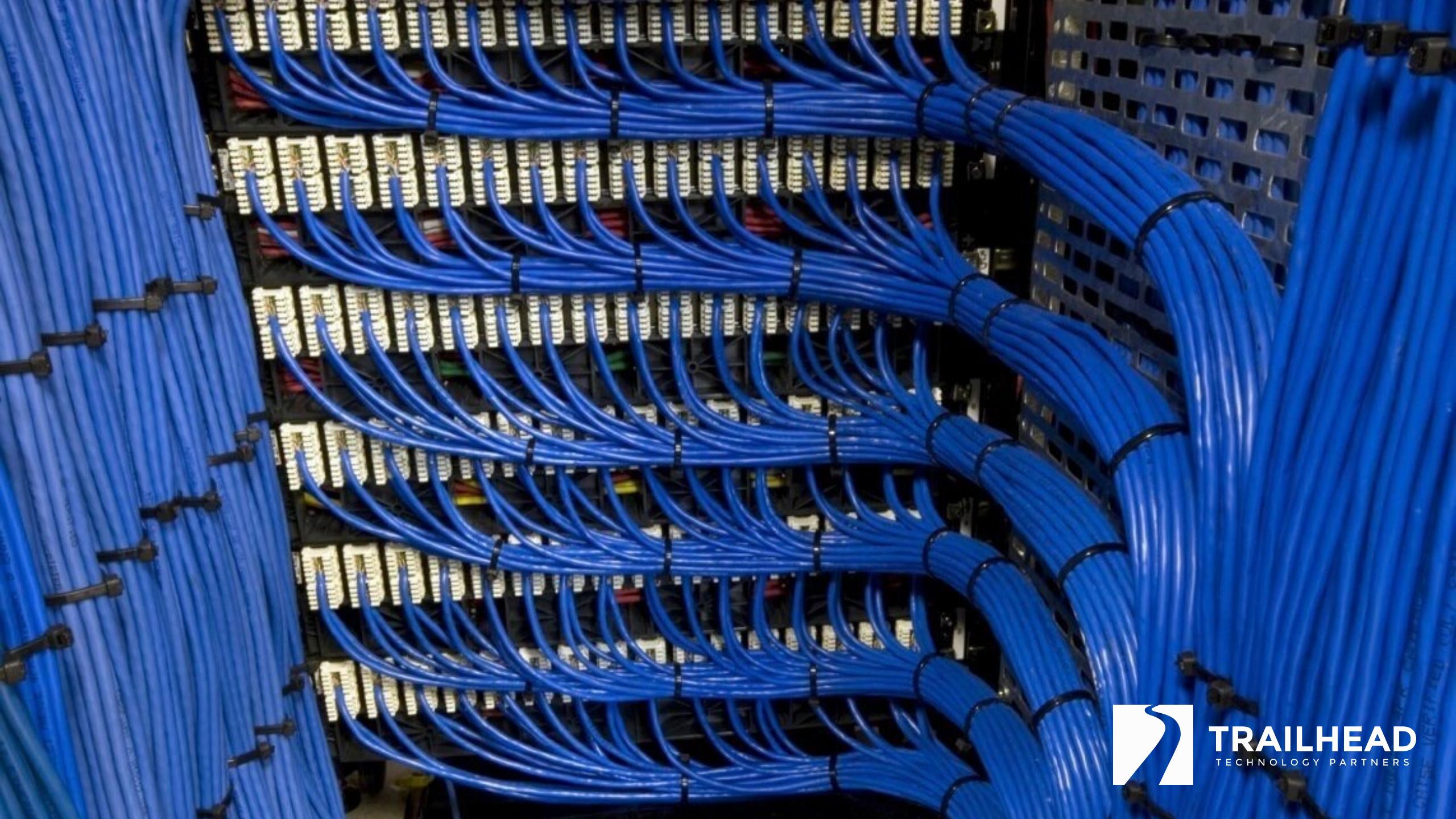




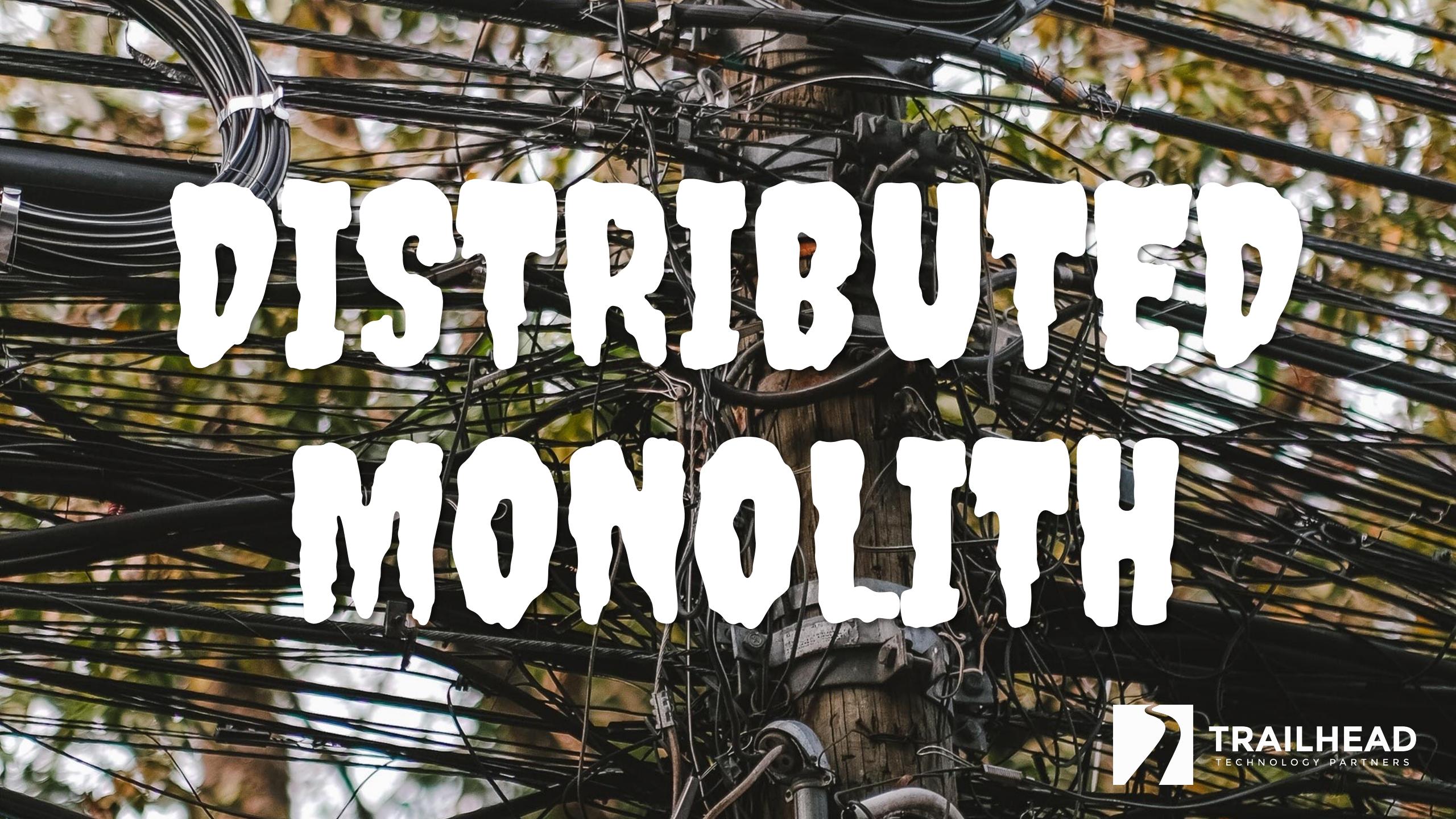
# Don't Build a Distributed Monolith

How to Avoid Doing Microservices Completely Wrong

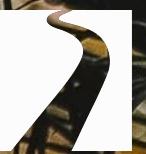
Jonathan "J." Tower



**TRAILHEAD**  
TECHNOLOGY PARTNERS



**DISTRIBUTED  
MONOLITH**



**TRAILHEAD**  
TECHNOLOGY PARTNERS

# Avoid the 10 Most Common Mistakes Made When Building Microservices

# Jonathan "J." Tower

Principal Consultant & Partner



**TRAILHEAD**  
TECHNOLOGY PARTNERS

- 🏆 Microsoft MVP in .NET
- ✉️ jtower@trailheadtechnology.com
- 🌐 trailheadtechnology.com/blog
- 🐦 jtowermi

**Free  
Consultation**



[bit.ly/th-offer](http://bit.ly/th-offer)

<https://github.com/jonathantower/distributed-monolith>



AUGUST 4 - 5, 2023 GRAND RAPIDS, MI



# BEER CITY CODE

A software conference for **software developers of all types**

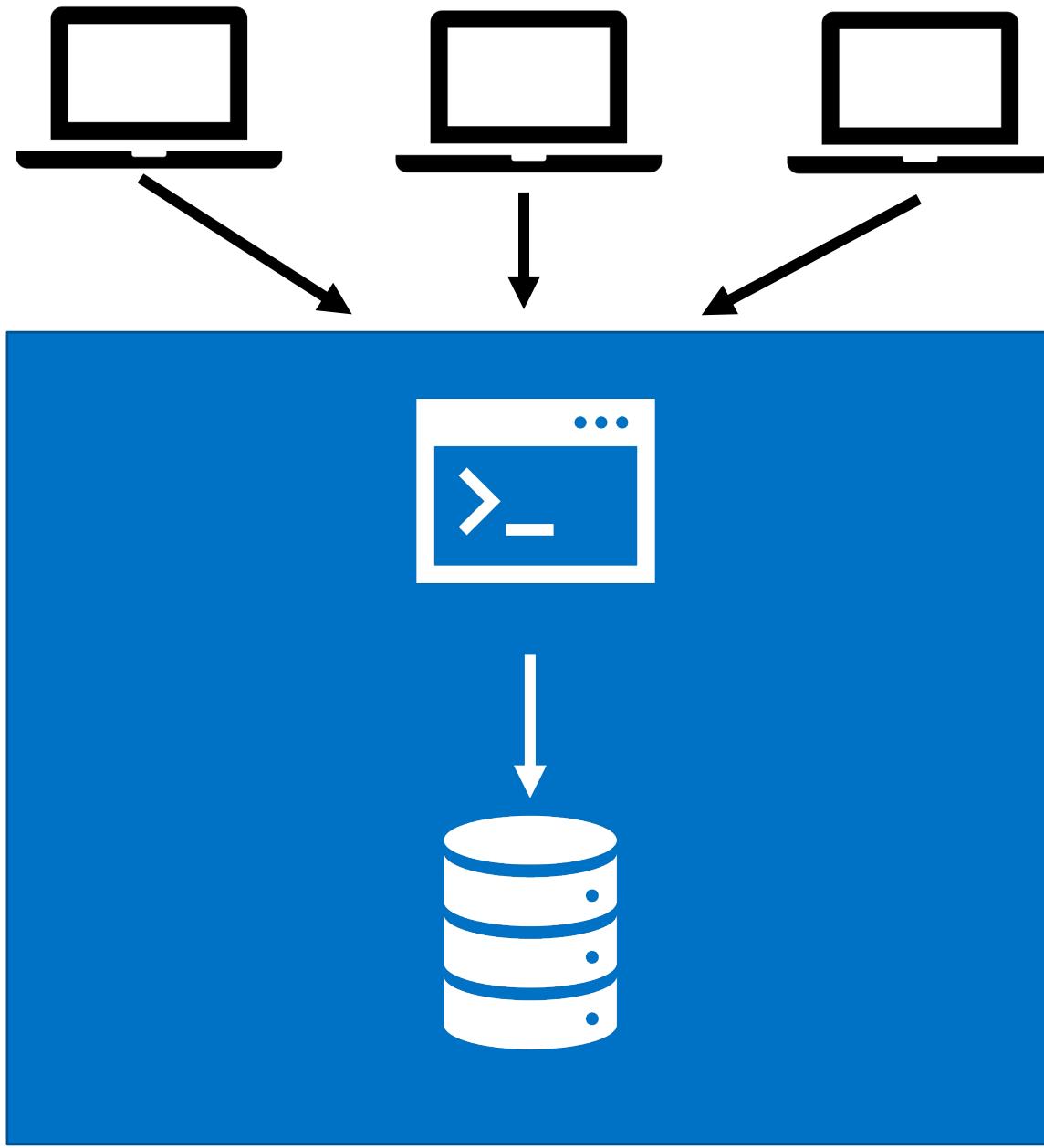
[beercitycode.com](http://beercitycode.com)

