

# Microservices

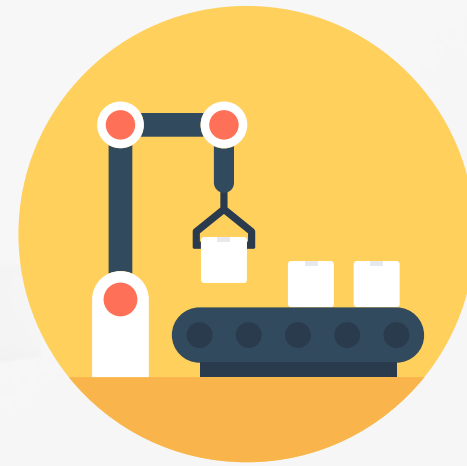
## – experiences from the front line

Stig Irming-Pedersen

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

# 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

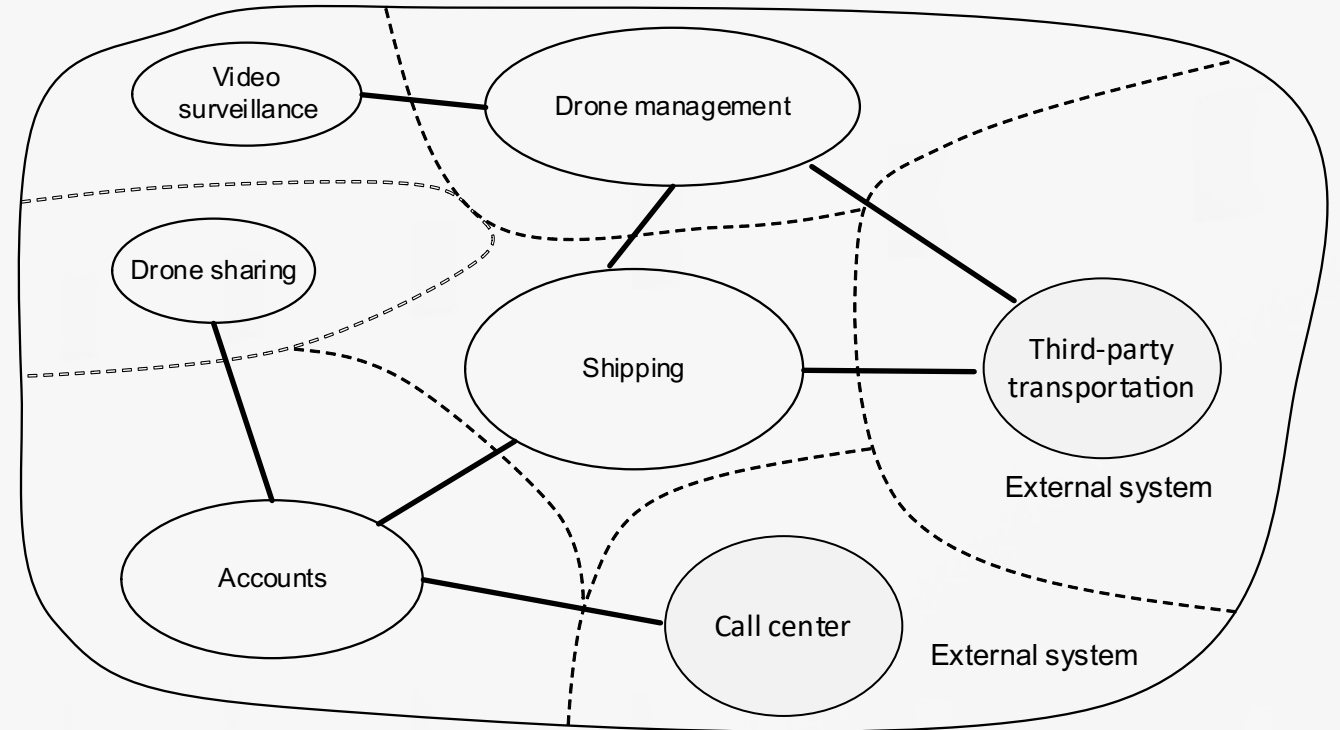


- Split software into smaller chunks
- ... and leverage great principles for software development:
  - Agile methodologies
  - Domain driven design
  - Clean code
  - Test driven development
  - Cross-functional teams
  - Continuous integration
  - Continuous delivery
  - DevOps mindset
  - Cloud native

