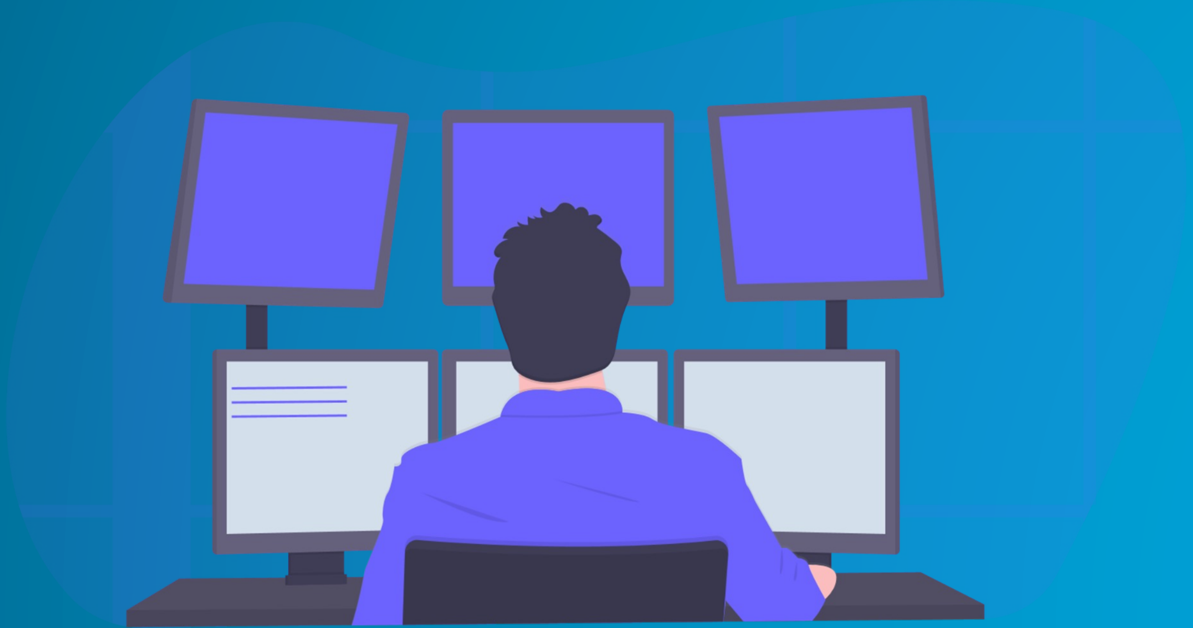
Stackable

Generationen-übergreifende Data Lakes mit Open Source Software aufbauen

Sönke Liebau

Stefan Igel



Stackable in a Nutshell

Founded

2020

OpenCore

b.telligent

IONOS

Stackable Data Platform

- Open Source
- Infrastructure as Code
- Cloud-native (Kubernetes)
- On-Premises, Cloud, Hybrid

