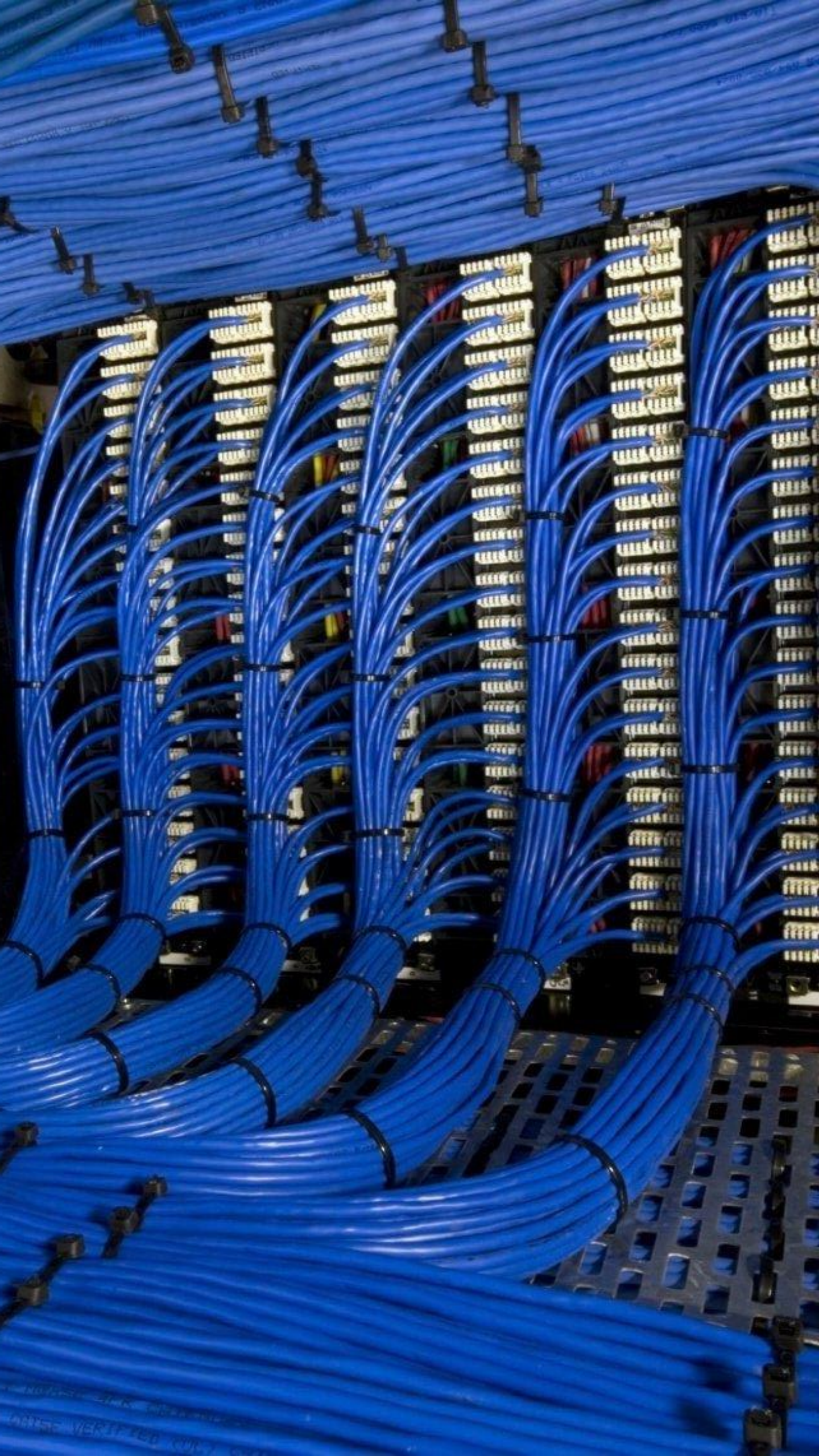
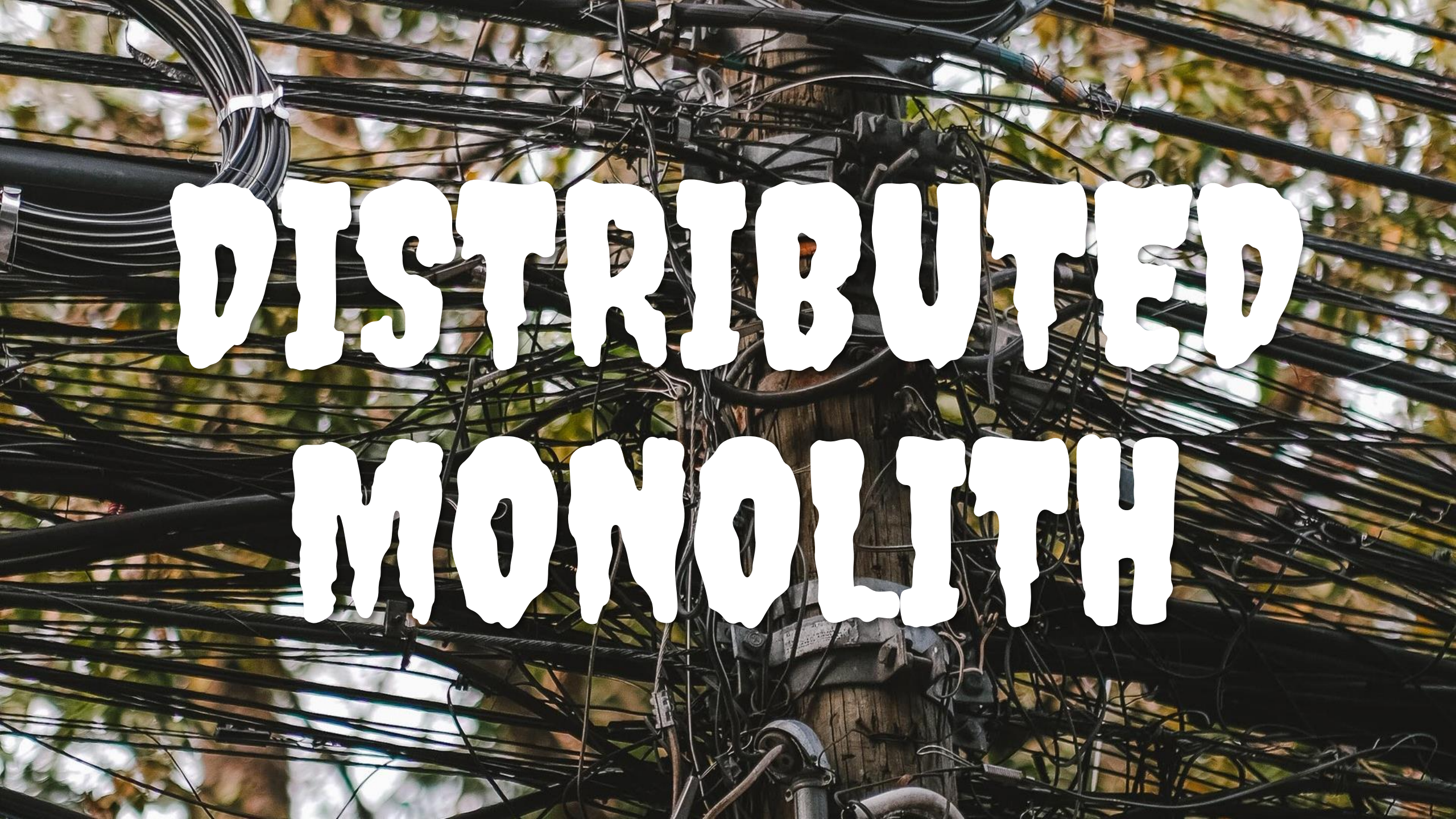
Don’t Build a Distributed Monolith

How to Avoid Doing Microservices Completely Wrong

Jonathan "J." Tower




DISTRIBUTED
MONOLITH

Avoid the **12 Most Common** Mistakes Made When Building Microservices

Hi, I'm J.

Jonathan "J." Tower

Principal Consultant & Partner

Trailhead Technology Partners

TRAILHEAD
TECHNOLOGY PARTNERS

trailheadtechnology.com

- Microsoft MVP in .NET
- Organizer of Beer City Code

- jtower@trailheadtechnology.com
- trailheadtechnology.com/blog
- jtowermi

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

BUT FIRST…If You Give $200, So Will I!

bit.ly/indy-water

"charity:water is a non-profit organization that provides clean and safe drinking water to people in developing nations. The organization was founded in 2006 and has helped fund 35,000 projects in 27 countries, benefiting over 9.5 million people.."
- Wikipedia