# 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

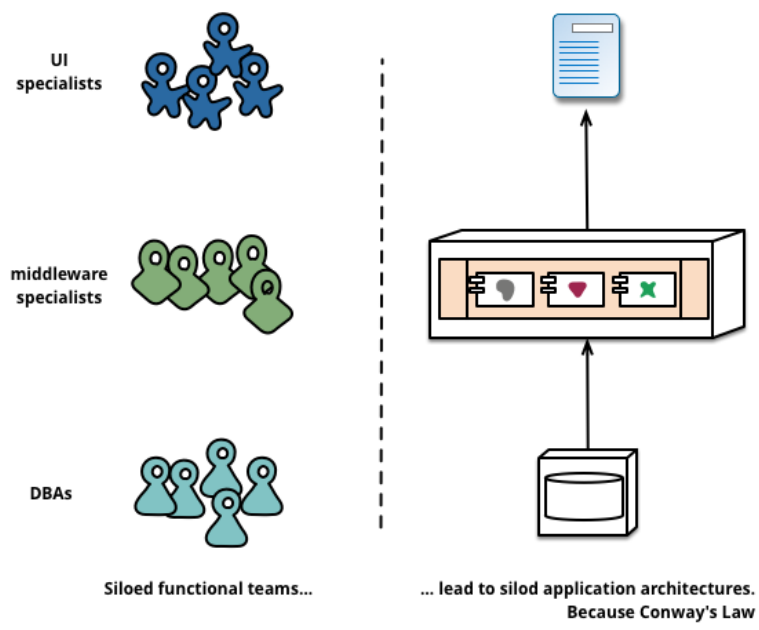
# Define boundaries



- 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

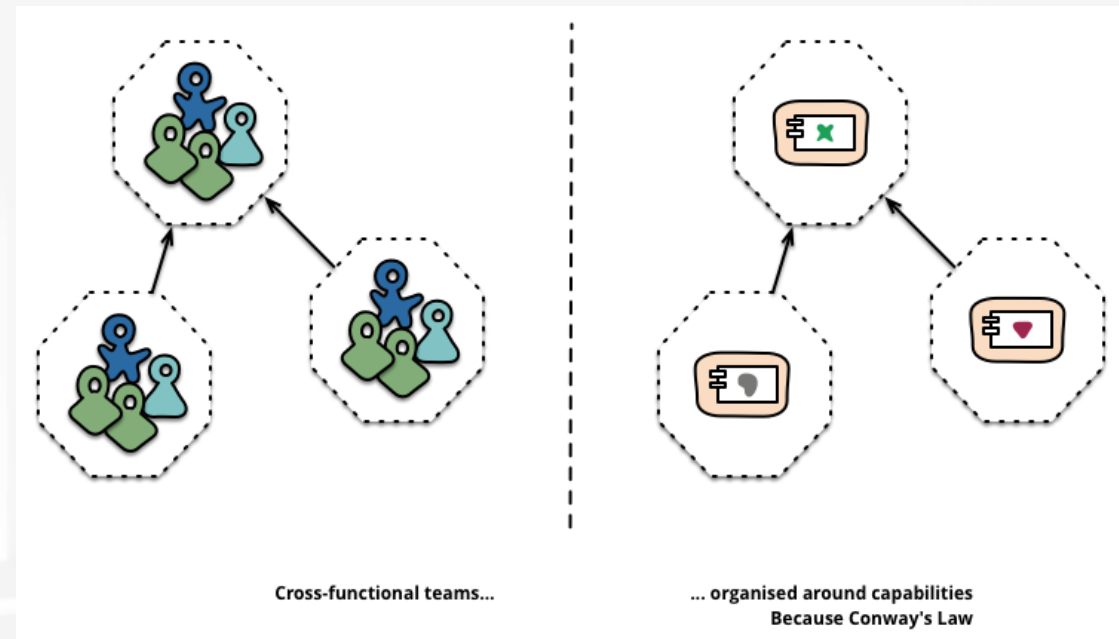


# Focused around organization



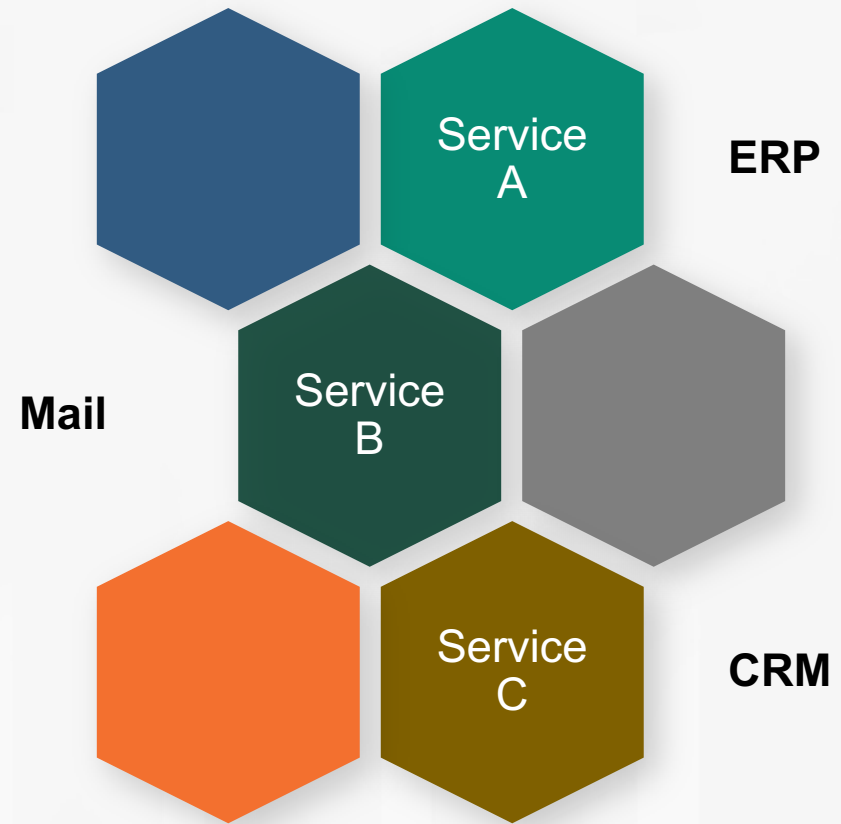
Conway's law:

