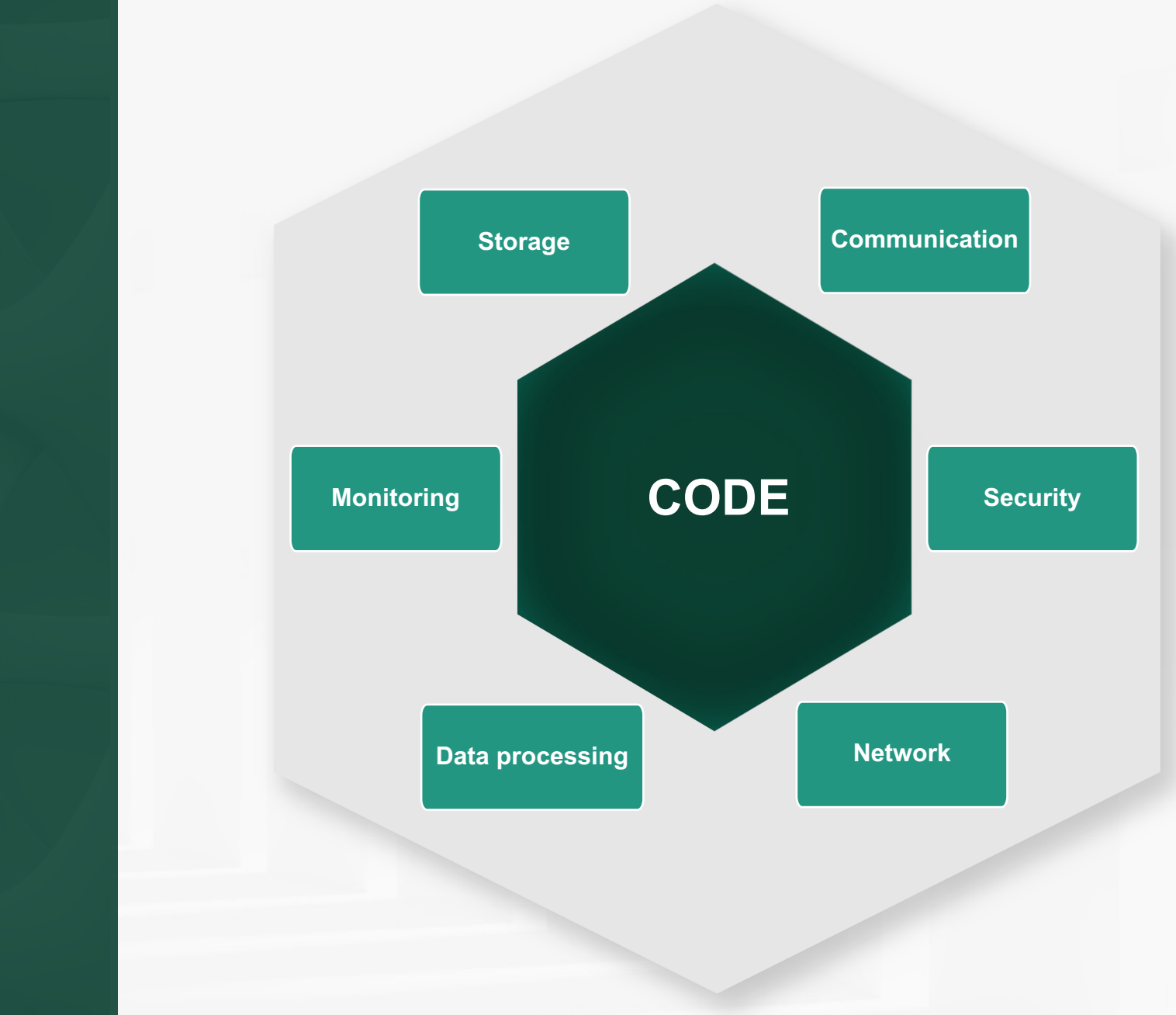


Advocate services => REST

Many external clients => REST

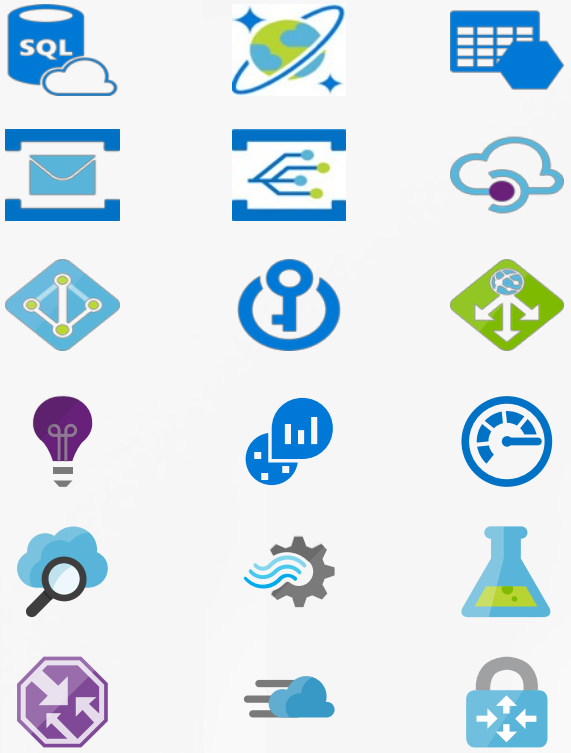
Integrating internal systems => Eventing

Components and infrastructure



Cloud components

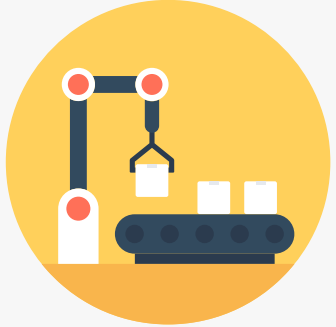
- Storage
- Communication
- Security
- Monitoring
- Processing
- Network