Our Customers

Danske Bank

Taboola™

T · ·

Dentsply Sirona

IONOS

opencorporates

Our Team: 20 People

International
in Germany & Europe

Our Services

- Product Support
- Big Data Consulting
- Trainings

Network - Collaborations

OSB Open Source Business ALLIANCE

KI BUNDESVERBAND

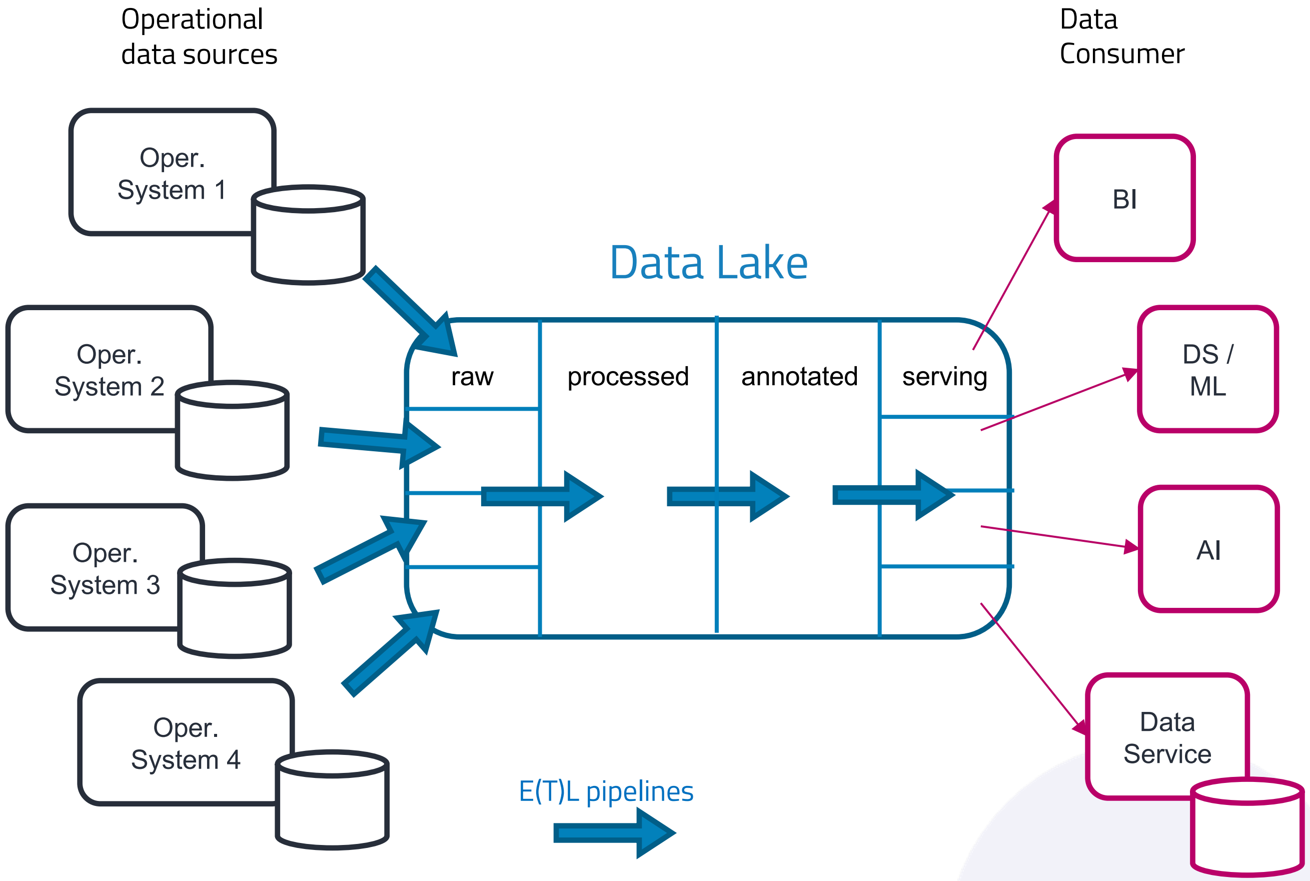
gaia-x

bitkom

eco

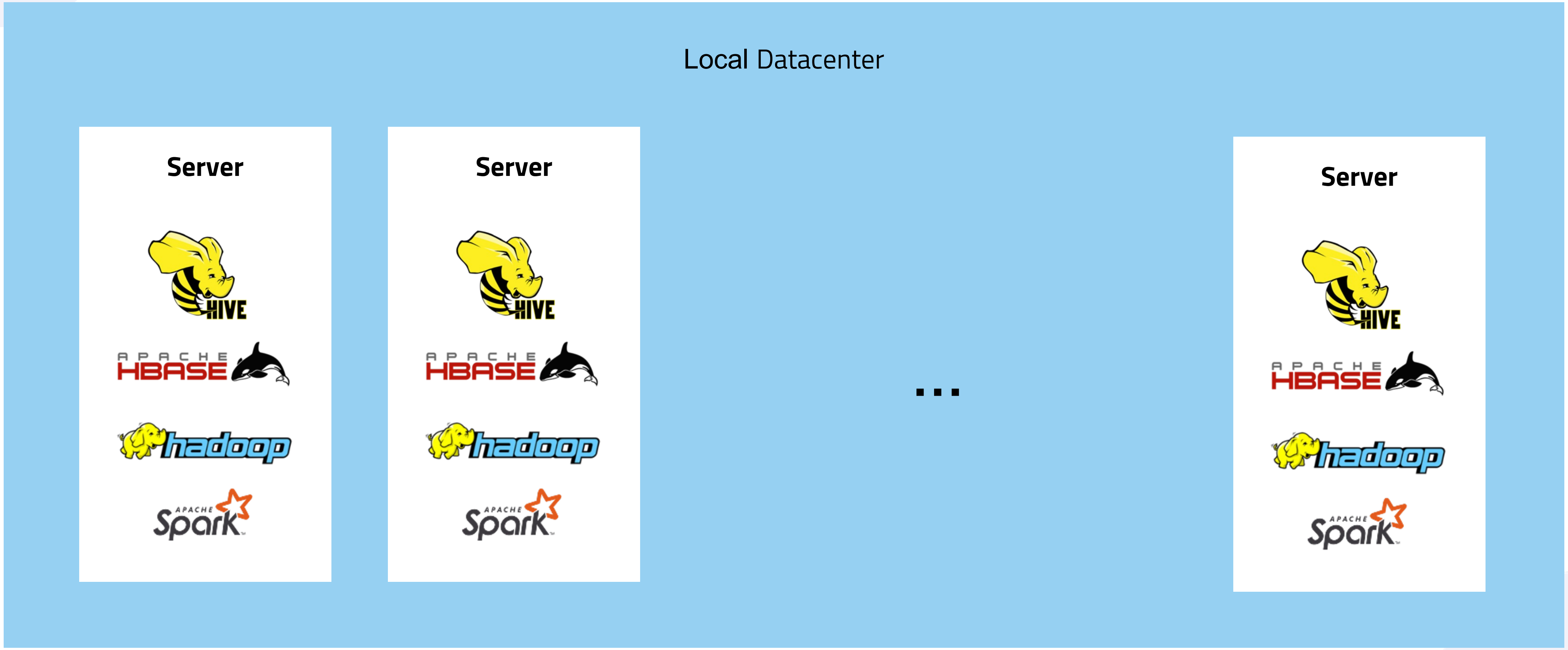
Data Lake – Characteristics and Logical Architecture

- A Data lake scales horizontally in storage and compute capacity
- A Data lake architecture is a derived hub and spoke architecture
- A Data lake is often structured into different zones
- Data from many op. source systems
- Data ingested 1:1 from source to data lake but transformed to optimized storage formats (e.g. Parquet)
- Data is loaded to scalable storage and read as files or data frames
- Queried by SQL-Engines with “Schema-on-read” approach
- Lakeshore marts as fit-for-purpose data marts used by apps and analytics
- A set of technologies is needed to ingest, transform, analyze, query and manage data in a data lake
- Central data governance, job scheduling / control, telemetry