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

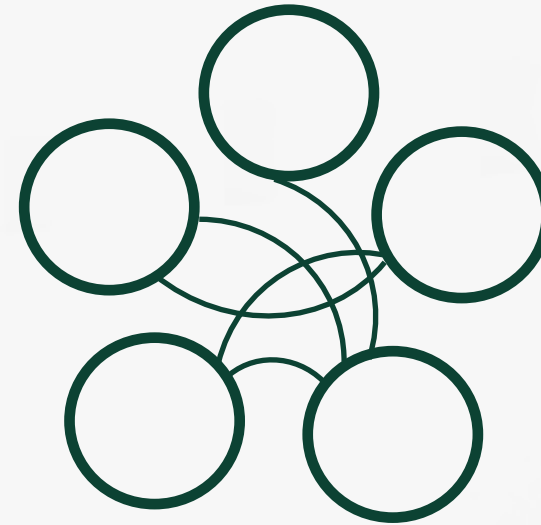


<https://martinfowler.com/articles/microservices.html>

# Service landscape

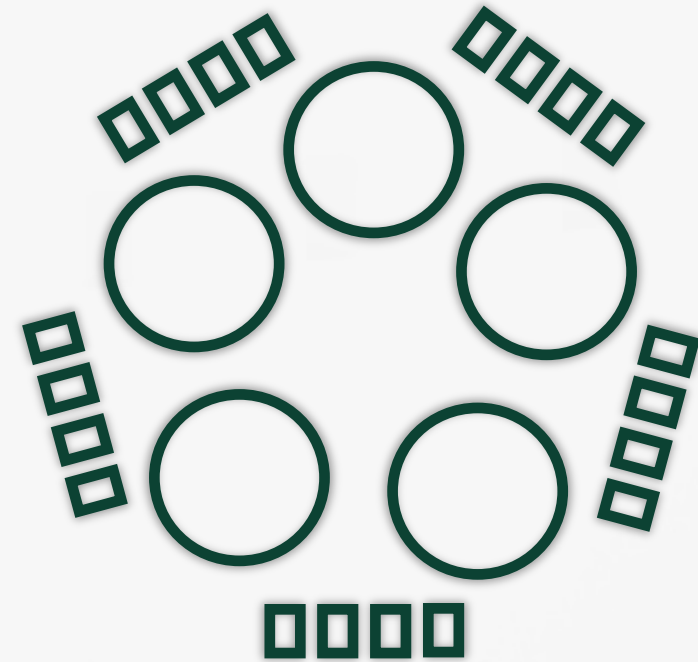


# APIs



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

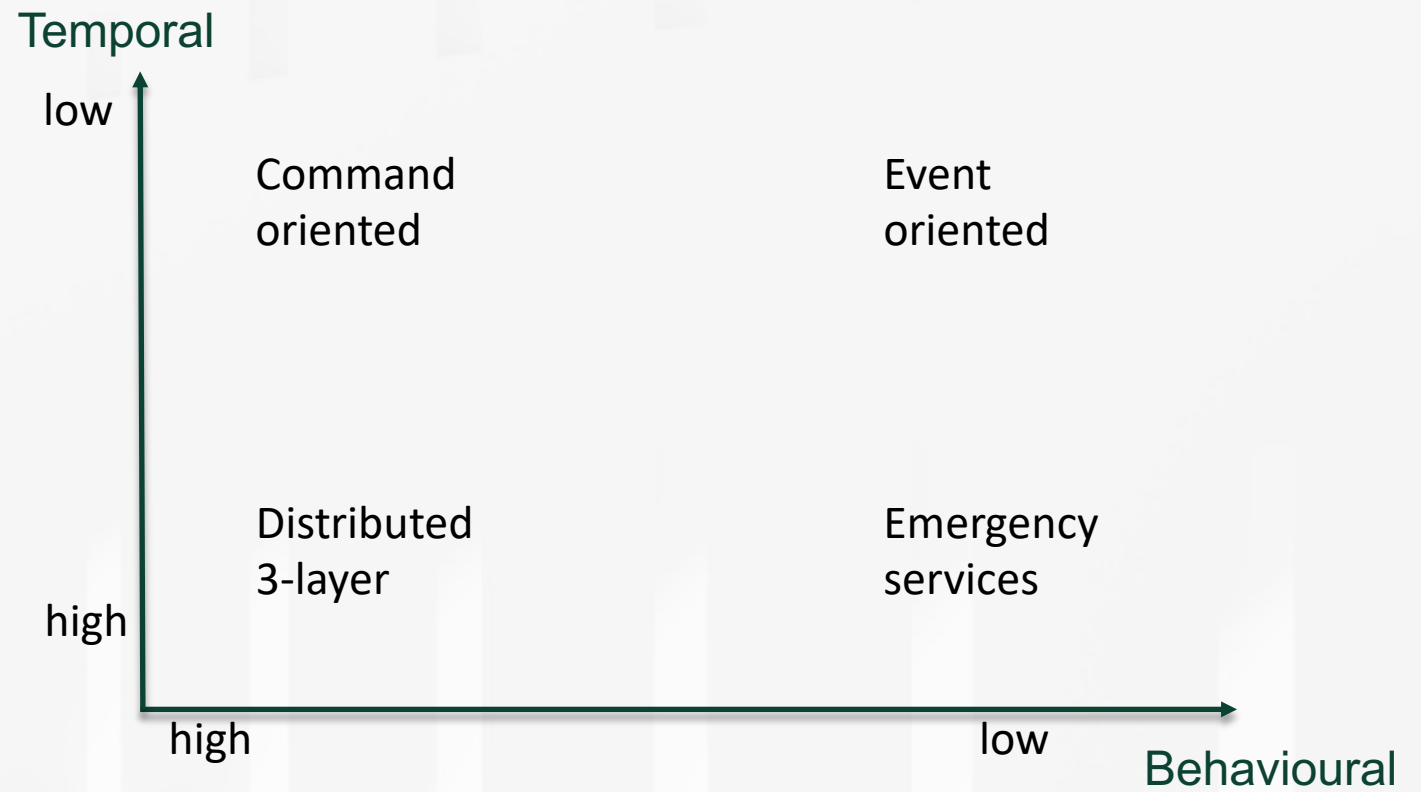
# Messaging



Communication patterns:

