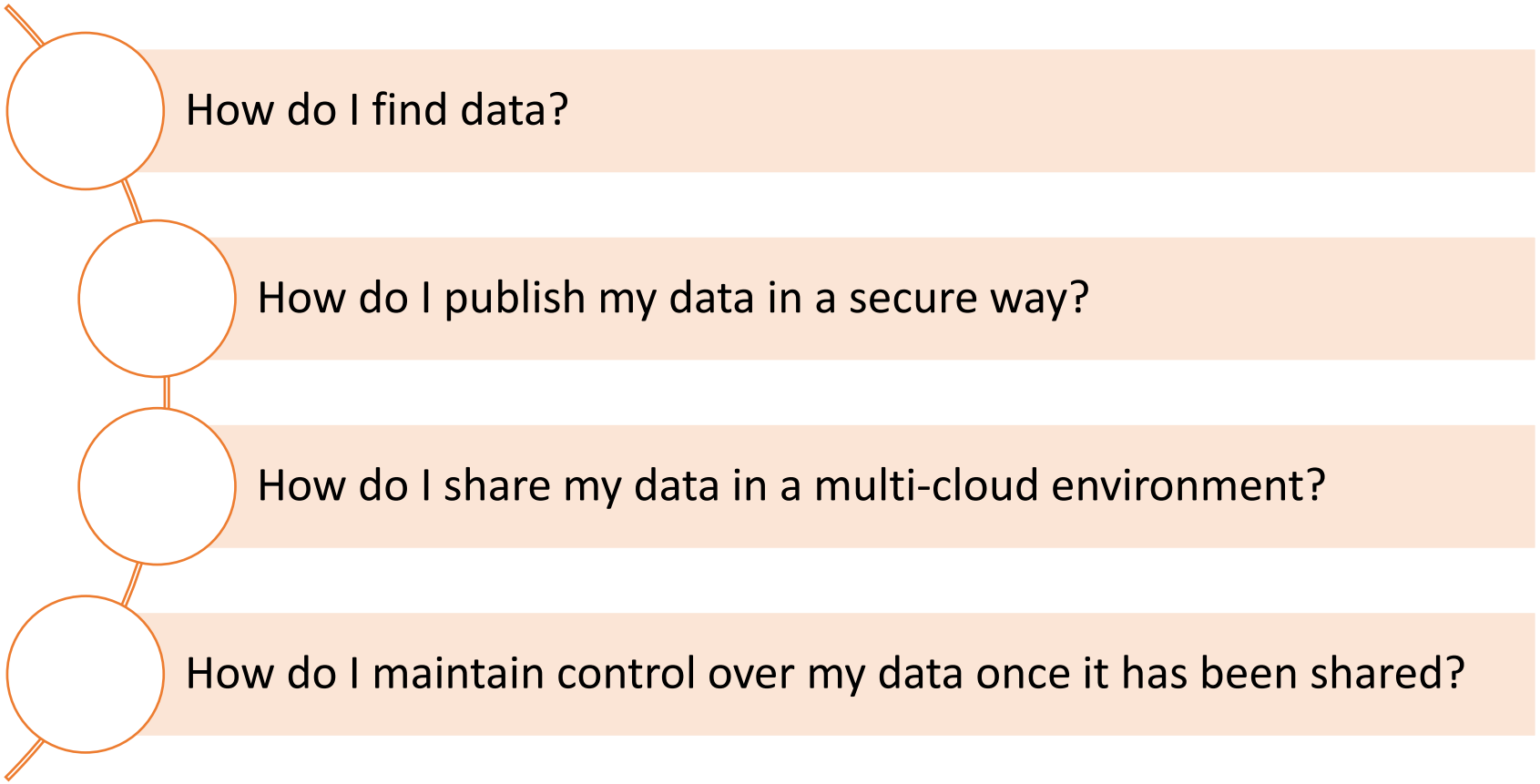
The Eclipse Dataspace Connector

Architecture and Principles

Jim Marino
Microsoft Corporation

Four technology challenges of data sharing



ECLIPSE
INCUBATION

The Eclipse Dataspace Connector

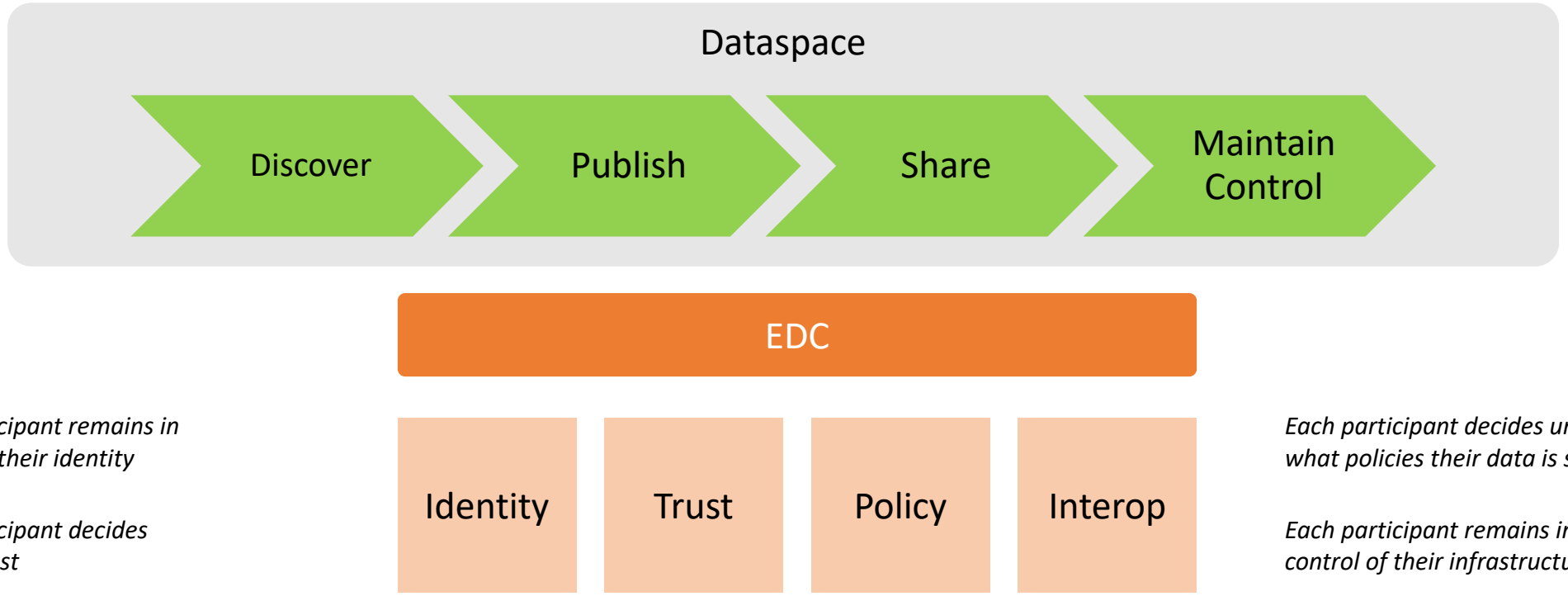
- A 100% open-source platform to address the problems of data sharing
- Provides the technology you need to create and participate in a **secure dataspace**
- Backed by global partners