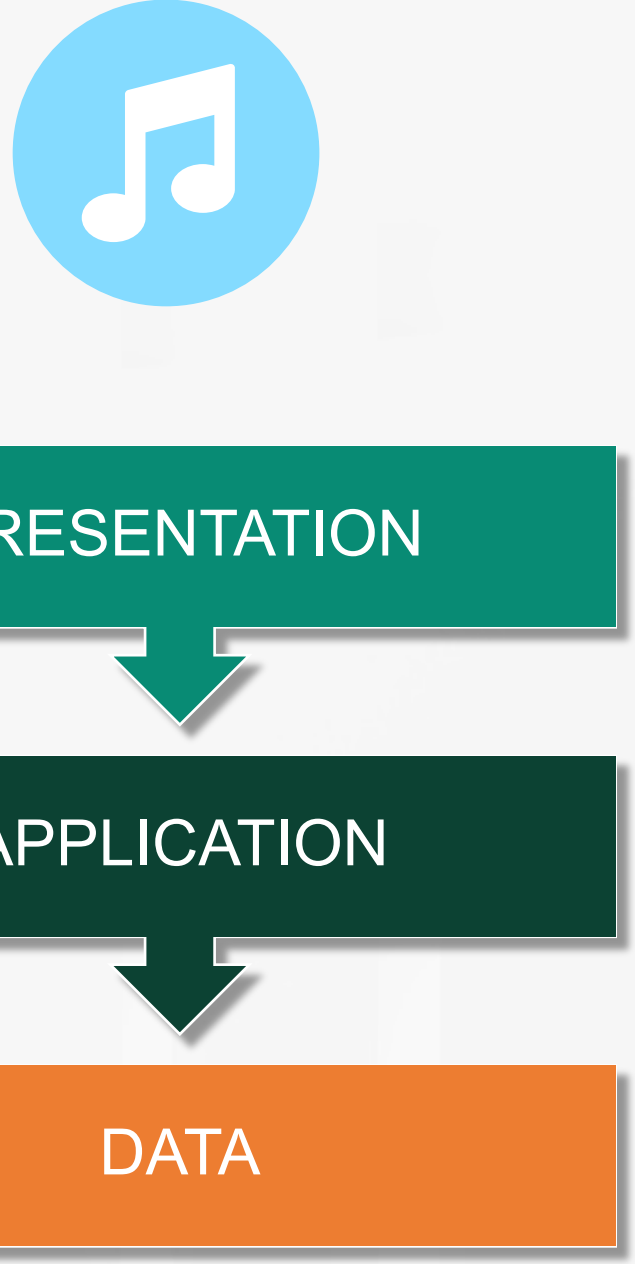
Azure advantages

- **Operation**: setup, monitoring, updates
- **SLA**: typical three 9s or more
- **Redundancy**: duplicated across instances
- **Replication**: multiple copies of data (backup)
- **Security**: access control and encryption
- **Hardening**: continuous improvements
- **Monitoring**: metrics and logs
- **Ecosystem**: recommendations, policies and governance

Global presence



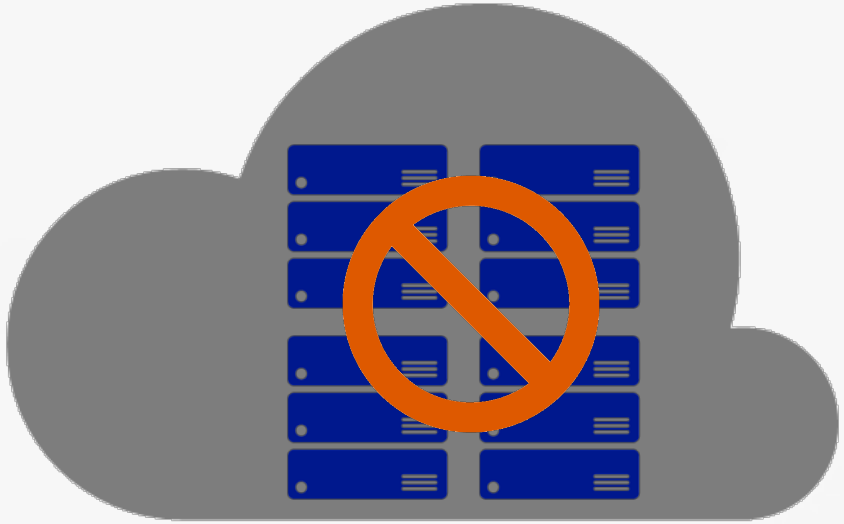
Splitting the monolith



Hosting levels

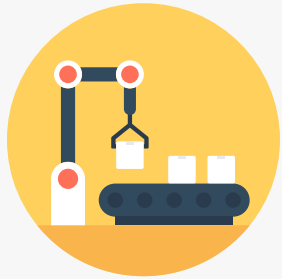
On-premise	Infrastructure as a Sevice (IaaS)	Platform as a service (PaaS)	Software as a service (SaaS)
Code	Code	Code	Code
Data	Data	Data	Data
Runtime	Runtime	Runtime	Runtime
Middleware	Middleware	Middleware	Middleware
Operating system	Operating system	Operating system	Operating system
Virtualization	Virtualization	Virtualization	Virtualization
Servers	Servers	Servers	Servers
Storage	Storage	Storage	Storage
Network	Network	Network	Network

Serverless

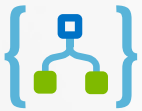


- **Code is run automatically**
  without explicit decision for capacity
- **Use back-end services**
  without knowledge of servers
- **Pay only for actual usage**
  not for reserved resources