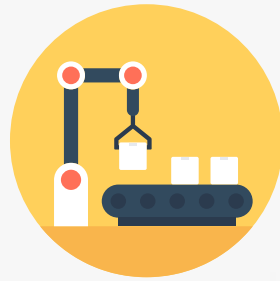
- Orchestrated (commands)
- Choreographed (events)

# Coupling



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

# How we chose com- munication



Advocate services => REST

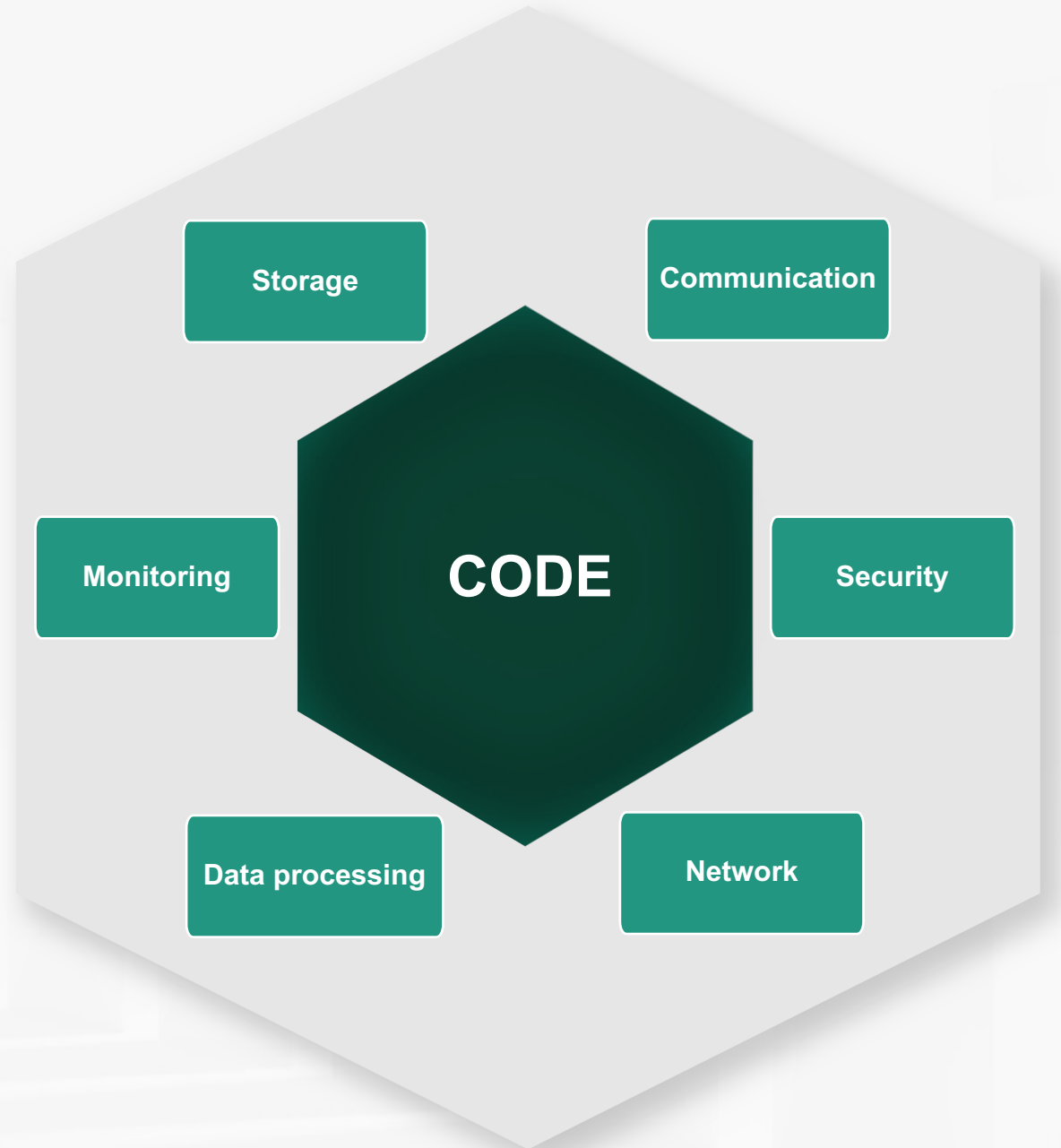


Many external clients => REST



Integrating internal systems => Eventing

# Components and infrastructure



# Cloud components

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



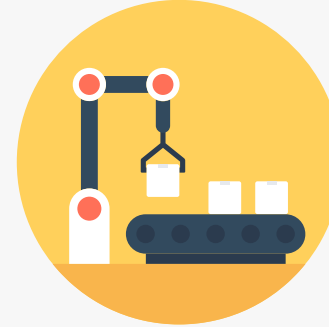


# Azure advantages



- **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



54 områder global 140 tilgængelig i 140 lande