Stackable

Data Lake Generation 1 - Combined storage & compute



Local Datacenter

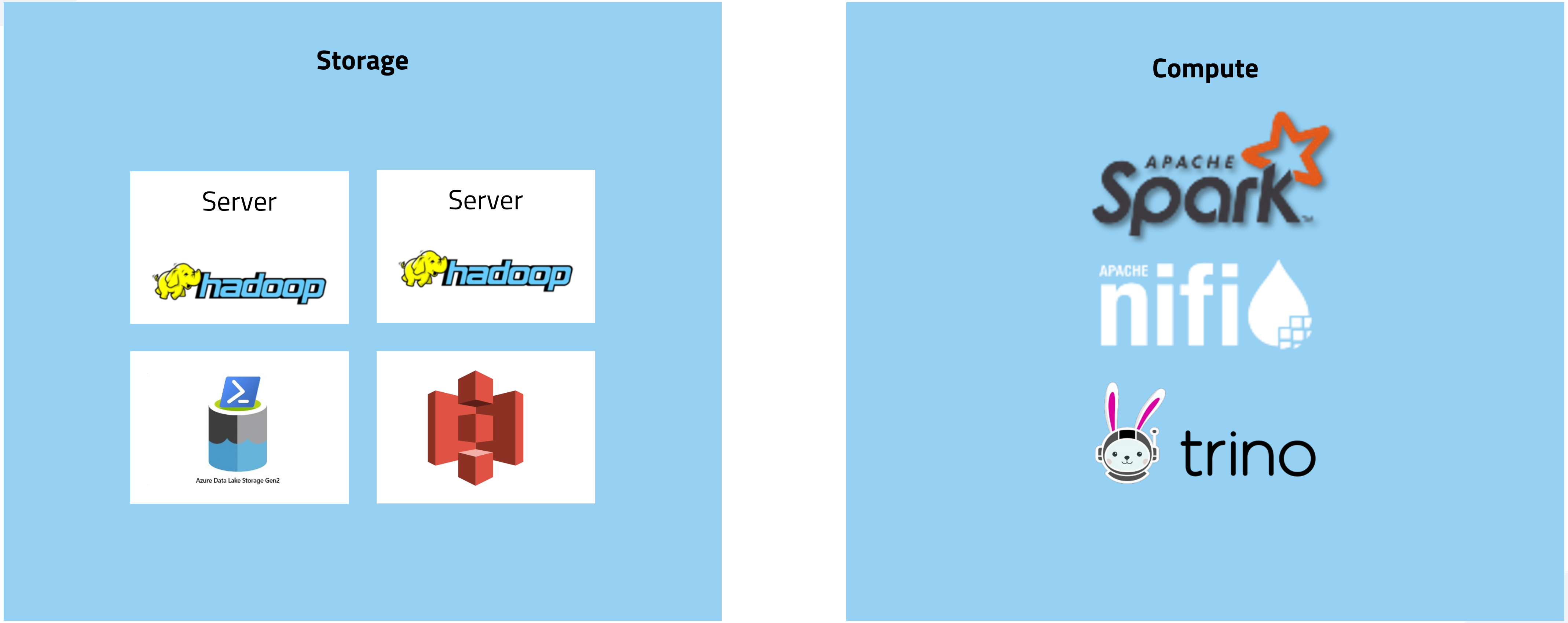
Server

Server

...

Server

Generation 2 – Separated storage & compute



Storage

Server

Server

Compute

Data Lake – The Reality of Zones

Data Lake – The risk of becoming a swamp

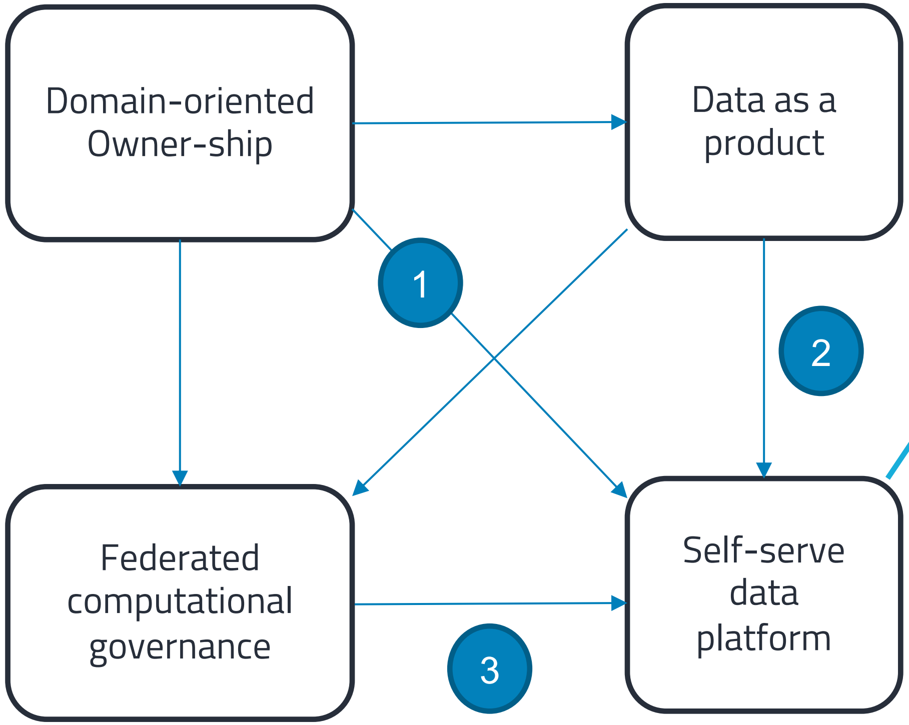
Stackable

Beyond the Lake - Data Mesh and its Principles*

Data Mesh is a decentralized sociotechnical approach to share, access, and manage analytical data in complex and large-scale environments – within or across organizations.

Zhamak Dehghani

Principals


Domain-oriented Owner-ship
Data as a product
Federated computational governance
Self-serve data platform
1
2
3