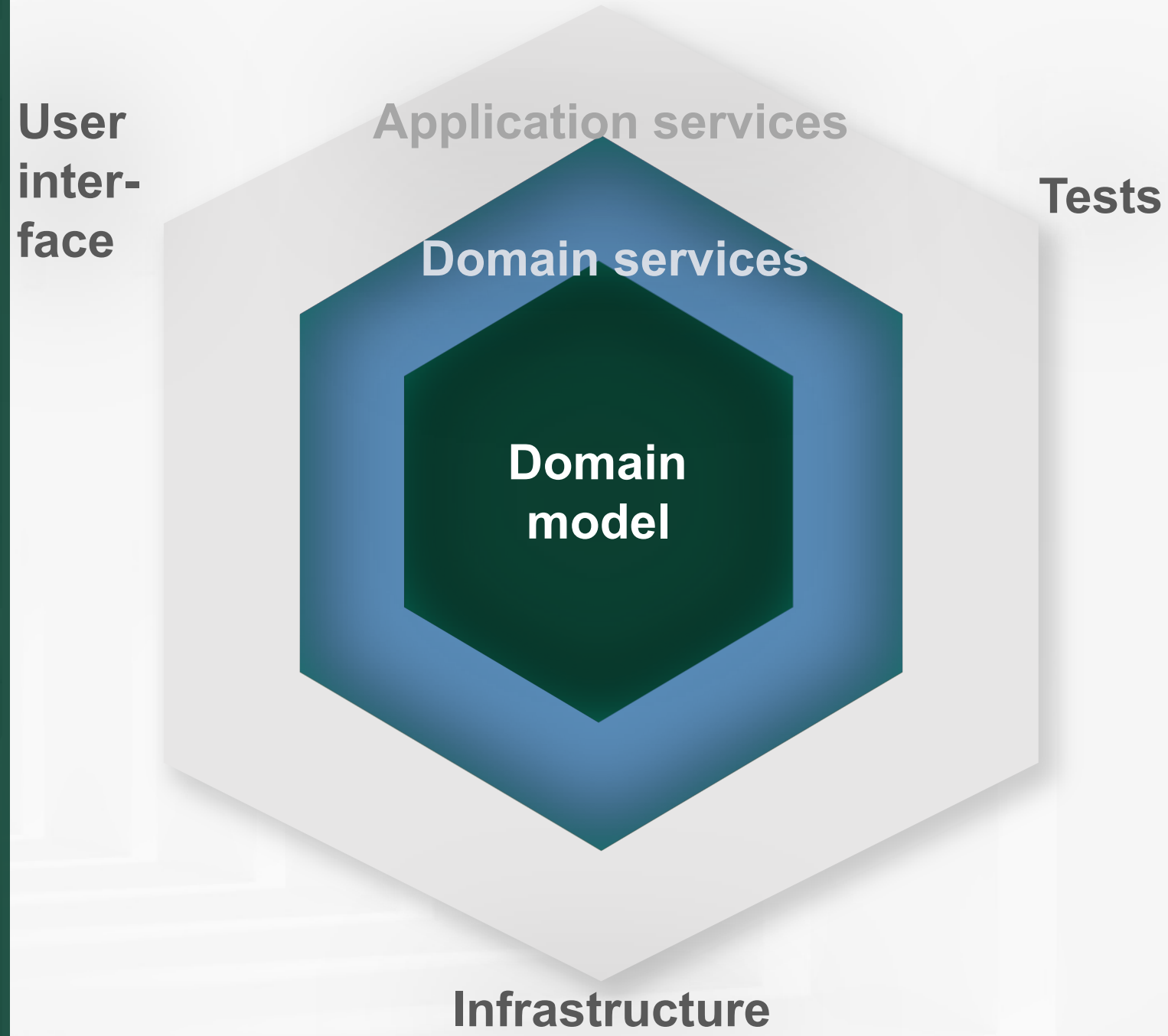
Chosen infrastructure and main components