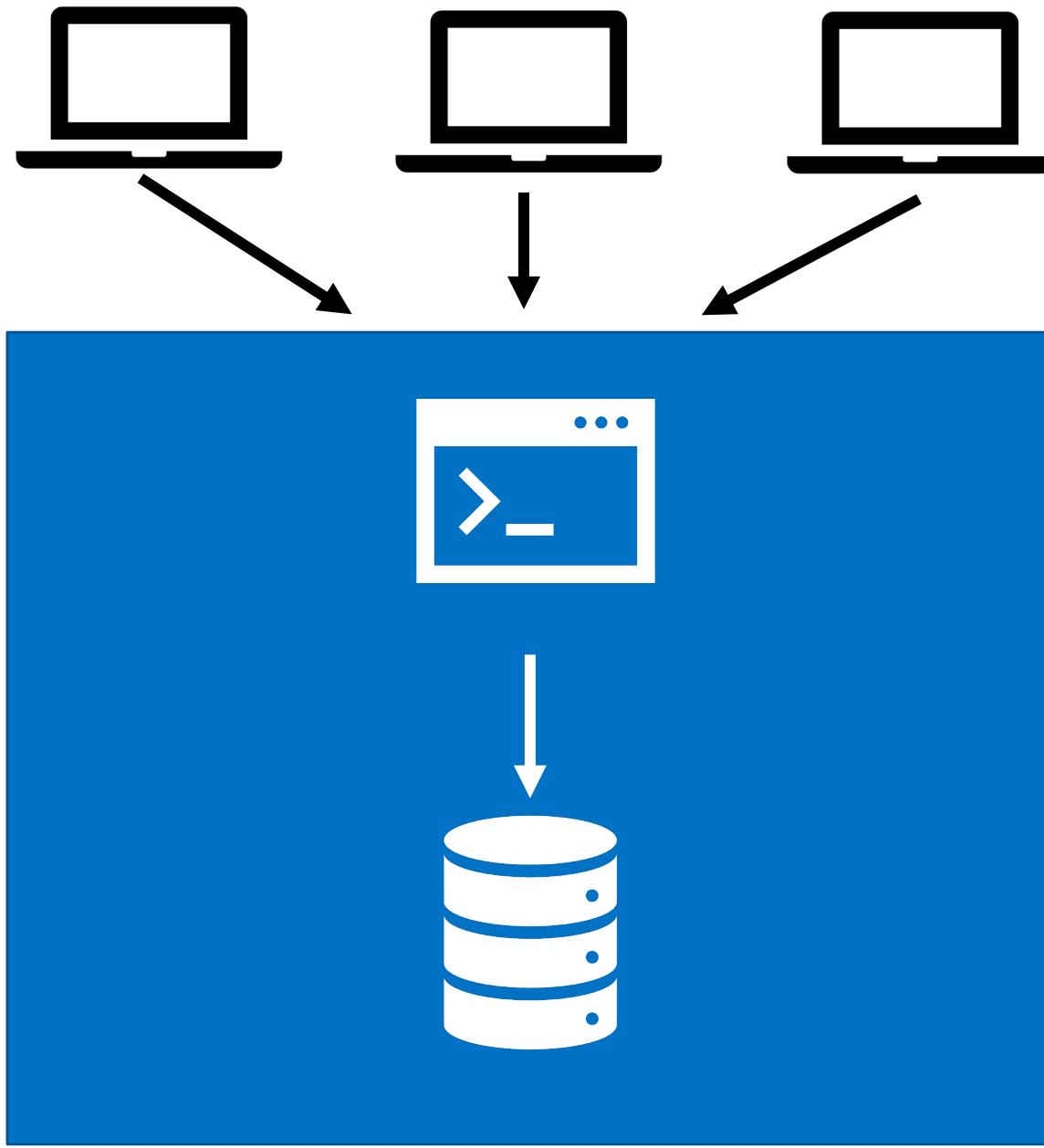
# Some Definitions

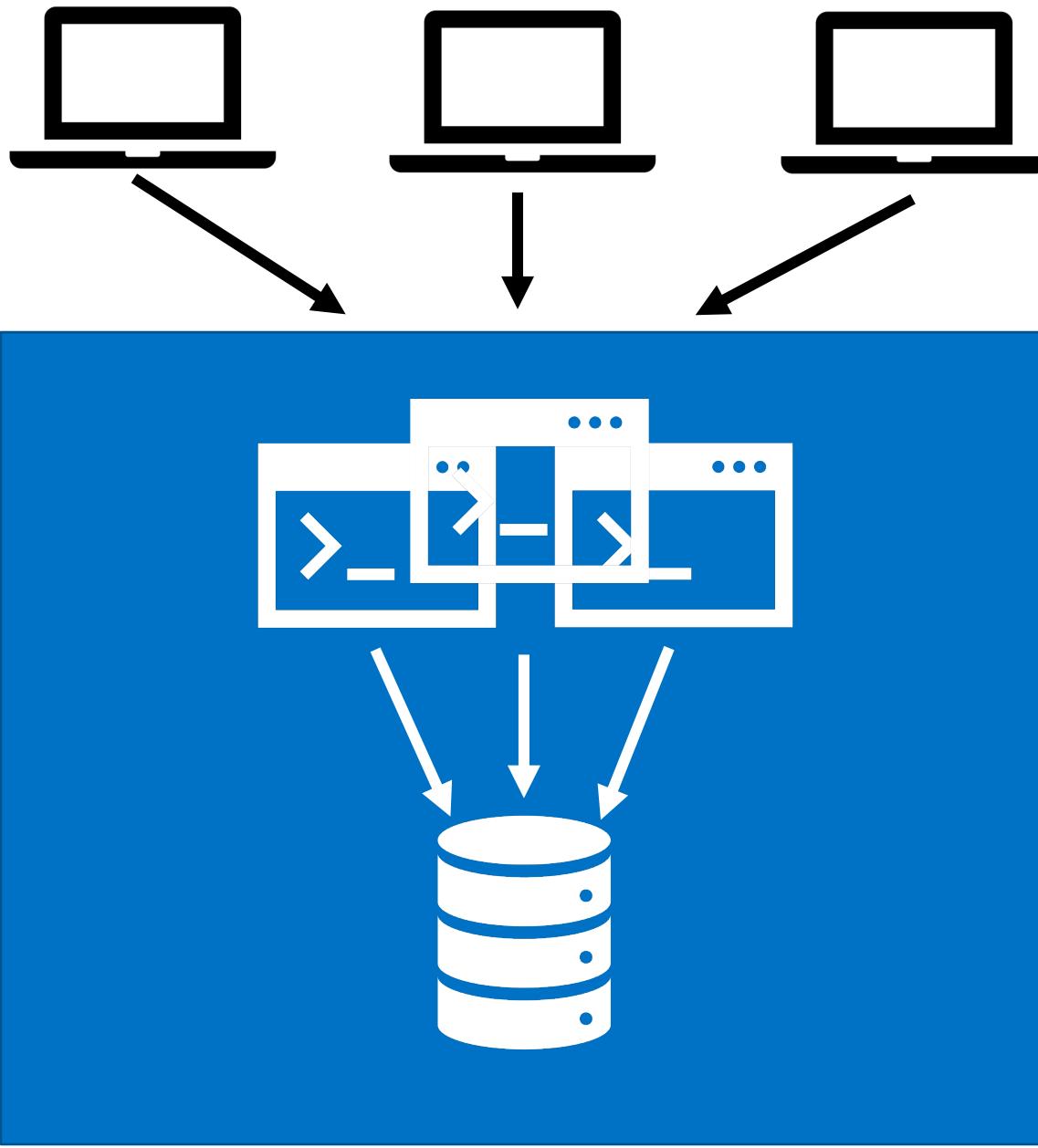
# Monolith

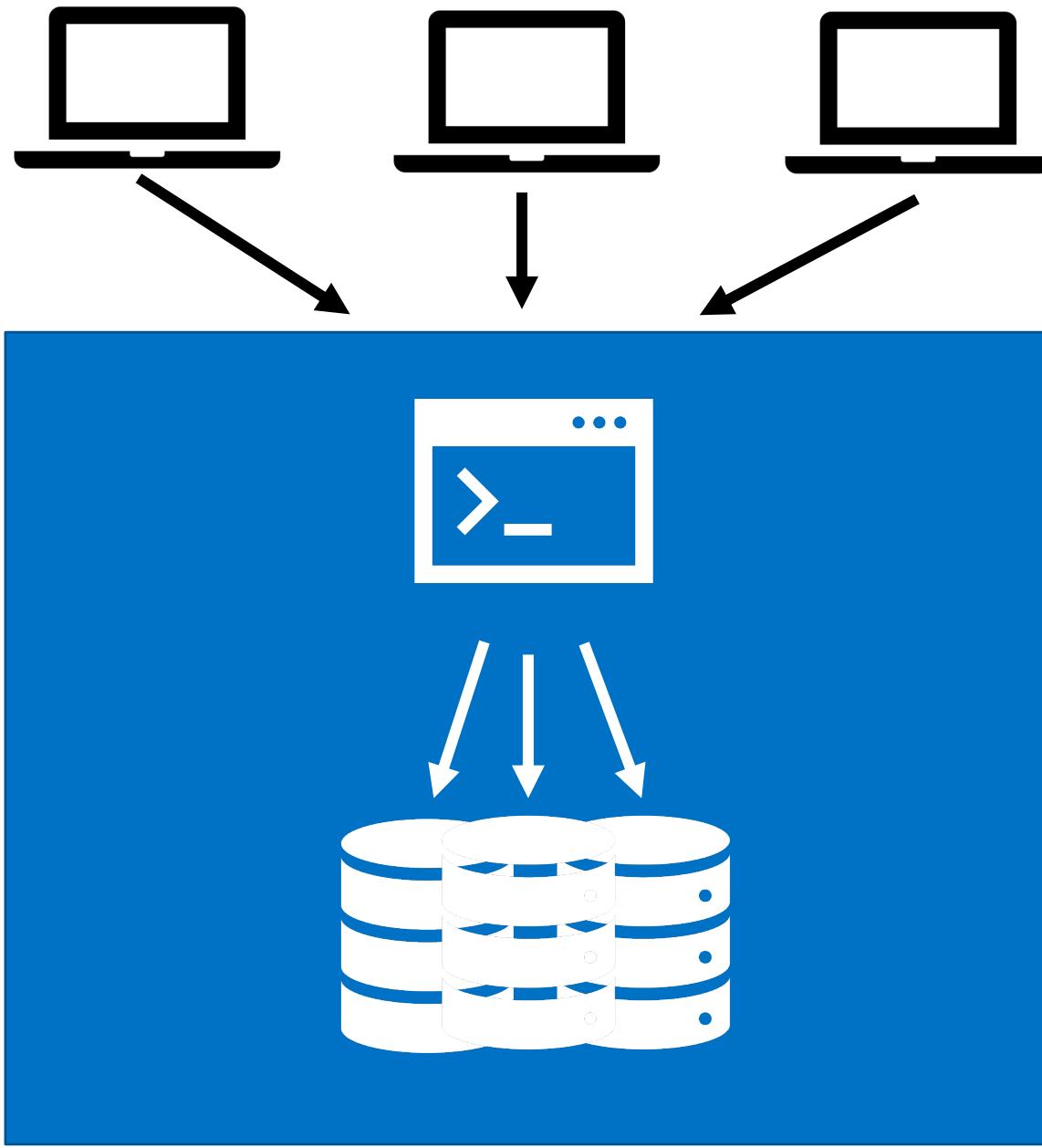


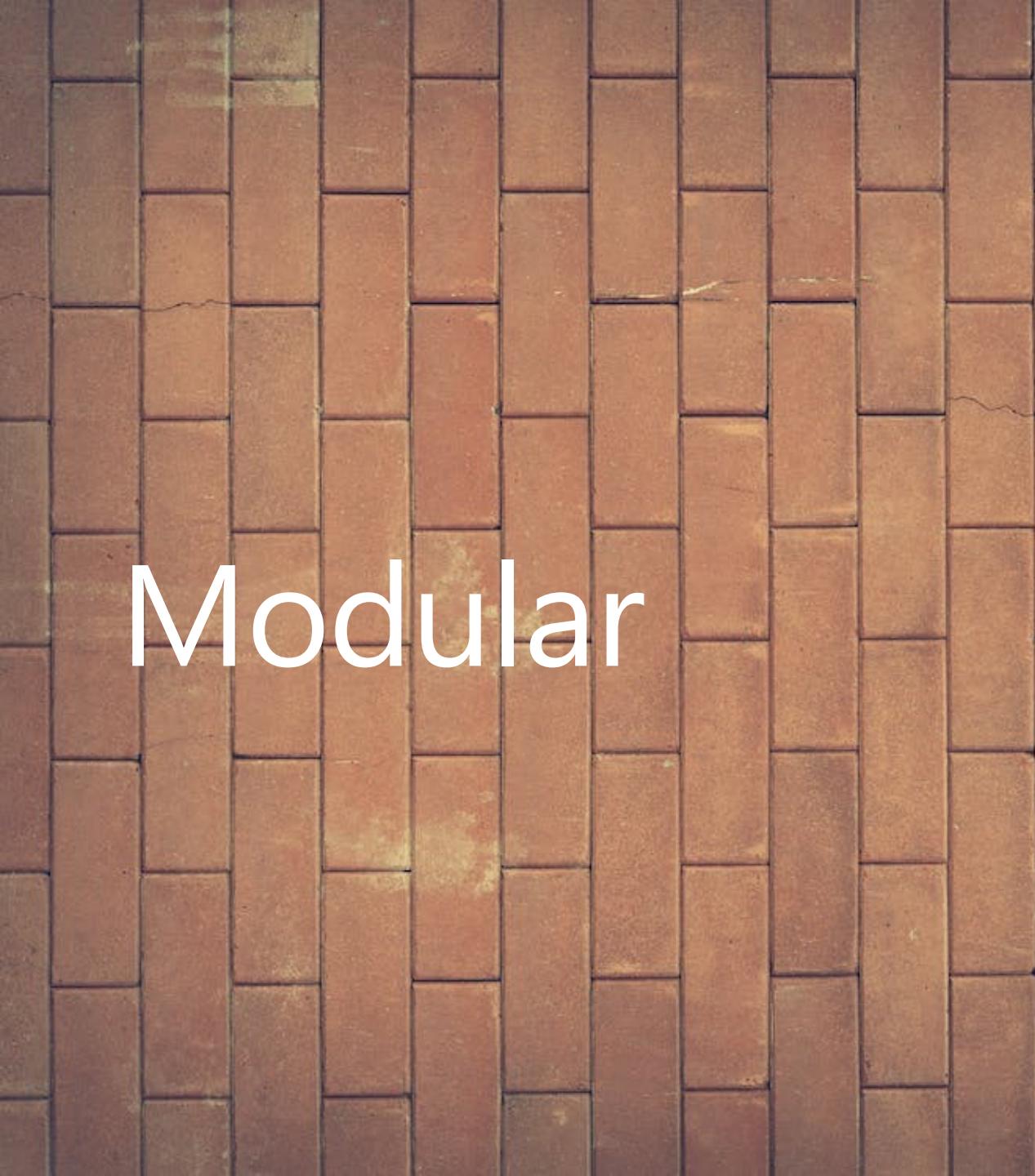
**TRAILHEAD**  
TECHNOLOGY PARTNERS











Modular



Ball Of Mud



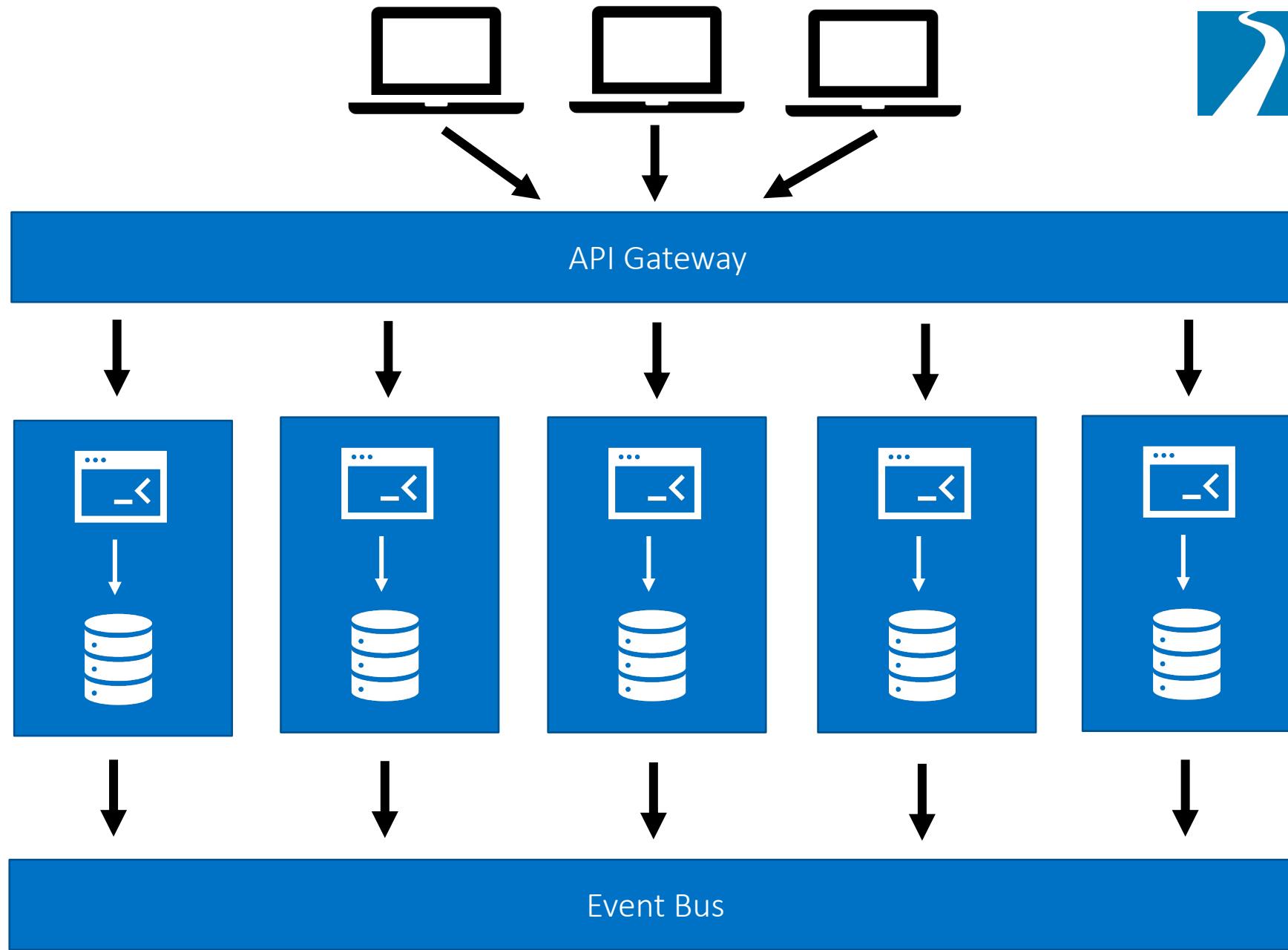
# Microservices



**TRAILHEAD**  
TECHNOLOGY PARTNERS



TRAILHEAD  
