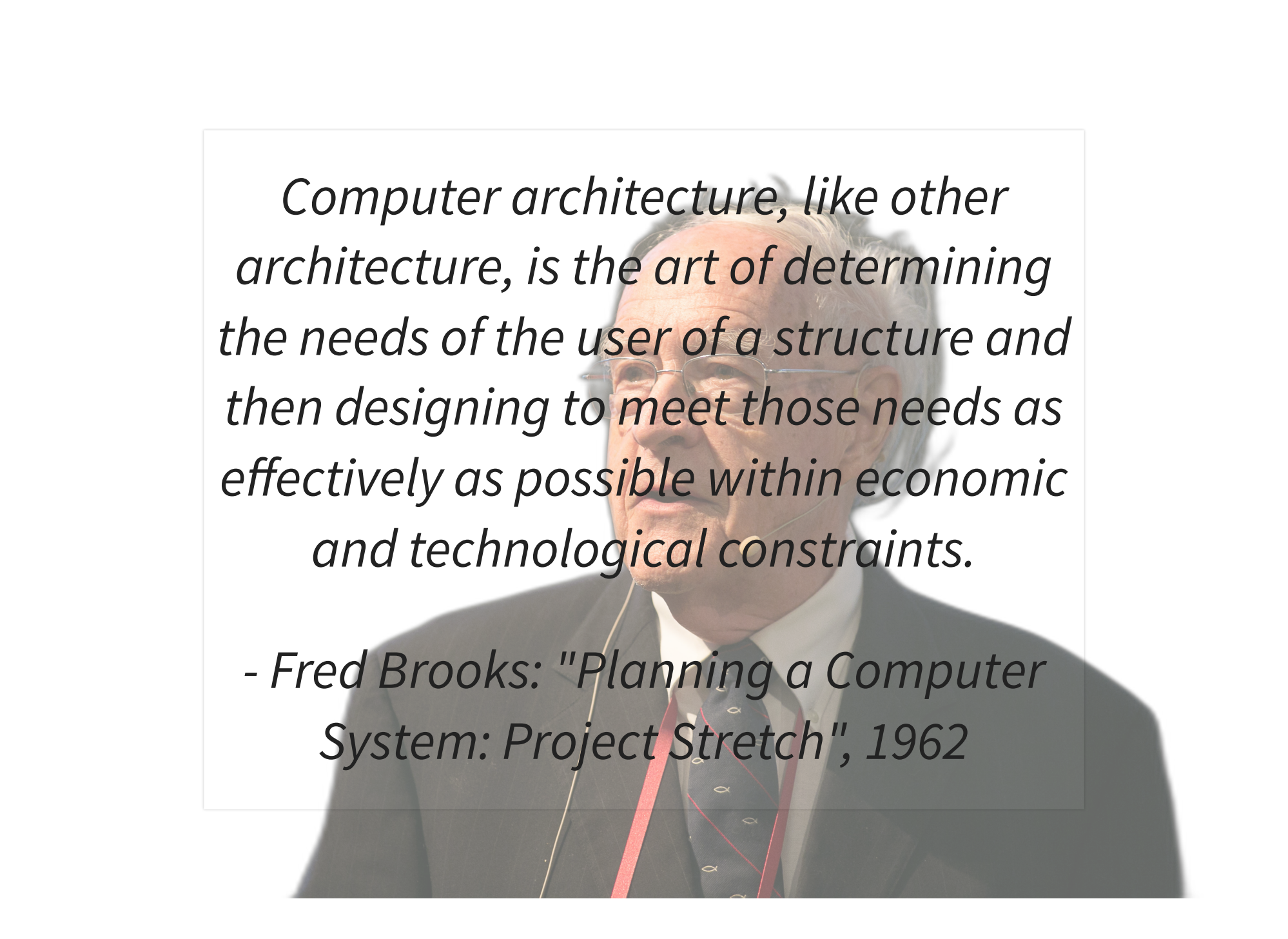
MICROSERVICES ARCHITEKTUR

(ANTI-)PATTERNS UND TECHNIKEN

ÜBER MICH

- Finn Rayk Gärtner
- Student an der Universität Bonn
- Fokus verteilte Systeme und Softwarearchitektur