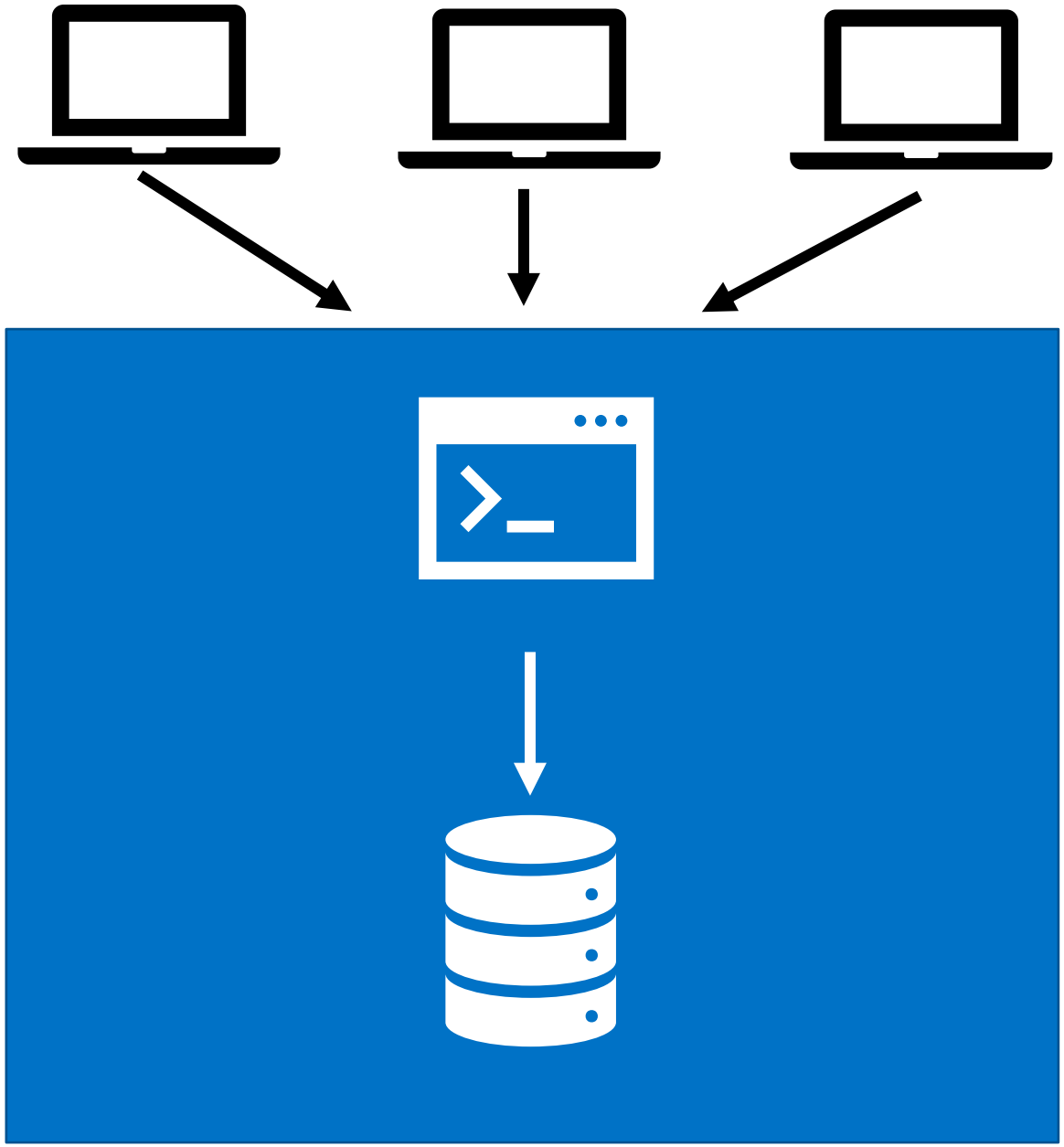
"4/4 Stars"
- CharityNavigator.org

charity: water

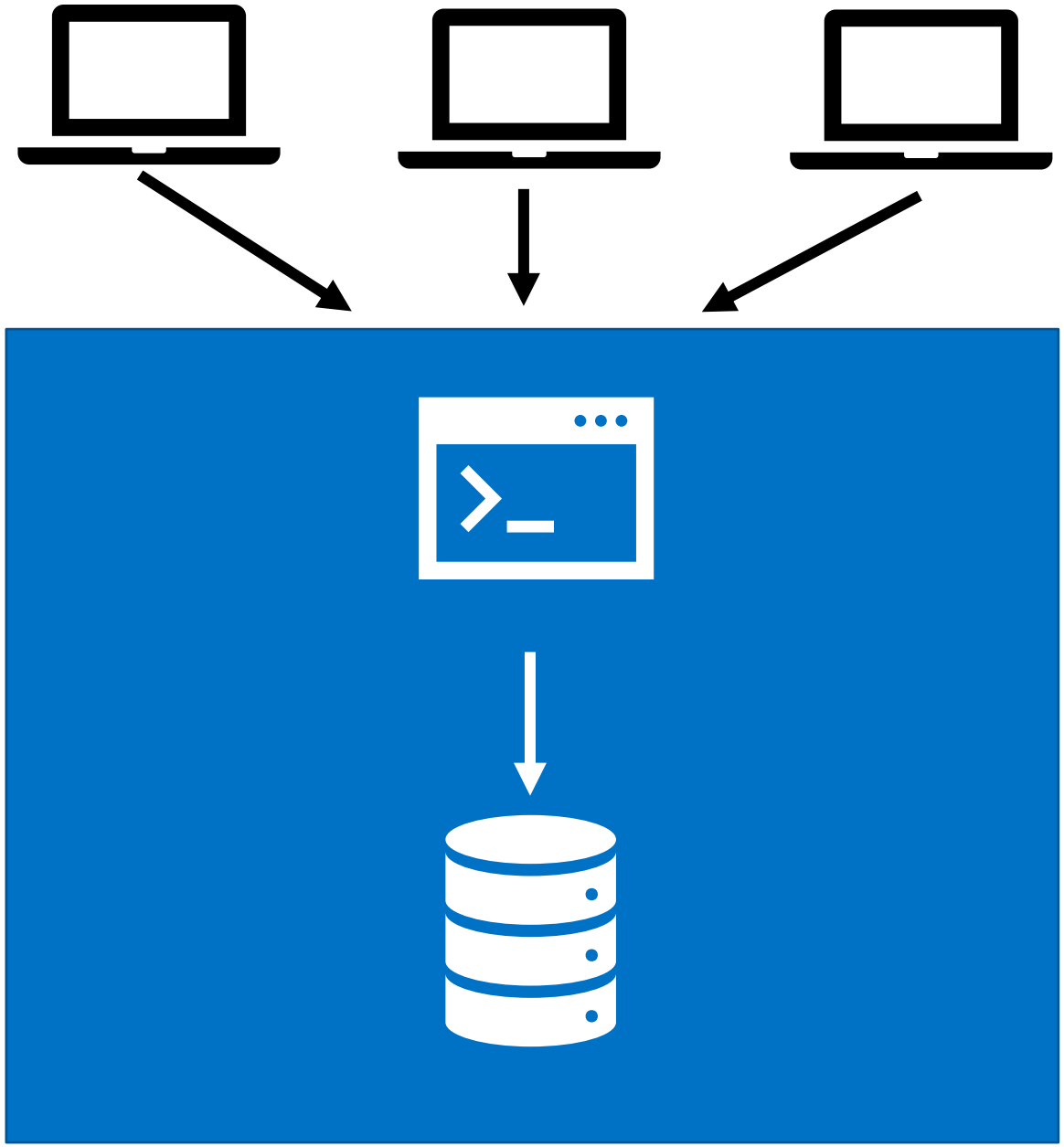
Some Definitions

TRAILHEAD
TECHNOLOGY PARTNERS

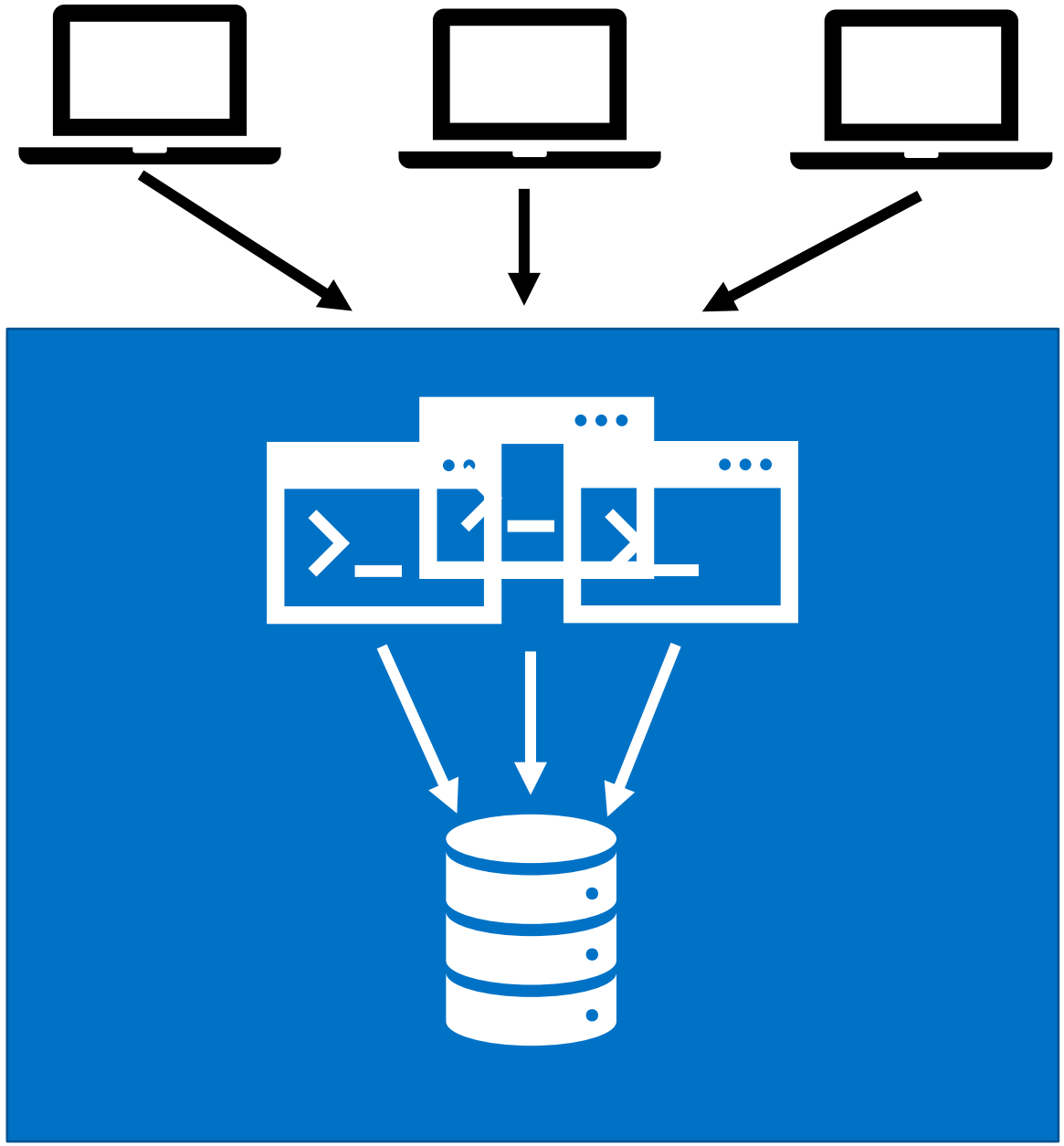

Monolith



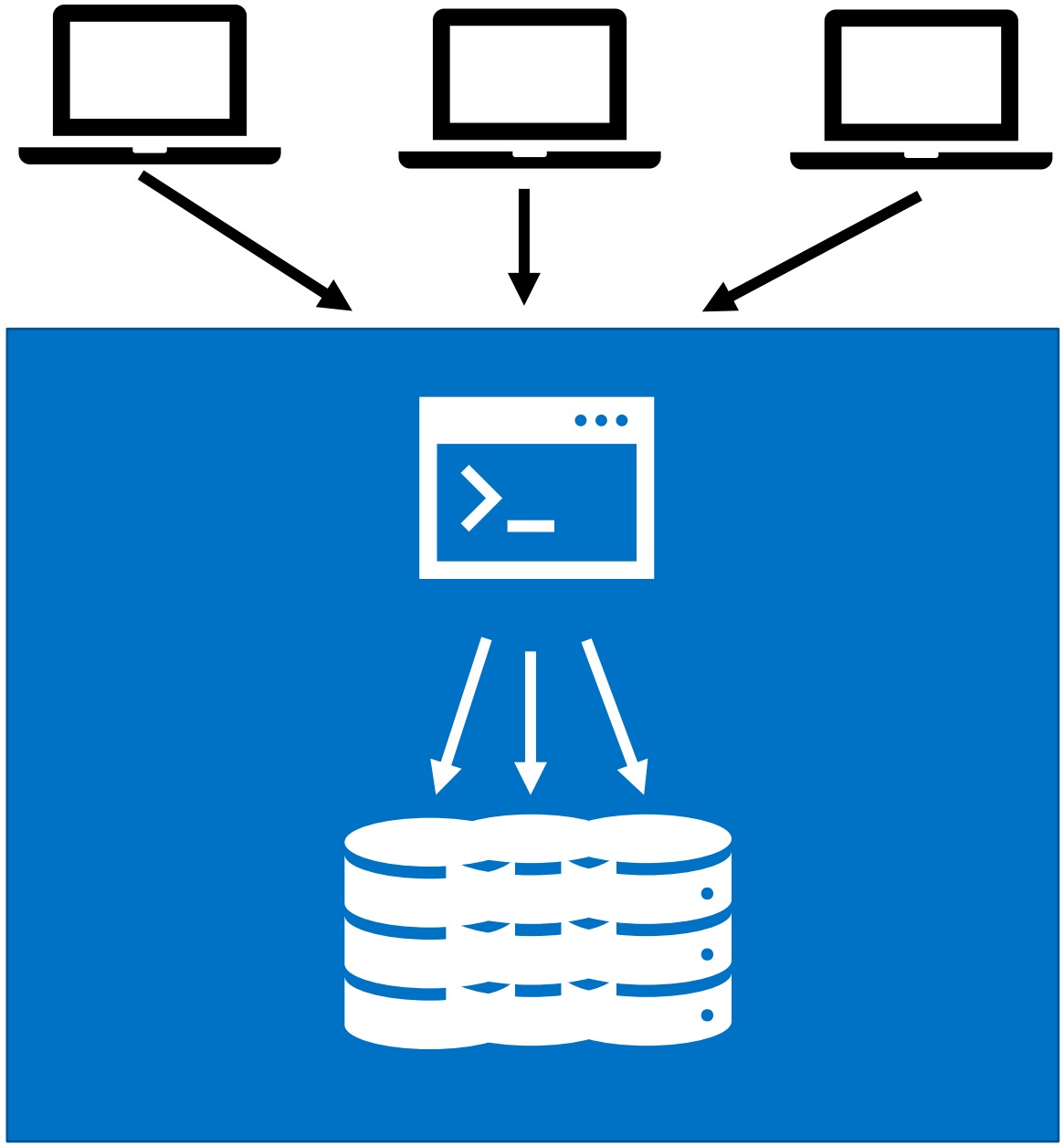
TRAILHEAD
TECHNOLOGY PARTNERS



TRAILHEAD
TECHNOLOGY PARTNERS

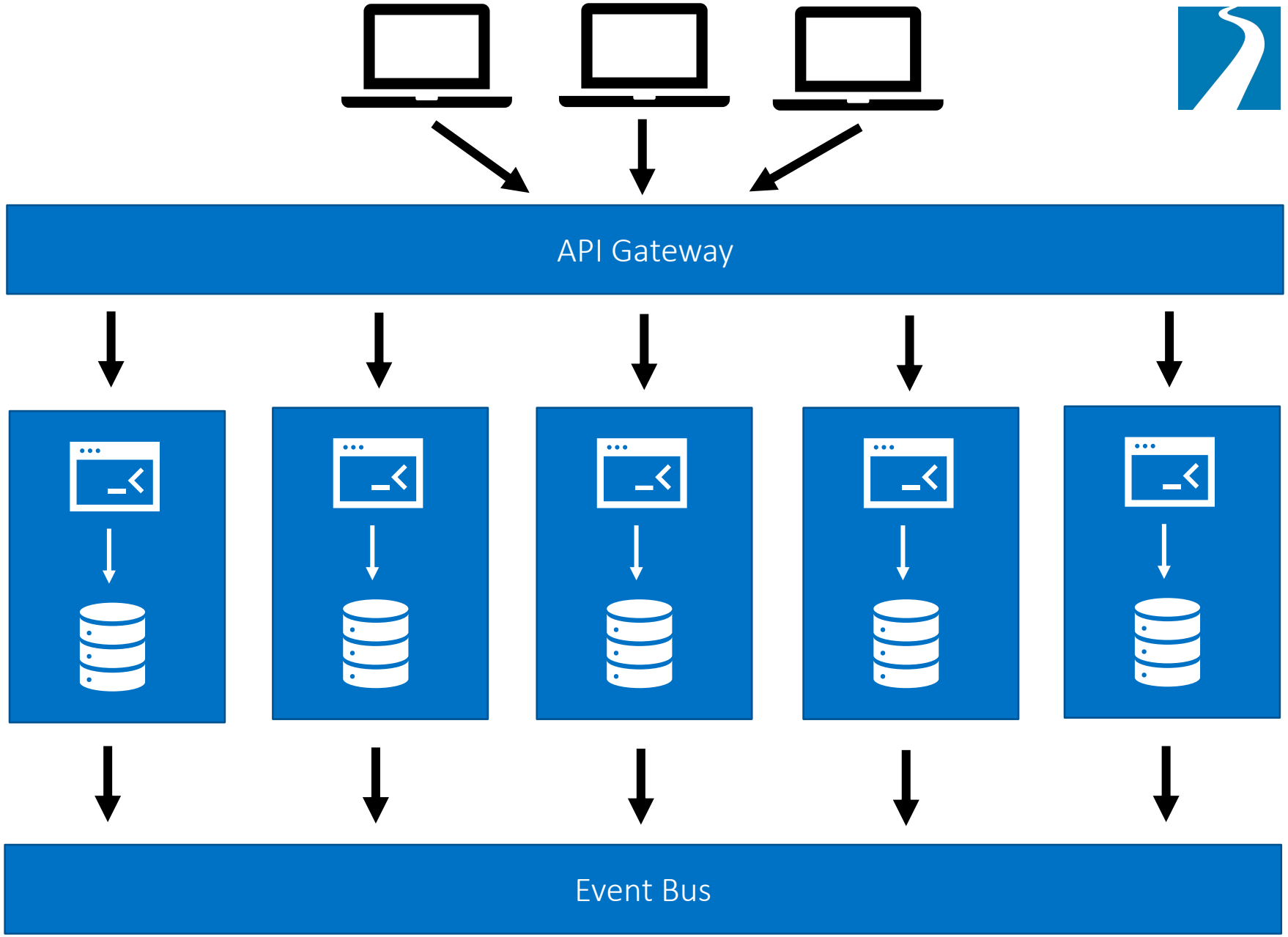


TRAILHEAD
TECHNOLOGY PARTNERS



TRAILHEAD
TECHNOLOGY PARTNERS

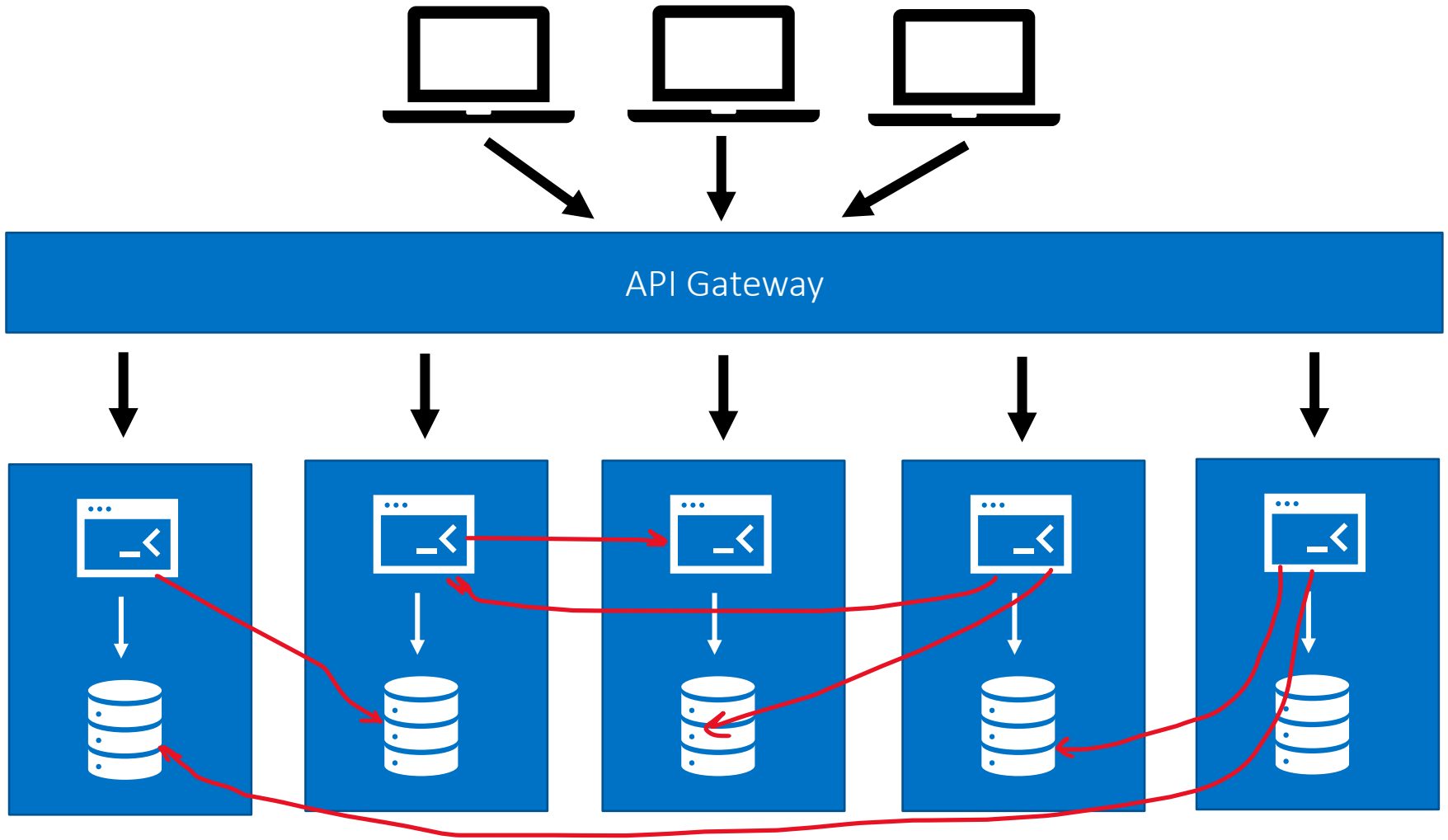
Microservices



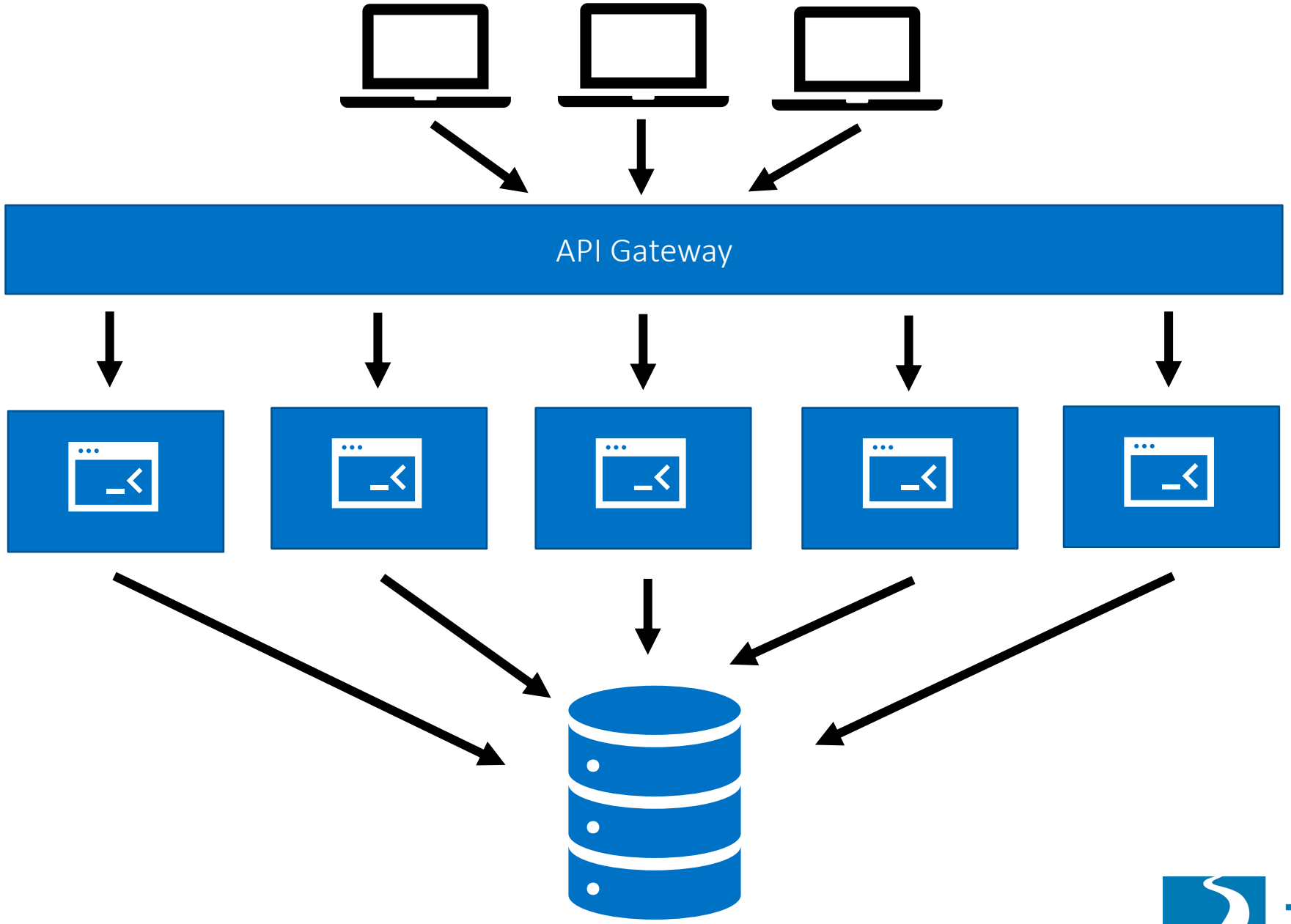
TRAILHEAD
TECHNOLOGY PARTNERS

Distributed Monolith

Distributed Monolith

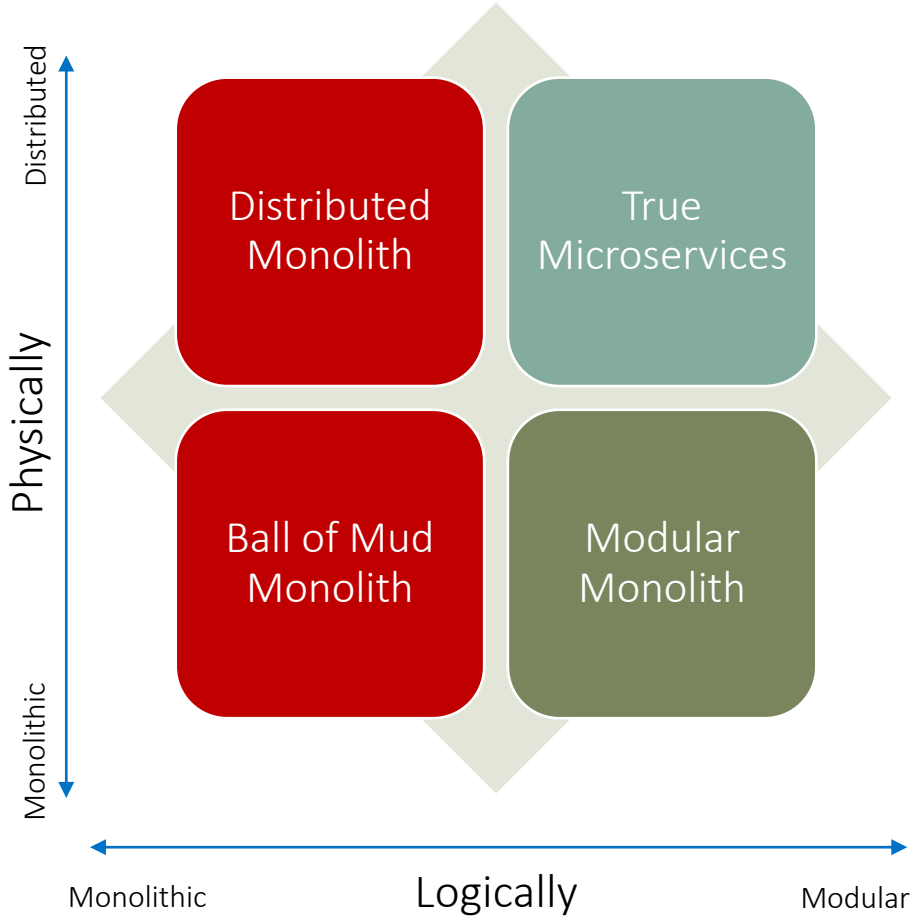


TRAILHEAD
TECHNOLOGY PARTNERS



TRAILHEAD
TECHNOLOGY PARTNERS

Good and Bad Monoliths



TRAILHEAD
TECHNOLOGY PARTNERS

12 Most Common Mistakes

Avoid Creating a “Monster”

TRAILHEAD
TECHNOLOGY PARTNERS

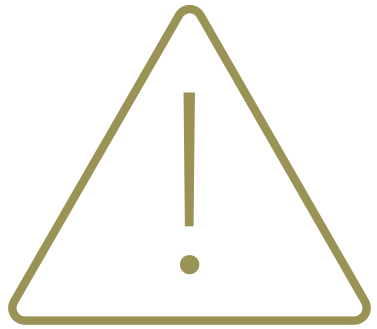