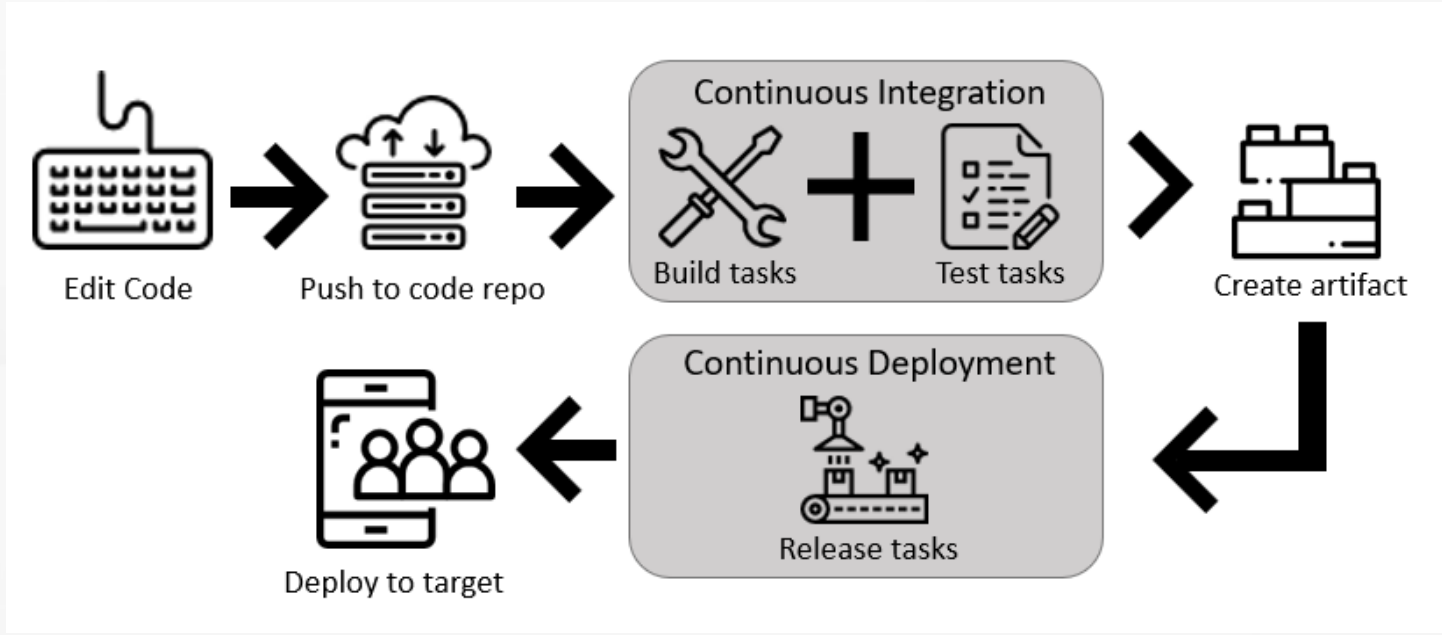
On-premise

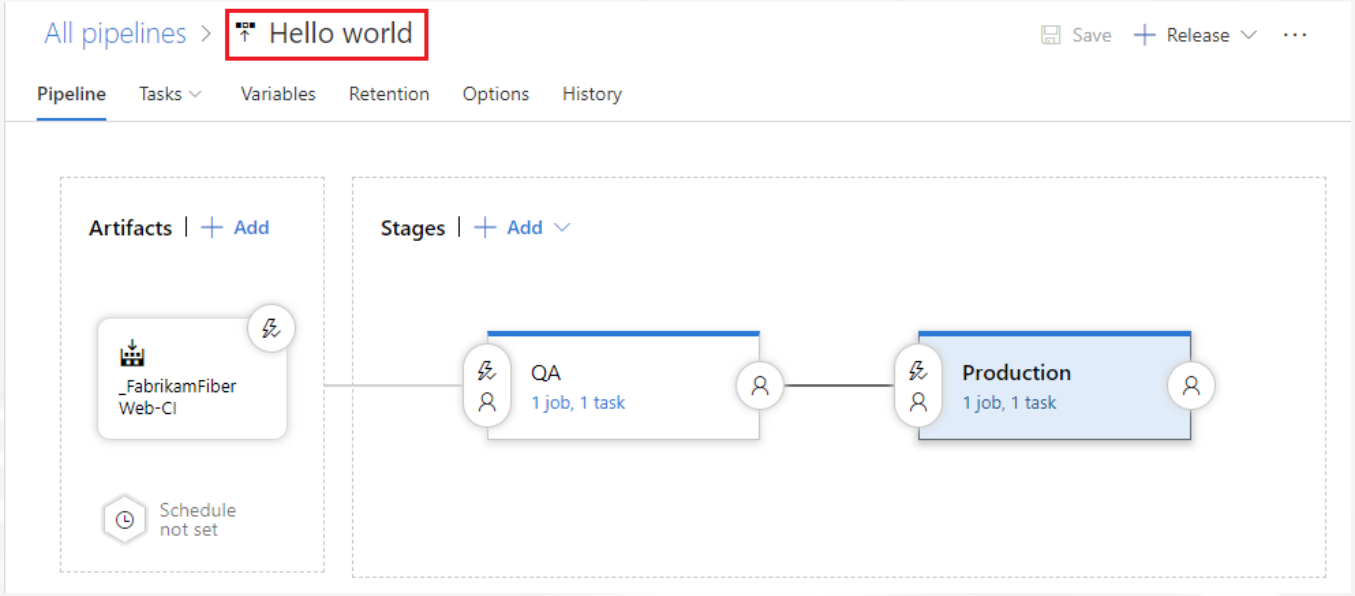


Hexagonal architecture



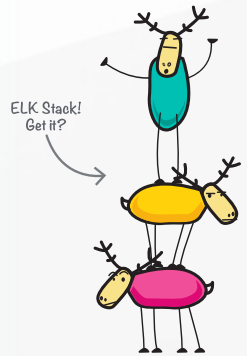
CI/CD





copenhagen software

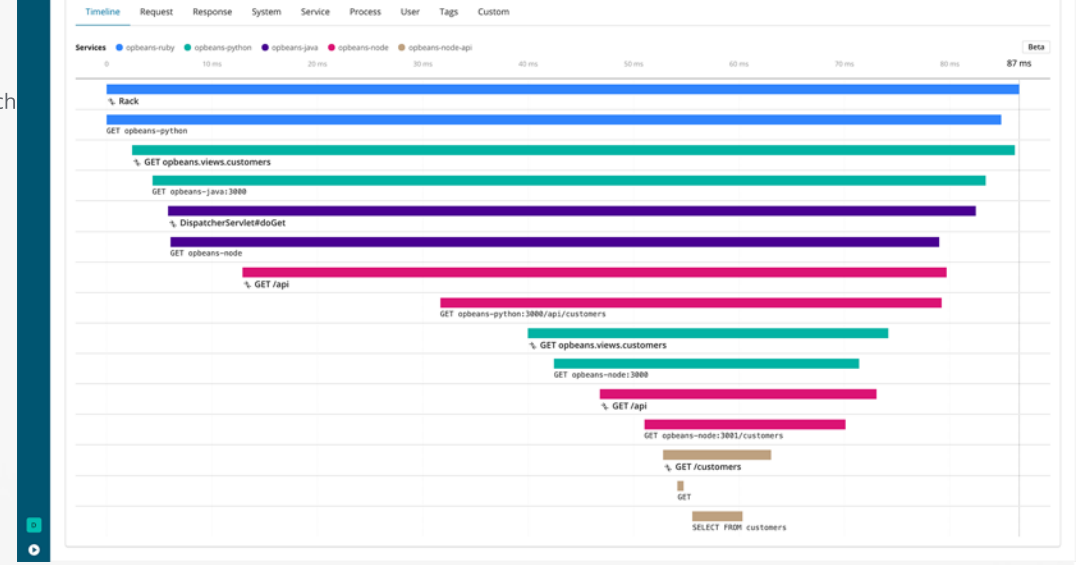
Monitoring

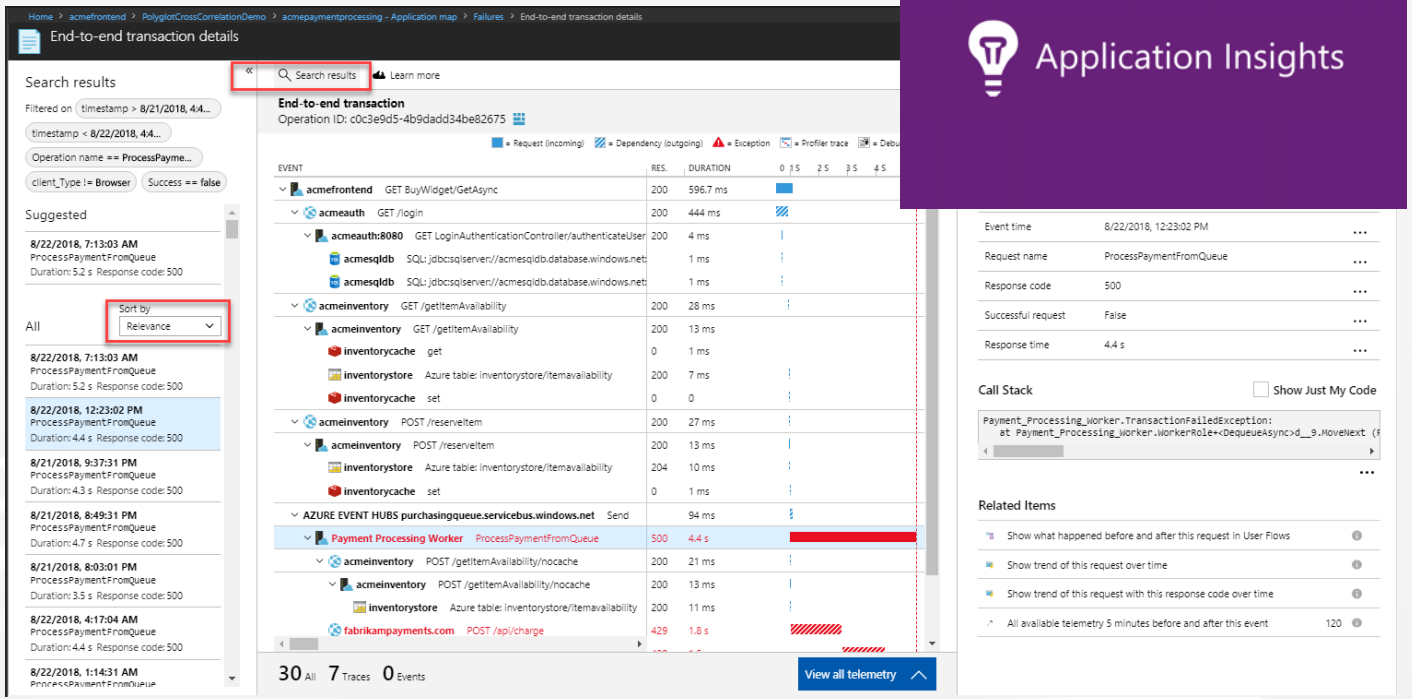


E Elasticsearch

L Logstash

K Kibana





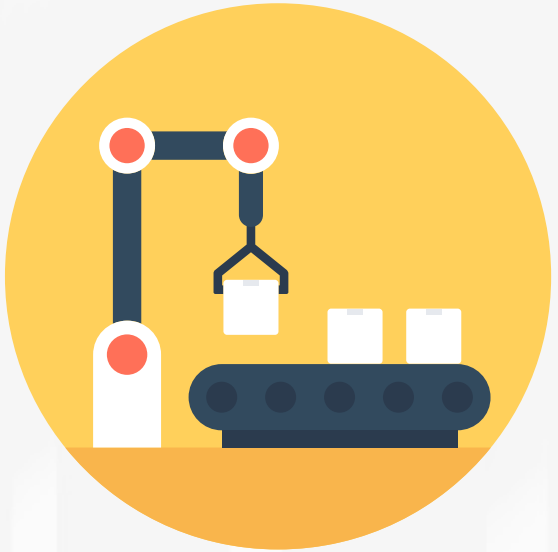
Application Insights

copenhagen software