SOFTWAREARCHITEKTUR



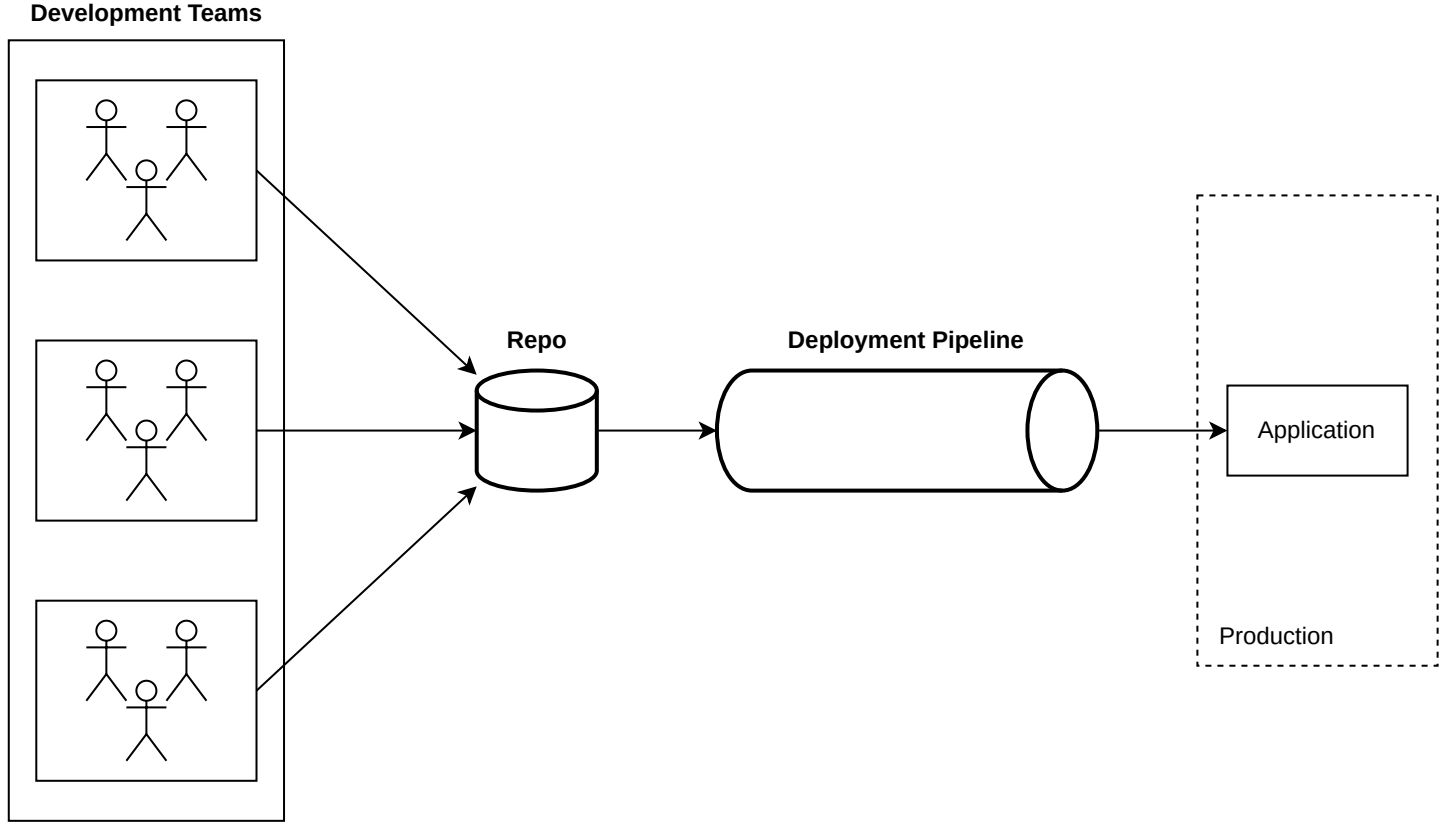
Computer architecture, like other architecture, is the art of determining the needs of the user of a structure and then designing to meet those needs as effectively as possible within economic and technological constraints.

- Fred Brooks: "Planning a Computer System: Project Stretch", 1962

-ILITIES

- Maintainability
- Evolvability
- Testability
- Scalability
- Understandability
- ...