Agenda

- What role does the EDC play in a Dataspace?
- EDC Architectural Principles and Design
- The EDC Foundation
- The Connector
- Policy Enforcement
- Federated Catalog Services

The four technology pillars of a dataspace

- A dataspace is a way for organizations to securely share data with other participants.
- Dataspaces are built on *identity*, *trust*, *policy*, and *interoperability*



Each participant remains in control of their identity

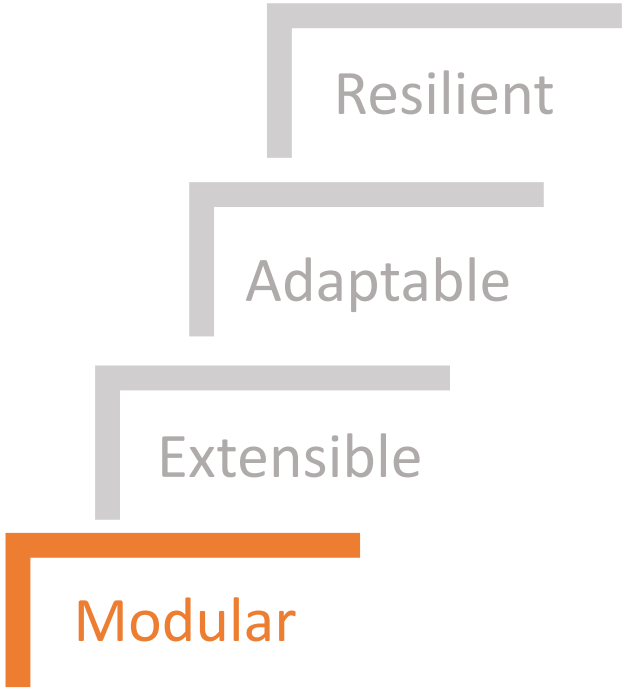
Each participant decides who to trust

Each participant decides under what policies their data is shared

Each participant remains in control of their infrastructure

ECLIPSE INCUBATION

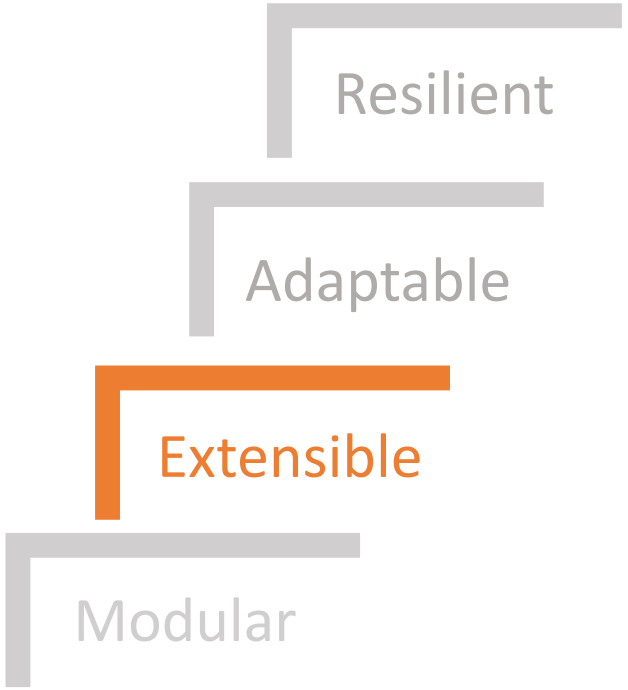
EDC architectural principals



- Written in Java
- All functionality is contributed as a module
- Lightweight, composable runtime
- Minimal dependencies

ECLIPSE
INCUBATION

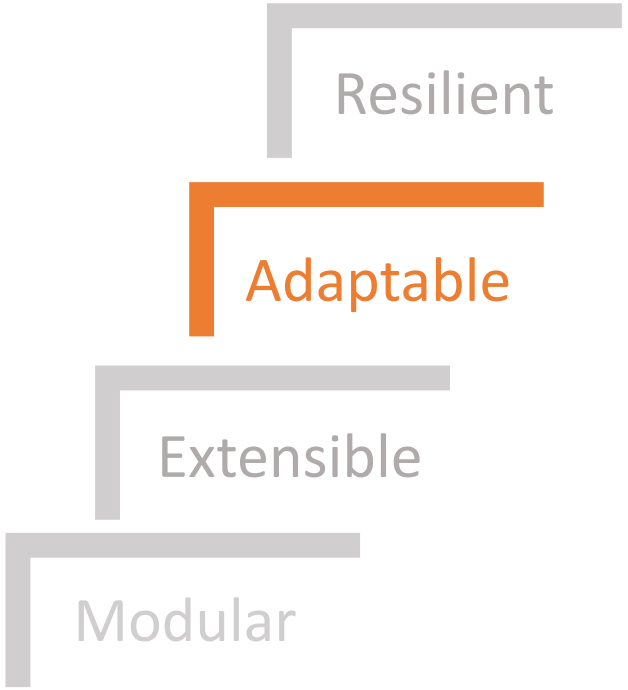
EDC architectural principals



- Defines extension points for all features
- Swap implementations, e.g., database, security
- Create your own features and capabilities
- Modules may combine to define higher-level extensibility points
  - APIs
  - RESTful extensions