\* To Azure Government Secret-områdeplaceringer er blevet offentliggjort

# Splitting the monolith



PRESENTATION



APPLICATION



DATA

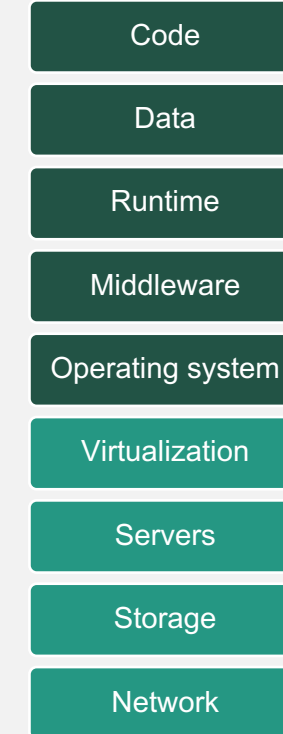
# Hosting levels



## On-premise



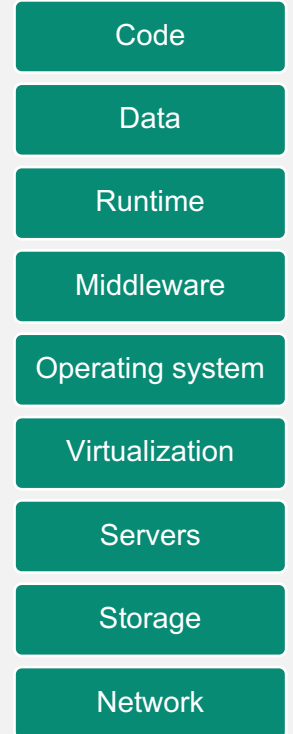
## Infrastructure as a Service (IaaS)



## Platform as a Service (PaaS)



## Software as a Service (SaaS)

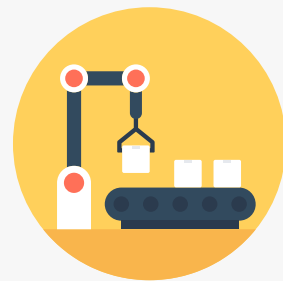


# Serverless

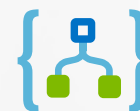


- **Code is run automatically**  
without explicit decision for capacity
- **Use back-end services**  
without knowledge of servers
- **Pay only for actual usage**  
not for reserved resources