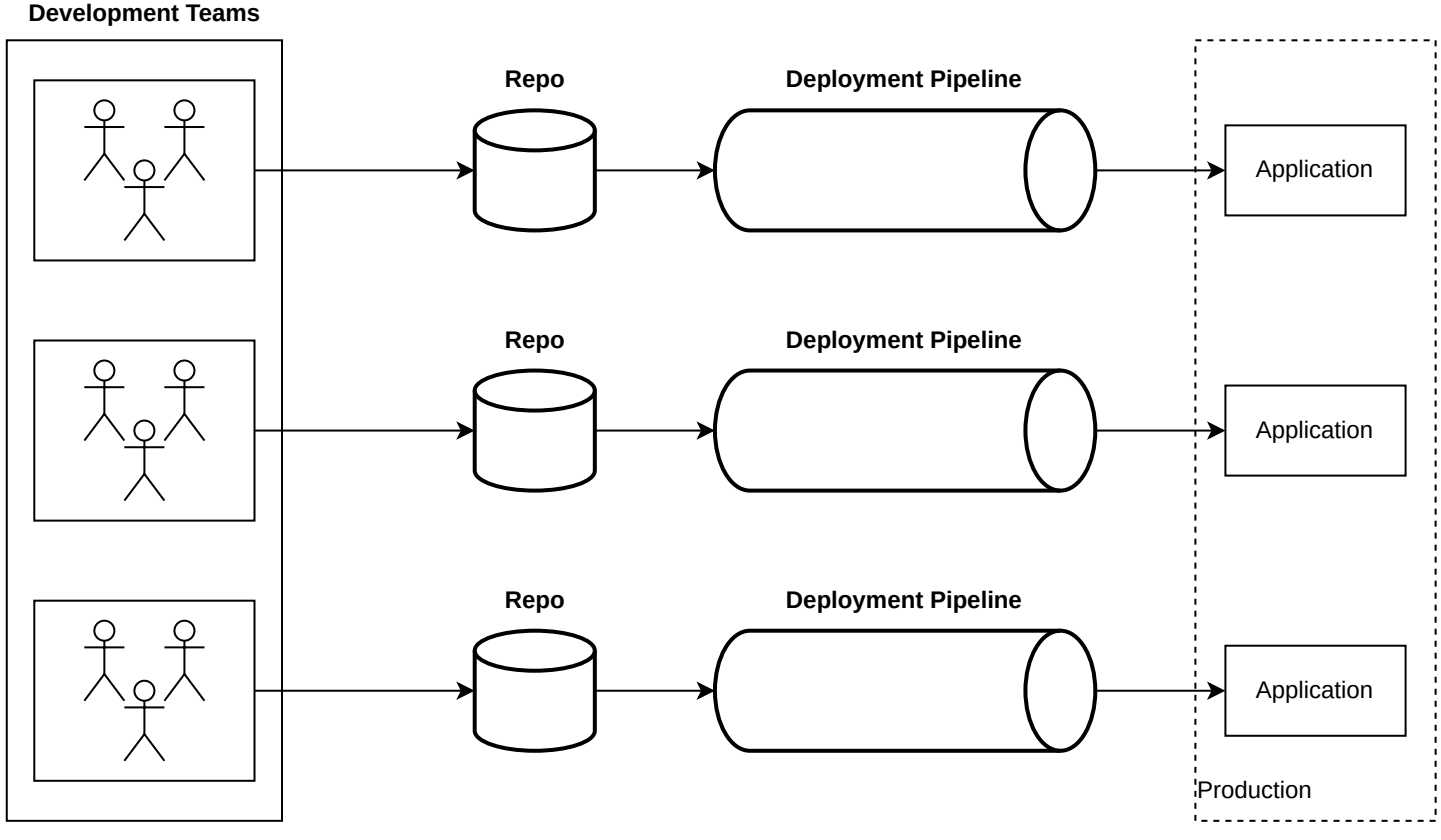
MONOLITH


Development Teams
Repo
Deployment Pipeline
Application
Production

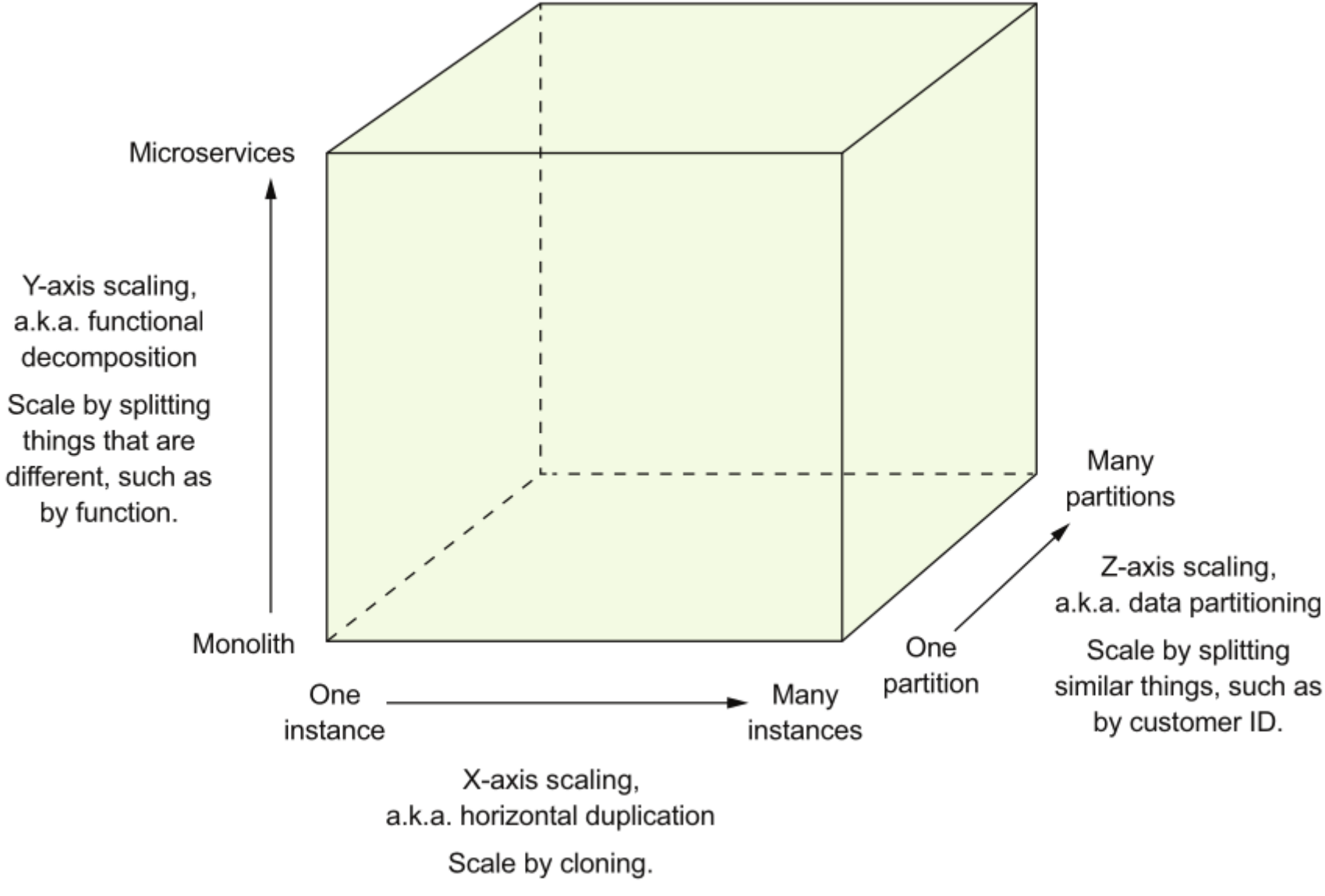
Organisationen, die Systeme entwerfen, […] sind gezwungen, Entwürfe zu erstellen, die die Kommunikationsstrukturen dieser Organisationen abbilden.

- Melvin E. Conway

MICROSERVICES



SKALIERUNG


Microservices
Y-axis scaling,
a.k.a. functional
decomposition
Scale by splitting
things that are
different, such as
by function.
Monolith
One
instance
Many
instances
X-axis scaling,
a.k.a. horizontal duplication
Scale by cloning.
One
partition
Many
partitions
Z-axis scaling,
a.k.a. data partitioning
Scale by splitting
similar things, such as
by customer ID.

FUNKTIONALE DEKOMPOSITION

Das Aufteilen eines Systems in Komponenten, die jeweils eine eigene Funktion besitzen.

MICROSERVICES - DEFINITION

Eine Architektur bestehend aus Komponenten, welche jeweils **unabhängig** getestet, deployed und verändert werden können und eine **eigene** und klar definierte Funktion besitzen.

LOOSE COUPLING

Design Time

- Teamübergreifende Änderungen
- Mehrfache Redeployments

Runtime

- Fehlerkaskaden
- Hohe Latenz

DESIGN TIME

- Gute Service Grenzen
- Gute APIs

API MANAGEMENT

- Versionierung
- Spezifikation (IDL)

```yaml
openapi: 3.0.0
info:
  title: Hello World API
  version: 1.0.0
servers:
  - url: https://example.com/api/v1
paths:
  /hello:
    get:
      summary: Returns a greeting message
      responses:
        '200':
          description: A greeting message
          content:
            text/plain:
```

```
syntax = "proto3";

package com.example.api.v1;

service HelloWorld {
  rpc GetGreeting (HelloRequest) returns (HelloRespons
}

message HelloRequest {}

message HelloResponse {
  string message = 1;
}
```