Self-serve data platform

Data Infrastructure optimized for infrastructure utilization and performance

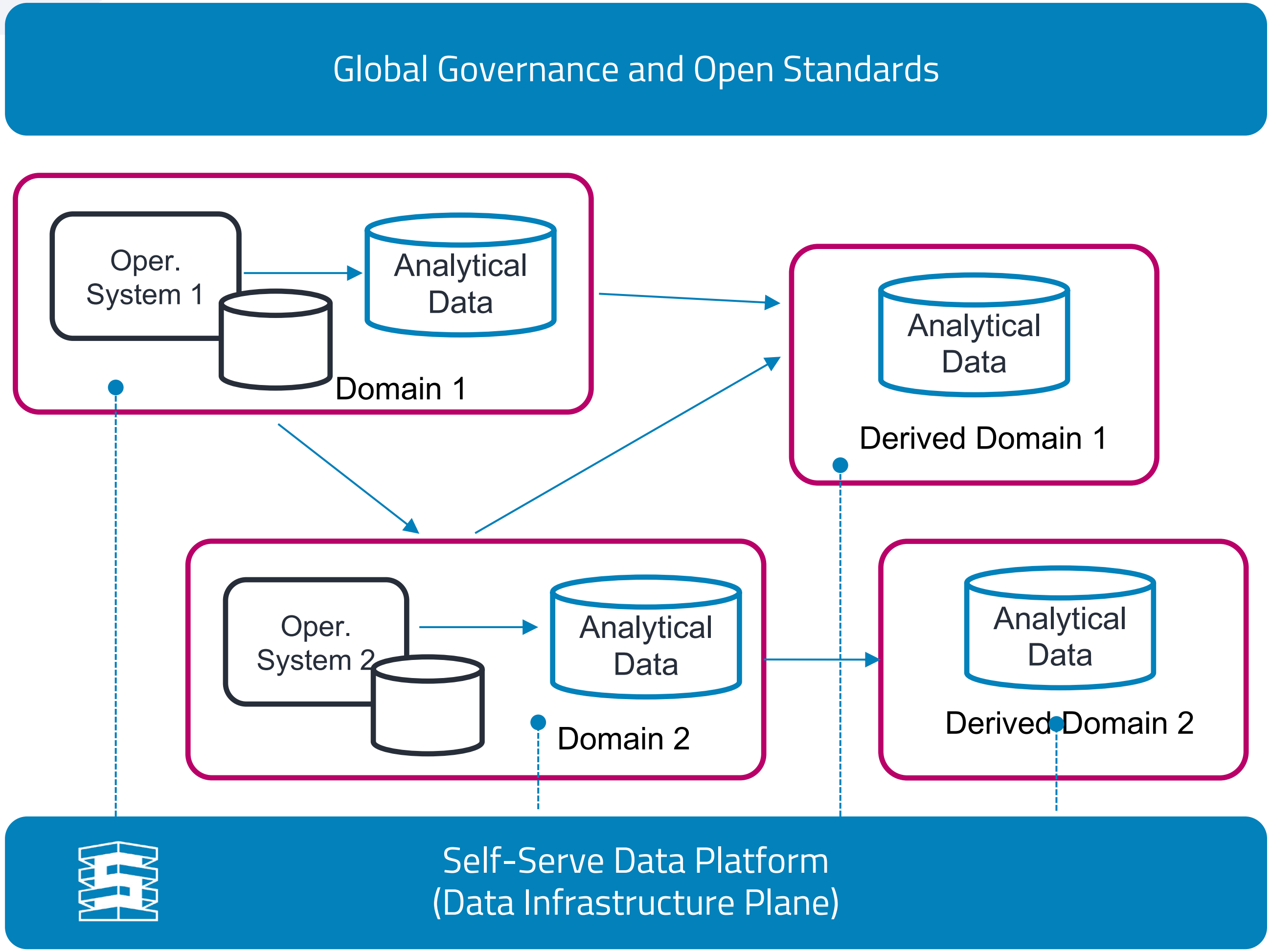
Purpose

1. Empower Domain Teams
2. Reduce data product cost of ownership
3. Mesh-level consistent and reliable Policy enforcement

*nach Zhamak Dehghani: Data Mesh, O'Reilly 2022

Stackable

Data Mesh Architecture – pivot the Data Lake*



- Domain driven design
- Data Products (APIs)
- De-centralized instead of centralized
- Mesh instead of Hub-and Spoke
- Discovery
- Raw data storage “at the source”

*nach Zhamak Dehghani: Data Mesh, O‘Reilly 2022

Stackable

Many of our customers ...

... already have an existing Data Platform

... with an Data Lake Generation 1 architecture

... installed on-premises based on Hadoop

What is our next generation data platform?
Stay on-prem or move to cloud?
Stay with Data Lake Gen 1? move to data Lake Gen 2? Or data mesh?

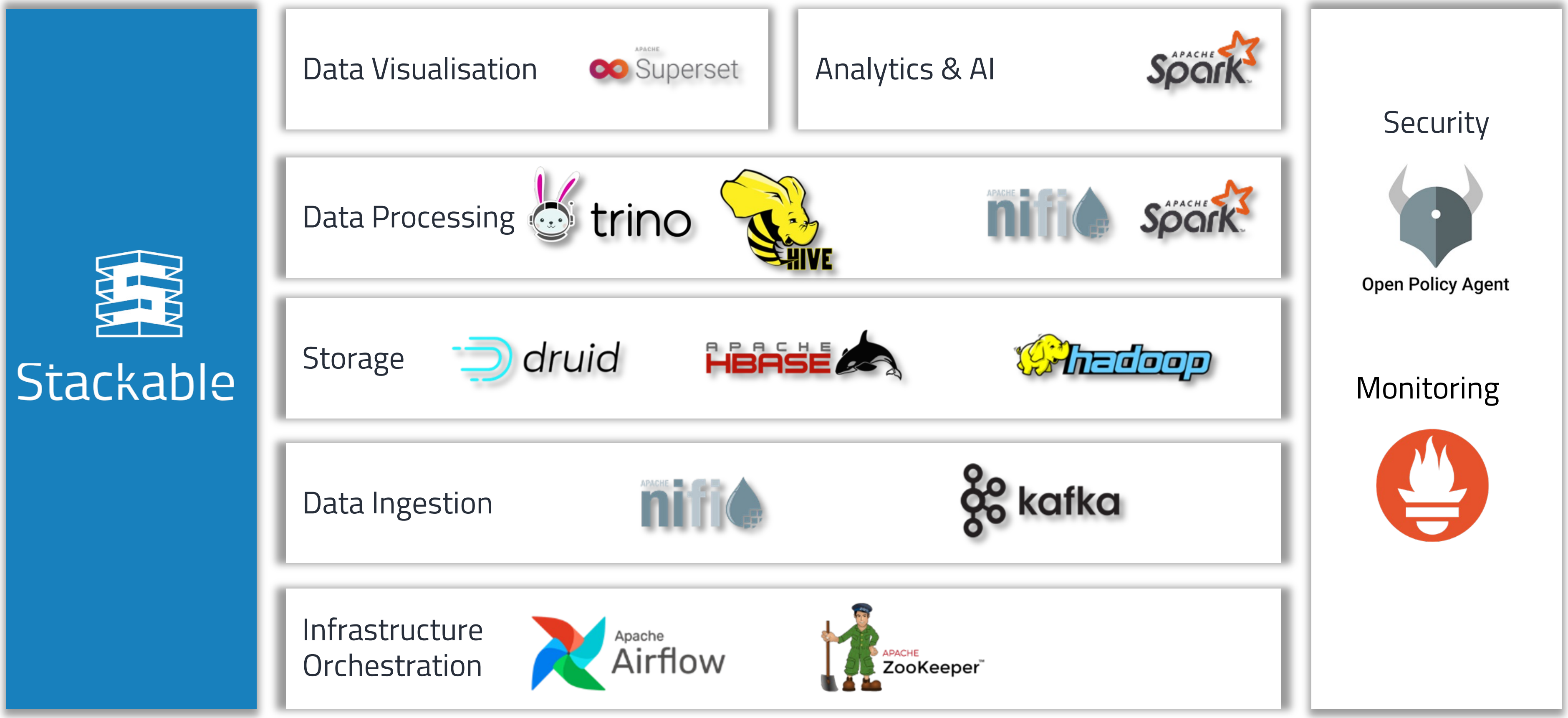
Stackable Data Platform wants to answer to their questions!