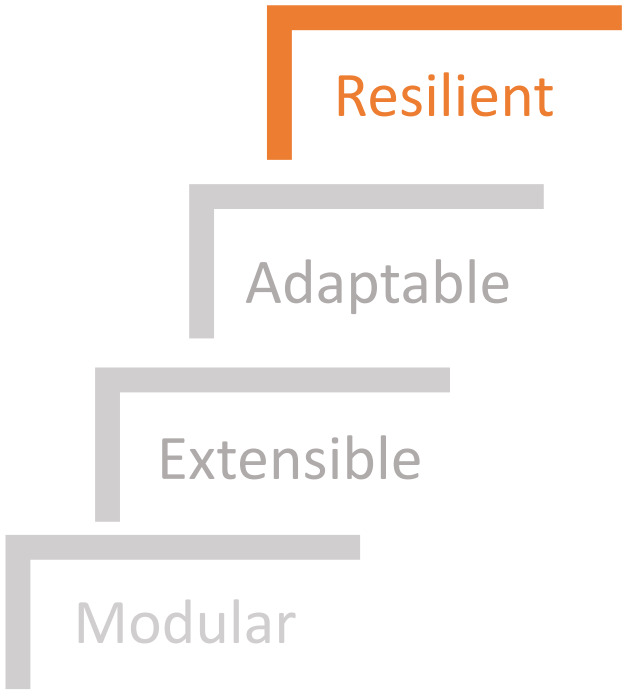
ECLIPSE
INCUBATION

EDC architectural principals



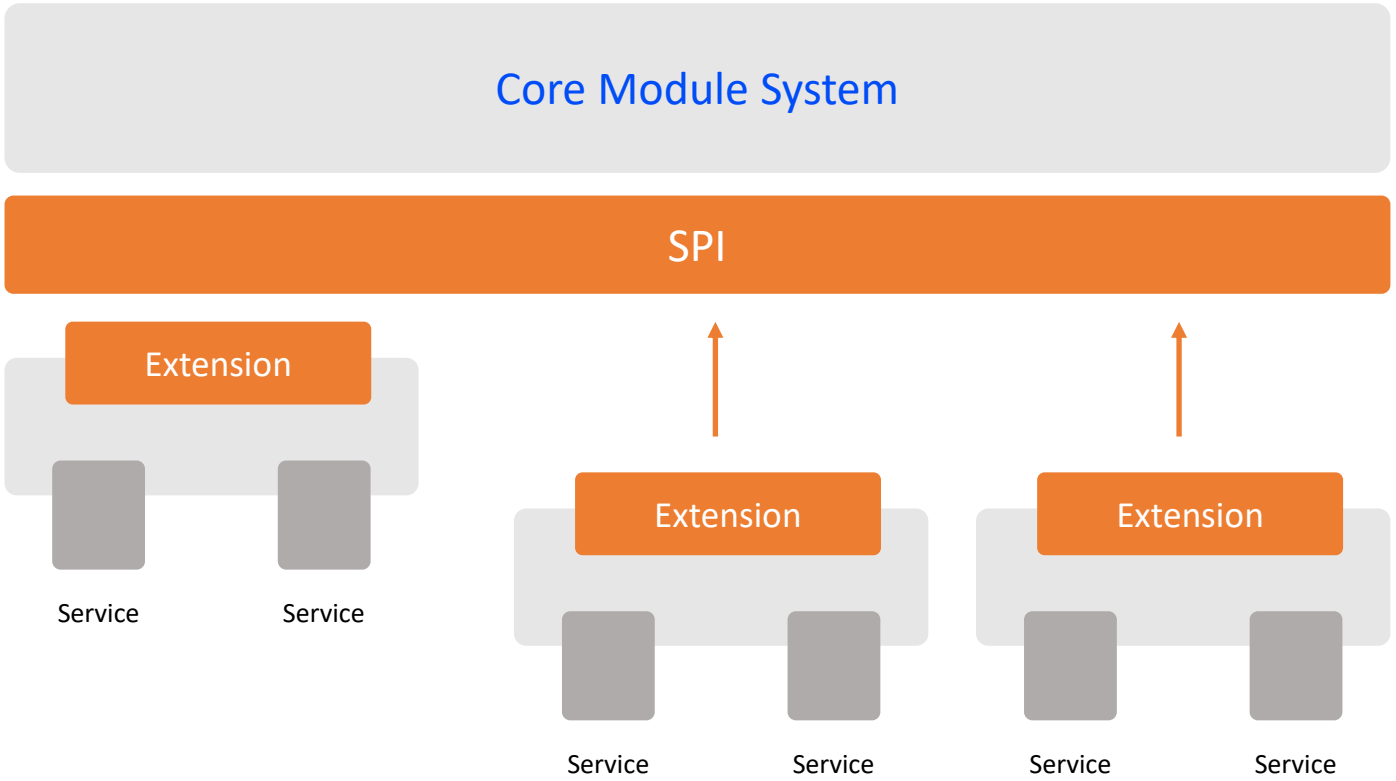
- Deploy to diverse environments
  - Cloud, on-premise, edge
- Deploy with different capabilities
- Scales up and down

EDC architectural principals



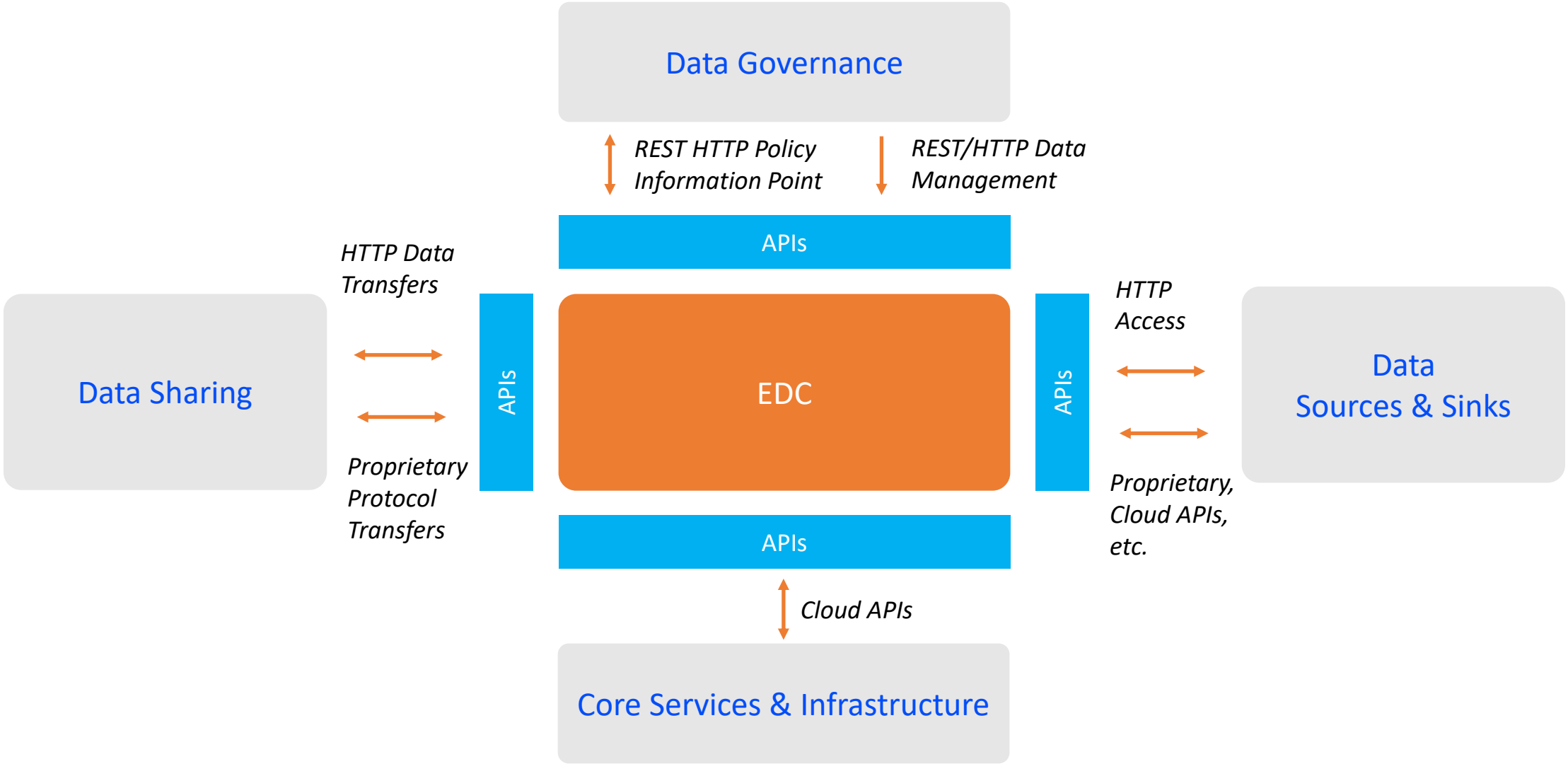
- Leverages high-availability infrastructure that you have already invested in
  - Cloud services
  - Data storage
  - Data transfer technologies