TECHNOLOGY PARTNERS



# Distributed Monolith

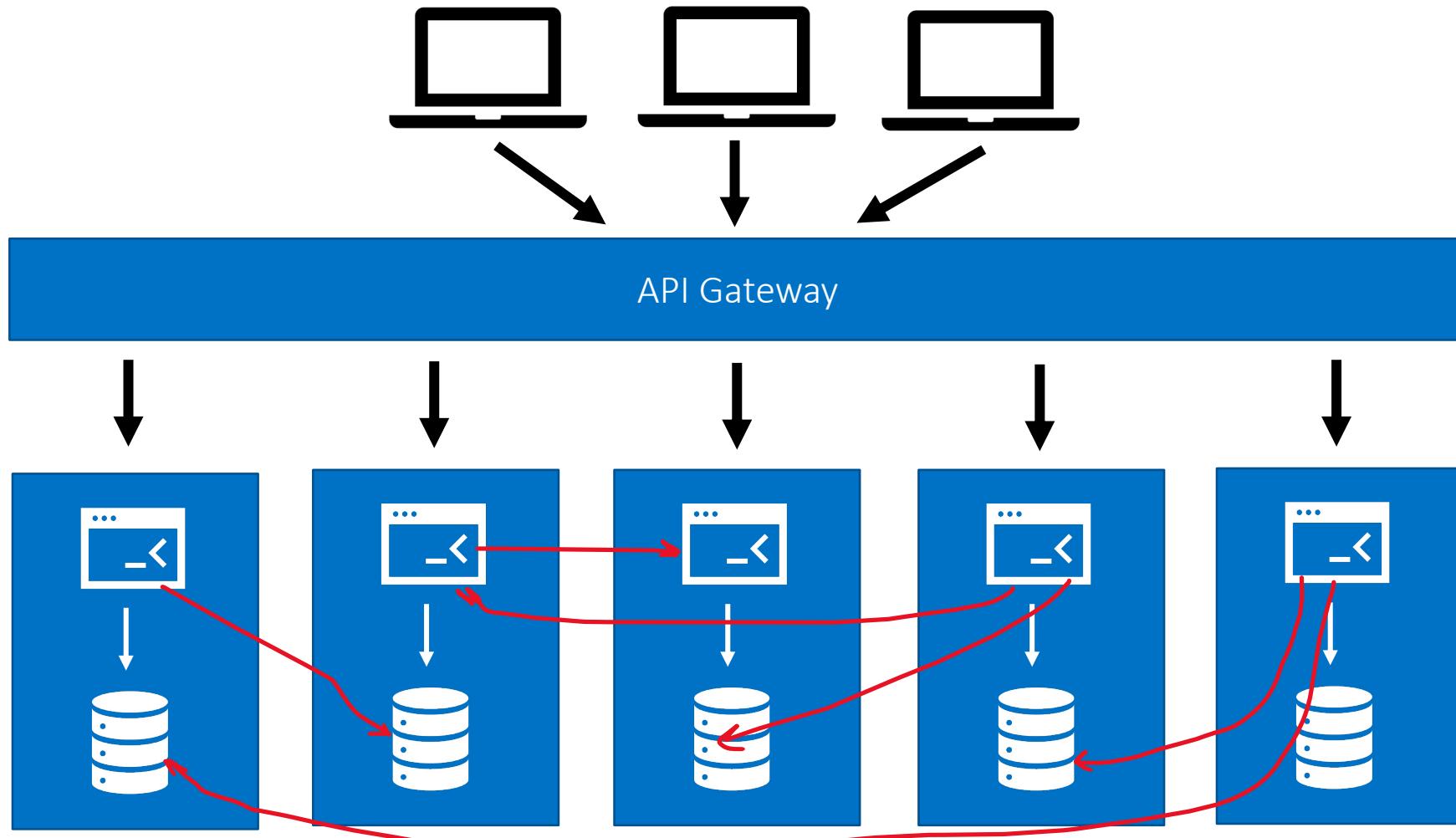


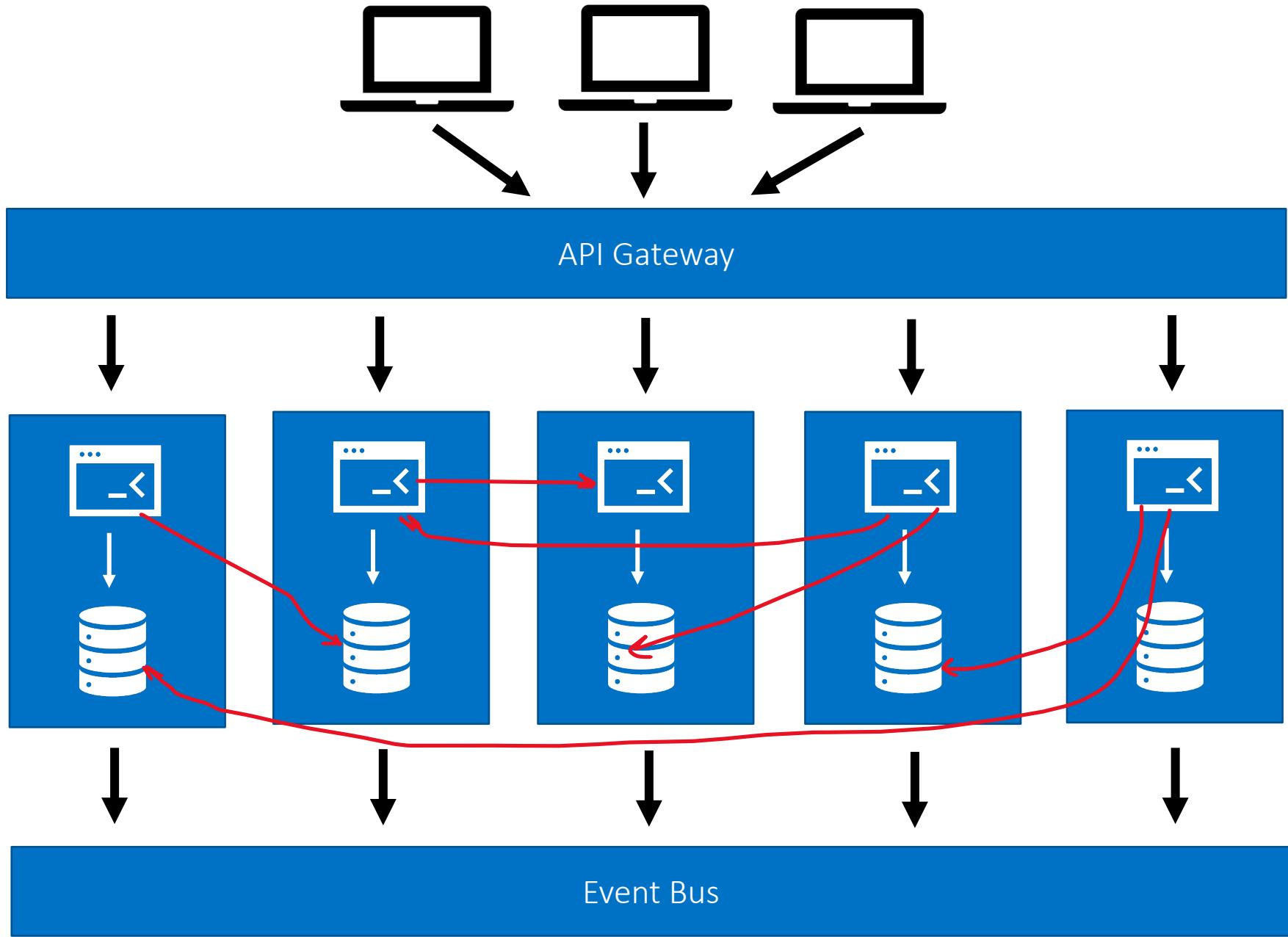
**TRAILHEAD**  
TECHNOLOGY PARTNERS

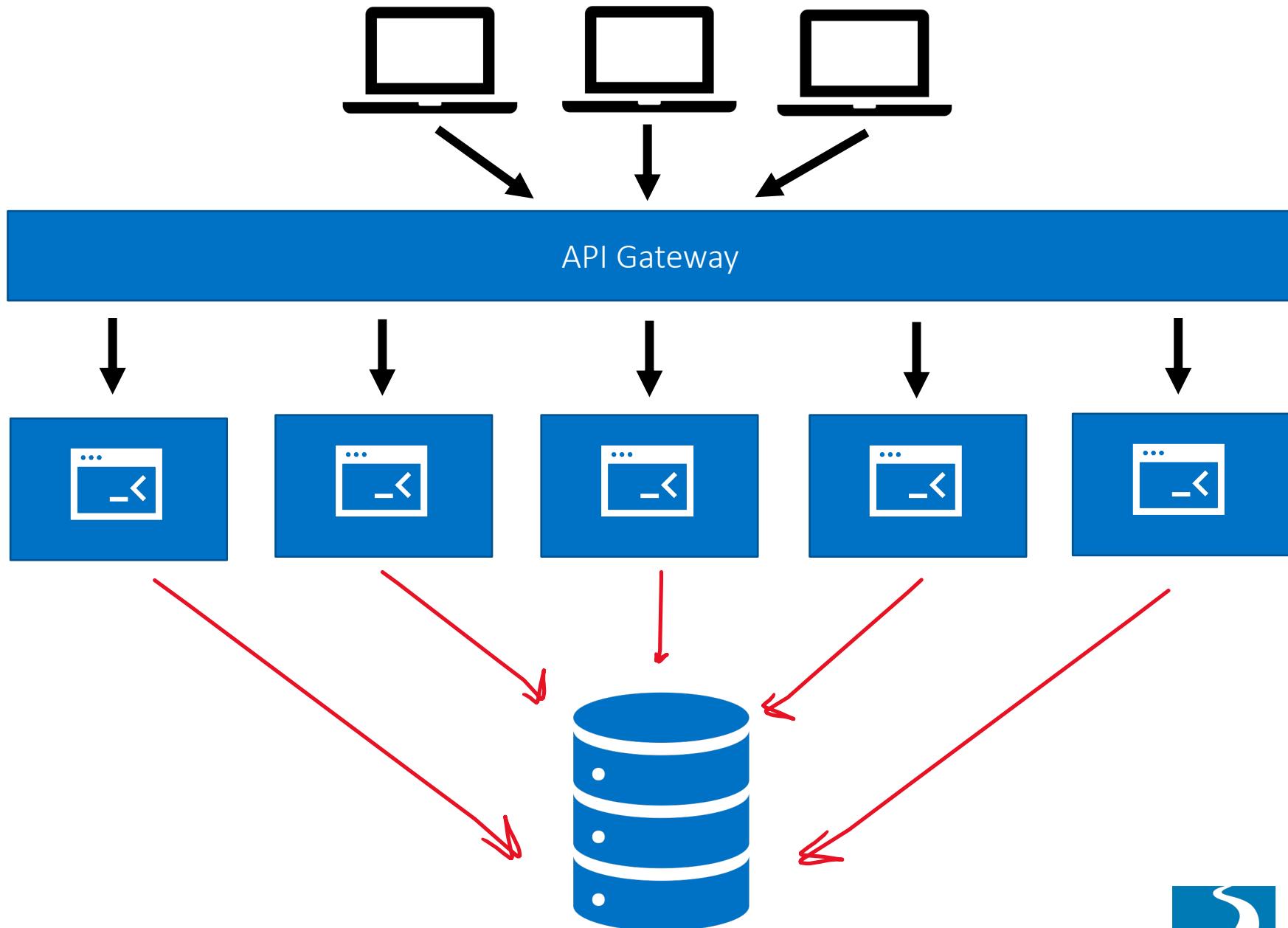
# Distributed Monolith



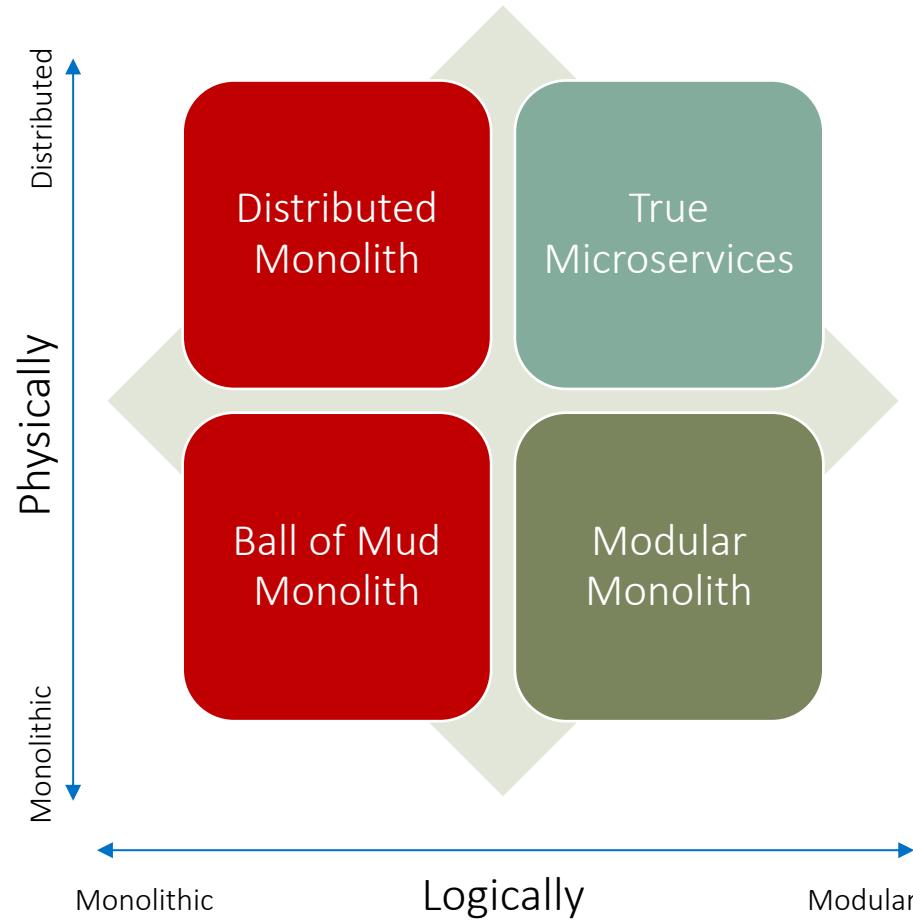
**TRAILHEAD**  
TECHNOLOGY PARTNERS







# Good vs Bad Monoliths





# 10 Most Common Mistakes

Avoid Creating a “Monster”