Stackable

Stackable Data Platform

Stackable

Stackable - an alternative Open-Source Data Platform



Stackable

Data Visualisation — Apache Superset

Analytics & AI — Apache Spark

Data Processing — trino, HIVE, Apache nifi, Apache Spark

Storage — druid, Apache HBASE, hadoop

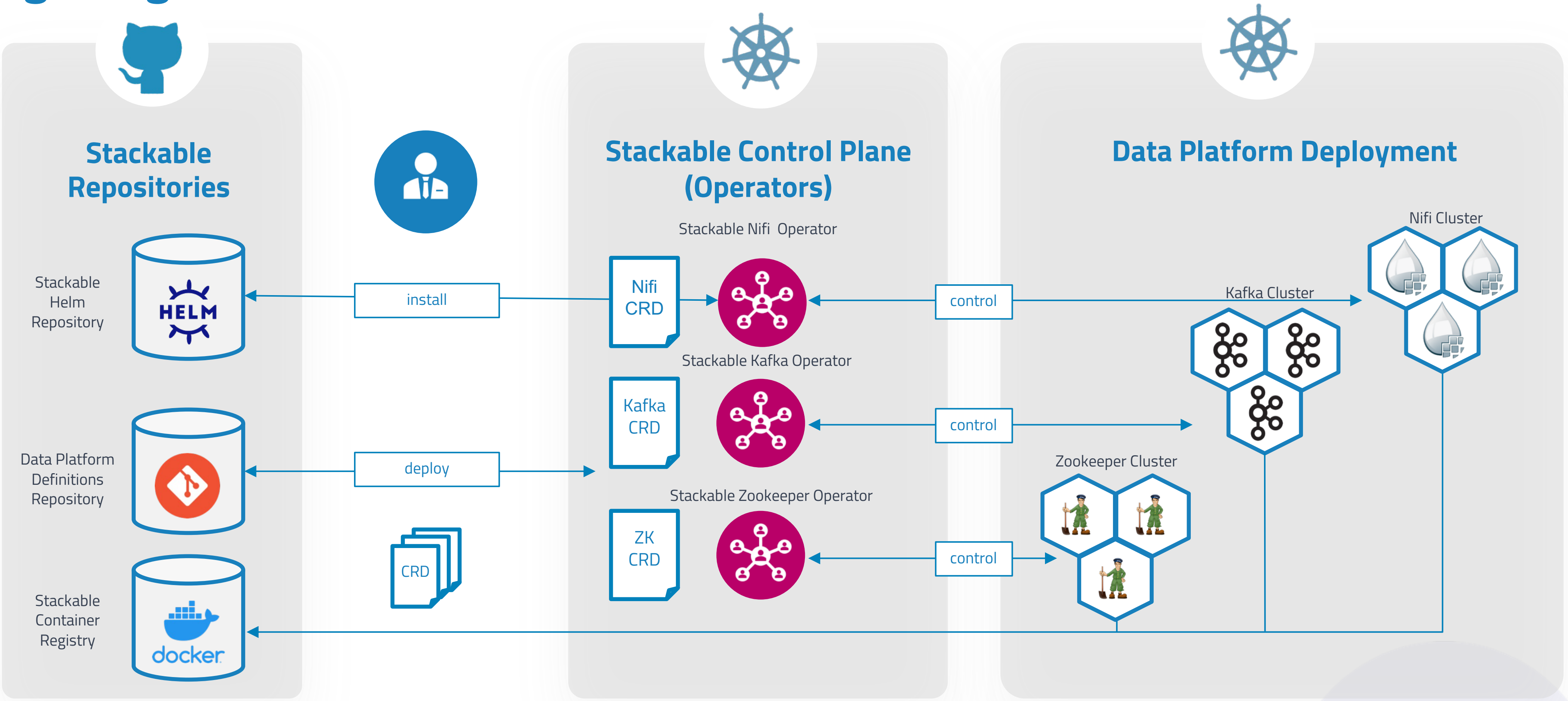
Data Ingestion — Apache nifi, kafka

Infrastructure Orchestration — Apache Airflow, Apache ZooKeeper

Security — Open Policy Agent

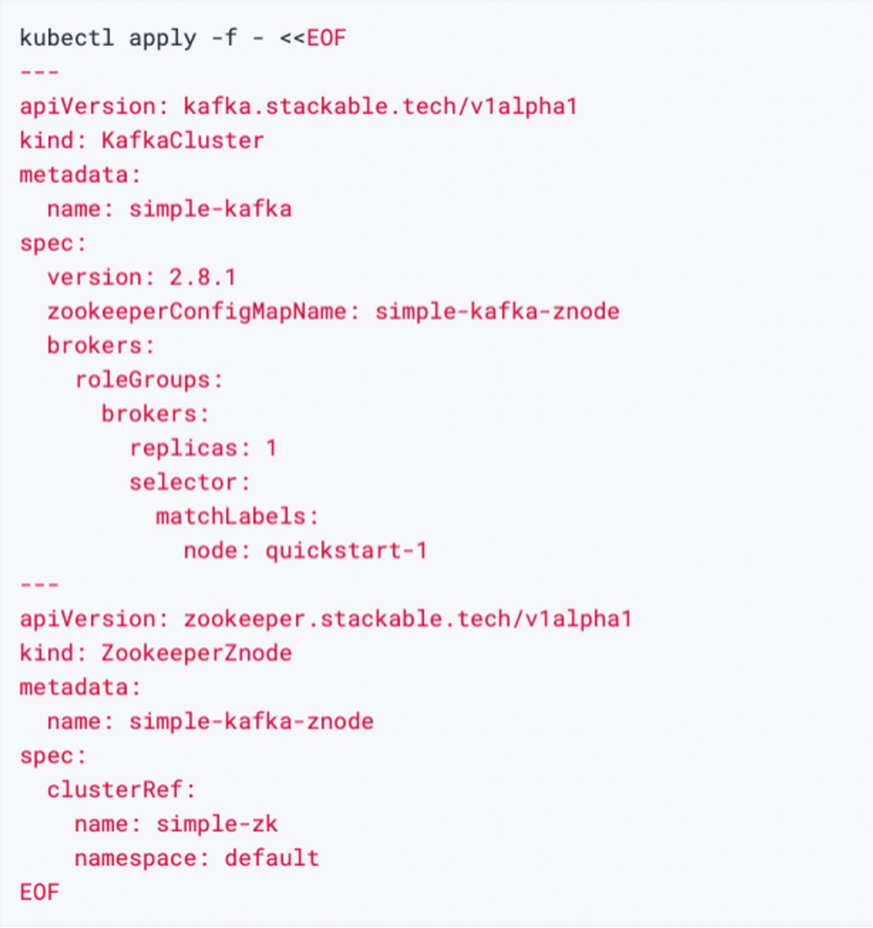
Monitoring

Streaming & Big Data Infrastructure as Code on Kubernetes



Stackable

How Kubernetes and K8S-Operators support Data Lake features



```
kubectl apply -f - <<EOF
---
apiVersion: kafka.stackable.tech/v1alpha1
kind: KafkaCluster
metadata:
  name: simple-kafka
spec:
  version: 2.8.1
  zookeeperConfigMapName: simple-kafka-znode
  brokers:
    roleGroups:
      brokers:
        replicas: 1
        selector:
          matchLabels:
            node: quickstart-1
---
apiVersion: zookeeper.stackable.tech/v1alpha1
kind: ZookeeperZnode
metadata:
  name: simple-kafka-znode
spec:
  clusterRef:
    name: simple-zk
    namespace: default
EOF
```

Example: Custom Resource Definition (CRD)

Operators are software extensions to Kubernetes that make use of custom resources to manage applications and their components.*