Assuming Microservices are Always Better

Problem #1

First Rule of Microservices: Don't Use Microservices

Have a "Really Good Reason" – Sam Newman

Monoliths aren't inherently bad



Microservices are *hard*

TRAILHEAD
TECHNOLOGY PARTNERS

Some Good Reasons to Microservice

A Need to Independently Deploy New Functionality with Zero Downtime

A Need to Isolate Specific Data and Data Processing Through Data Partitioning

A Need to Enable a High Degree of Team Autonomy

Source: https://blog.dreamfactory.com/when-to-use-microservices-sam-newman-and-martin-fowler-share-their-knowledge

TRAILHEAD
TECHNOLOGY PARTNERS

The Big Trade Off

Microservices

Monoliths

TRAILHEAD
TECHNOLOGY PARTNERS

Tight Coupling of Services

Problem #2

TRAILHEAD
TECHNOLOGY PARTNERS

Easy To Tightly Couple Accidentally

Synchronous calls
(time coupling)

Shared message
definitions

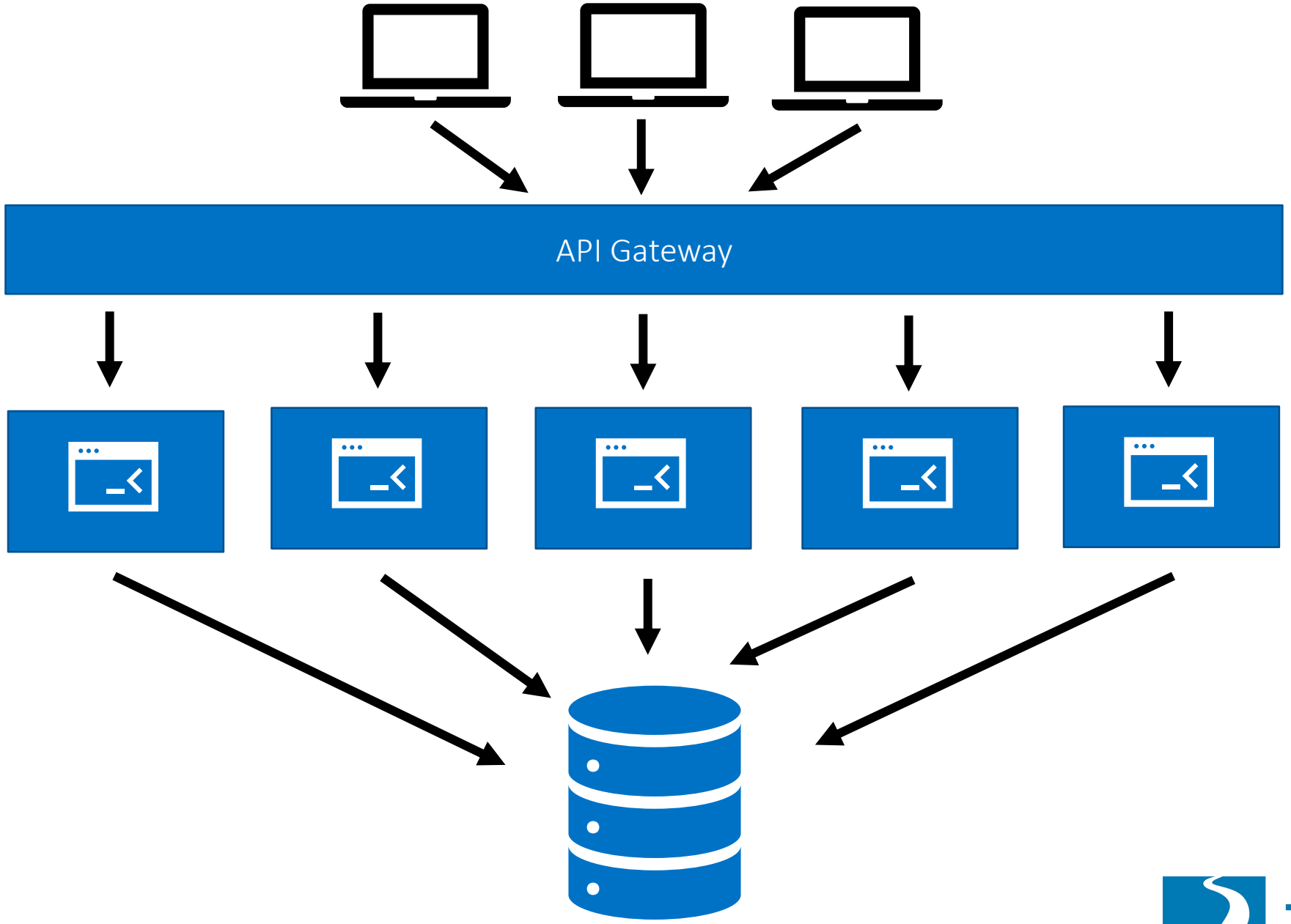
Shared object models
(DTOs, models)