EDC design: The Foundation



ECLIPSE
INCUBATION

EDC APIs


Data Governance
REST HTTP Policy Information Point
REST/HTTP Data Management
APIs
HTTP Data Transfers
Data Sharing
APIs
EDC
APIs
HTTP Access
Data Sources & Sinks
Proprietary Protocol Transfers
Proprietary, Cloud APIs, etc.
APIs
Cloud APIs
Core Services & Infrastructure

EDC design: Dataspace Services

Registry

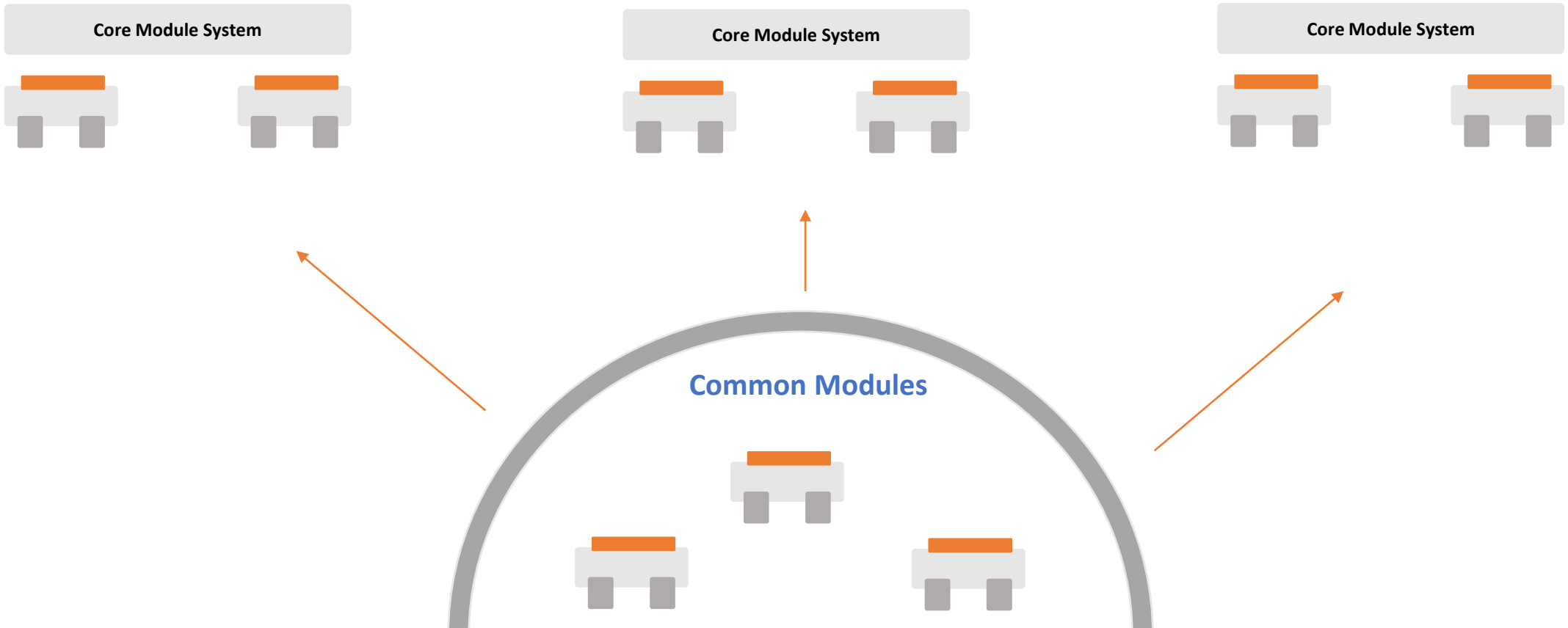
Registration and discovery

Catalog Services

Publish and search

Connector

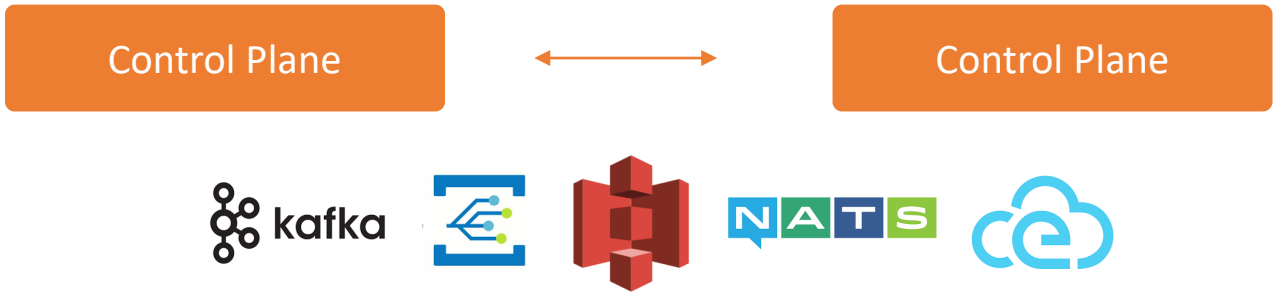