# Assuming Microservices are Always Better

Problem #1

# First Rule of Microservices: Don't Use Microservices

Have a “Really Good Reason” – Sam Newman



Monoliths aren't inherently bad or un-scalable



Microservices are *hard* to do well



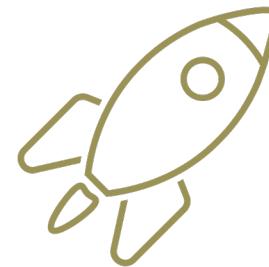
*Wrong reasons create distributed monoliths*

# Some Good Reasons to Microservice

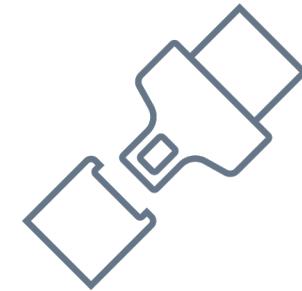
“Microservice architectures **buy you options**” – James Lewis



More Scalability  
Options

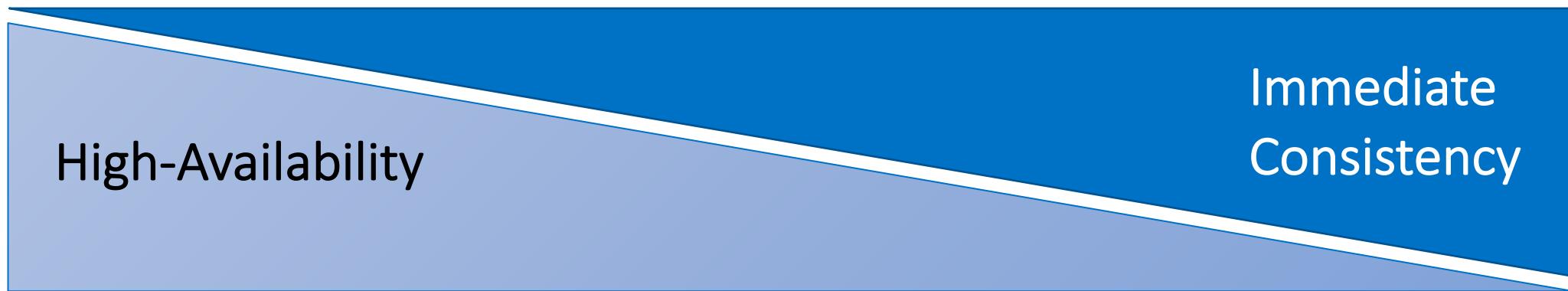


Independent  
Deployability



Isolate Surface Area  
of Failure

# The Big Trade Off



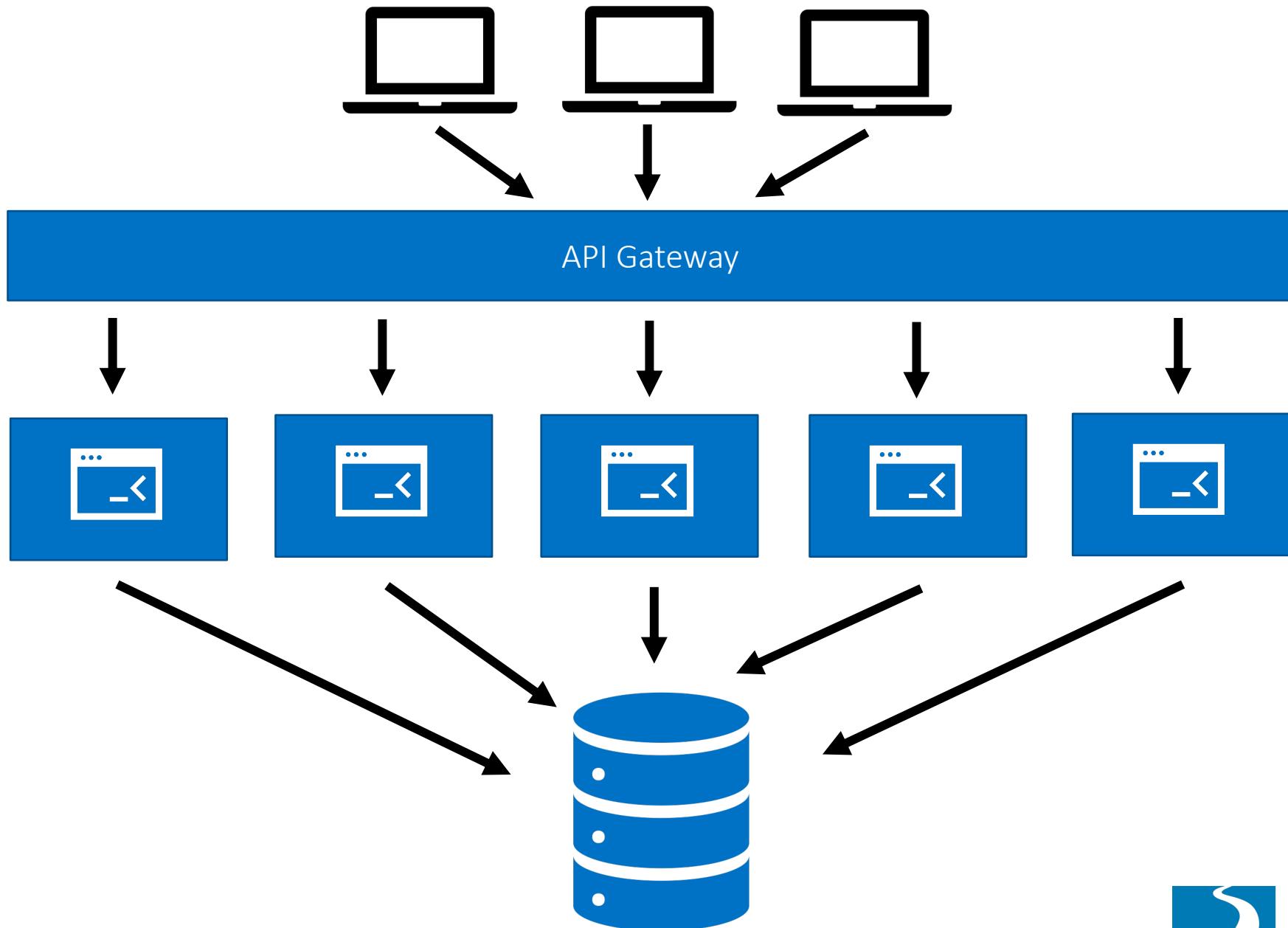
Microservices

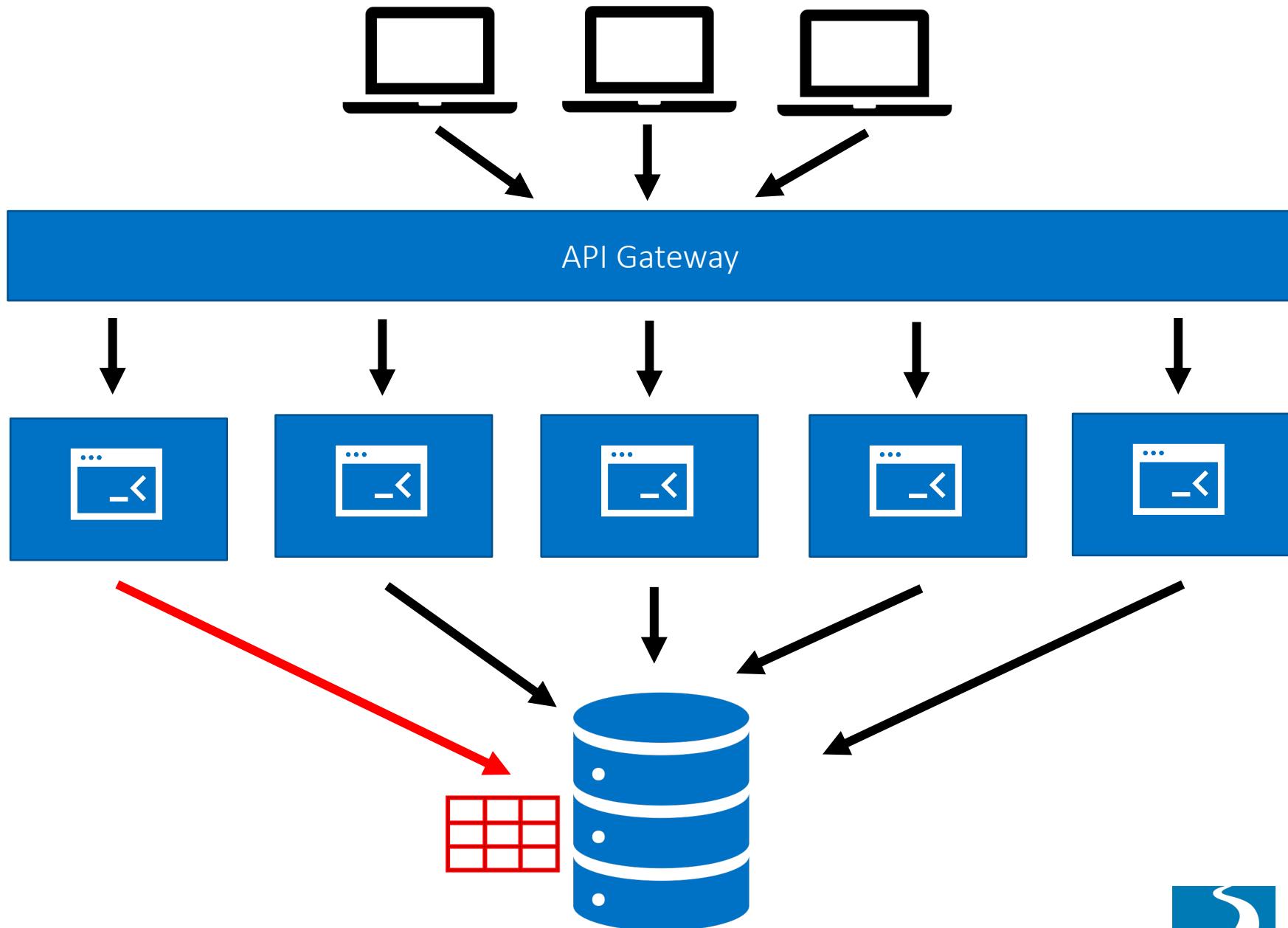
Monoliths

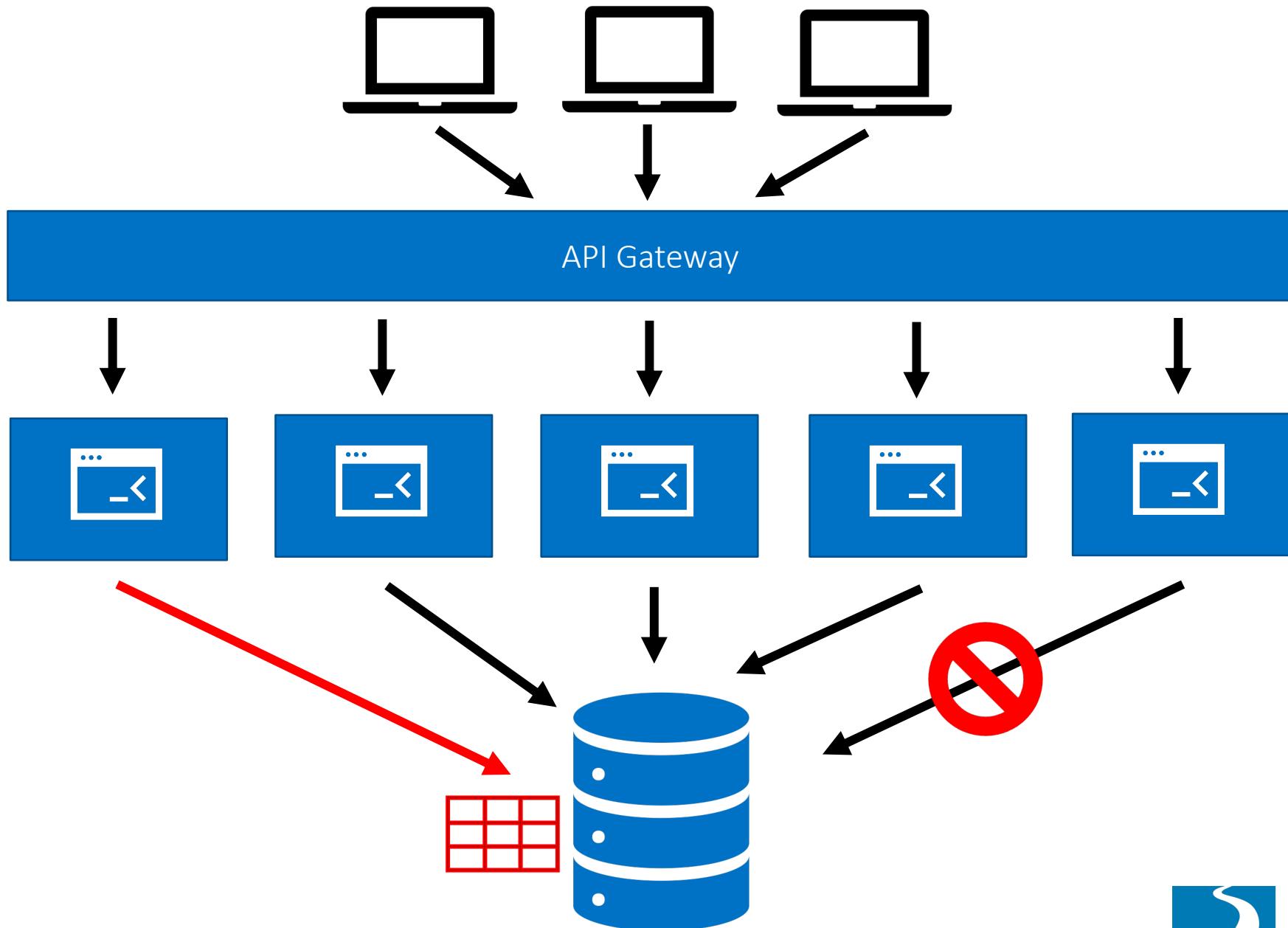
# Shared Data Store or Models

Problem #2









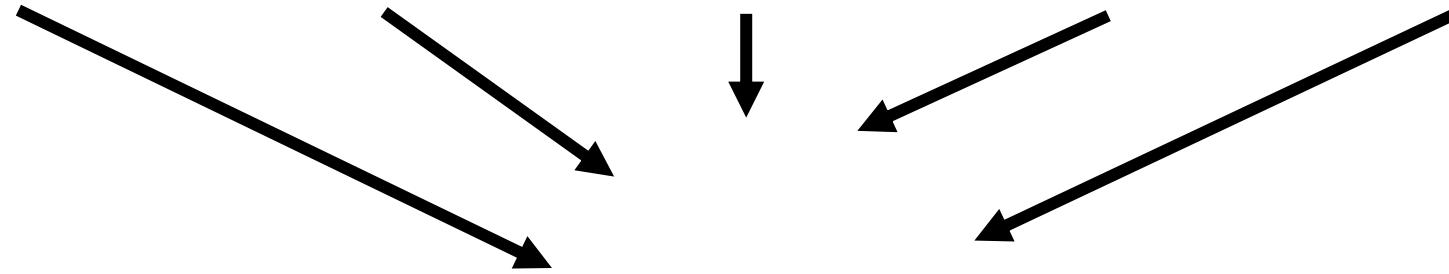
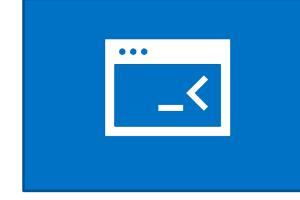
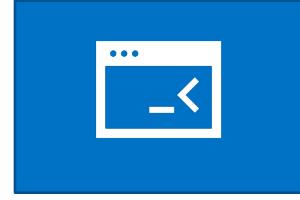
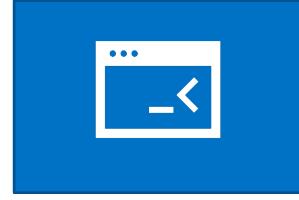
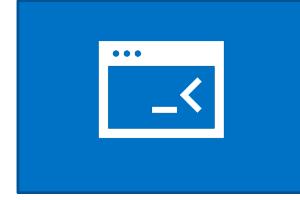
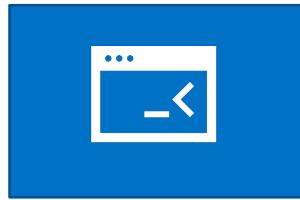
Reservations

...

Maintenance

...

Housekeeping



```
HotelRoom {  
    CurrentlyOccupied : bool  
    NumberofBeds : int  
    MinutesToClean : float  
    RepairHistory: [  
        { RepairDate: ... },  
        { RepairDate: ... },  
    ]  
}
```



**TRAILHEAD**  
TECHNOLOGY PARTNERS

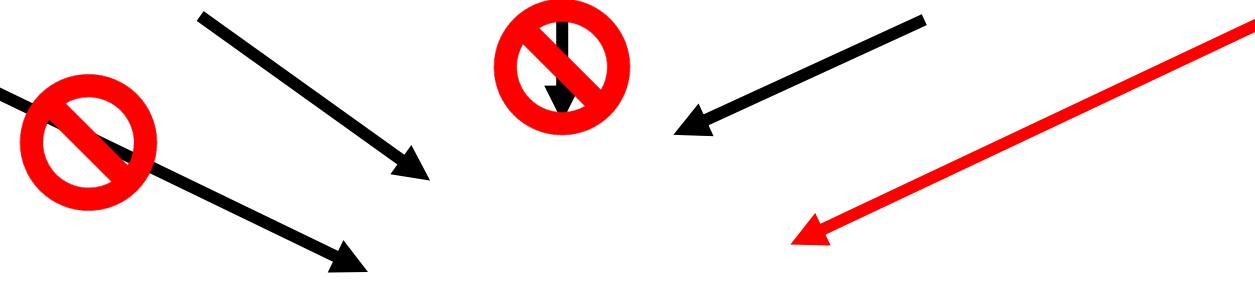
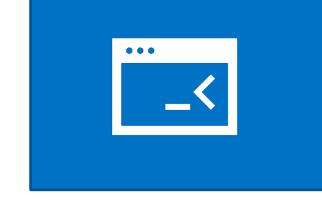
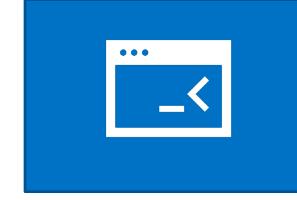
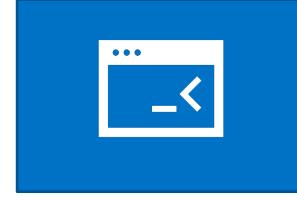
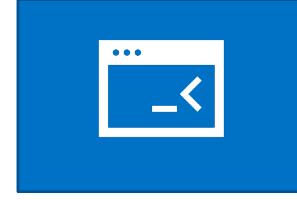
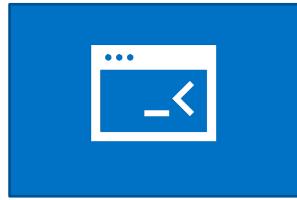
Reservations

...

Maintenance

...

Housekeeping

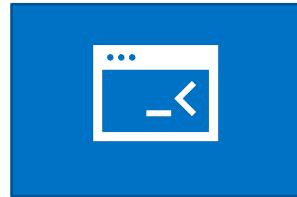


```
HotelRoom {  
    CurrentlyOccupied : bool  
    NumberofBeds : int  