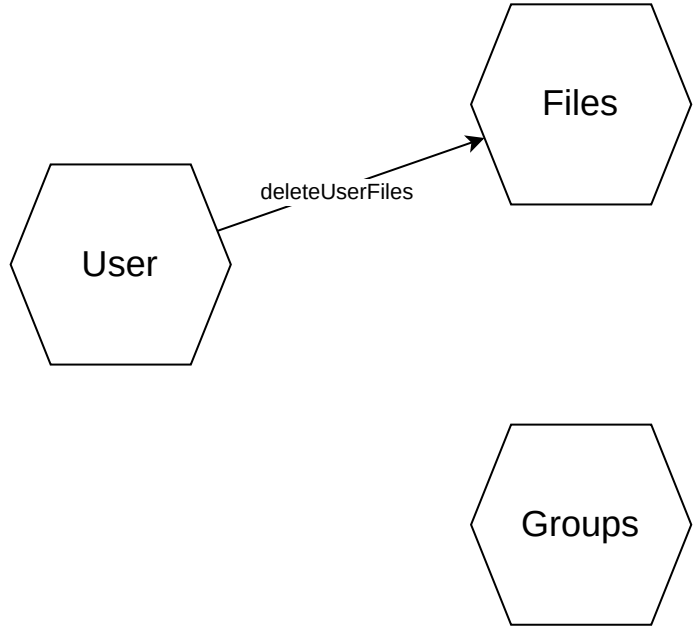
```
namespace java com.example.api.v1

service HelloWorld {
  string GetGreeting() throws (1:exception Error)
}

exception Error {
  1: string message
}
```

RUNTIME COUPLING

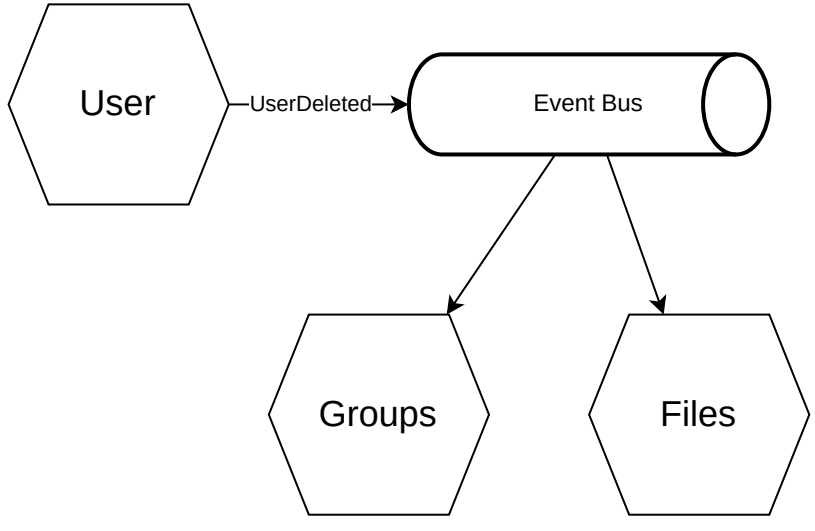
- Eine Datenbank pro Service
- Circuit Breaker
- Asynchrone Kommunikation

	1 zu 1	**1 zu Viele**
Synchron	Request/Response	--
Asynchron	One-way notifications	Pub/Sub