Shared helper classes

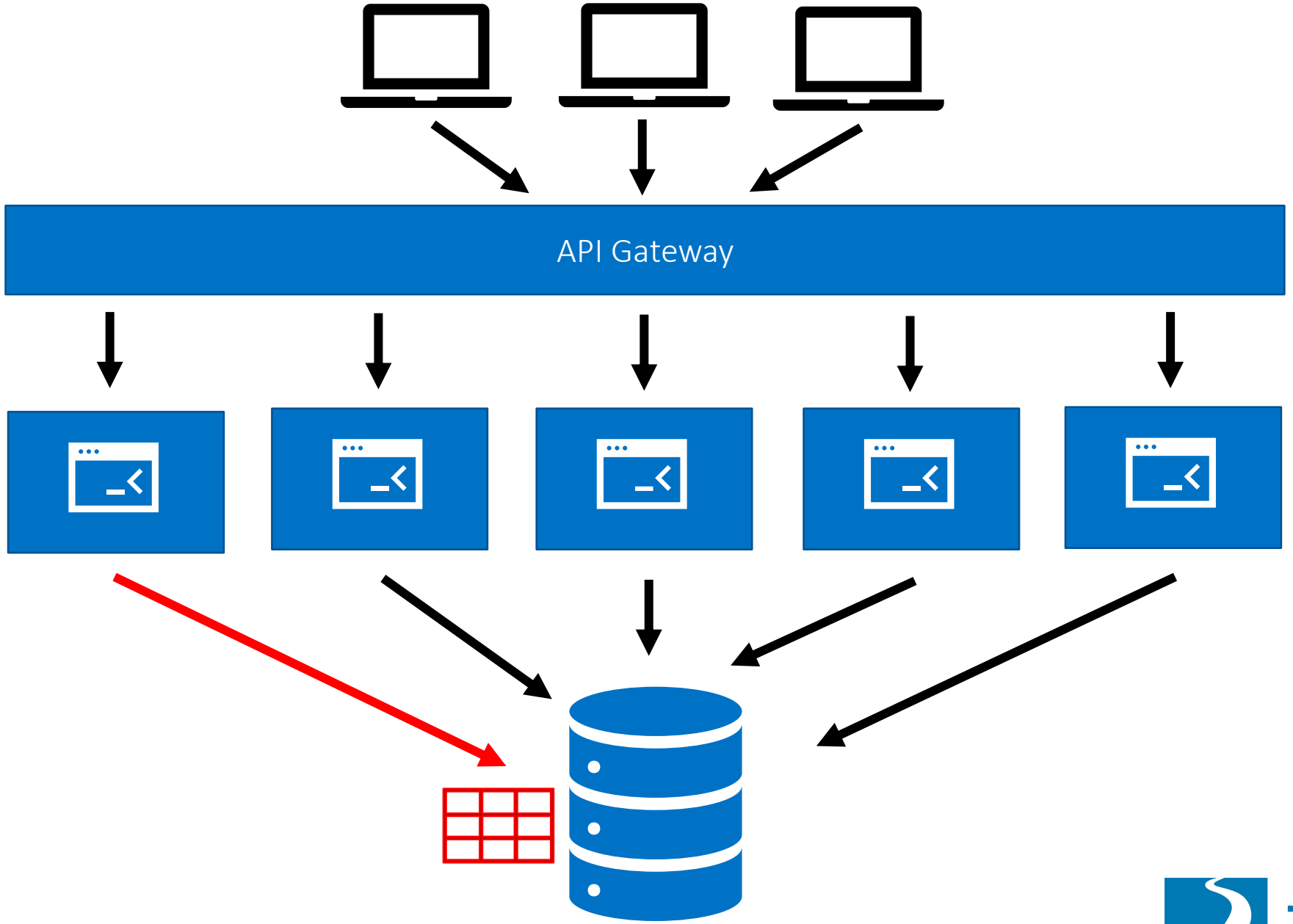
TRAILHEAD
TECHNOLOGY PARTNERS

Shared Data Store or Models

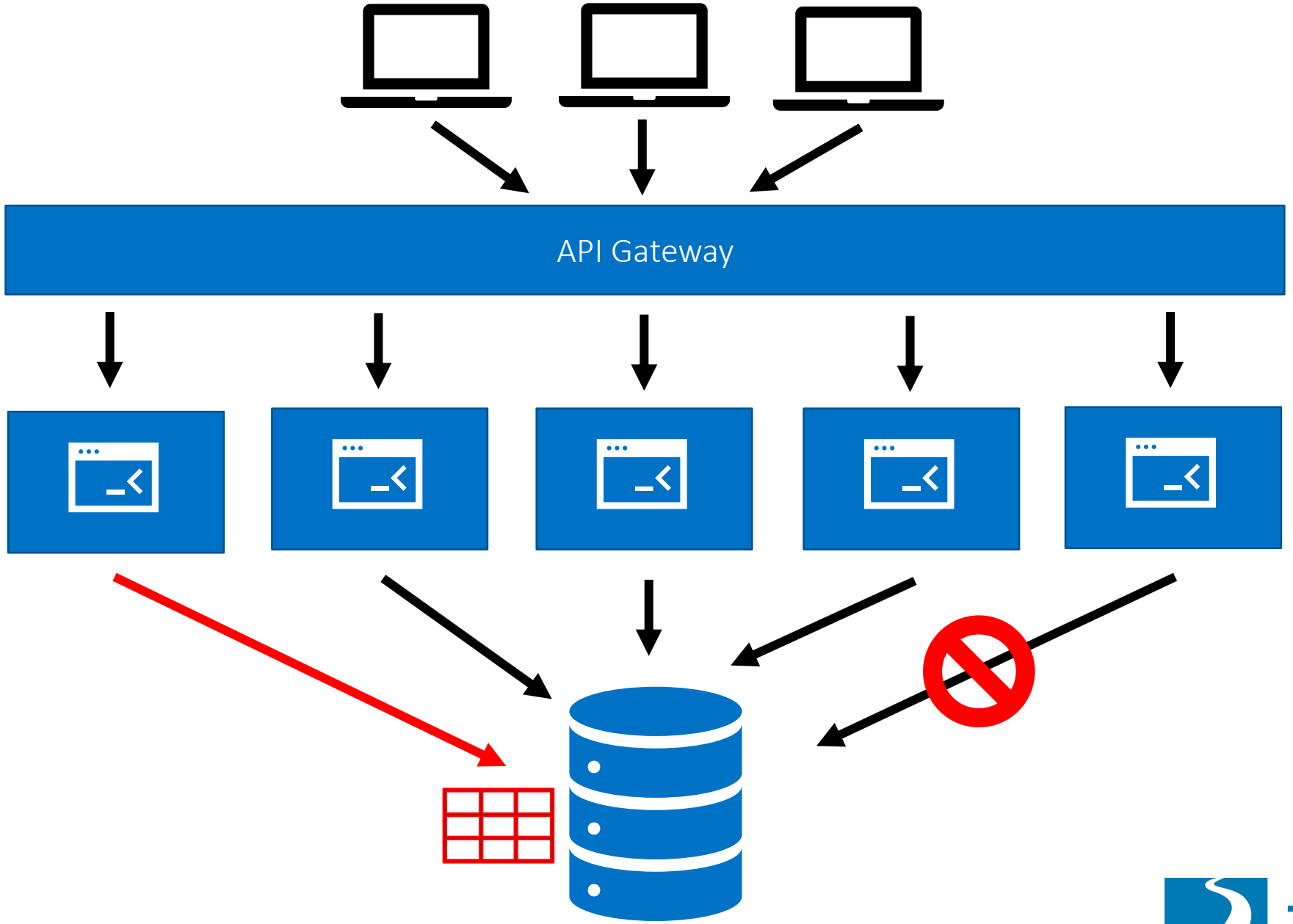
Problem #3



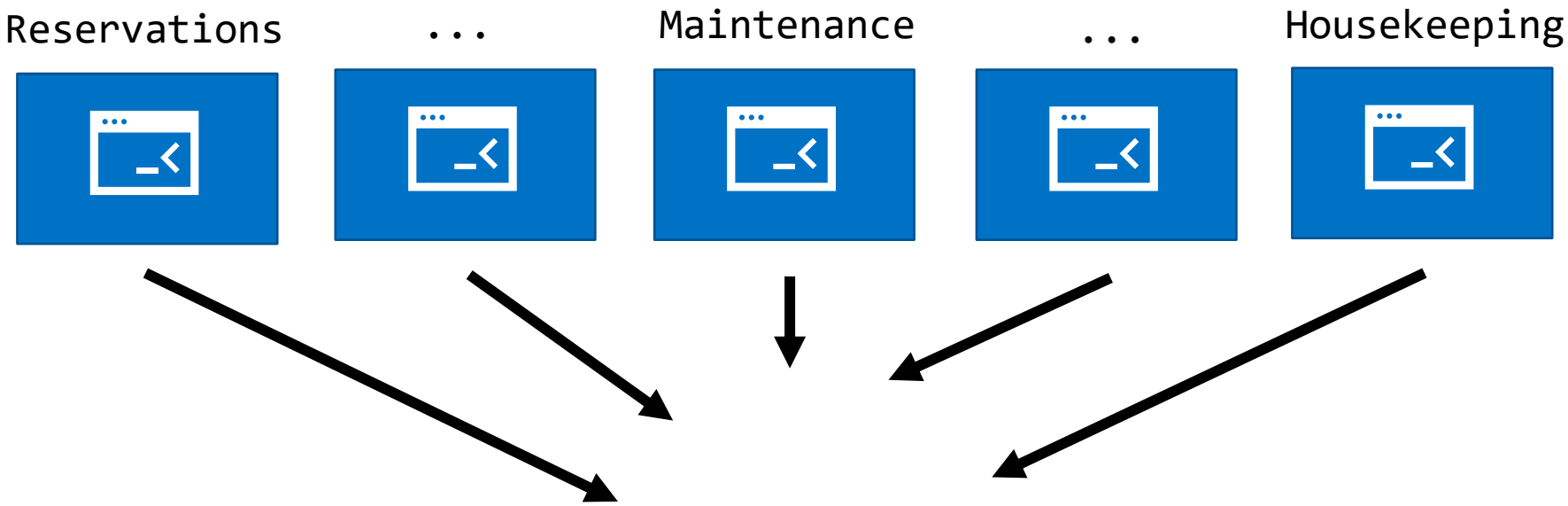
TRAILHEAD
TECHNOLOGY PARTNERS



TRAILHEAD
TECHNOLOGY PARTNERS

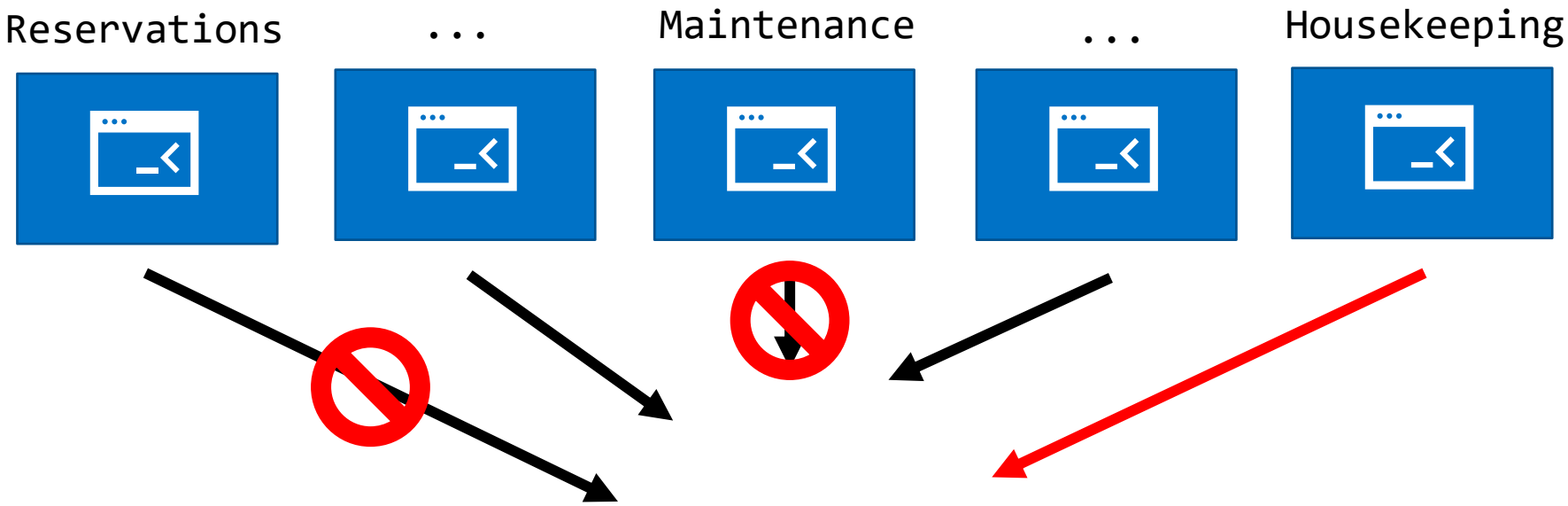


TRAILHEAD
TECHNOLOGY PARTNERS



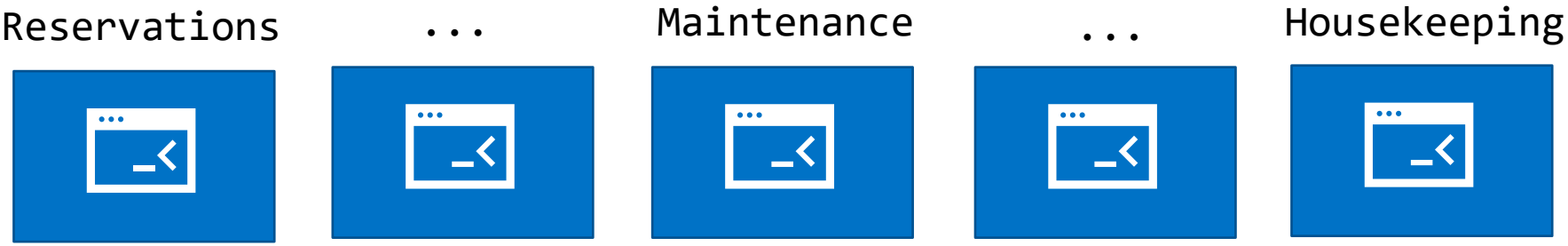

```
HotelRoom {
  CurrentlyOccupied : bool
  NumberofBeds : int
  MinutesToClean : float
  RepairHistory: [
    { RepairDate: ... },
    { RepairDate: ... },
  ]
}
```