    MinutesToClean : int float  
    RepairHistory: [  
        { RepairDate: ... },  
        { RepairDate: ... },  
    ]  
}
```



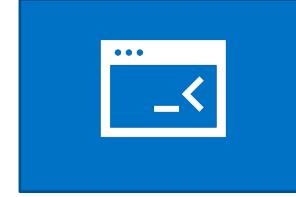
**TRAILHEAD**  
TECHNOLOGY PARTNERS

Reservations



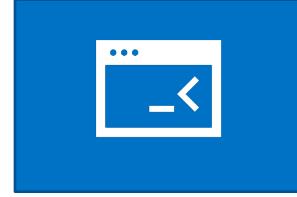
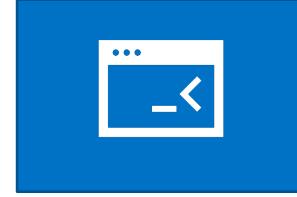
...

Maintenance



...

Housekeeping



```
HotelRoom {  
    RepairHistory: [  
        { RepairDate: ... },  
        { RepairDate: ... },  
    ]  
}
```

```
HotelRoom {  
    CurrentlyOccupied : bool  
    NumberofBeds : int  
}
```

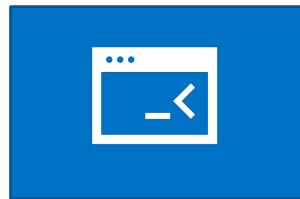


```
HotelRoom {  
    MinutesToClean : int  
}
```

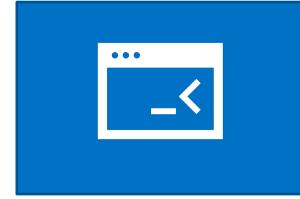


**TRAILHEAD**  
TECHNOLOGY PARTNERS

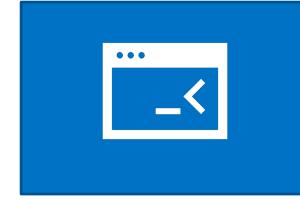
Reservations



Admin

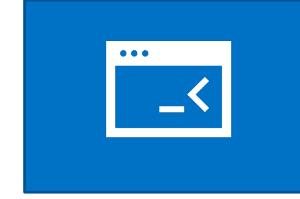


Maintenance



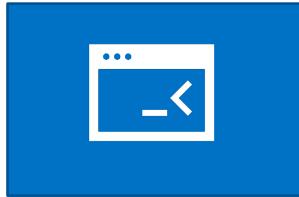
...

Housekeeping

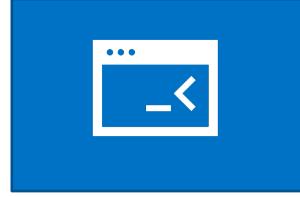


Event Bus

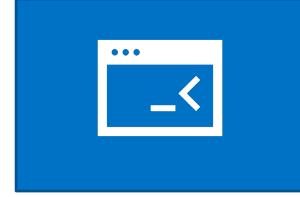
Reservations



Admin

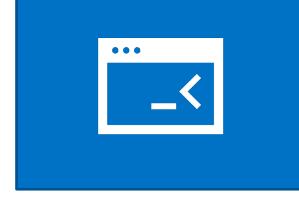


Maintenance



...

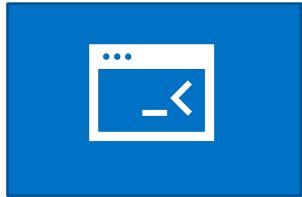
Housekeeping



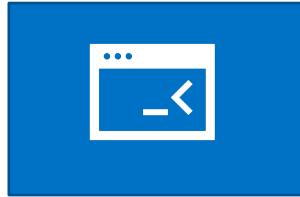
```
HotelRoom {  
    RoomNumber: int  
}
```

Event Bus

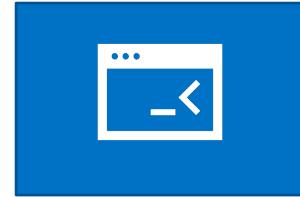
Reservations



Admin



Maintenance



...

Housekeeping

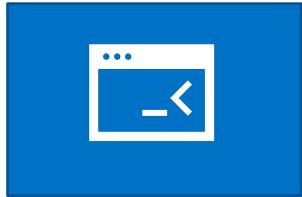


```
HotelRoom {  
    RoomNumber: int  
}
```

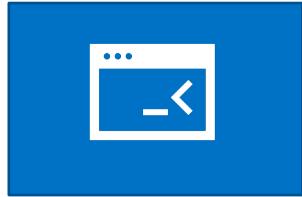


Event Bus

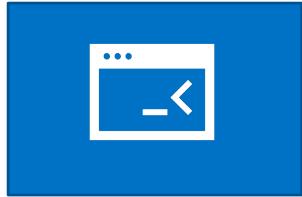
Reservations



Admin

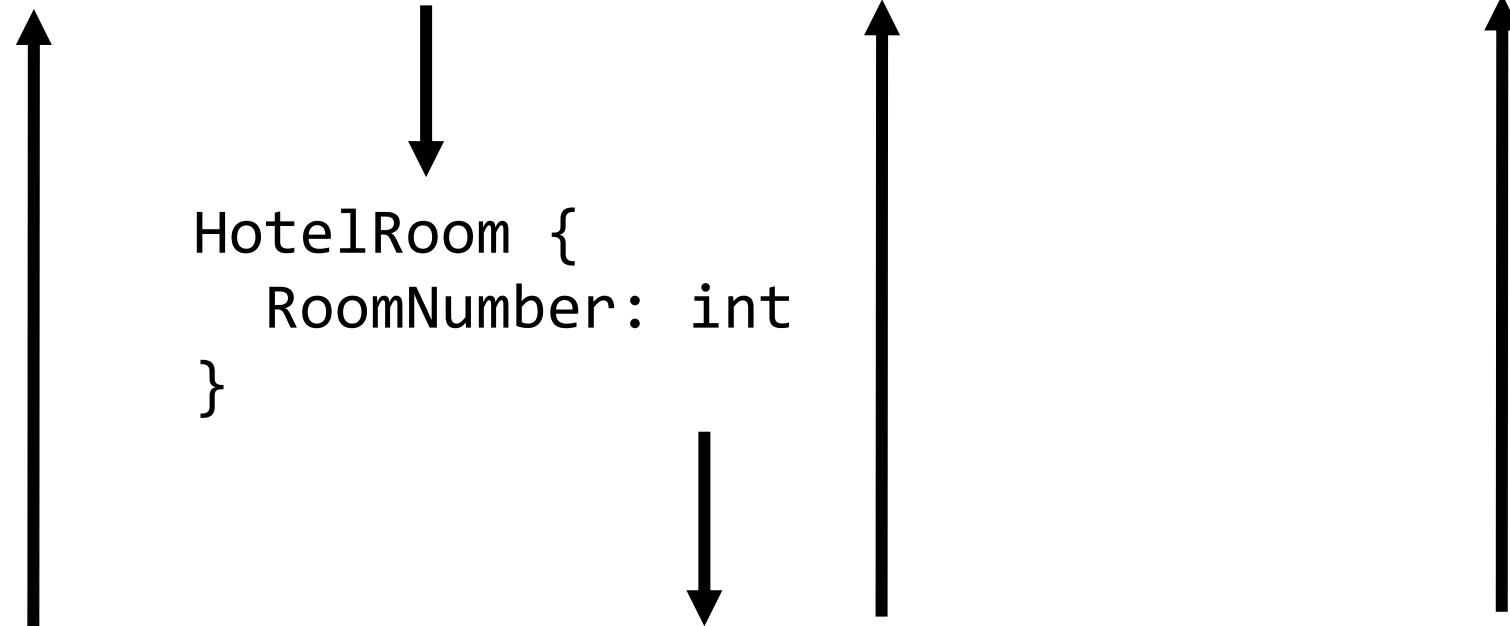


Maintenance

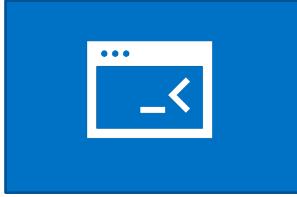


...

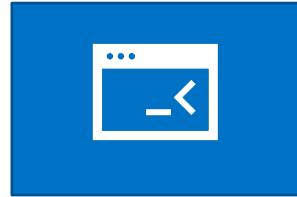
Housekeeping



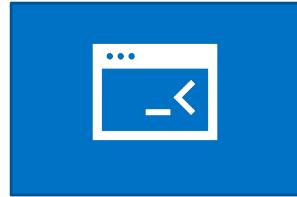
Reservations



Admin



Maintenance



...