# Chosen infrastructure and main components



On-premise



# Hexagonal architecture

User  
inter-  
face

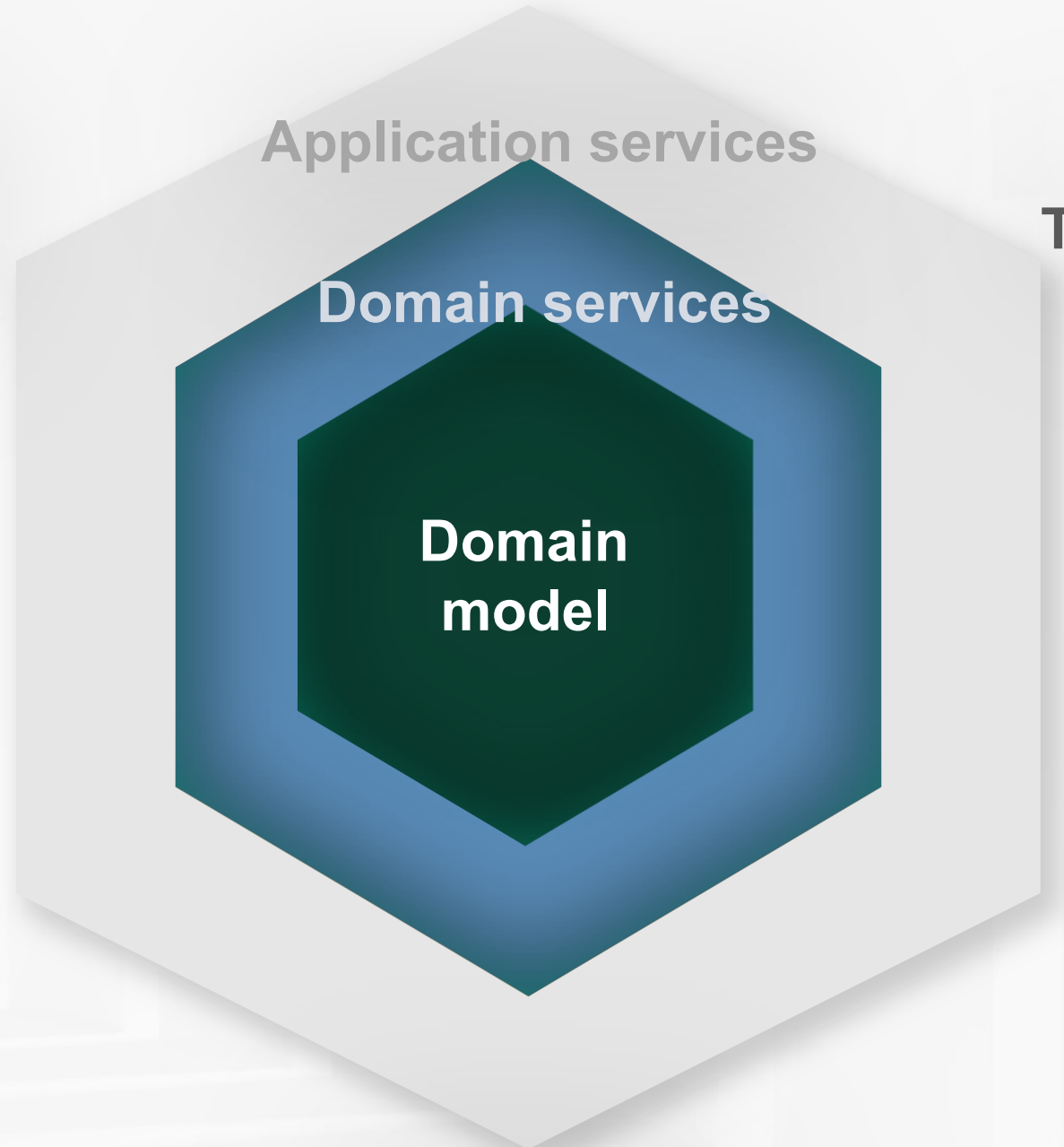
Application services

Tests

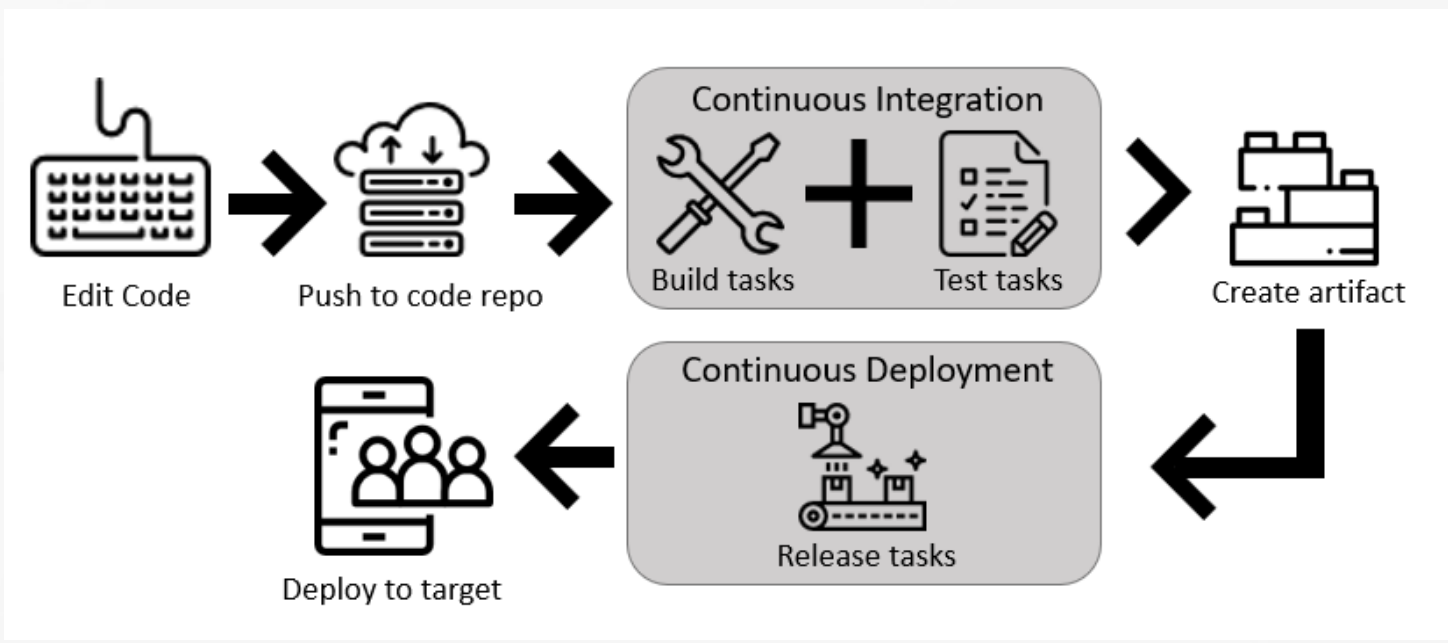
Domain services

Domain  
model

Infrastructure



# CI/CD



All pipelines > **Hello world** Save + Release ...