A *resource* is an endpoint in the Kubernetes API that stores a collection of API objects of a certain kind**

- Scalability of compute resources is managed by K8S
- Ship platform components as containers managed by operators
- Storage: S3 and HDFS Operators or external
- Portable, reduces vendor lock-in

- Infrastructure-as-code via CRDs
- Service Discovery
- Central secret management (certificates) by Secret Operator
- Flexible authorization (as code) through Open Policy Agent Operator
- Unified telemetry (Monitoring, Logging, Alerting) configurable via CRDs

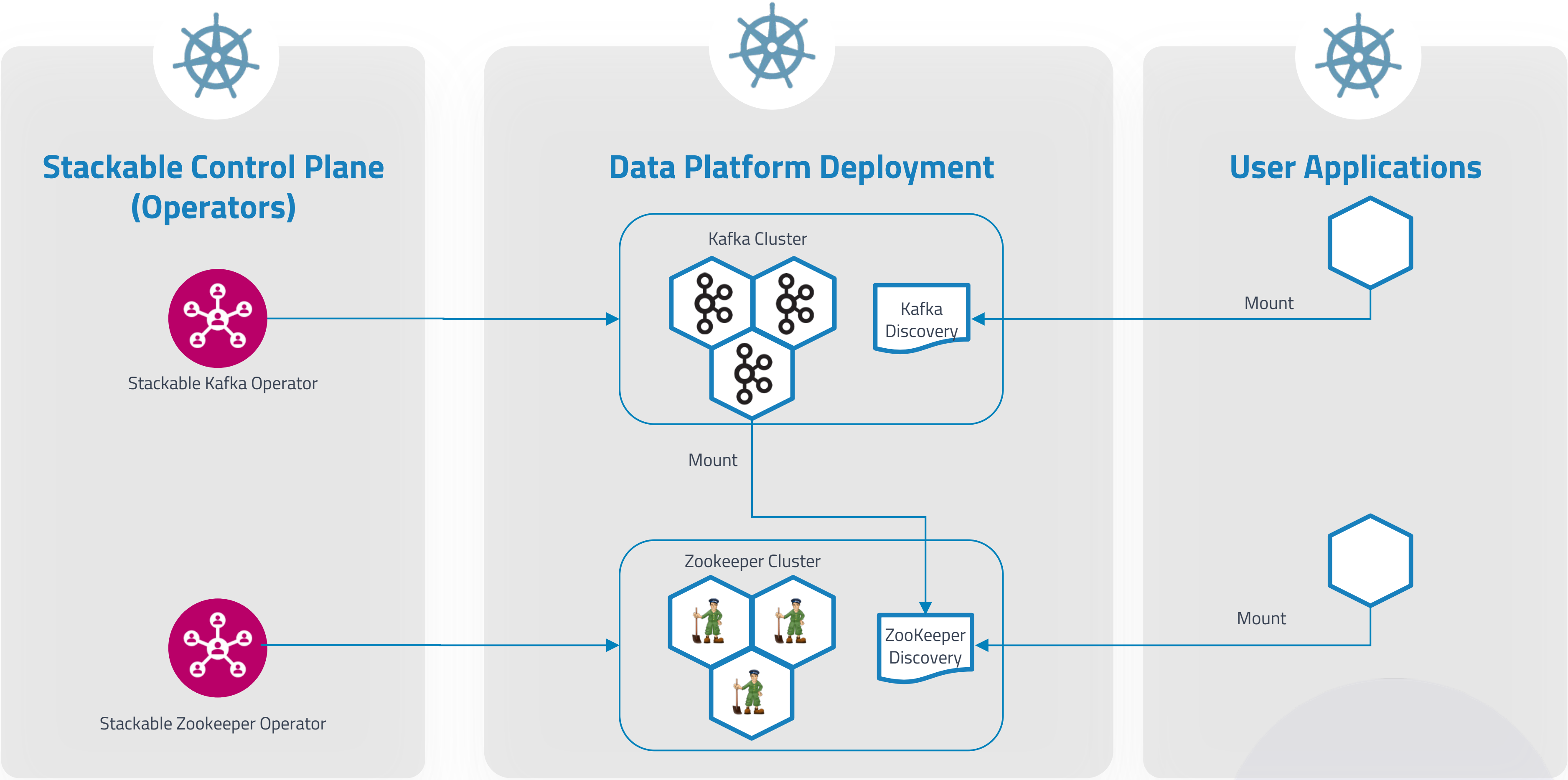
Stackable

*https://kubernetes.io/docs/concepts/extend-kubernetes/operator/,
**https://kubernetes.io/docs/concepts/extend-kubernetes/api-extension/custom-resources/

Service Discovery

Kubernetes offers native functionality that can be used for service discovery.

This has the additional benefit of providing automatic service restarts when services change that they depend on.



Stackable

Security - Authentication

Automatic TLS handling

- Including handling of expired certificates and automated restarts of affected services
- Client certificates on-demand
- Integration with company CA possible
- Automatic creation of keystores in required formats

Source: https://web.mit.edu/kerberos/

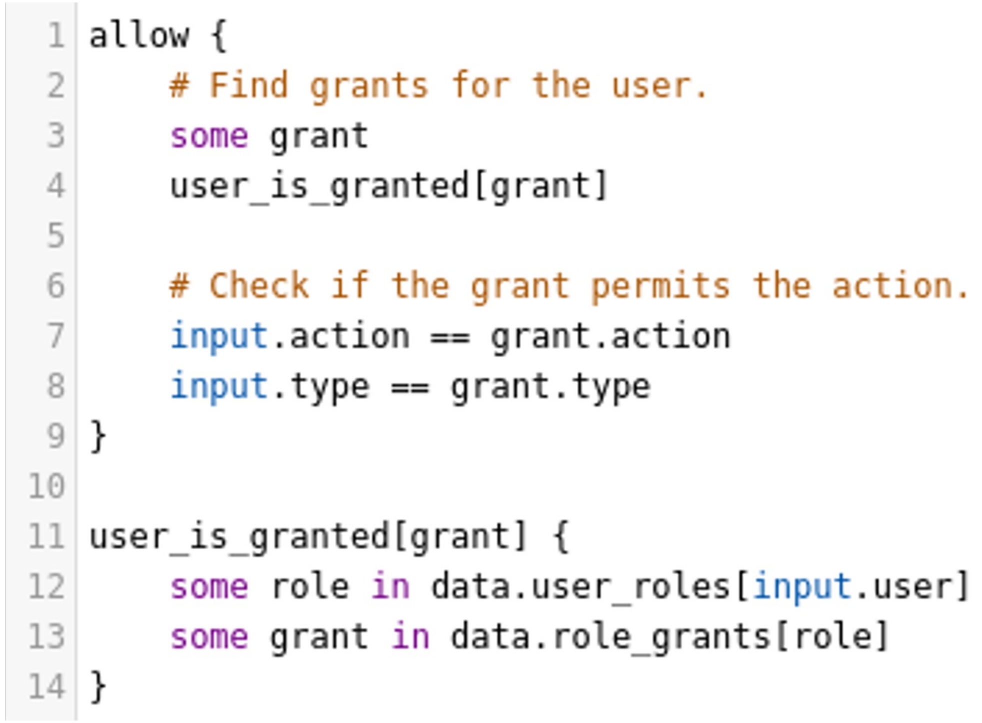
Automatic Kerberos handling

- As far as we know this is a ***world first*** in Kubernetes
- Automatic creation of principals
- Automatic creation of keytabs