KLASSISCHE KOMMUNIKATION



EVENT-BASIERTE KOMMUNIKATION



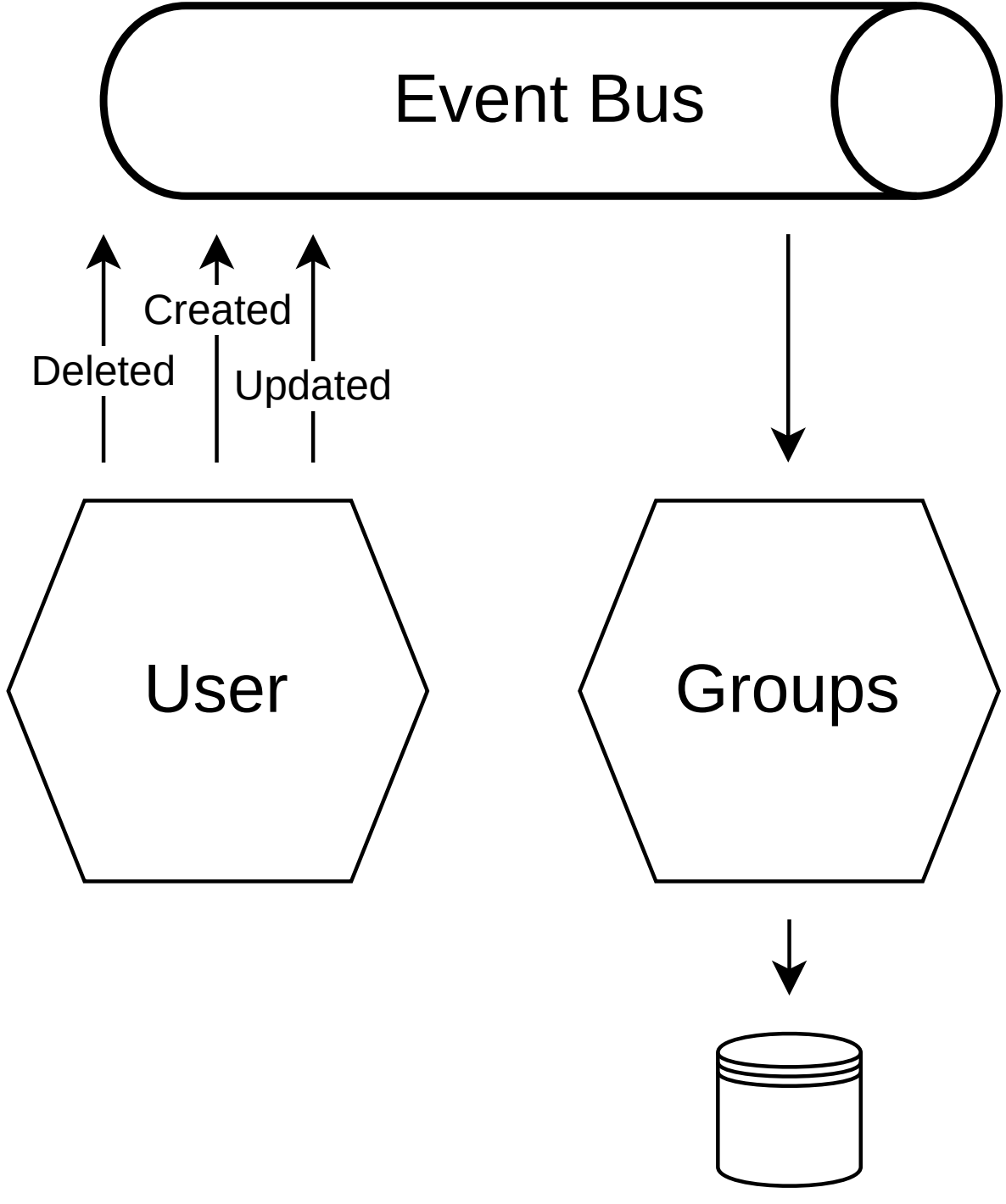
PROBLEME

- 2PC mit DB und Bus
- ACID vs BASE
- Querying über mehrere Services

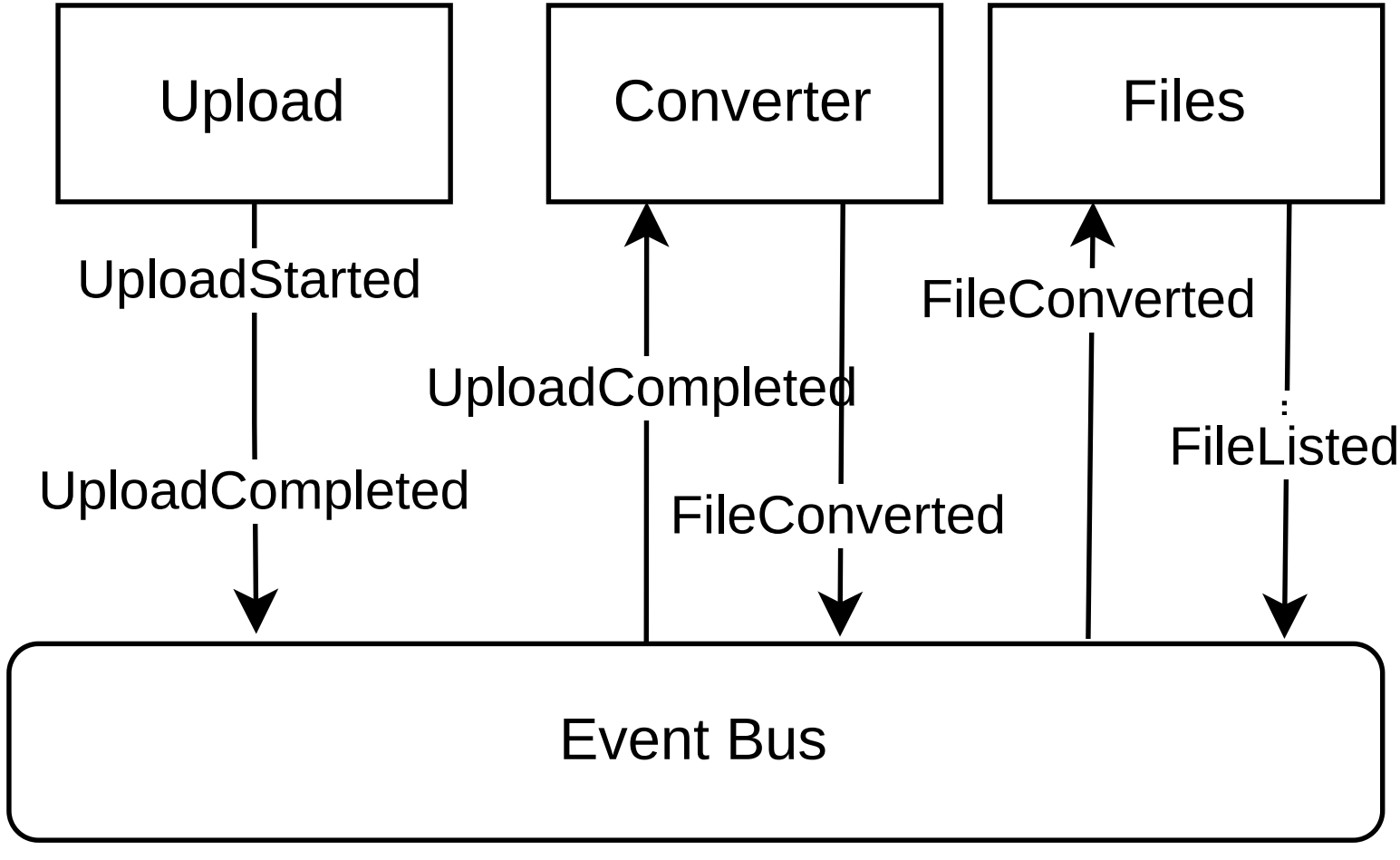
EVENTS

- Unveränderbar
- Einmalig
- CUD - Events
- Metadaten
- IDL