Housekeeping



```
HotelRoom {  
    RoomNumber: int  
    CurrentlyOccupied : bool  
    NumberofBeds : int  
}
```

```
HotelRoom {  
    RoomNumber: int  
    RepairHistory: [  
        { RepairDate: ... },  
        { RepairDate: ... },  
    ]  
}
```

```
HotelRoom {  
    RoomNumber: int  
    MinutesToClean : int  
}
```

Event Bus



**TRAILHEAD**  
TECHNOLOGY PARTNERS

# Microservices That Are Too Big

Problem #3

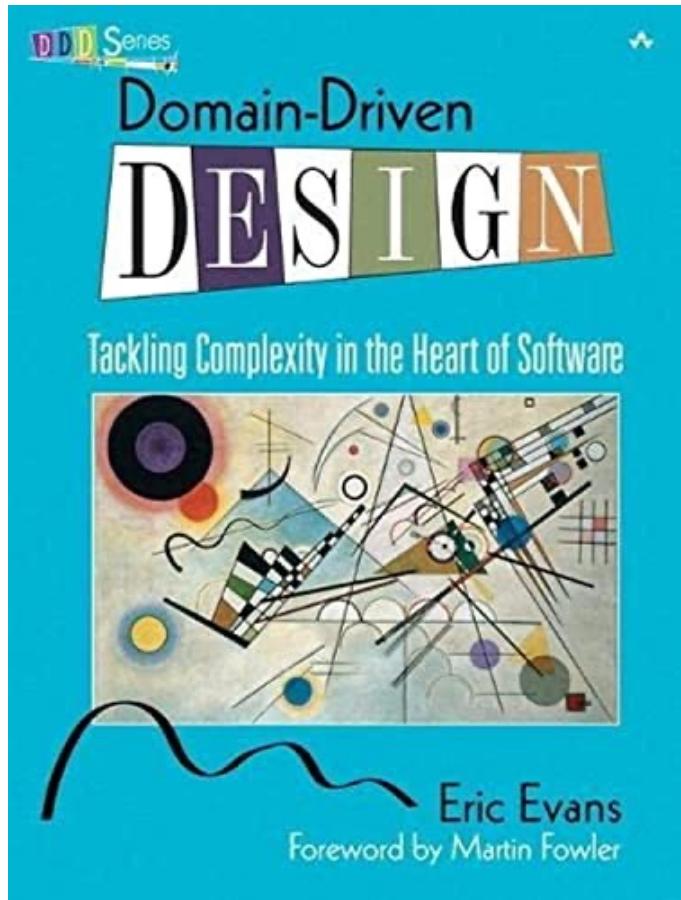


# Domain Driven Design (DDD)

Domain  
Subdomain  
Bounded Context

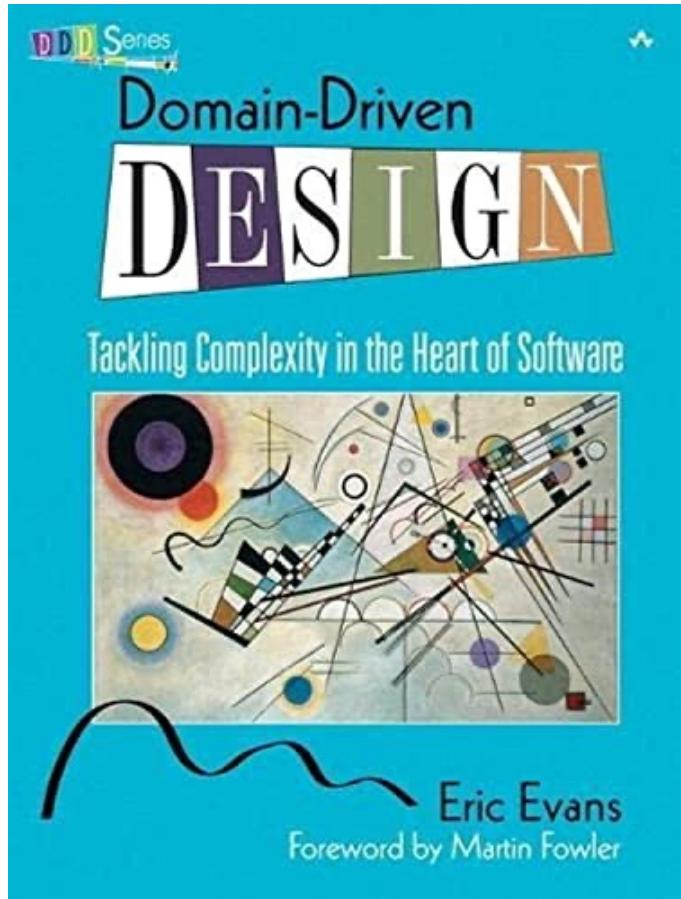


# Domain Driven Design



Domain  
Subdomain  
Bounded Context

# Domain Driven Design



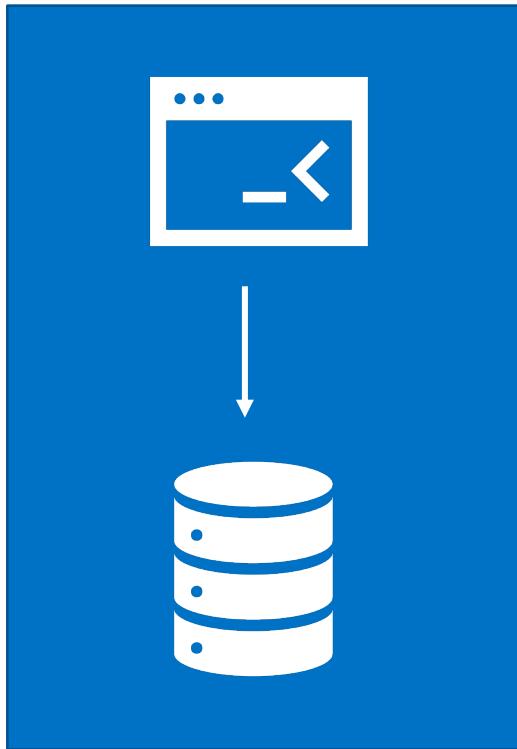
Domain  
Subdomain  
Bounded Context

J's ***SIMPLE RULE: smallest possible microservices without chatty communication between services***

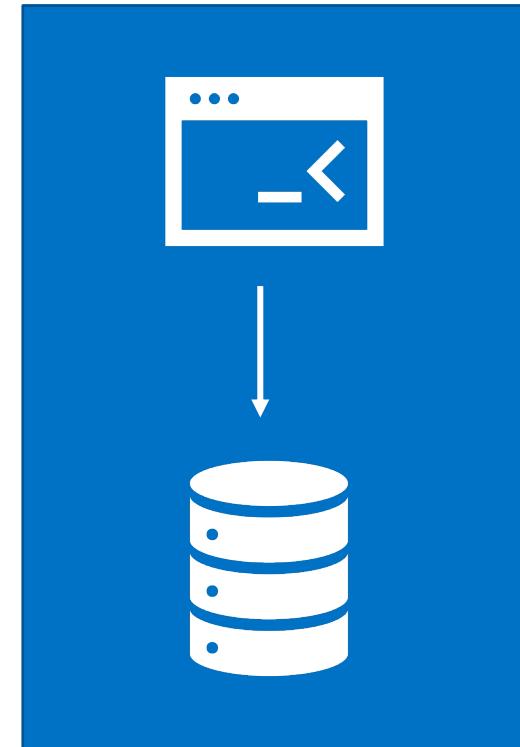
# Microservices That Are Too Small

Problem #4

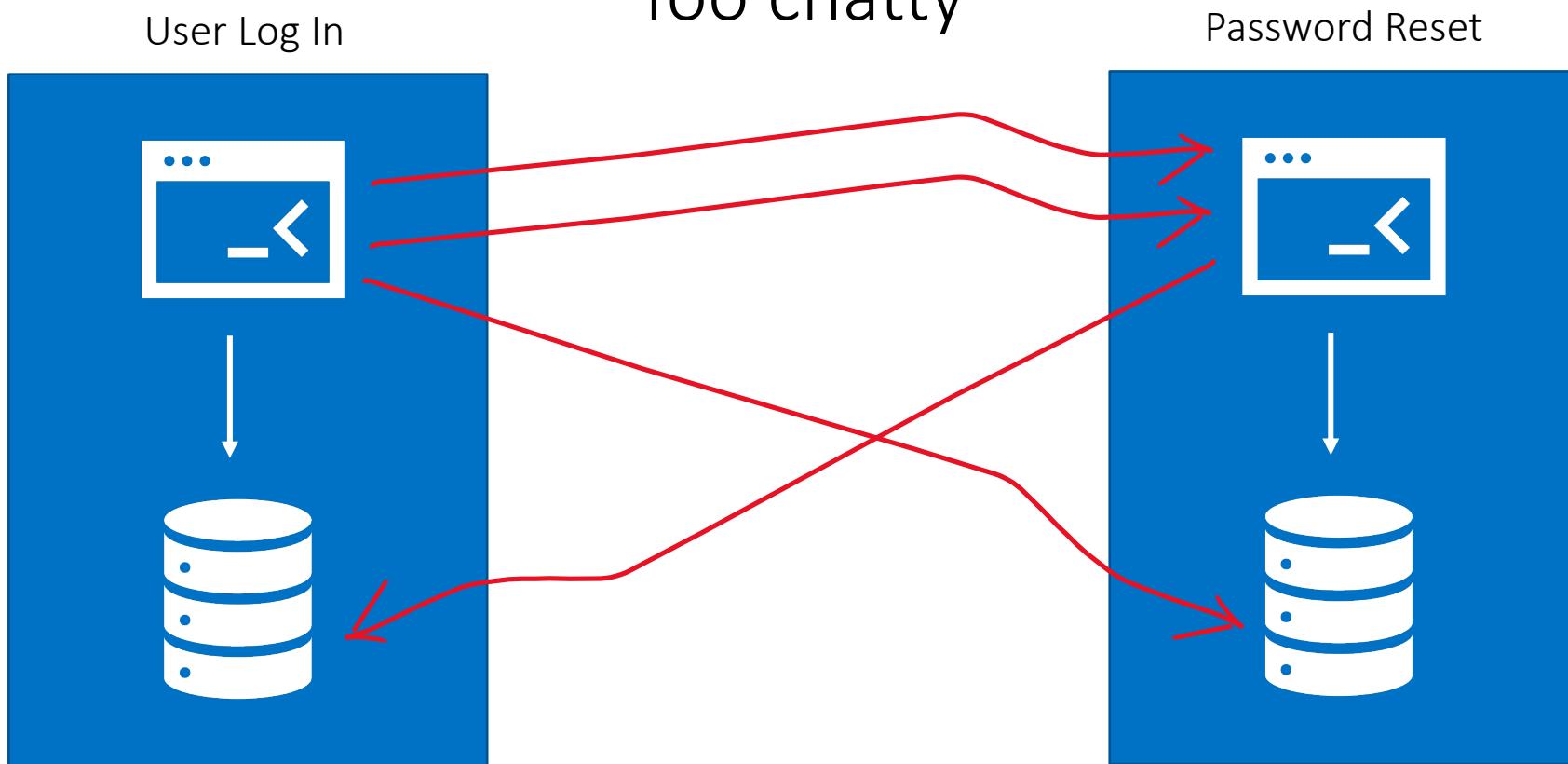
User Log In



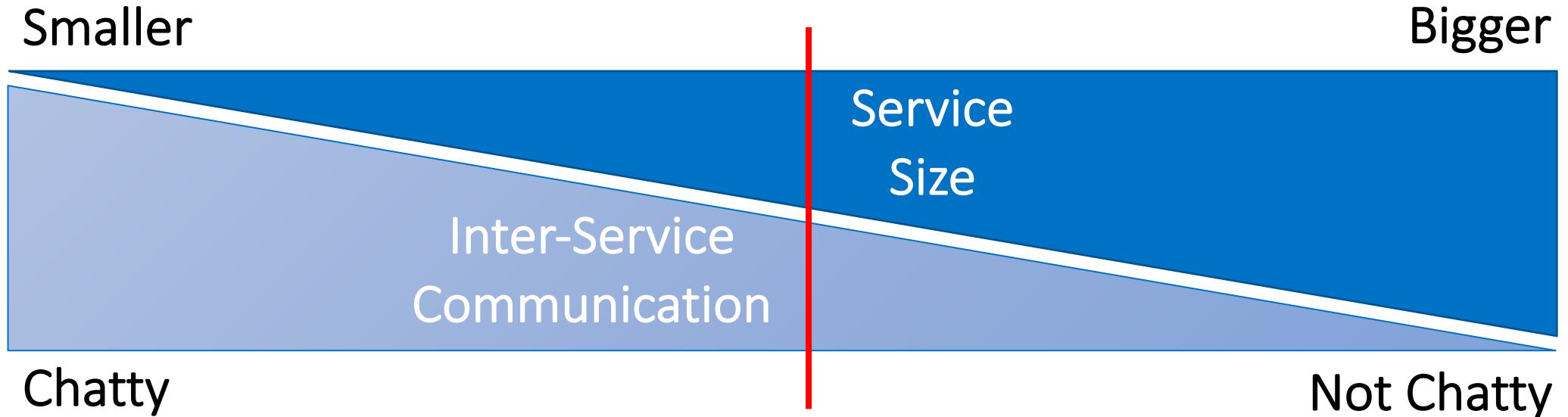
Password Reset



# Too chatty



# Balance



# Starting from Scratch

Problem #5



# Greenfield is Actually Harder

Easier to partition an existing, "brownfield" system

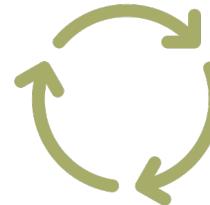
## **Brownfield → Microservices Advantages:**

1. Code and relationships to examine
2. People to talk to who know the system
3. A system that already works
4. Baseline to compare to refactoring