Contract negotiation and data sharing



ECLIPSE
INCUBATION

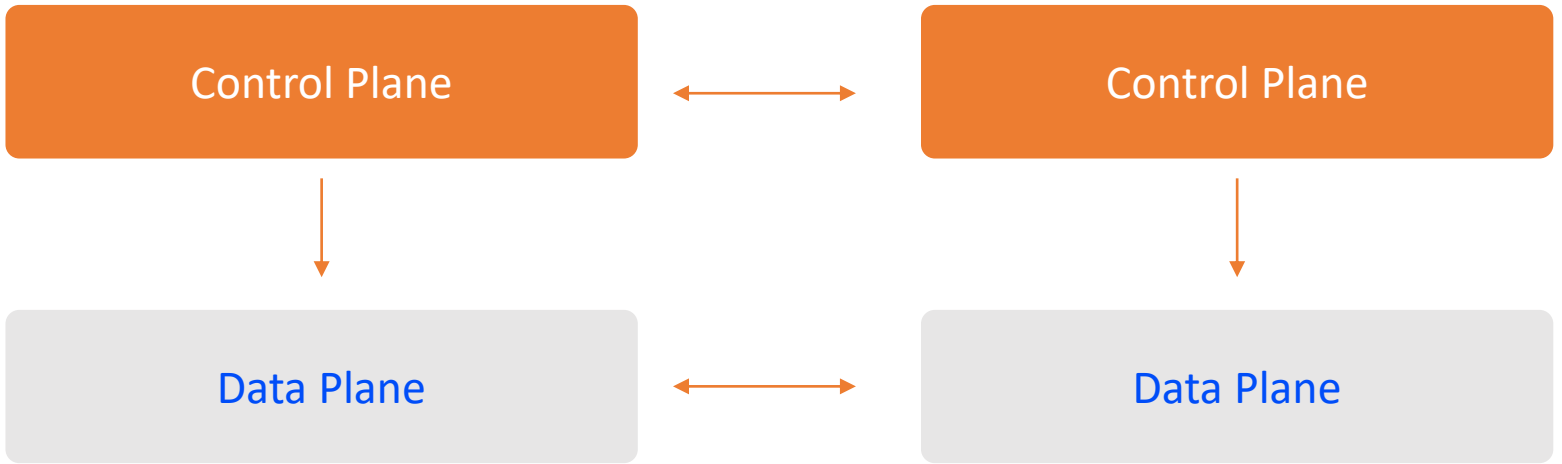
The Connector

Sharing Data



The Connector: Control Plane and Data Plane

- The Connector is divided into two logical subsystems, a **control plane** and a **data plane**

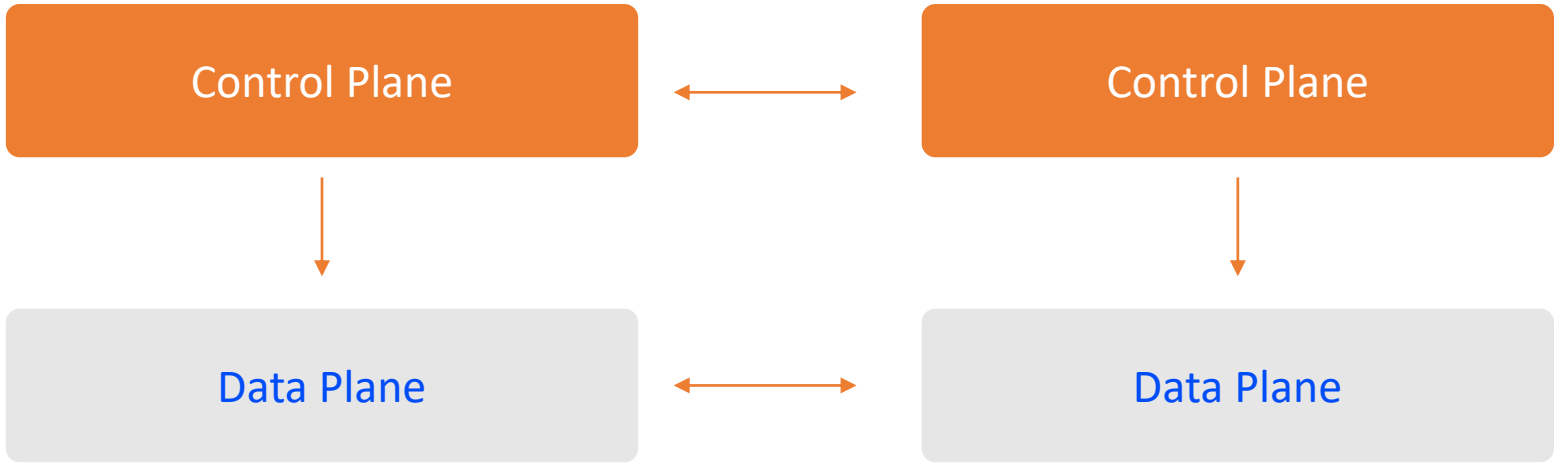


- Verification
- Contract negotiation
- Oversee policy enforcement
- Manages provisioning

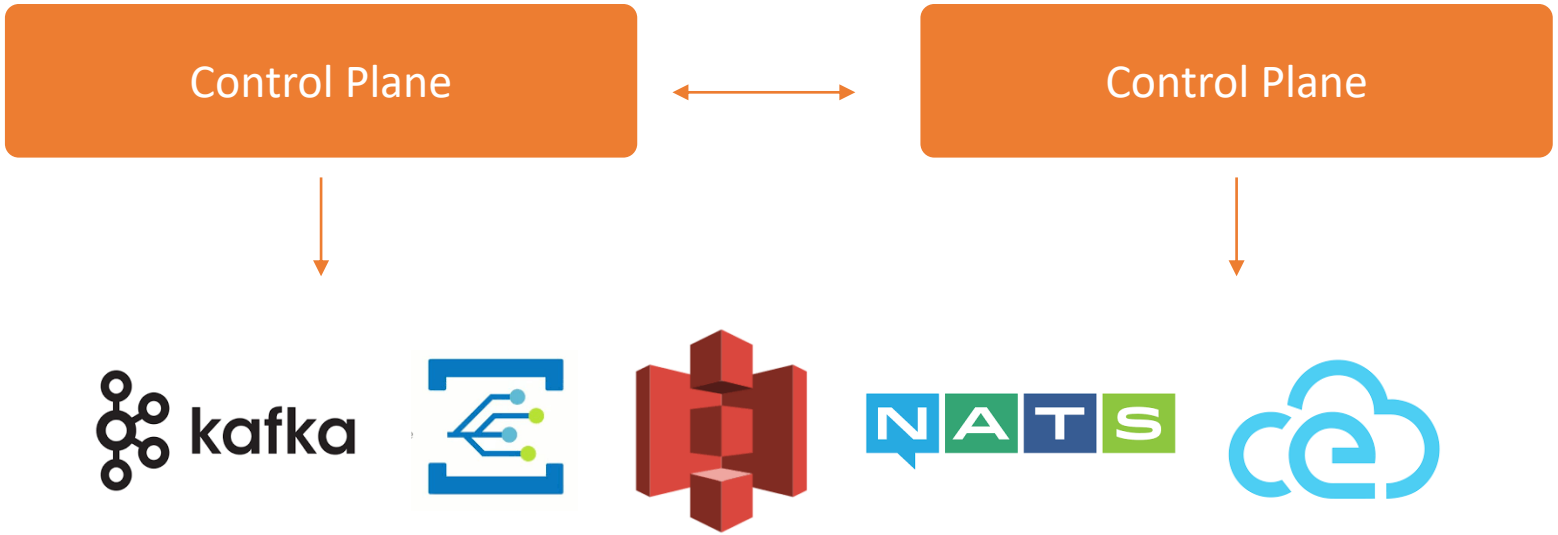
- Moves bits
- Big Data
- Streaming
- Events