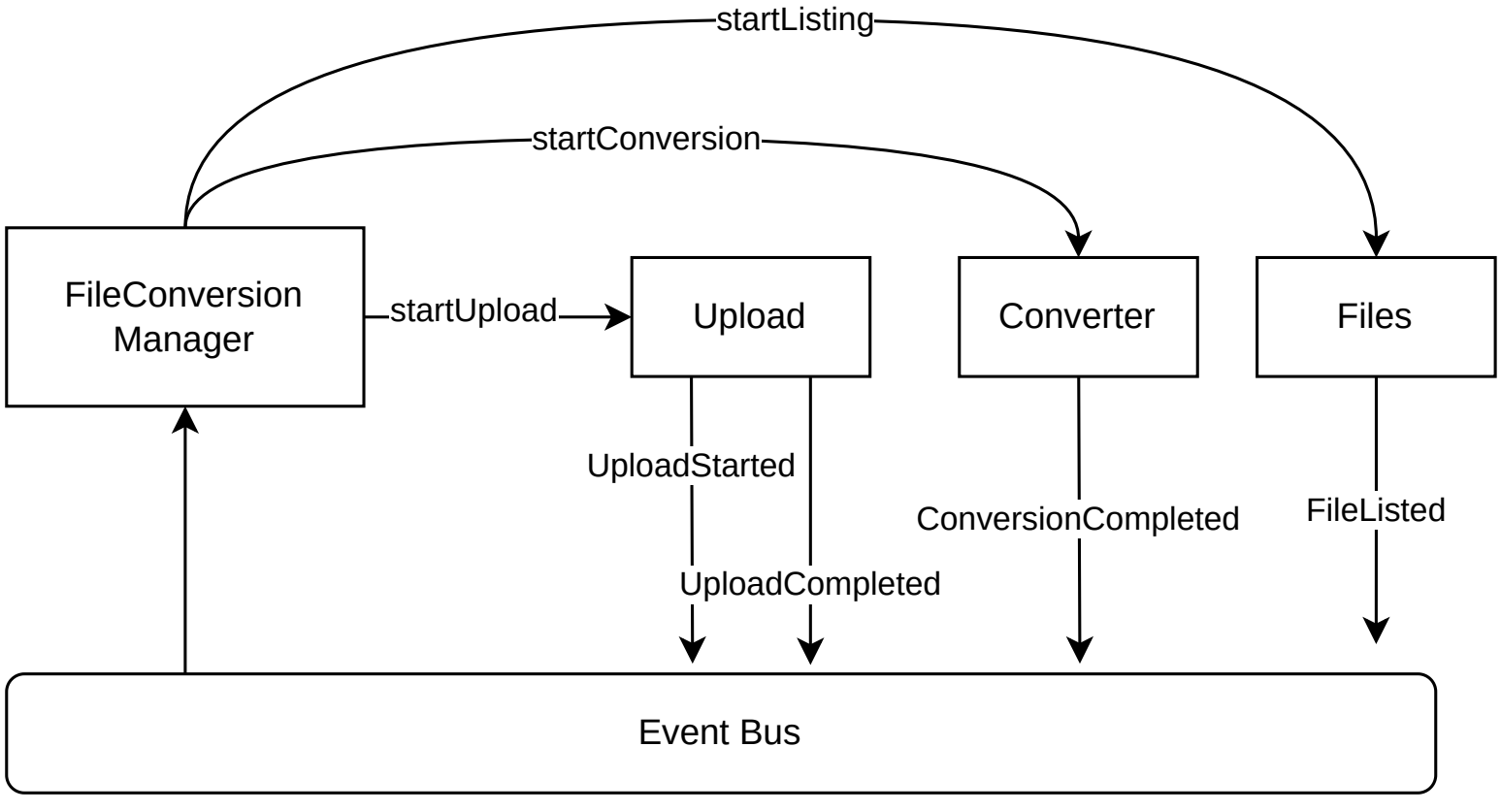
EVENT SOURCING


Event Bus
Deleted
Created
Updated
User
Groups

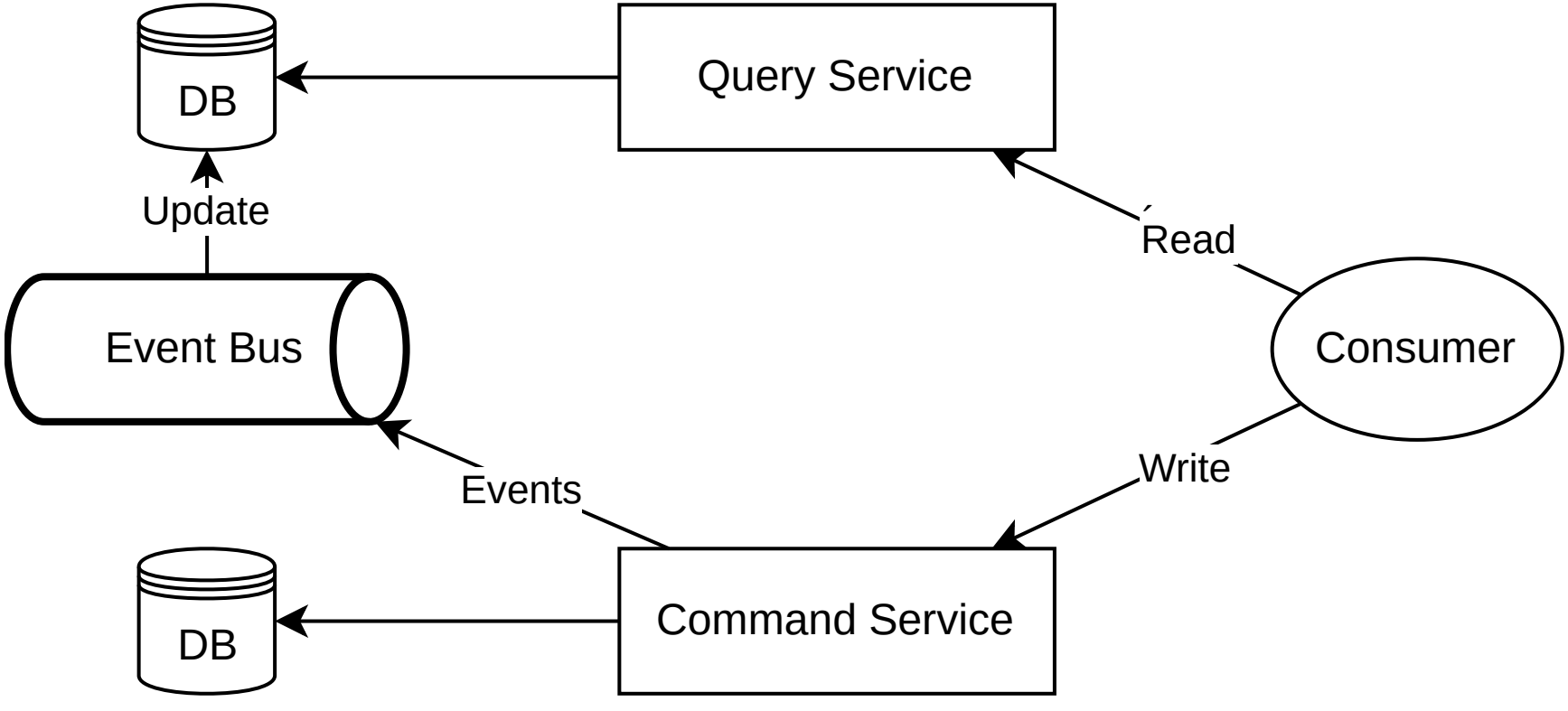
SAGA (CHOREOGRAPHIE)



SAGA (ORCHESTRIERT)



CQRS


DB
Query Service
Update
Read
Event Bus
Consumer
Events
Write
DB
Command Service

OBSERVABILITY

Logs

- Spezifisch für einen Service

Metrics

- Kennzahlen
- Alerts

Traces

- "verteilte Logs"
- Latenzen

VISUALISIERUNG

- Grafana (LGTM Stack)
- Kibana (ELK Stack)

WEITERE RESSOURCEN

- Microservices Patterns, Chris Richardson
- microservices.io
- Microservices for the Enterprise