Device production

ERP transition halted,
but microservices lives on

Enabling new technologies
has proved it's worth

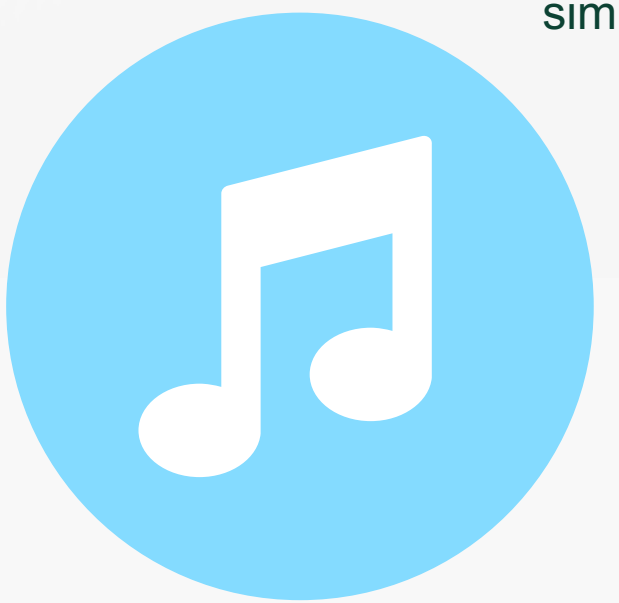


Lack of organizational structure
for one of the microservices

Microservices is now a strategy
at management level

copenhagen software

Content delivery

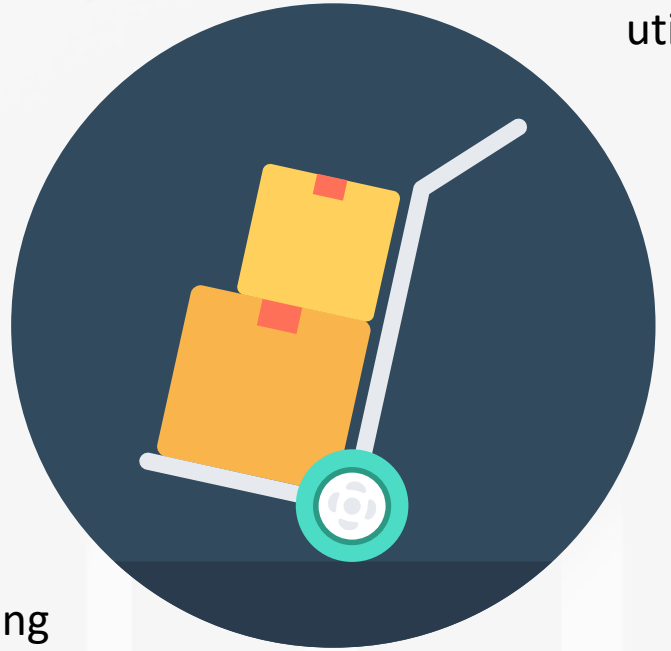


- Micro services for music streaming
- REST APIs in ASP.NET, C#
- 100.000+ simultaneous users
- 10 services
- Continuous delivery
- 6 developers
- 10.000 unit tests
- No testers

Parallel distribution

Events are driving integration work

Serverless is being utilized



Lack of monitoring is starting to hurt

Shim-services valuable for SaaS solutions

Microservice highlights

- Bounded by strategic business capability
- Focused development by cross-functional team
- Agile process with rapid feedback
- Quality ensured by continuous integration and automated testing
- Fast deployment with continuous delivery and infrastructure-as-code
- Operation using devops mindset
- Use suiting technology in every case
- Elastic scalability using cloud platforms

Want to know more?

- https://copenhagensoftware.com/hvad-er-microservices/
- @CPHSoft and @StigIP on Twitter