ECLIPSE
INCUBATION

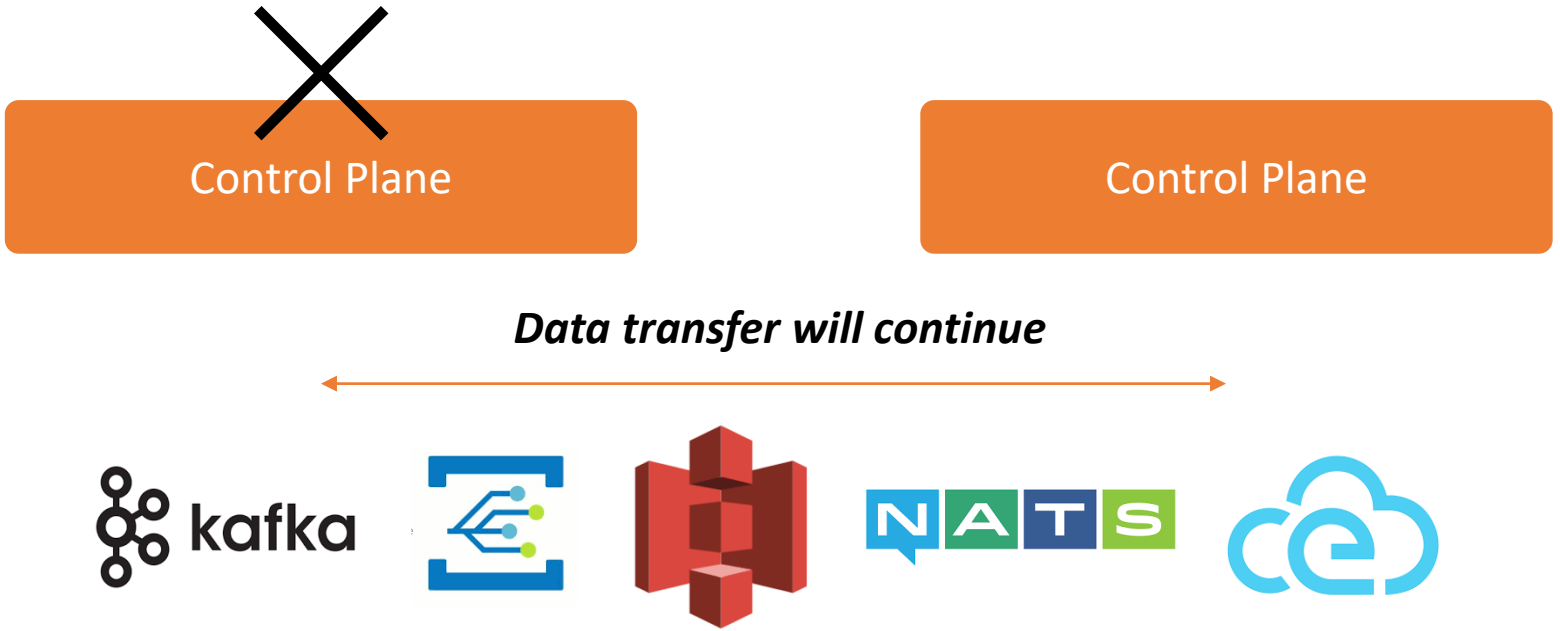
Why did we do this?