Security – Authorization as Code

Open Policy Agent

Source: https://cncf-branding.netlify.app/projects/opa/



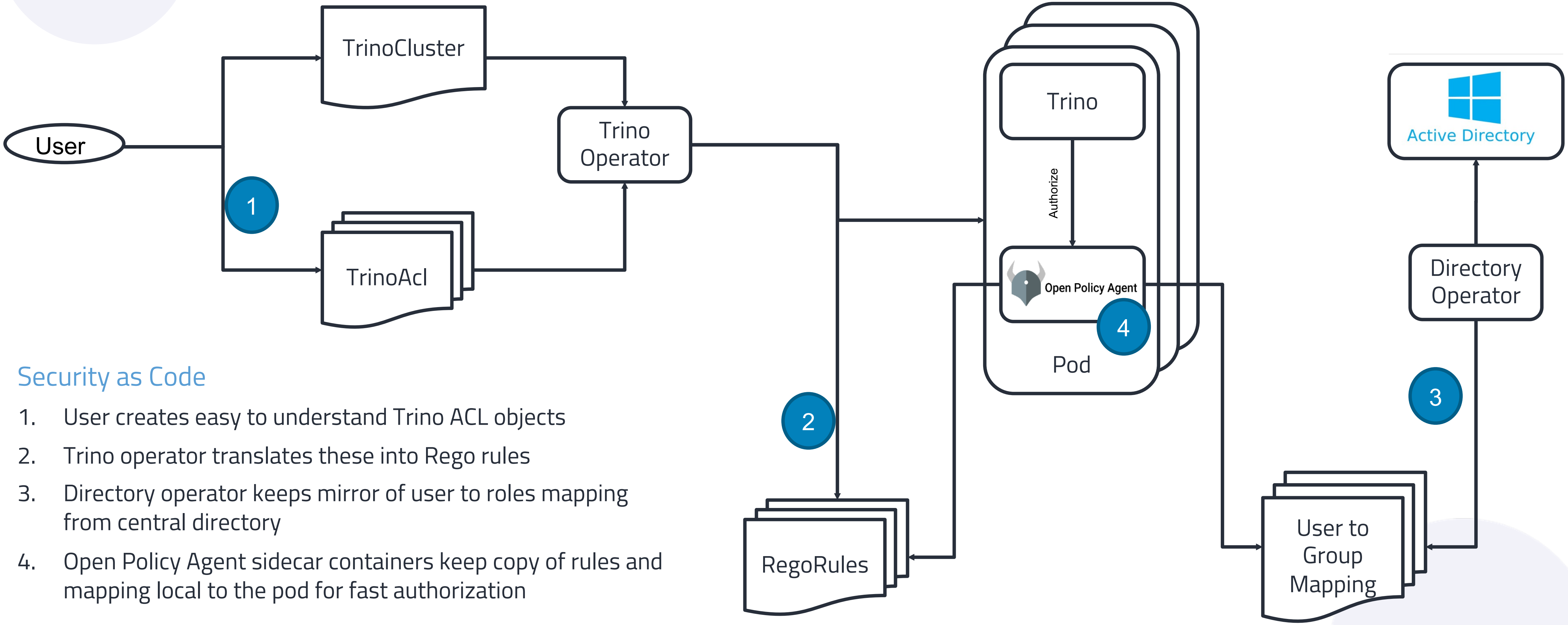
```
allow {
    # Find grants for the user.
    some grant
    user_is_granted[grant]

    # Check if the grant permits the action.
    input.action == grant.action
    input.type == grant.type
}

user_is_granted[grant] {
    some role in data.user_roles[input.user]
    some grant in data.role_grants[role]
}
```

- Policies-as-Code (“Rego Rules”)
- Authorization plugins added to the components where possible
    - Trino
    - Apache Druid
    - Apache Kafka

- Group lookup done once!
    - We’re adding a dedicated way to look up groups
    - No more configuring a dozen tools with the same settings

Stackable

Security as Code – Putting it all together



Security as Code

1. User creates easy to understand Trino ACL objects
2. Trino operator translates these into Rego rules
3. Directory operator keeps mirror of user to roles mapping from central directory
4. Open Policy Agent sidecar containers keep copy of rules and mapping local to the pod for fast authorization

Data Lake gen 1 with K8s - challenges

- Due to data locality paradigm a lot of effort was put into running calculations where the data is - K8S is not really interested in this
- A lot of the early data (or big data) tools are from a different era of computing
  - Stable Network
  - Bare Metal access
  - "Simple" DNS
  - Predictable Restarts
  - …
- Complexity from "back then" is not gone, it is just hidden - until …

Stackable

How K8s and Operators support data mesh features

Challenges to address

Self-Serve Data Platform

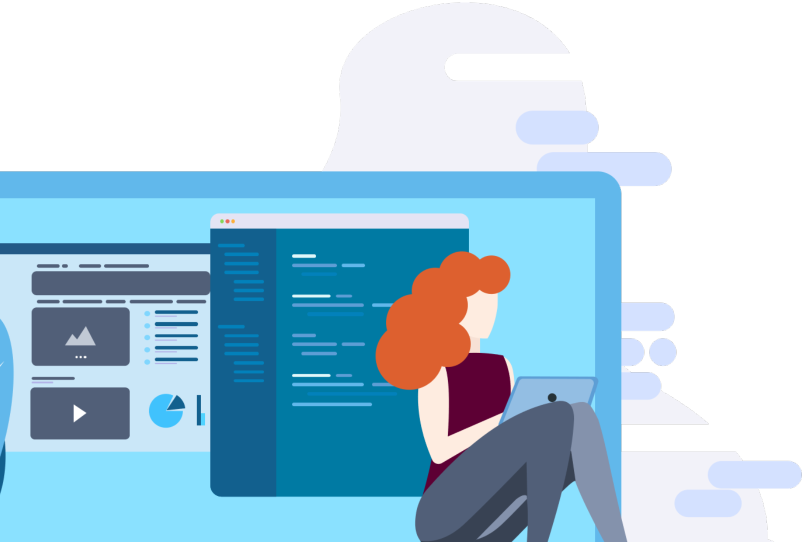
Distributed data architecture will lead to