TRAILHEAD
TECHNOLOGY PARTNERS




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

TRAILHEAD
TECHNOLOGY PARTNERS




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

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

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

TRAILHEAD
TECHNOLOGY PARTNERS

Microservices That Are Too Big

Problem #4

Domain Driven Development

Domain
 Subdomain
  Bounded Context

Domain Driven Development

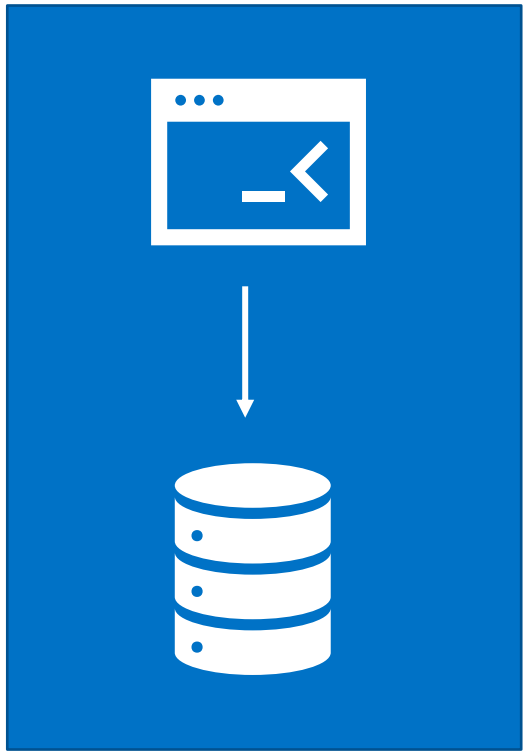
Domain
 Subdomain
  Bounded Context

Smallest possible microservices without chatty communication between services

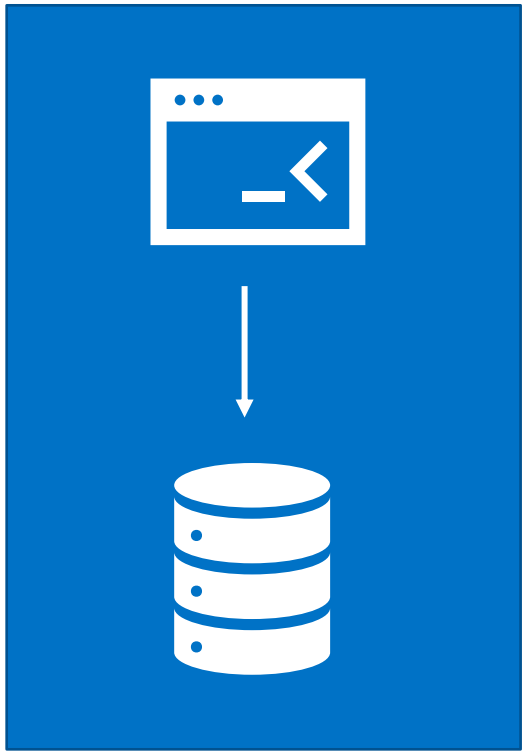
Microservices That Are Too Small

Problem #5

User Log In

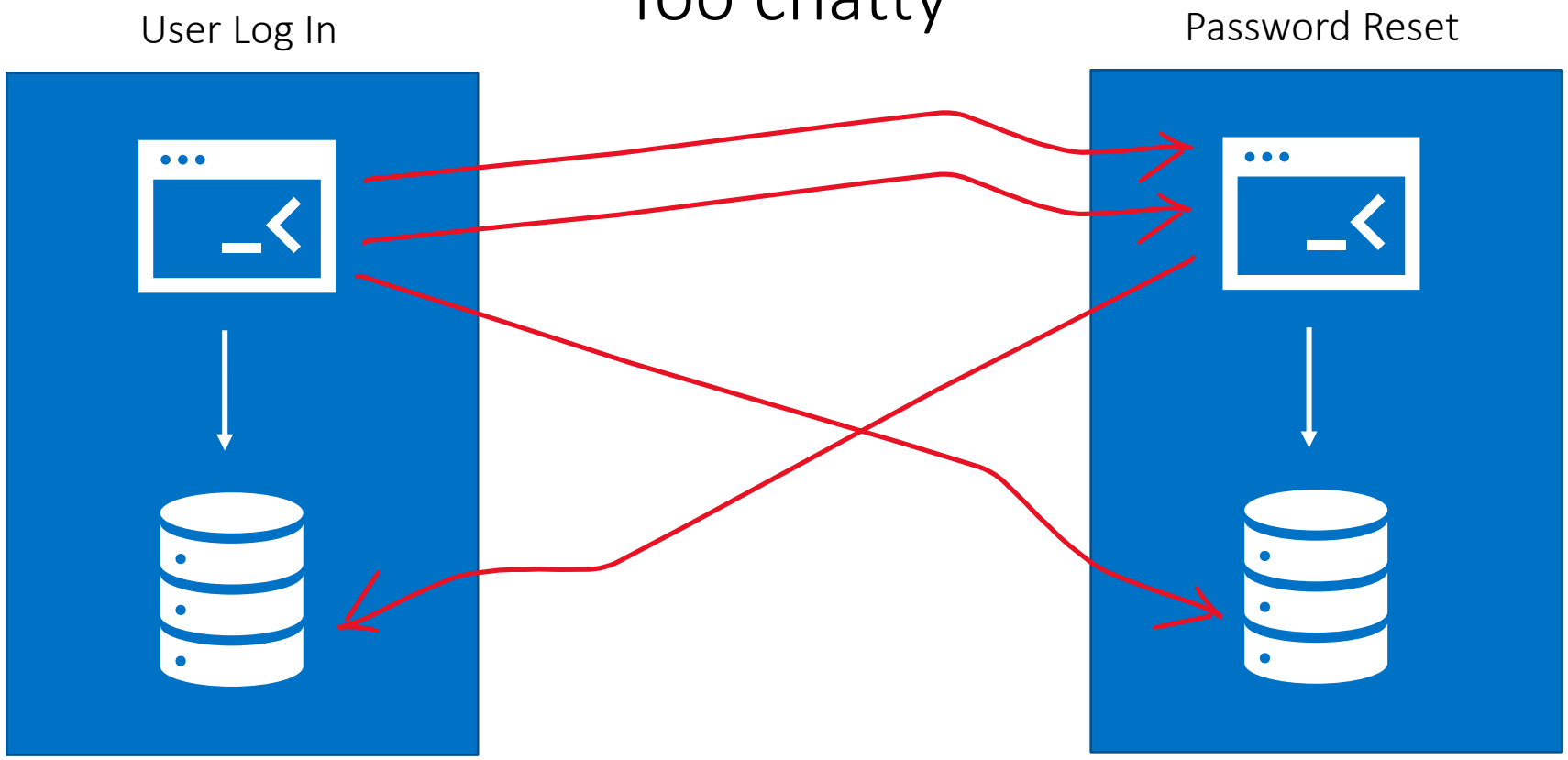


Password Reset



TRAILHEAD
TECHNOLOGY PARTNERS

Too chatty



TRAILHEAD
TECHNOLOGY PARTNERS

Starting from Scratch

Problem #6

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 Approaches: Monolith to Microservices

Big bang

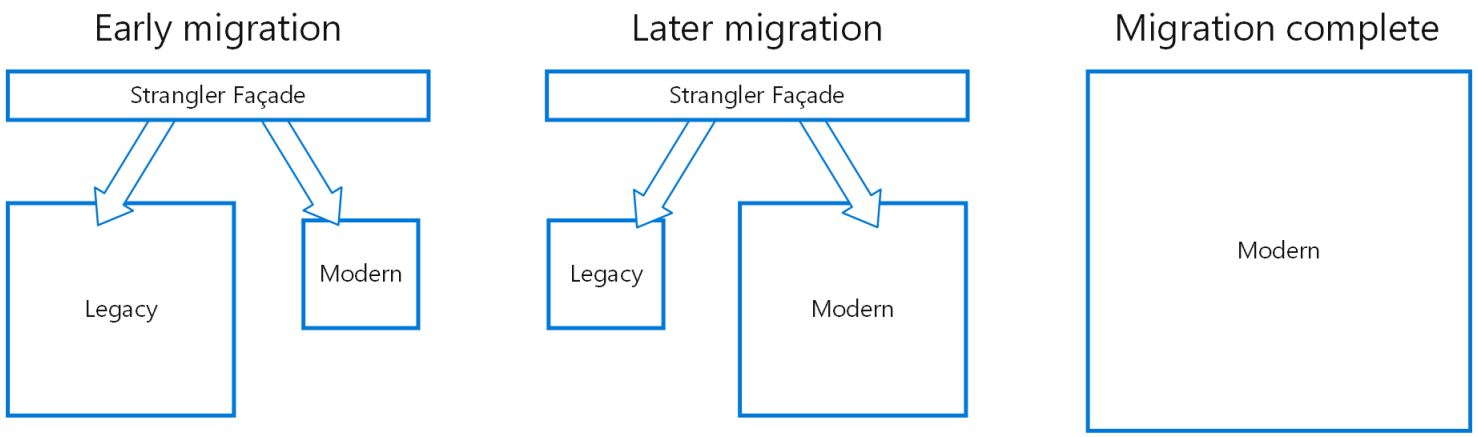
Evolution

“Strangler fig” pattern

Source: https://blog.dreamfactory.com/when-to-use-microservices-sam-newman-and-martin-fowler-share-their-knowledge

TRAILHEAD
TECHNOLOGY PARTNERS

Strangler Fig Pattern



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

TRAILHEAD
TECHNOLOGY PARTNERS

Reminder…If You Give $200, So Will I!

bit.ly/indy-water

"charity:water is a non-profit organization that provides clean and safe drinking water to people in developing nations. The organization was founded in 2006 and has helped fund 35,000 projects in 27 countries, benefiting over 9.5 million people.."
- Wikipedia

"4/4 Stars"
- CharityNavigator.org

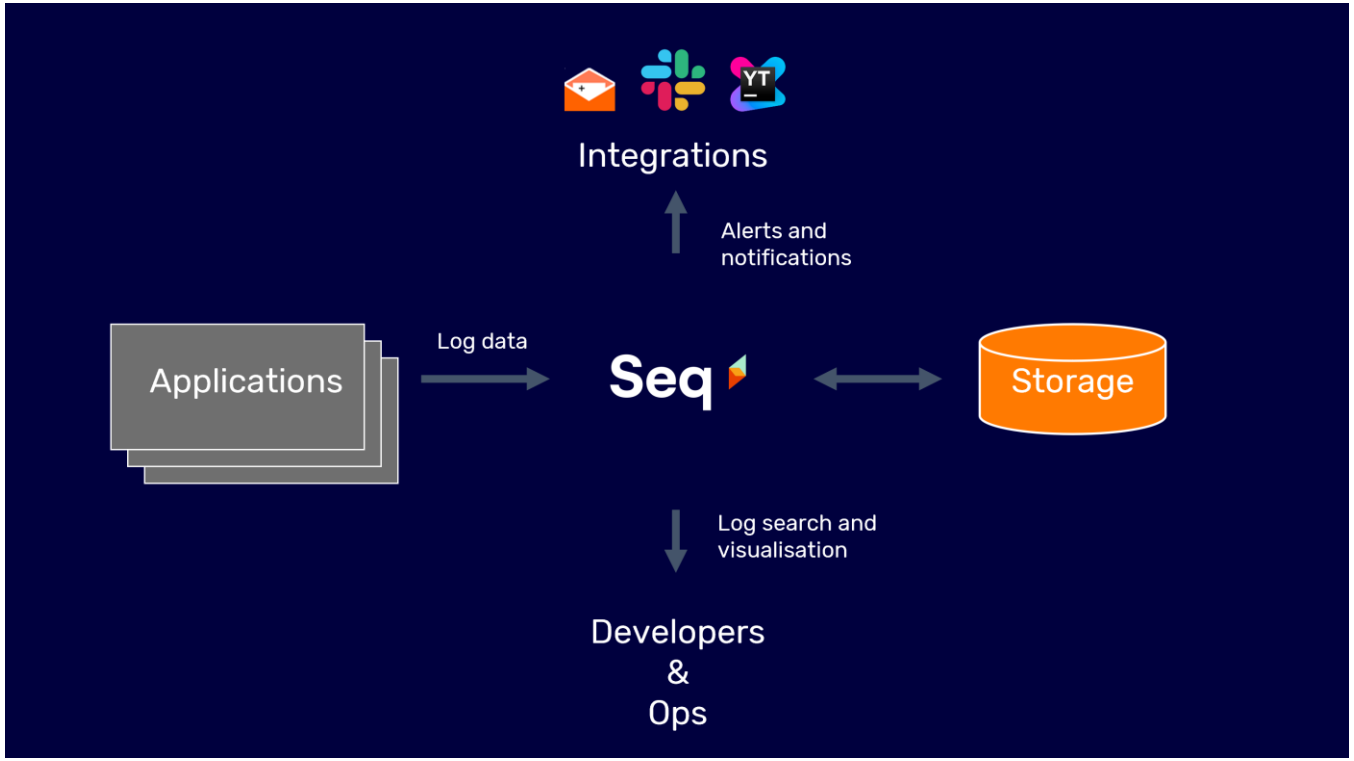
charity: water

Decentralized Logging

Problem #7

Debugging Distributed Systems Is Hard

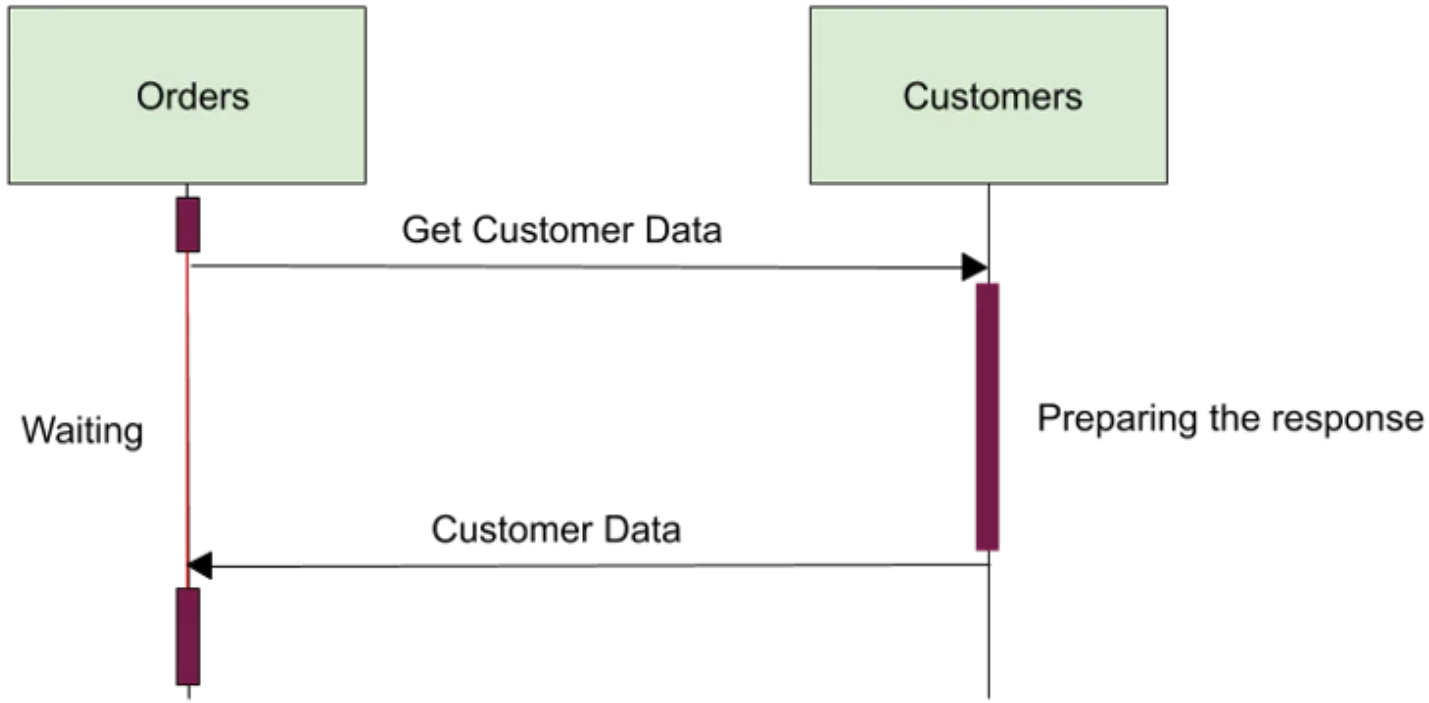
- Centralized without coupling
- Third party solutions like Seq
  - Seq: “Intelligent search, analysis, and alerting server built specifically for modern structured log data”
  - Supports .NET, Java, NodeJS, Ruby, Go, Python, more.
  - Inherently fault tolerant, embraces eventual consistency