Pipeline Tasks Variables Retention Options History

Artifacts | + Add

Stages | + Add

\_FabrikamFiber Web-CI

QA 1 job, 1 task

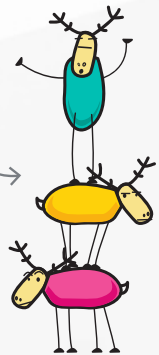
Production 1 job, 1 task

Schedule not set



# Monitoring

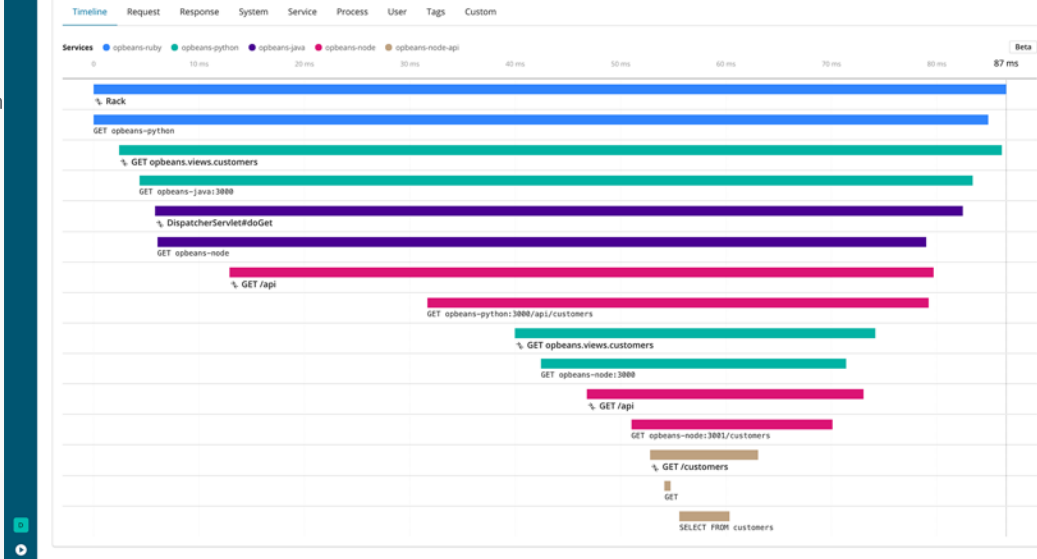
ELK Stack!  
Get it? →



**E** Elasticsearch

**L** Logstash

**K** Kibana



Home > acmefrontend > PolyglotCrossCorrelationDemo > acmepaymentprocessing - Application map > Failures > End-to-end transaction details

### End-to-end transaction details

Search results

Filtered on (timestamp > 8/21/2018, 4:4...  
timestamp < 8/22/2018, 4:4...  
Operation name == ProcessPayme...  
client\_Type != Browser Success == false

Suggested

8/22/2018, 7:13:03 AM  
ProcessPaymentFromQueue  
Duration: 5.2 s Response code: 500

Sort by  
Relevance

8/22/2018, 7:13:03 AM  
ProcessPaymentFromQueue  
Duration: 5.2 s Response code: 500

8/22/2018, 12:23:02 PM  
ProcessPaymentFromQueue  
Duration: 4.4 s Response code: 500

8/21/2018, 9:37:31 PM  
ProcessPaymentFromQueue  
Duration: 4.3 s Response code: 500

8/21/2018, 8:49:31 PM  
ProcessPaymentFromQueue  
Duration: 4.7 s Response code: 500

8/21/2018, 8:03:01 PM  
ProcessPaymentFromQueue  
Duration: 3.5 s Response code: 500

8/22/2018, 4:17:04 AM  
ProcessPaymentFromQueue  
Duration: 4.4 s Response code: 500

8/22/2018, 1:14:31 AM  
ProcessPaymentFromQueue

End-to-end transaction  
Operation ID: c0c3e9d5-4b9dadd3-4be82675

EVENT	RES.	DURATION	0	1	2	3	4	5
acmefrontend GET BuyWidget/GetAsync	200	596.7 ms						
acmeauth GET /login	200	444 ms						
acmeauth:8080 GET LoginAuthenticationController/authenticateUser	200	4 ms						
acmesqldb SQL: jdbc:sqlserver://acmesqldb.database.windows.net	1 ms							
acmesqldb SQL: jdbc:sqlserver://acmesqldb.database.windows.net	1 ms							
acmeinventory GET /getItemAvailability	200	28 ms						
acmeinventory GET /getItemAvailability	200	13 ms						
inventorycache get	0	1 ms						
inventorystore Azure table: inventorystore/ItemAvailability	200	7 ms						
inventorycache set	0	0						
acmeinventory POST /reserveItem	200	27 ms						
acmeinventory POST /reserveItem	200	13 ms						
inventorystore Azure table: inventorystore/ItemAvailability	204	10 ms						
inventorycache set	0	1 ms						
AZURE EVENT HUBS purchasingqueue.servicebus.windows.net Send	94 ms							
Payment Processing Worker ProcessPaymentFromQueue	500	4.4 s						
acmeinventory POST /getItemAvailability/nocache	200	21 ms						
acmeinventory POST /getItemAvailability/nocache	200	13 ms						
inventorystore Azure table: inventorystore/ItemAvailability	200	11 ms						
fabrikampayments.com POST /api/charge	429	1.8 s						

30 All 7 Traces 0 Events

View all telemetry

Event time: 8/22/2018, 12:23:02 PM

Request name: ProcessPaymentFromQueue

Response code: 500

Successful request: False

Response time: 4.4 s

Call Stack

Payment\_Processing\_worker.TransactionFailedException:  
at Payment\_Processing\_worker.WorkerRole<<DequeueAsync>>d\_\_9.MoveNext() (f

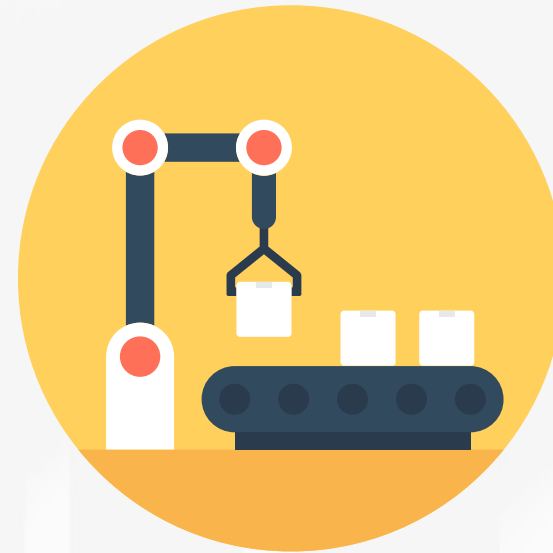
Related Items

- Show what happened before and after this request in User Flows
- Show trend of this request over time
- Show trend of this request with this response code over time
- All available telemetry 5 minutes before and after this event