# Three “Brownfield” Migration Approaches



Big bang

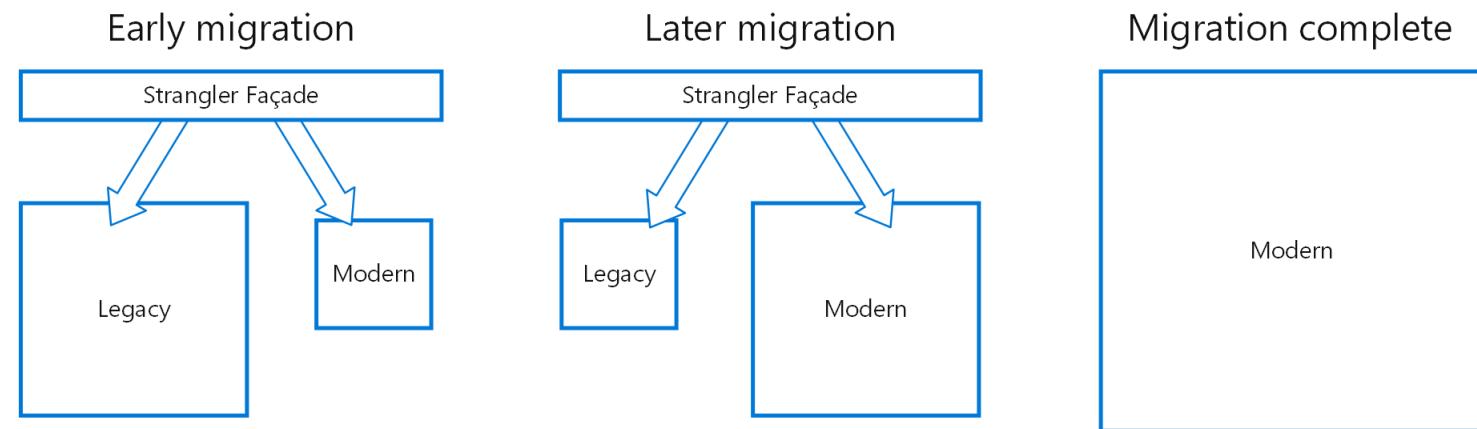


Evolution



“Strangler fig” pattern

# Strangler Fig Pattern



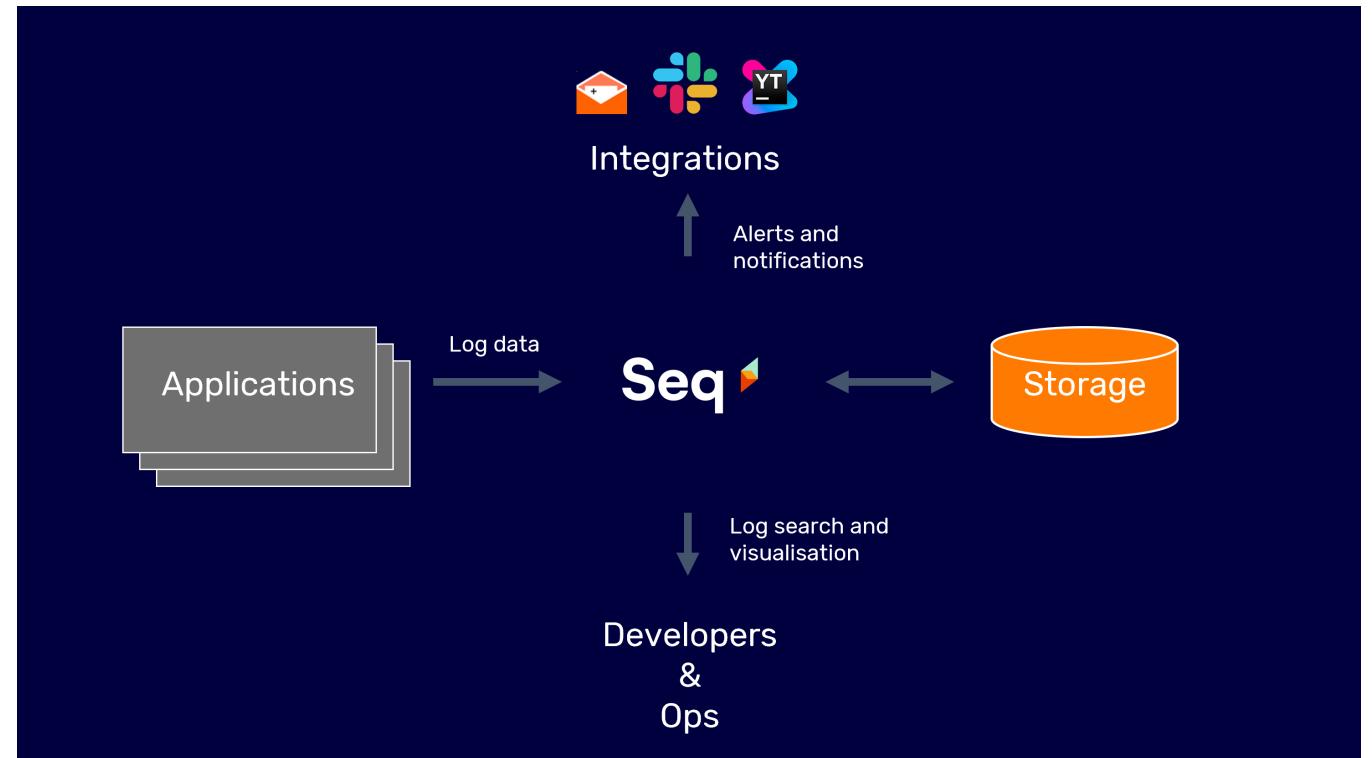
Source: <https://docs.microsoft.com/en-us/azure/architecture/patterns/strangler-fig>

# Coupling Through Cross-Cutting Concerns

Problem #6

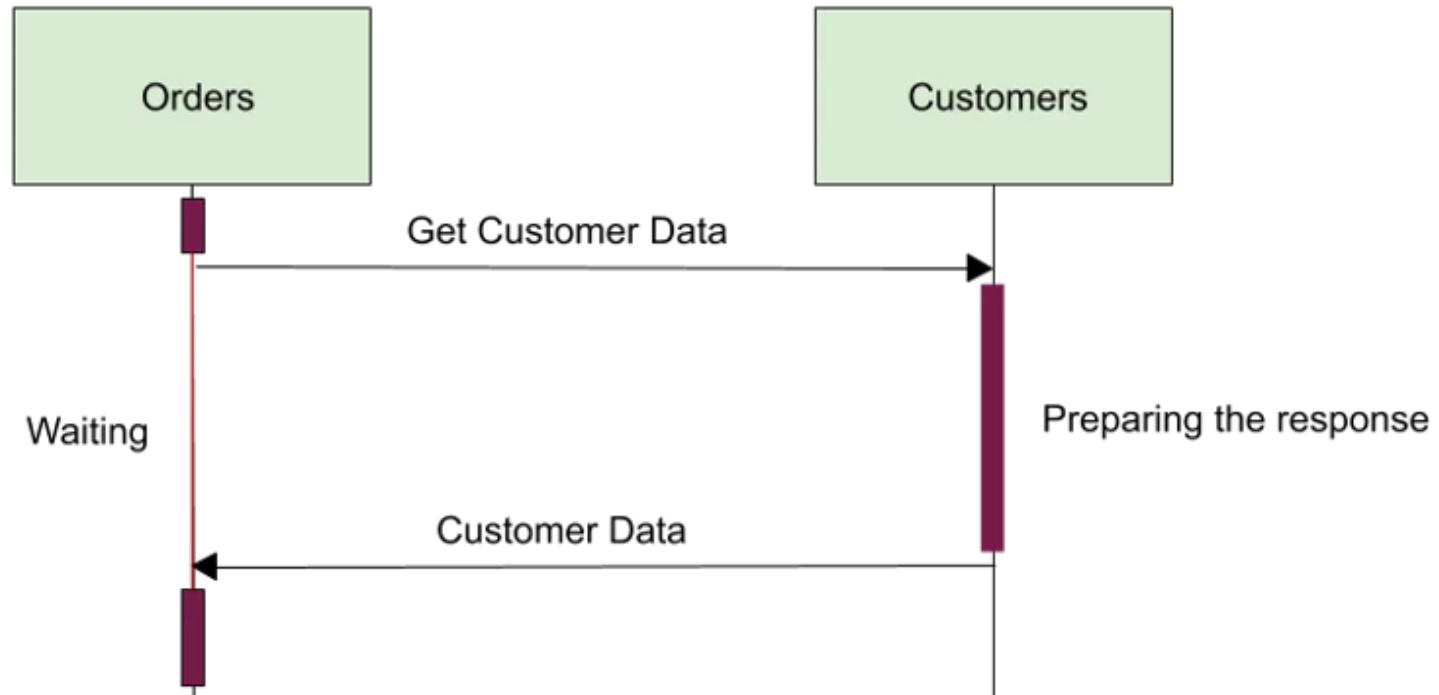
# Example: Distributed Logging Is Hard

- Centralized without coupling
- Third party solutions like Seq
  - Seq: “Intelligent search, analysis, and alerting server built specifically for modern structured log data”
  - Supports .NET, Java, NodeJS, Ruby, Go, Python, more.
  - Inherently fault tolerant, embraces eventual consistency

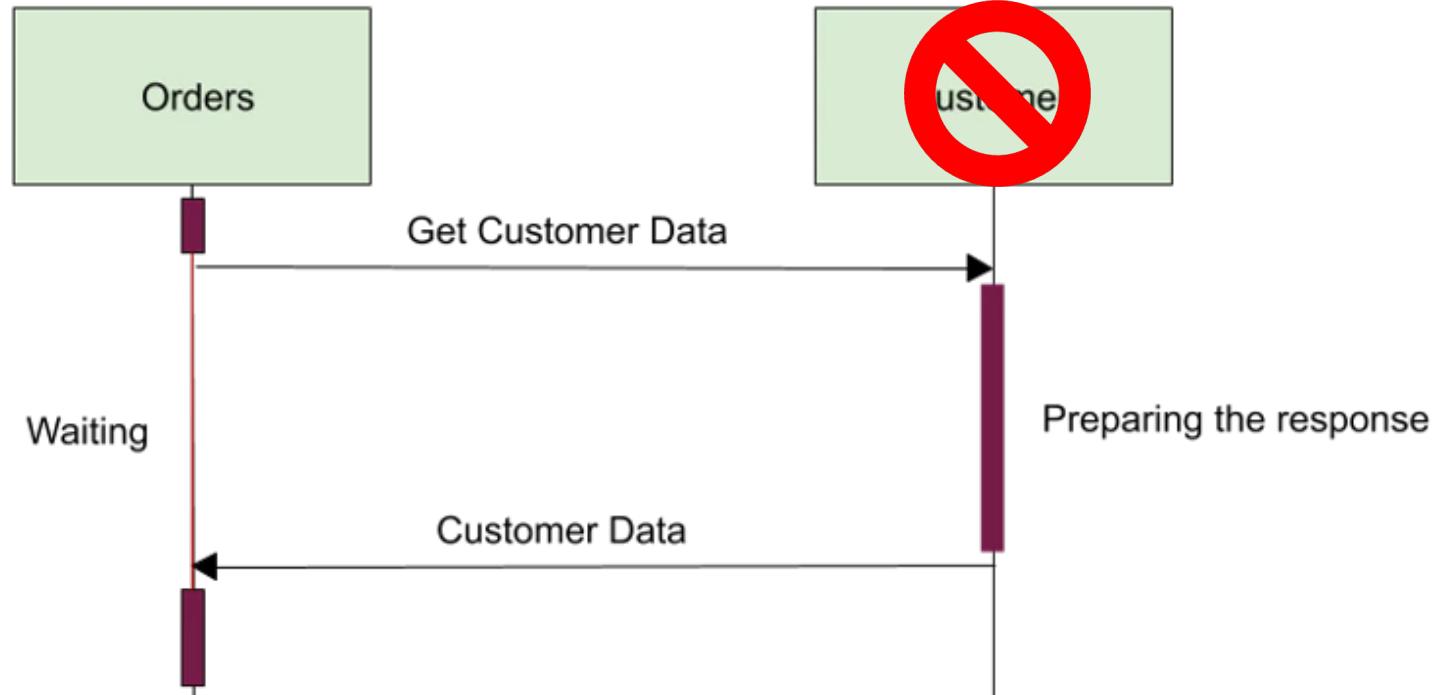


# Use of Synchronous Communication

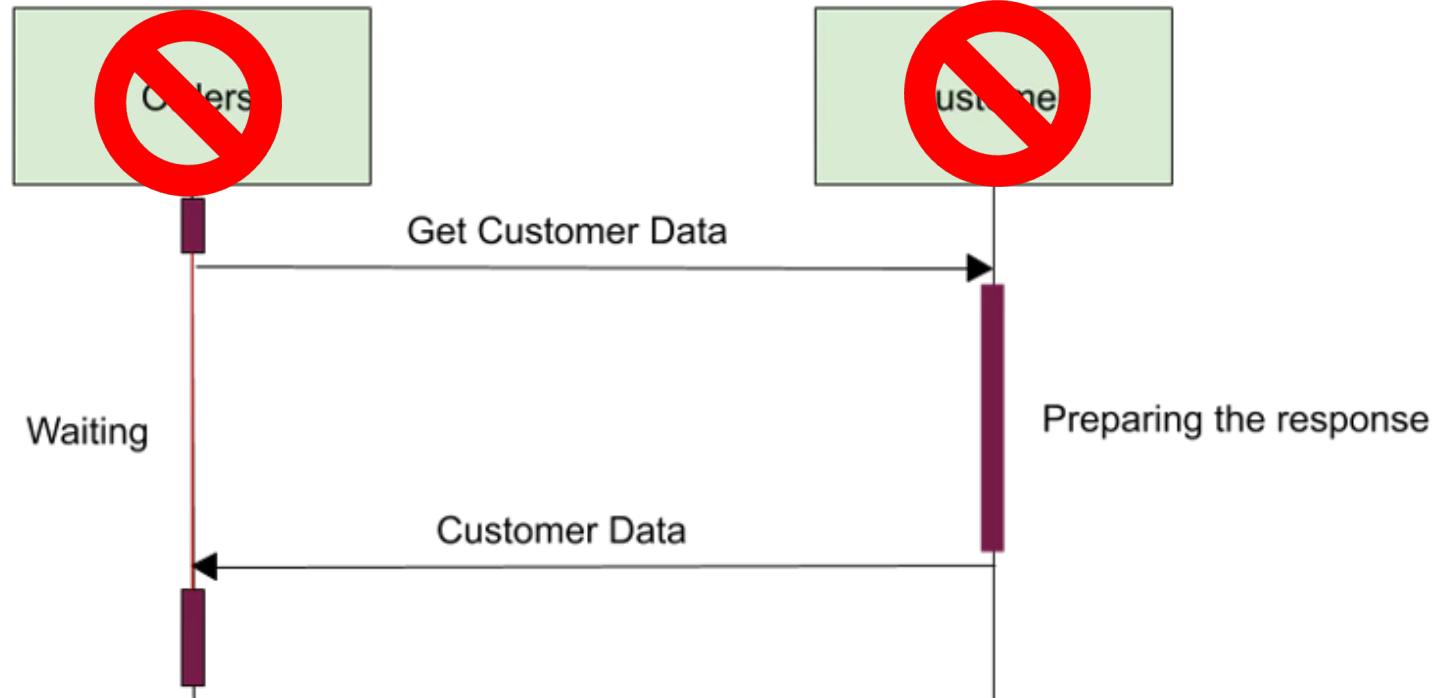
Problem #7



Source: <https://www.capitalone.com/tech/software-engineering/how-to-avoid-loose-coupled-microservices/>



Source: <https://www.capitalone.com/tech/software-engineering/how-to-avoid-loose-coupled-microservices/>



Source: <https://www.capitalone.com/tech/software-engineering/how-to-avoid-loose-coupled-microservices/>

# Breaking Changes to Event Contracts

Problem #8



# Rules for Event Changes

01

No new **required** fields, only optional fields (with documented default values).