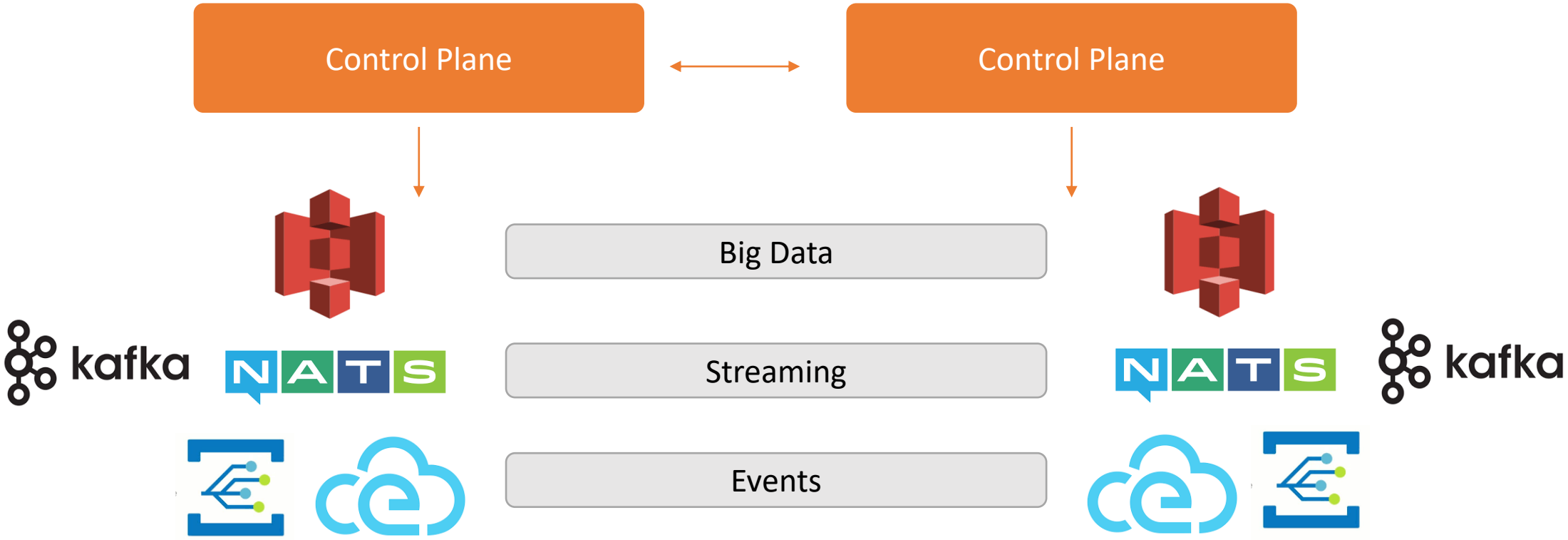
Leverage existing infrastructure



High Availability

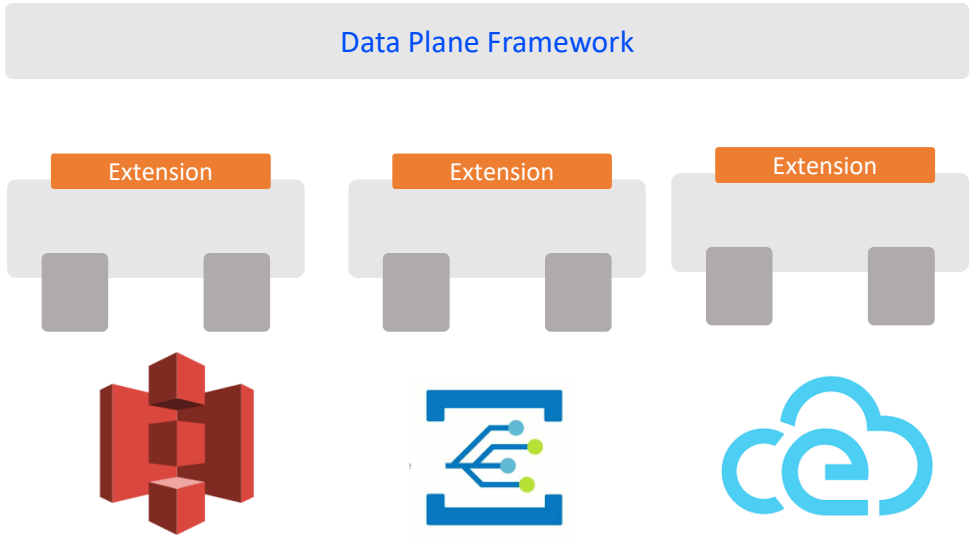


One size does not fit all



The Data Plane Framework (DPF)

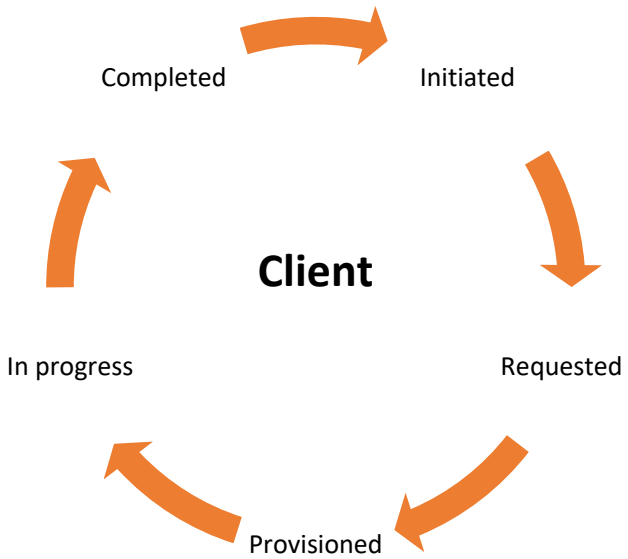
- A dynamic routing data plane
  - Optimized for big data and eventing
  - N-way transfers
- Built on the EDC foundation
- Initial release in Milestone 2

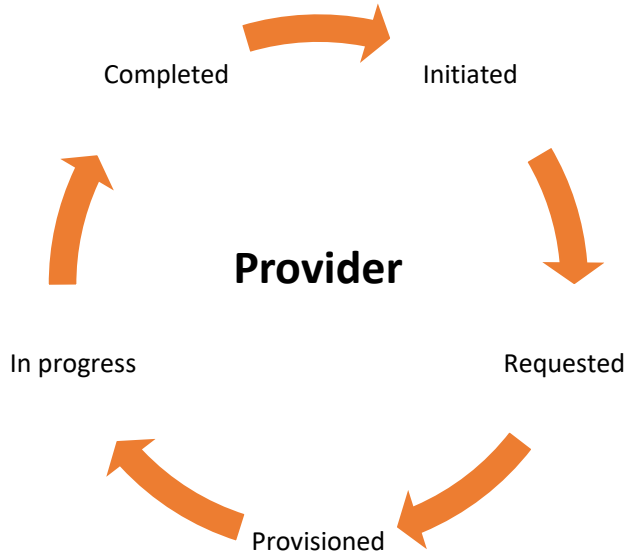


ECLIPSE
INCUBATION

The Connector: Asynchrony

- The Connector is an asynchronous system
- Requests asynchronously transition through a series of predefined states on the client and provider connectors





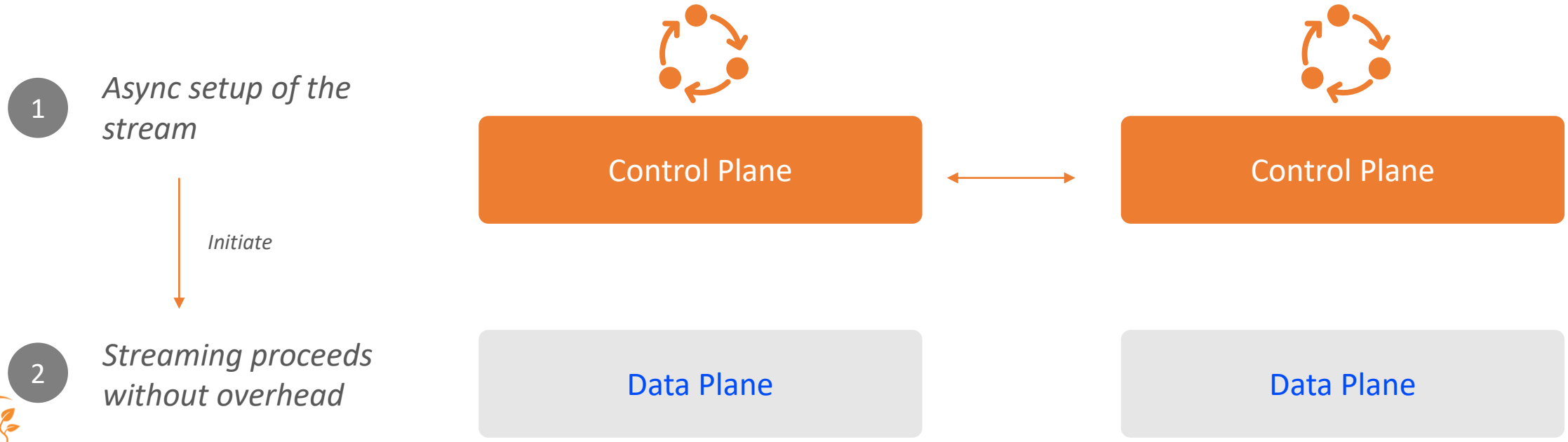
ECLIPSE
INCUBATION

Virtues of asynchrony

- Allows for potentially lengthy data preparation
  - Infrastructure provisioning
  - Policy provisioning
  - Data pre-processing
- Reliability through idempotent retries
- High availability via clustered workers
- Natural throttling
- Fairness by distributing processing across states

Does this impact data transfer performance?