TRAILHEAD TECHNOLOGY PARTNERS

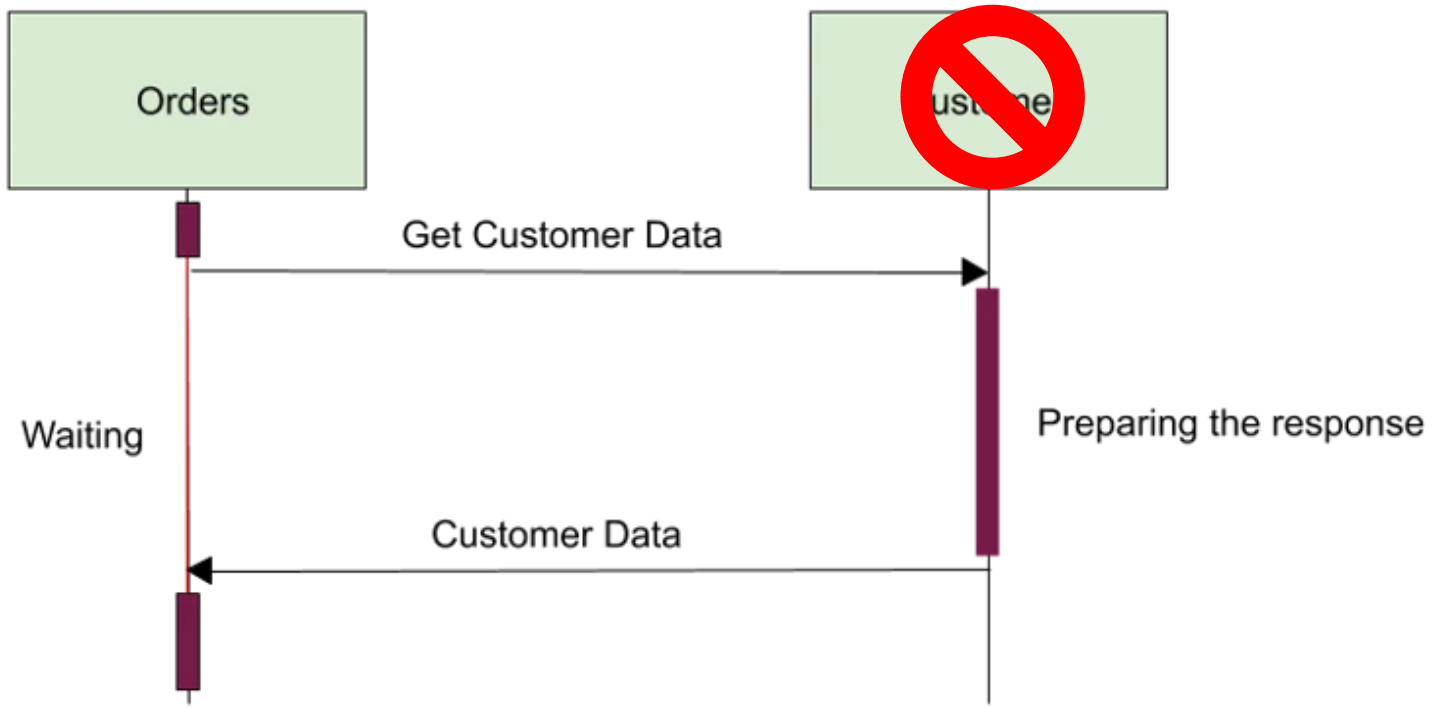
Synchronous Communication

Problem #8



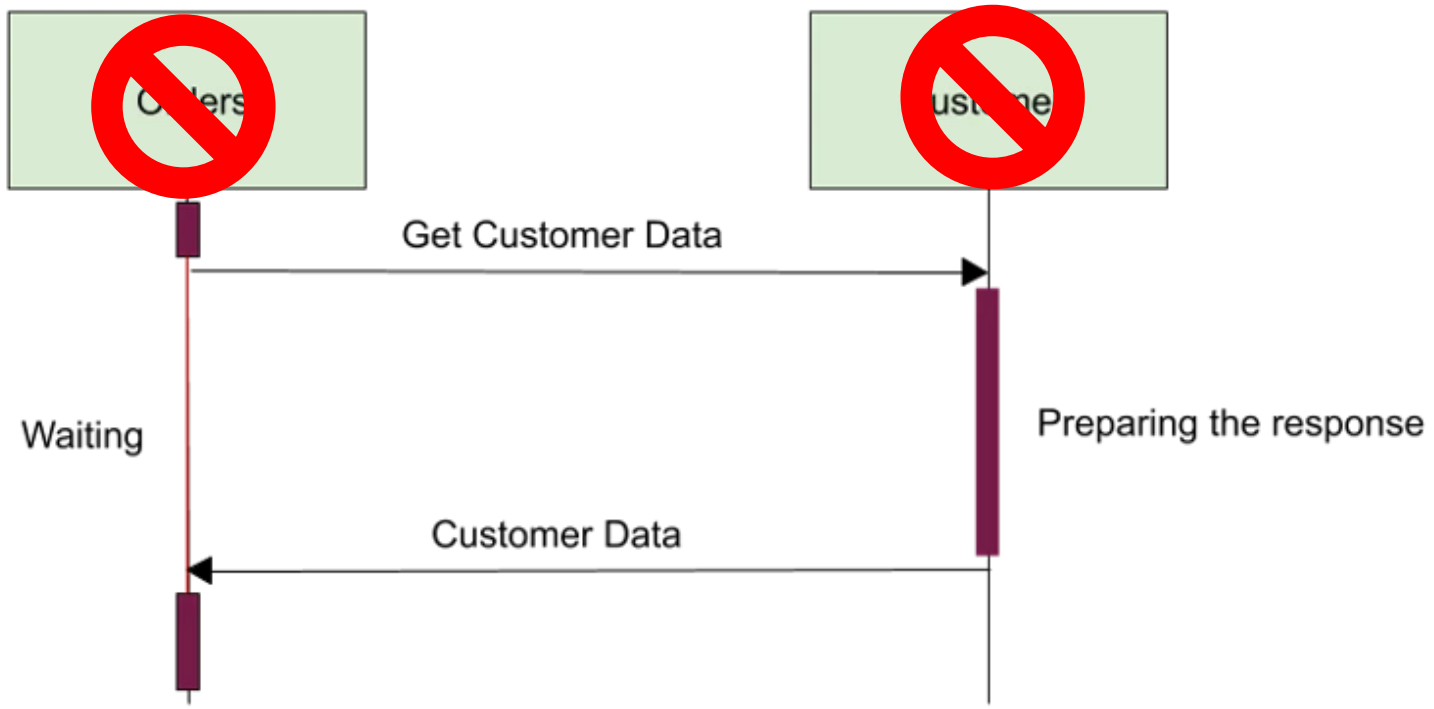
Source: https://www.capitalone.com/tech/software-engineering/how-to-avoid-loose-coupled-microservices/

TRAILHEAD
TECHNOLOGY PARTNERS



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

TRAILHEAD
TECHNOLOGY PARTNERS



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

TRAILHEAD
TECHNOLOGY PARTNERS

Breaking Changes to Event Contracts

Problem #9

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 #10

Prerequisite: Automated Build and Release

Time consuming

Prone to human error

TRAILHEAD
TECHNOLOGY PARTNERS

Mismatched Team Organization

Problem #11

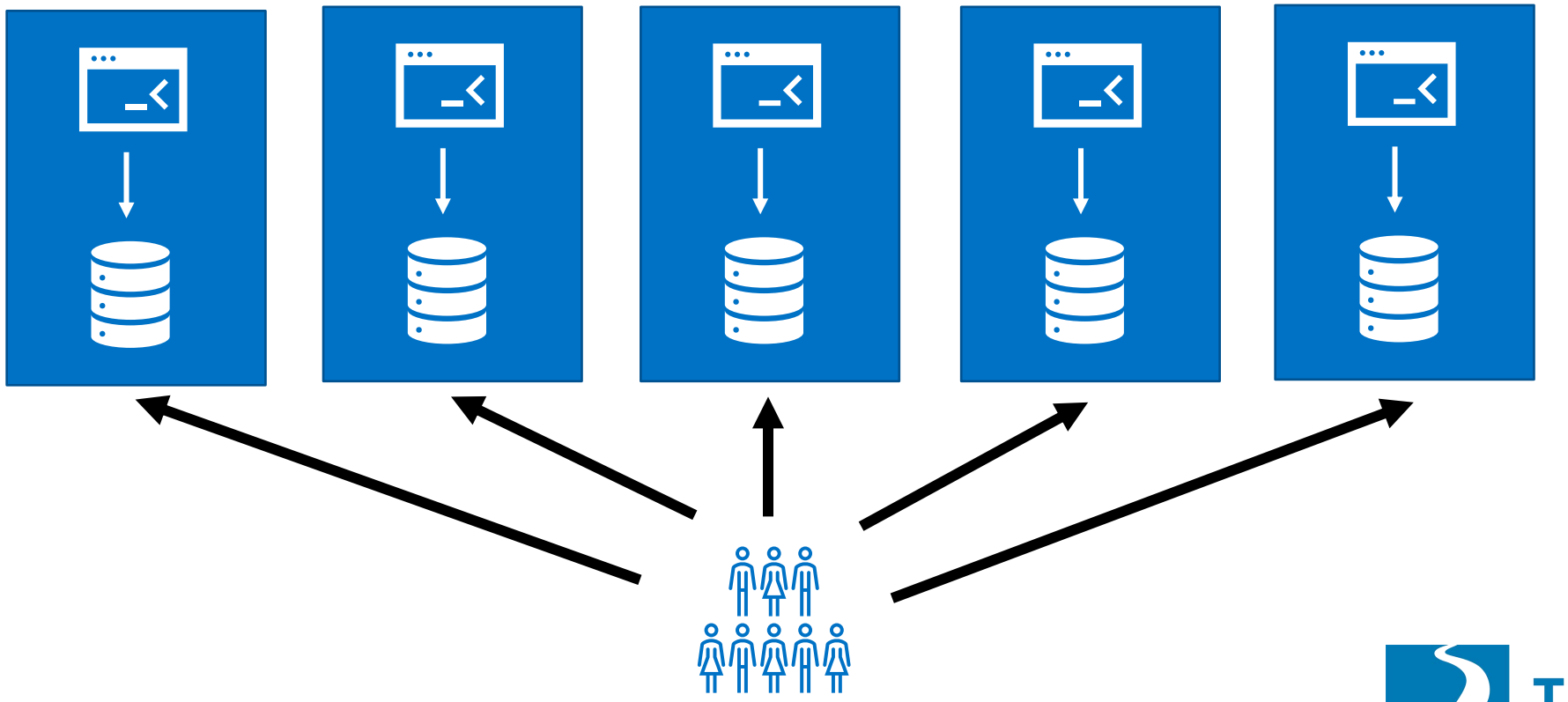
Conway's Law

"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

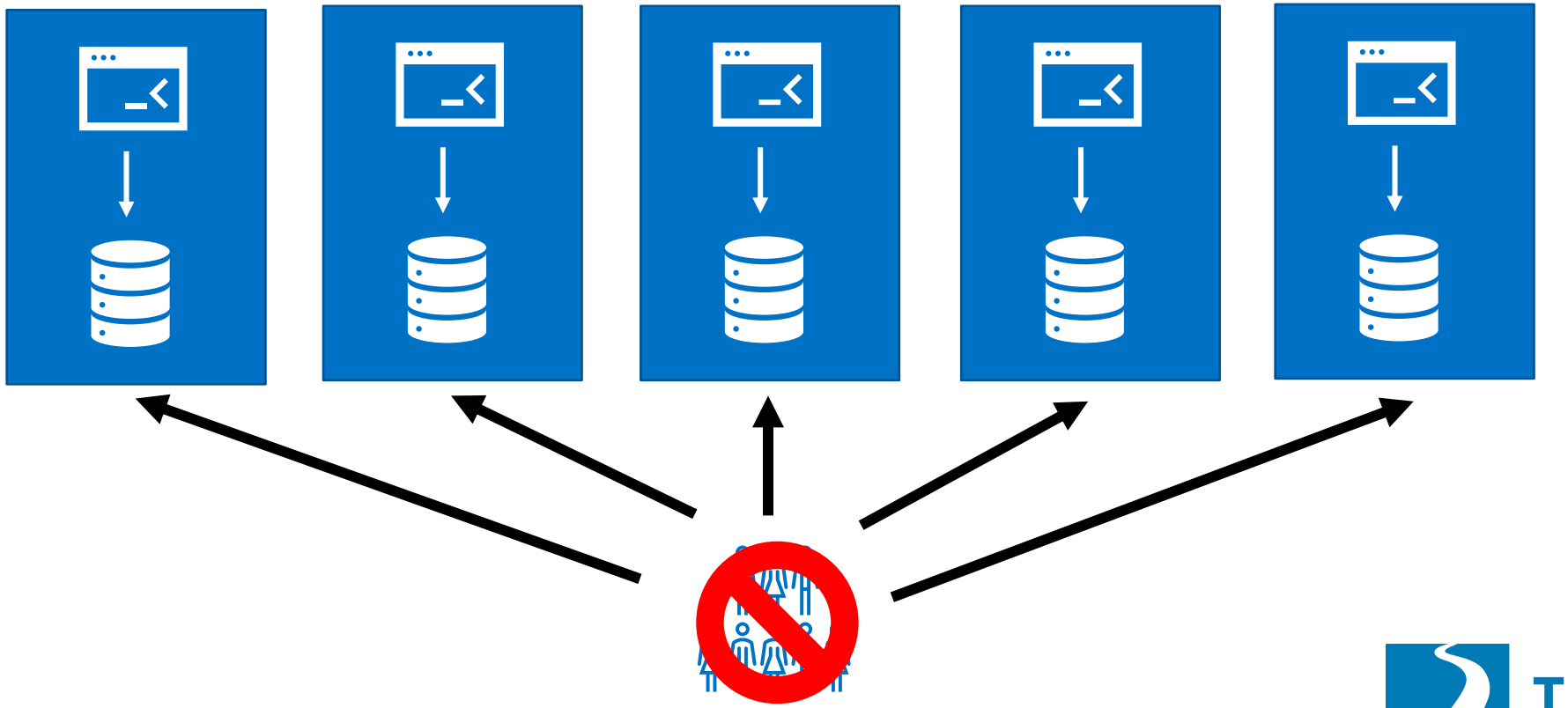
IOW: if you have four groups working on a compiler, you'll get a 4-pass compiler.