- duplication of efforts in each domain
- Increased cost of operation
- Inconsistencies and incompatibilities across domains

Benefits

- Standardized Logging, Monitoring, Auditing
- Similar Operators (same look and feel)
- More providing standards and examples than running a central platform

- Easy to run many instances at the same time
- Easy to define entire stacks and deploy them multiple times
  - Every team has its own stack to run
  - Can be easily shared with other teams



Stackable

Summary

- Data platform architectures have evolved over time together with the enabling technologies
- Kubernetes has been a great paradigm shift
  - K8S provides a scalable compute platform for data workloads
  - Operators allow enforcement of standards
  - Containers and K8S facilitate data lakes and meshes
  - Some tweaks necessary to enable data lake gen 1 technologies
- Modern data platforms can be setup vendor-independent by open-source tools

Thank you

Contact

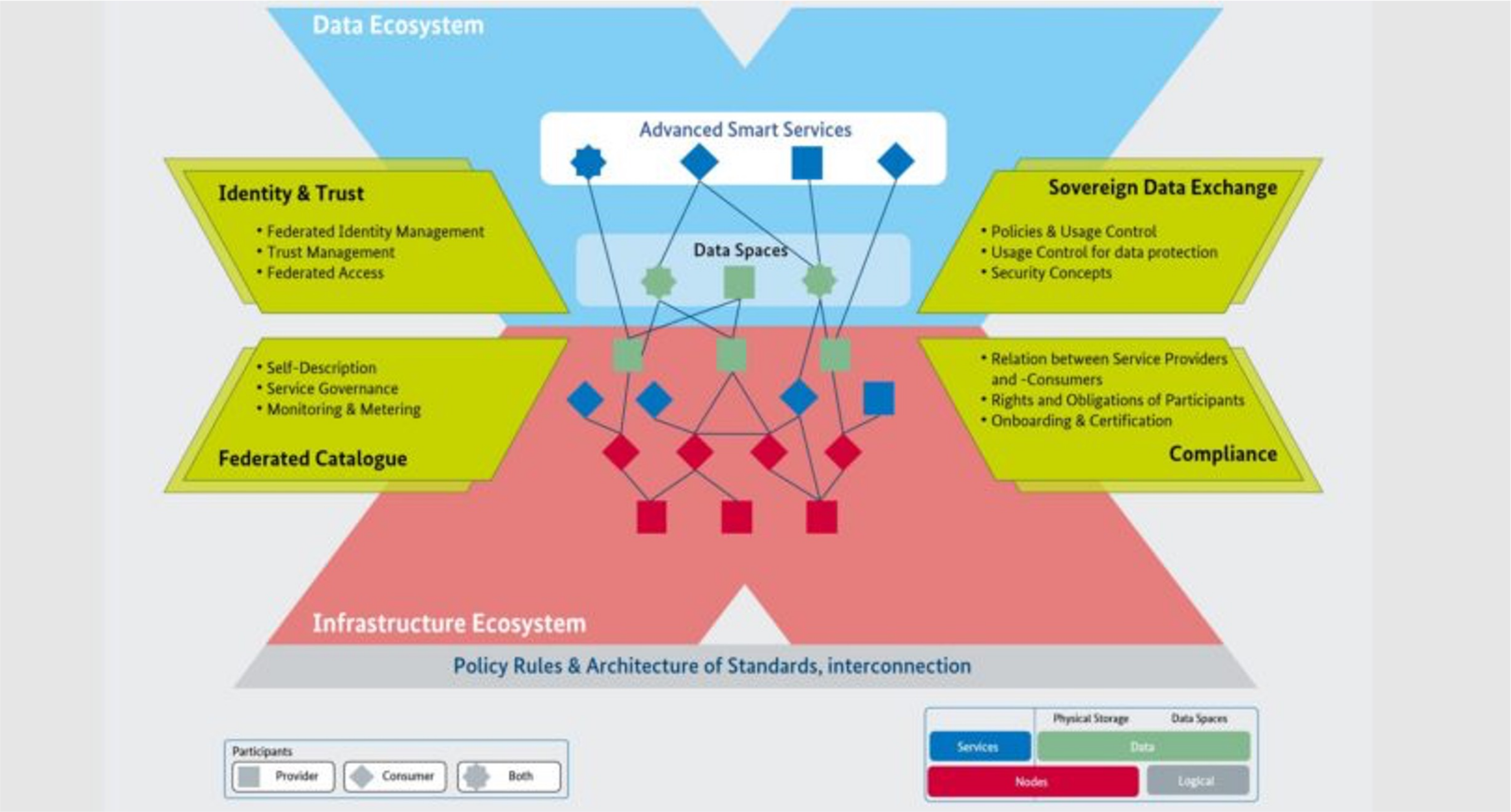
Dr. Stefan Igel
stefan.igel@stackable.de
+49 (160) 6171731
linkedin.com/in/stefan-igel

Sönke Liebau
Sönke.liebau@stackble.de
+49 (179) 7940878

Stackable

What’s next? Gaia-X Sovereign Data Spaces

GAIA-X
data-infrastructure.eu



conceptual

Cross-organization data mesh

Federation Services

Data Sovereinty

Data Products

Governance

Technical Architecture

API based

Revival of Compute-to-Data

K8S part of the reference architecture (SCS stack)

Stackable

marispace-x Building a maritime Data Space and Connect the dots.

GAIA-X Lighthouse project

- Drive the digitization of the ocean
- Facilitate digital collaboration in marine research
- Develop a smart maritime dataspace including Cloud- , Fog- and Edge-Computing
- Leveraging GAIA-X Federation Services for data sovereignty
- Funded by BMWK

Stackable Role

- Dataspace Platform Service Layer
- Data Storage & Compute
- Data Security & Governance
- GAIA-X Interoperability

Consortium

- Universität Kiel
- Universität Rostock
- GEOMAR Helmholtz Zentrum
- Fraunhofer IGD
- EGEOS GmbH
- TrueOcean GmbH
- MacArtney Germany
- IONOS SE
- Stackable GmbH

Use Cases

- Internet of Underwater Things (IoUT)
- Offshore Wind – renewable energy
- Marine protection – ammunition in the sea
- Bio climate protection - decarbonization

Learn more

Stackable

https://marispacex.com/