# Device production

ERP transition halted,  
but microservices lives on

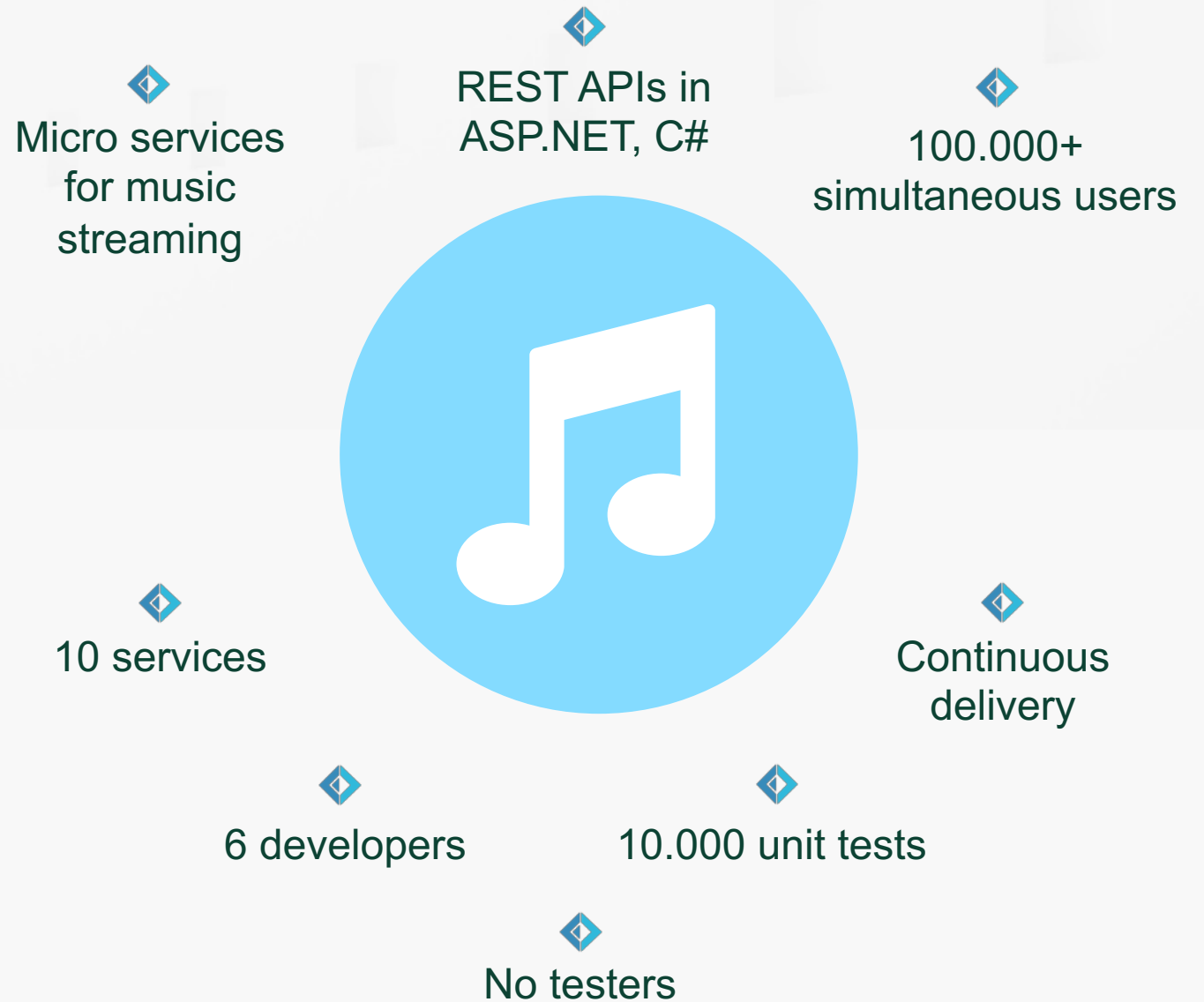
Enabling new technologies  
has proved it's worth



Lack of organizational structure  
for one of the microservices

Microservices is now a strategy  
at management level

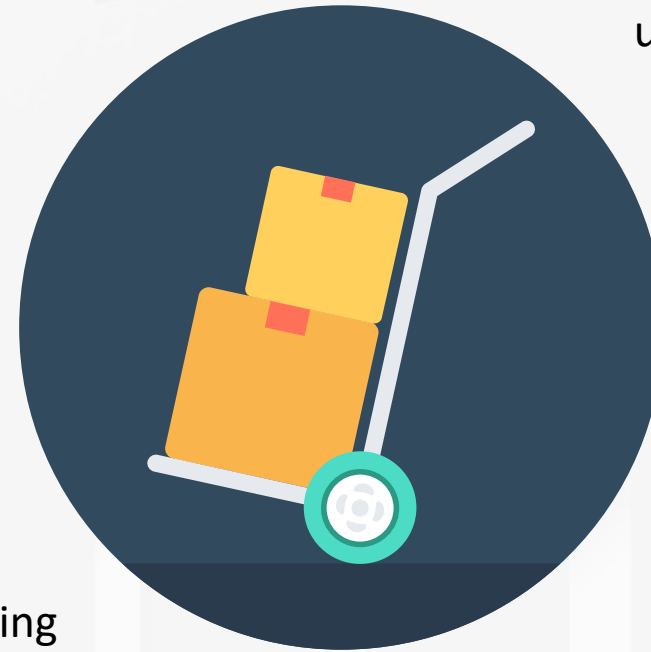
# Content delivery



# 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