Single Team



TRAILHEAD
TECHNOLOGY PARTNERS

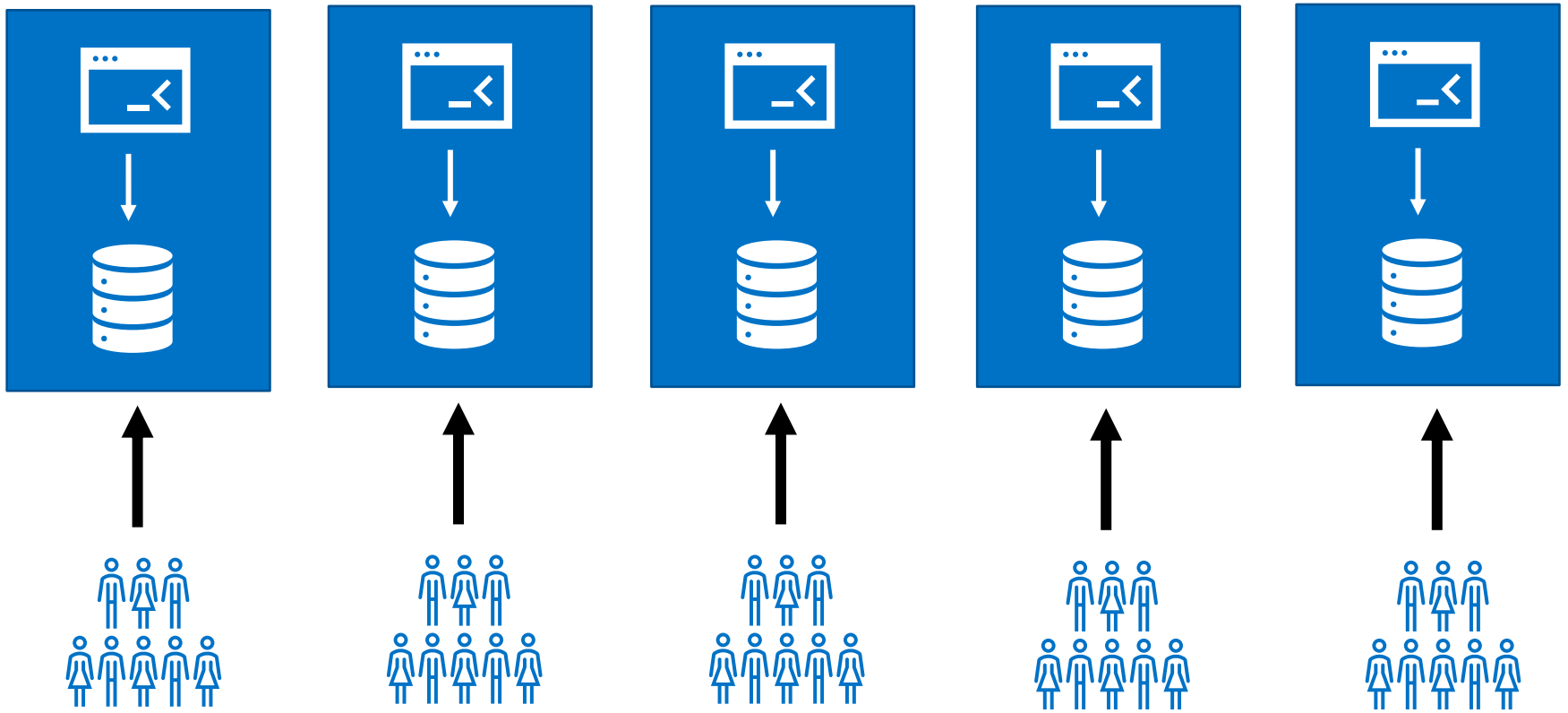
Single Team



TRAILHEAD
TECHNOLOGY PARTNERS

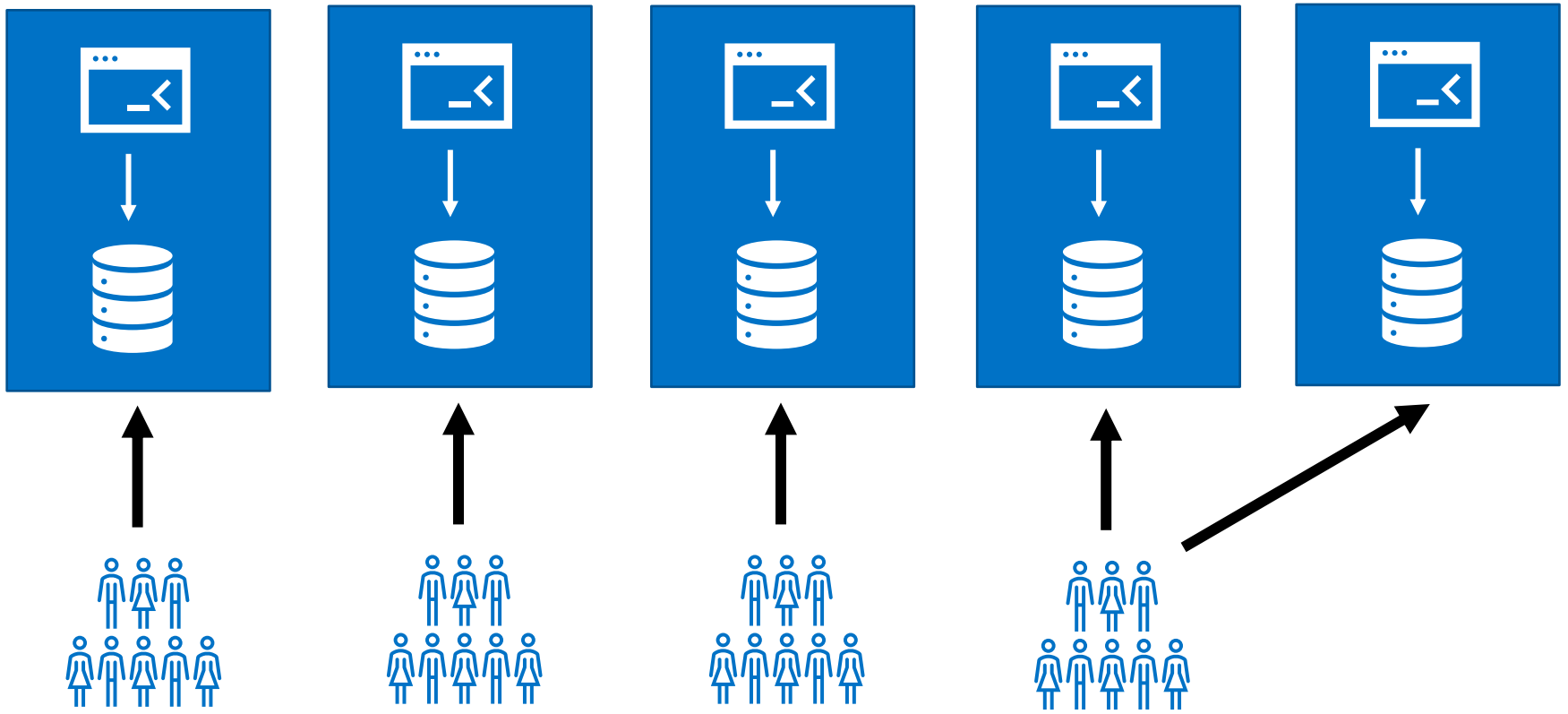
TRAILHEAD
TECHNOLOGY PARTNERS

Team Per Service



TRAILHEAD
TECHNOLOGY PARTNERS

Team Per Service



Unencapsulated Services

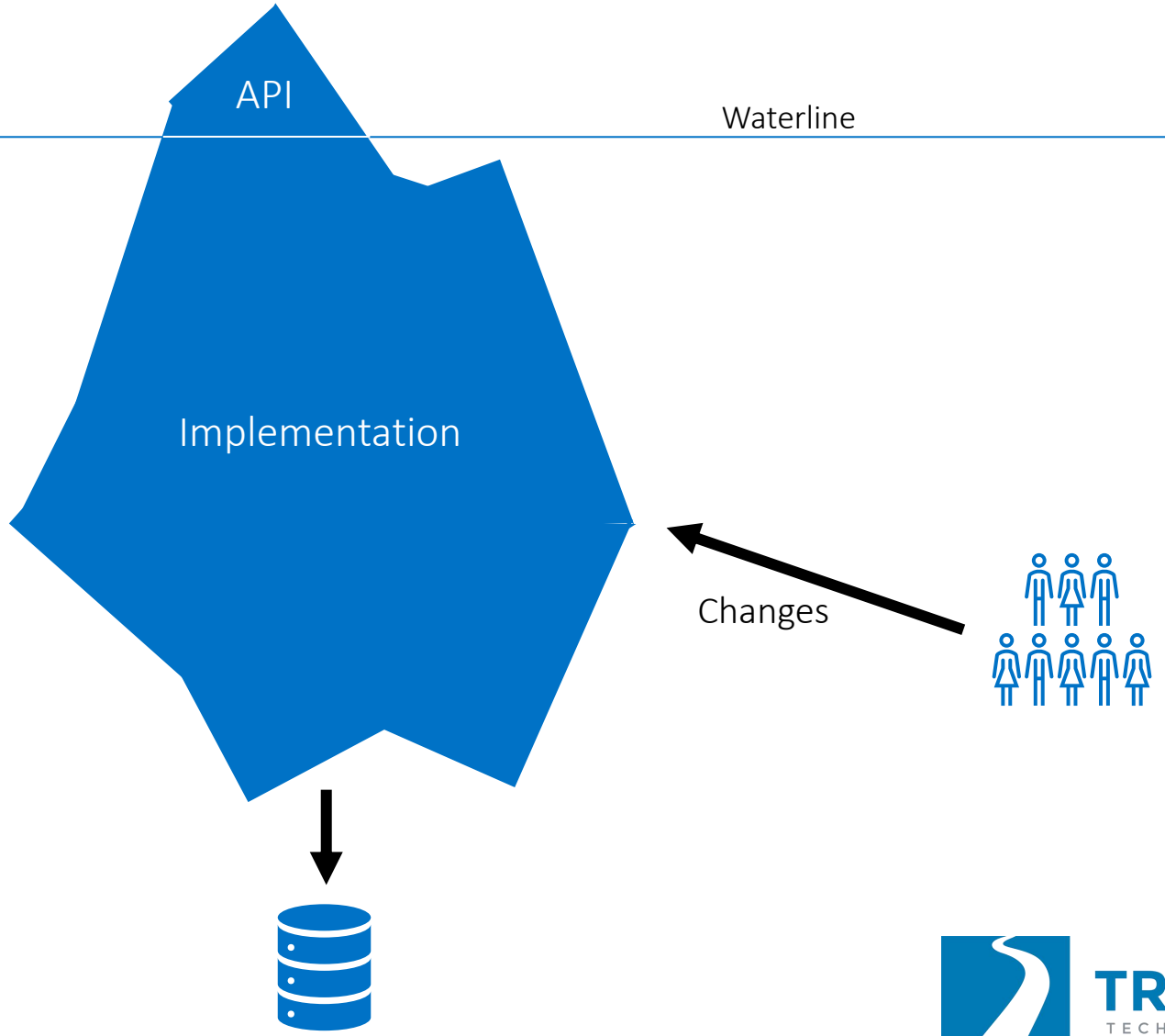
Problem #12

Iceberg Services

Services ENCAPSULATE significant business logic

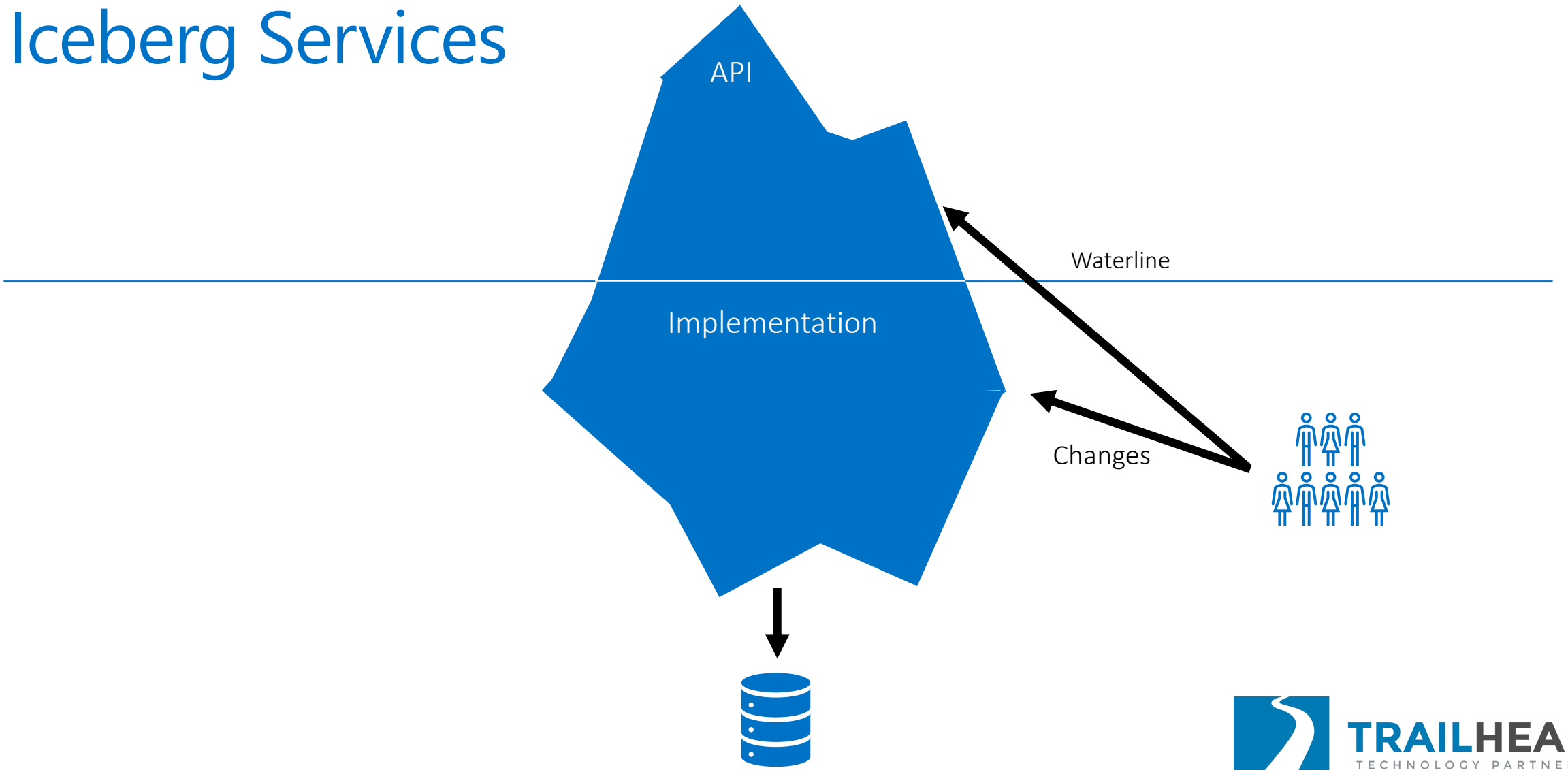
Small, stable API

Large implementation



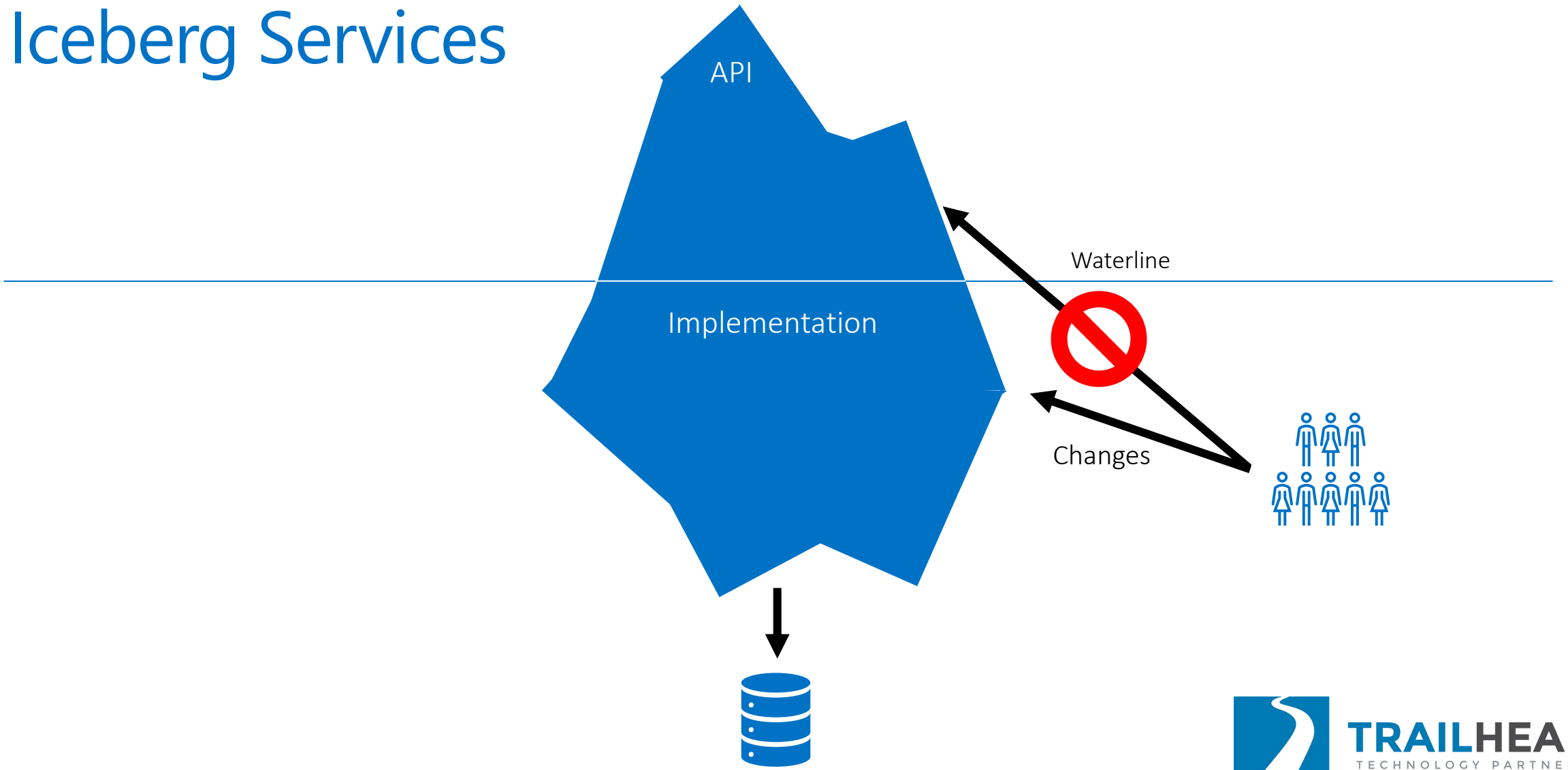
TRAILHEAD
TECHNOLOGY PARTNERS

Iceberg Services



TRAILHEAD
TECHNOLOGY PARTNERS

Iceberg Services



TRAILHEAD
TECHNOLOGY PARTNERS

Summing Up

1. For a lightweight application or single team, a monolithic system often suits better

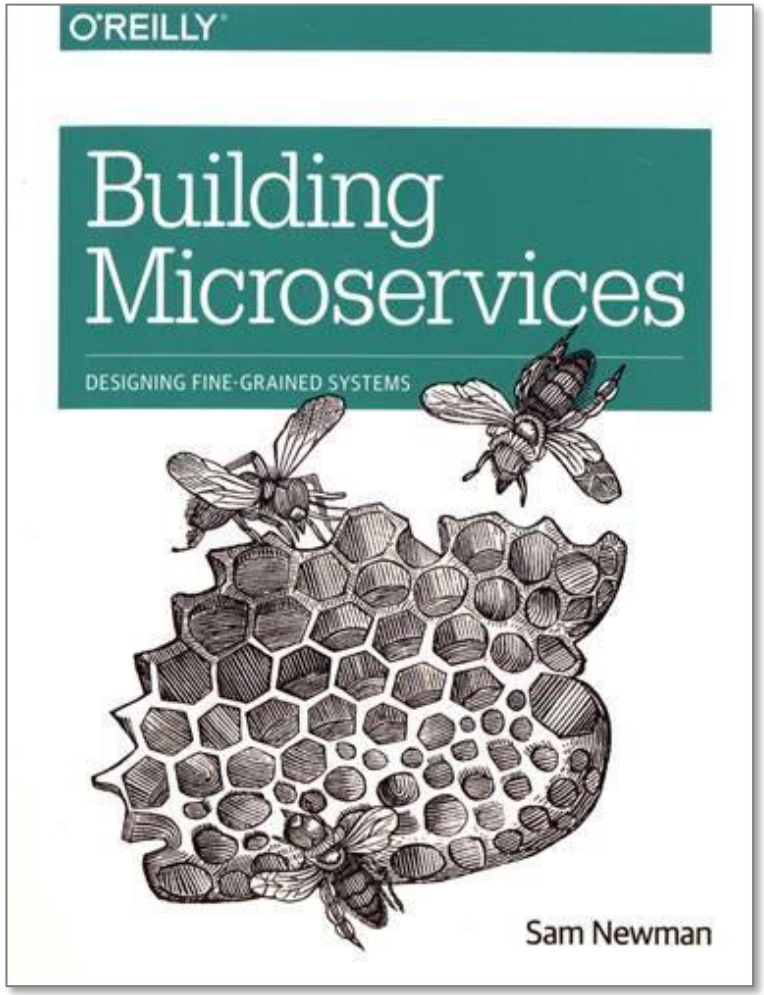
2. For a complex, evolving application with clear domains and separate teams, microservices will be best

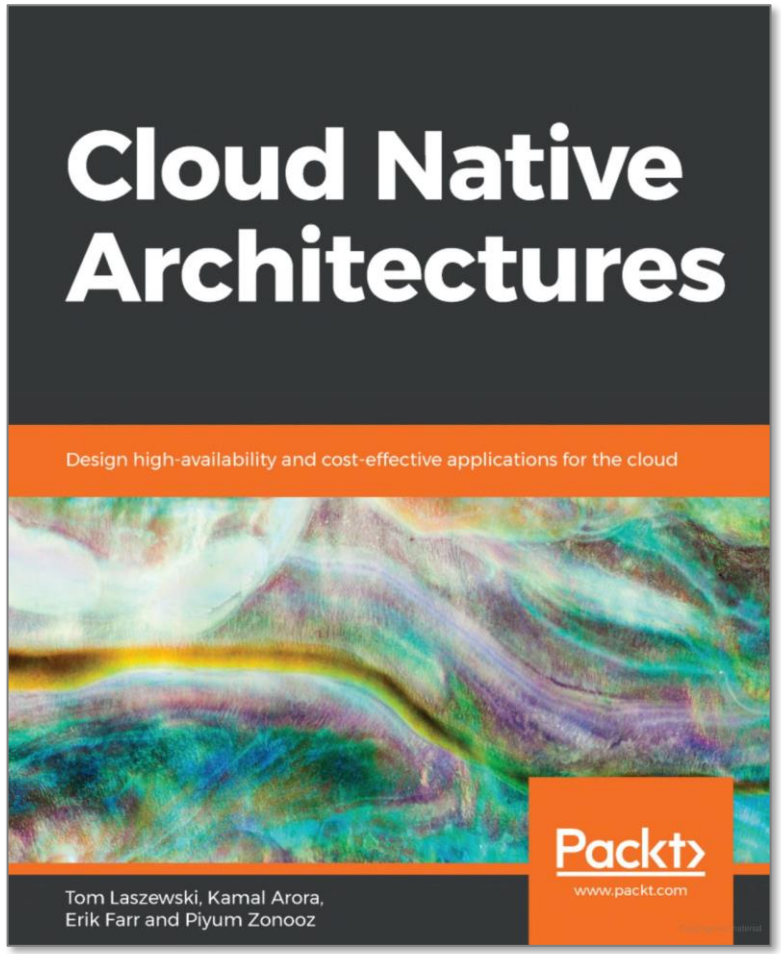
3. Don't try microservices without "a really good reason". Monoliths can be good!

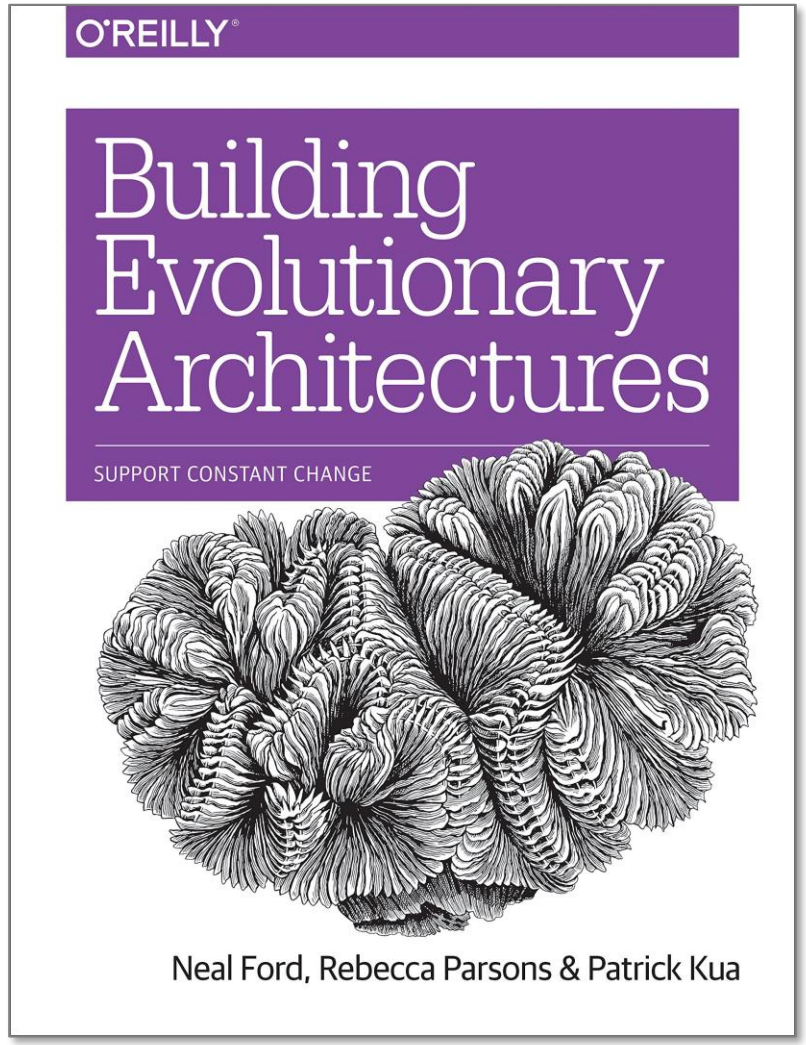
4. Avoid pitfalls of the distributed monolith

TRAILHEAD
TECHNOLOGY PARTNERS

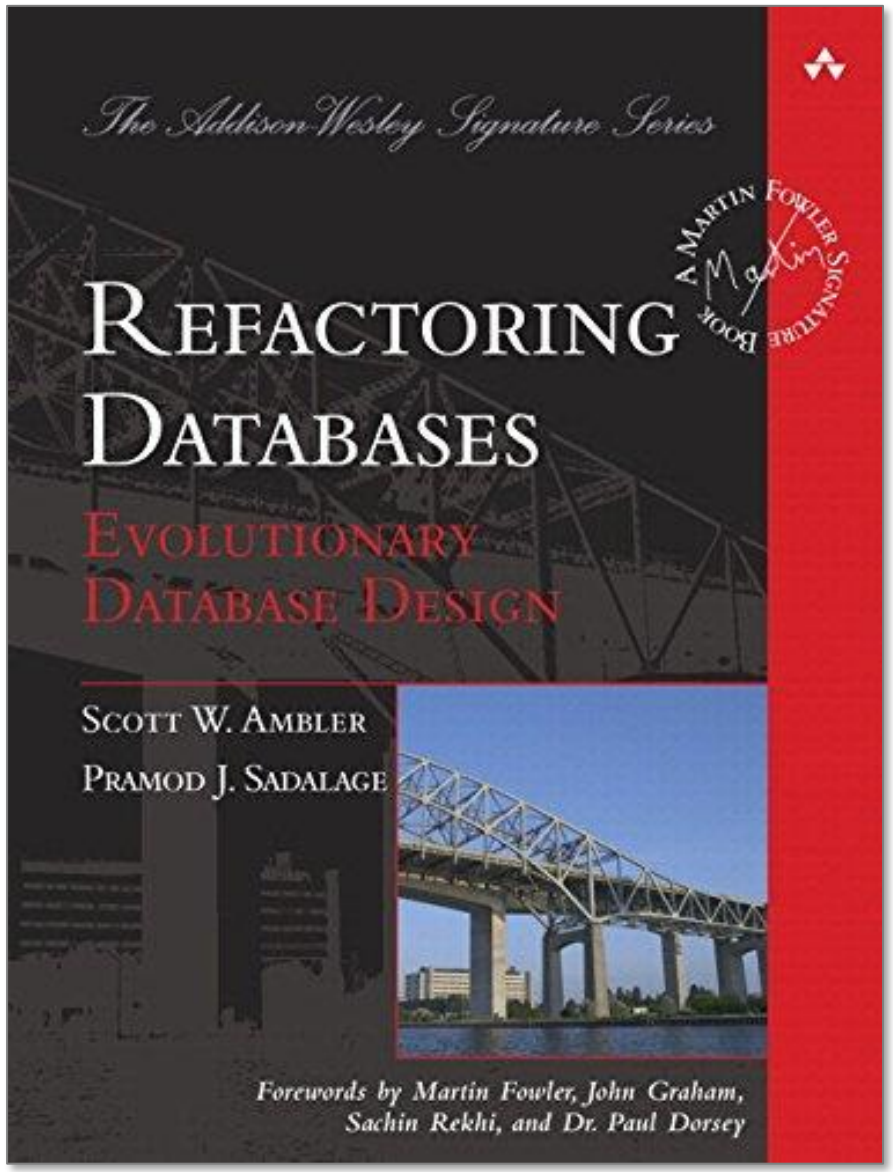
Further Reading

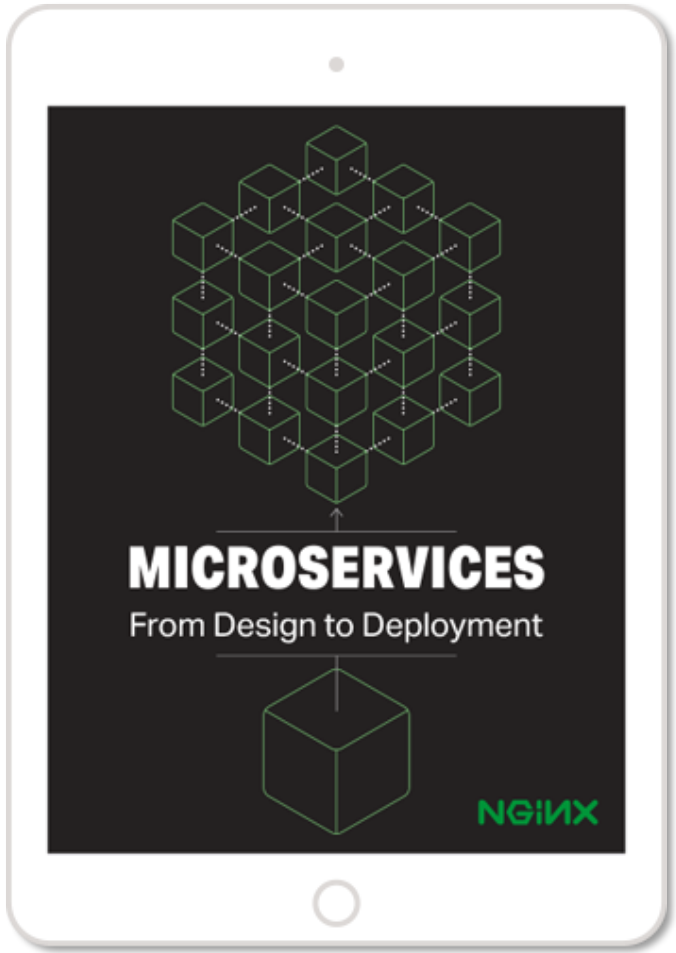