02

Unrecognized fields are **ignored** (but forwarded)

03

Consumers of optional fields use **default** values when missing

04

When 1-3 cannot be satisfied, it's a **new event type**

# Not Automating Build and Release

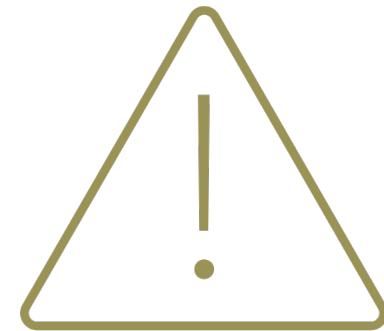
Problem #9



# Prerequisite: Automated Build and Release



Time consuming



Prone to human error

# Mismatched Teams

Problem #10



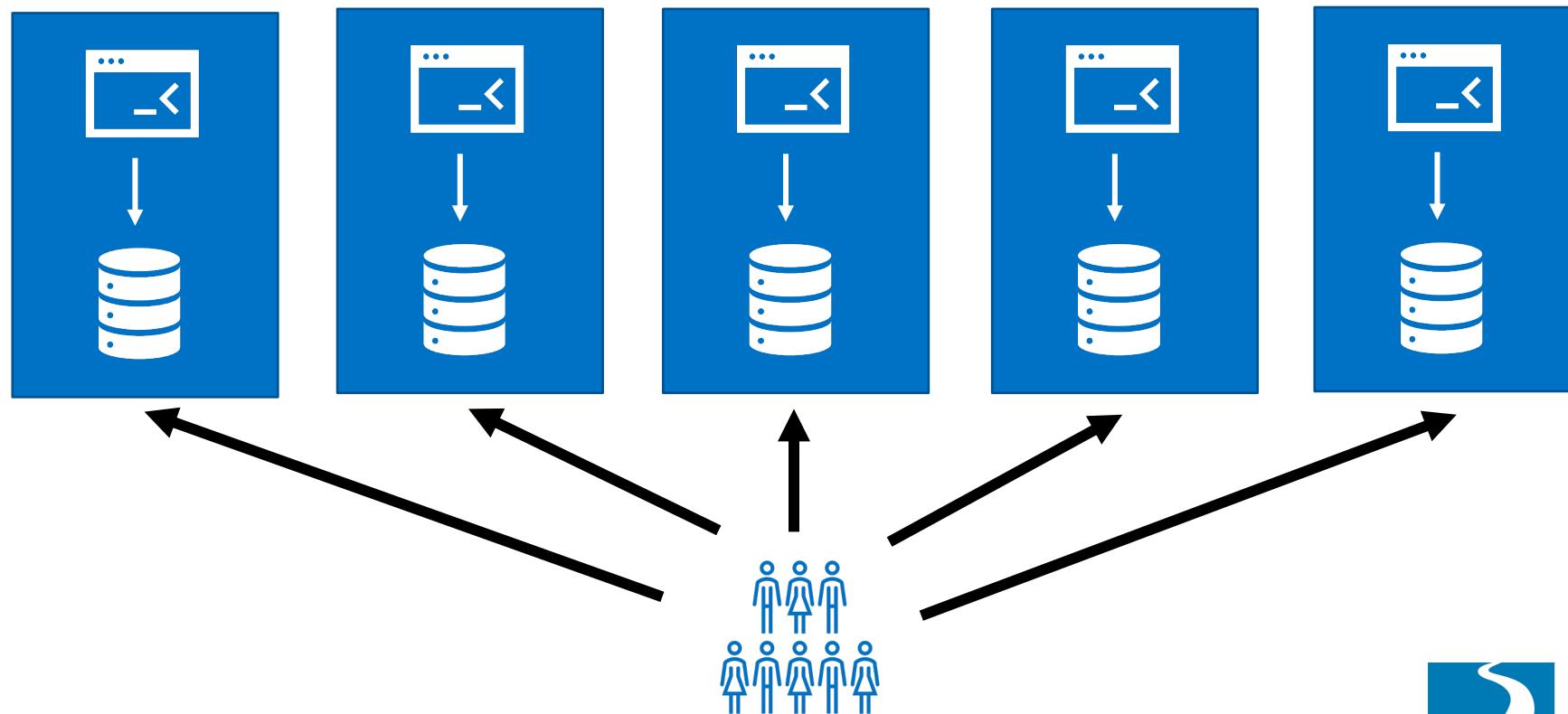
# Conway's Law

"Any organization that designs a system (defined broadly) will produce a design whose **structure** is a **copy** of the organization's **communication structure**."

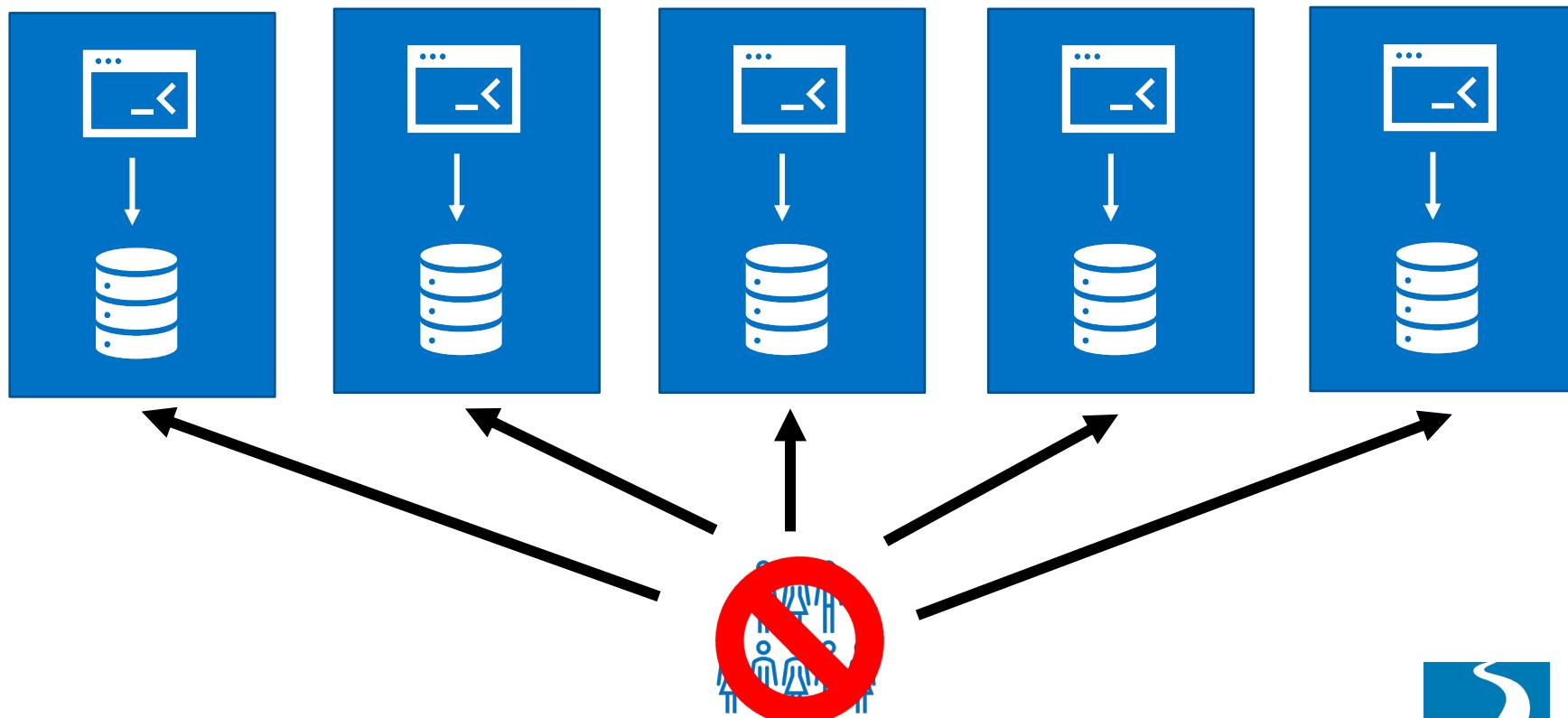
- Melvin E. Conway

IOW: if you have four groups working on a compiler, you'll get a 4-pass compiler.

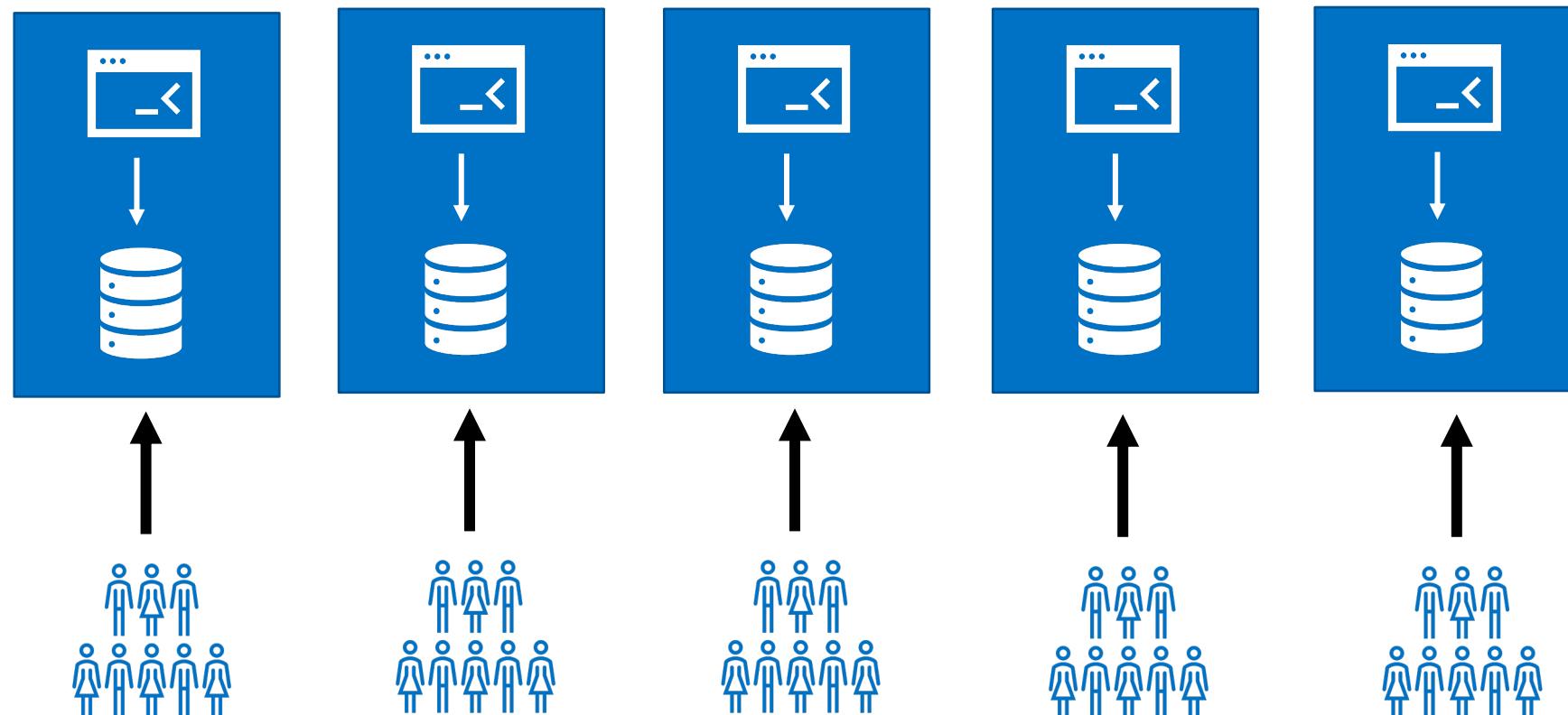
# Single Team



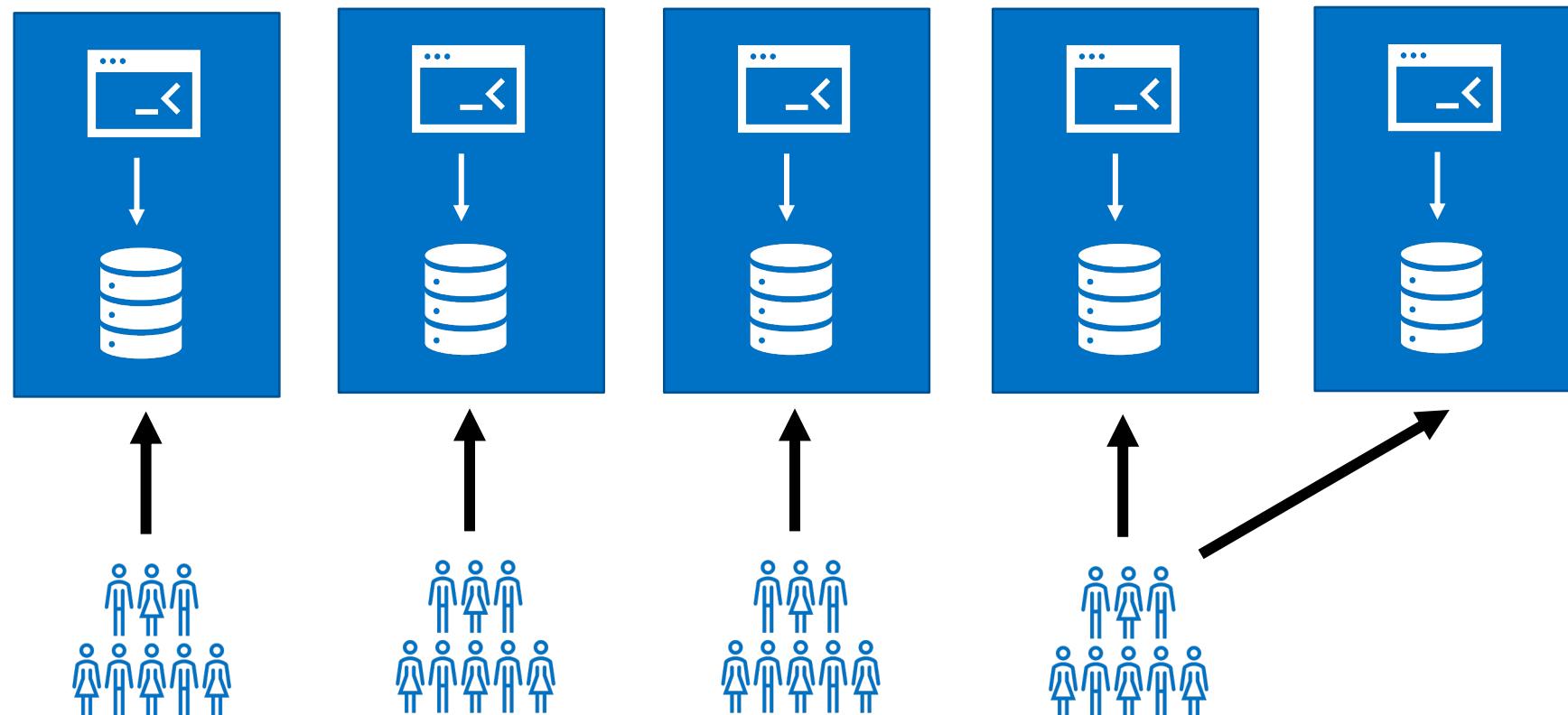
# Single Team



# Team Per Service



# Team Per Service





AUGUST 4 - 5, 2023 GRAND RAPIDS, MI



# BEER CITY CODE

A software conference for **software developers of all types**

[beercitycode.com](http://beercitycode.com)

# Summing Up

1. Monoliths can be good!
2. Only use microservices with “a really good reason”.
3. Use monoliths for lightweight or highly integrated applications with a single team; use microservices for evolving applications with clear domains and separate teams.
4. When implementing microservices, avoid the pitfalls that lead to a distributed monolith.



# Thanks! Questions?

## Jonathan "J." Tower

- ✉ jtower@trailheadtechnology.com
- 🌐 trailheadtechnology.com/blog
- 🐦 jtowermi

<https://github.com/jonathantower/distributed-monolith>

# Free Consultation



bit.ly/th-offer