TRAILHEAD
TECHNOLOGY PARTNERS





TRAILHEAD
TECHNOLOGY PARTNERS

Online Resources

https://martinfowler.com/microservices

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

https://docs.microsoft.com/en-us/dotnet/architecture/cloud-native/introduce-eshoponcontainers-reference-app

https://docs.microsoft.com/en-us/dotnet/architecture/microservices/

TRAILHEAD
TECHNOLOGY PARTNERS

Recap

- Definitions
  - Monolith
  - Microservices
  - Distributed Monolith
- 12 Most Common Mistakes
- Further Reading
- Q&A

Last Chance...If You Give $200, So Will I!

bit.ly/indy-water

"charity:water is a non-profit organization that provides clean and safe drinking water to people in developing nations. The organization was founded in 2006 and has helped fund 35,000 projects in 27 countries, benefiting over 9.5 million people.."
- Wikipedia

"4/4 Stars"
- CharityNavigator.org

charity: water

Thank You! Questions?

Jonathan "J." Tower

Principal Consultant & Partner

Trailhead Technology Partners

TRAILHEAD
TECHNOLOGY PARTNERS

trailheadtechnology.com

- Microsoft MVP in .NET
- Organizer of Beer City Code

- jtower@trailheadtechnology.com
- trailheadtechnology.com/blog
- jtowermi

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