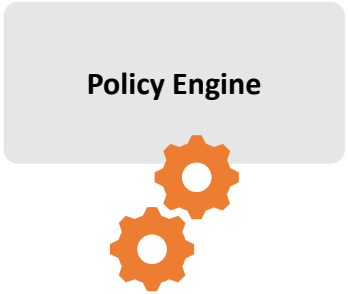
- Remember, the data plane is separate from the control plane
- Let's see how this works with streams…



Policy Enforcement

Maintaining Control Over Data

“These assets should only be shared with my partners”



“Data must remain in Europe”

ECLIPSE
INCUBATION

Two aspects of control

- The role of policies when publishing data
- Runtime policy enforcement

Assets and policies

- Data is represented as an ***asset.***
- All assets are associated with ***policies***
- If we had to define separate access control and usage policies for each ***asset*** that would be
  - Tedious and error prone
  - A security risk
  - Difficult for the ***data officer*** to set corporate standards
  - Overly complex when defining different policies for the same ***asset*** (e.g., for different audiences)
- The ***contract definition*** solves these issues

Contract Definition

- Top-down design where policies are “attached” to assets
  - Decouple policy authoring from particular assets
  - Provides flexibility and can be simplified when needed

- Contains
  - **Access control policy** (private) - “my partners”
  - **Contract usage policy** (advertised) – “data must stay in Europe”
  - **Asset selector** – “applies to these assets…”

Publishing data

The ***data officer*** defines a ***contract definition*** in the EDC system

This ***contract definition*** is for all parts of the ***asset*** (***asset selector***)

- Example: Can be accessed by only a given member company's partners (***access policy***)
- Example: Must be stored in Europe and used only for maintenance purposes (usage policy)

The ***data owner*** creates an **asset entry** in the EDC system

The ***asset entry*** is not the actual ***asset***, rather it points to where the ***asset*** is stored (e.g., Object Storage)

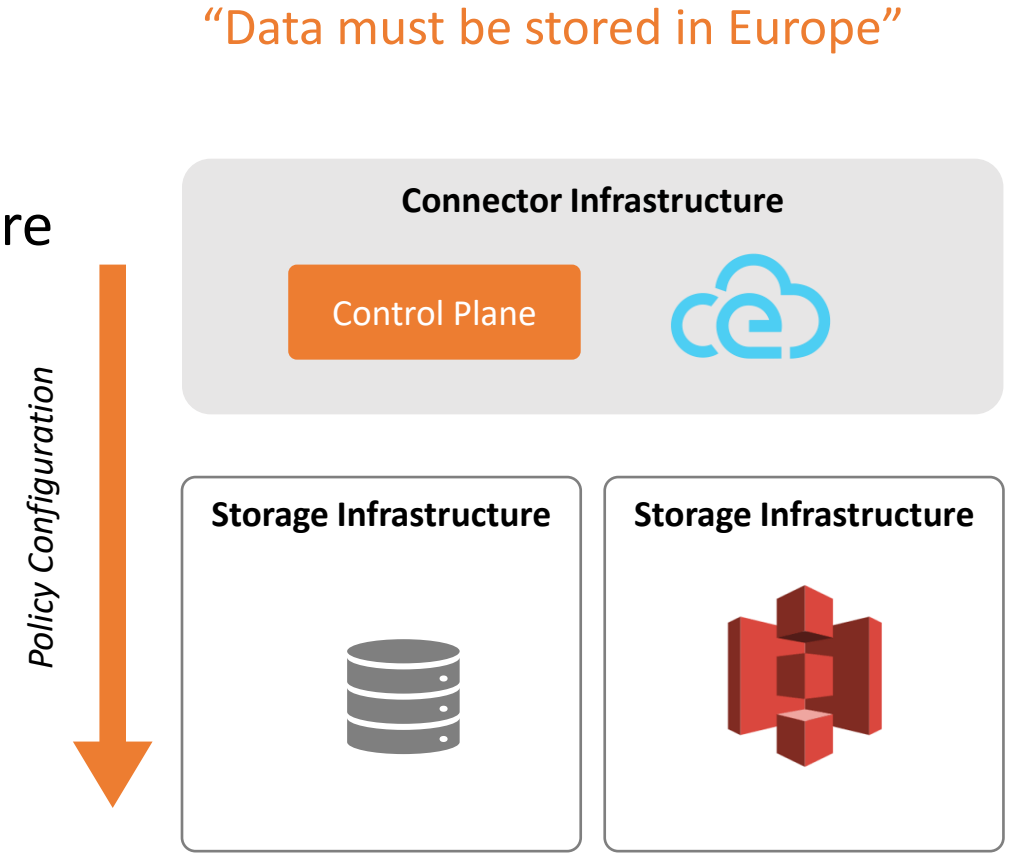
The EDC automatically "associates" the ***asset*** with ***contract definitions*** in the system

The ***asset*** is now available to other ***participants*** that satisfy the policies contained in associated ***contract definitions***

ECLIPSE INCUBATION

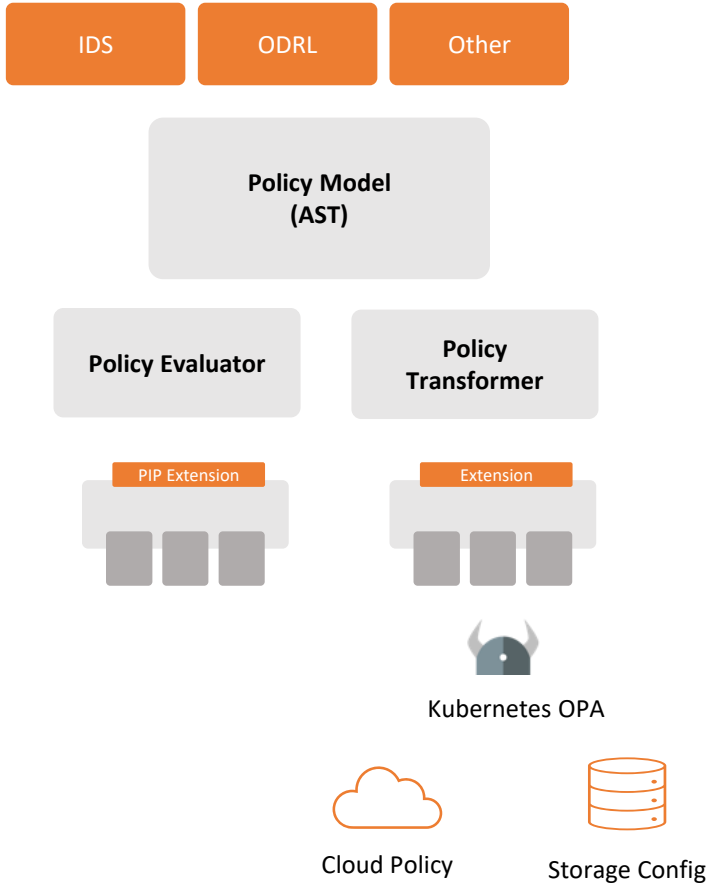
Runtime policy enforcement principles

- There is no single way to enforce policy
  - Lenient vs strict requirements
  - Data traverses diverse compute infrastructure
  - Some obligations cannot be automated
- Requires wholistic coordination
  - May reach to all levels of your technical infrastructure



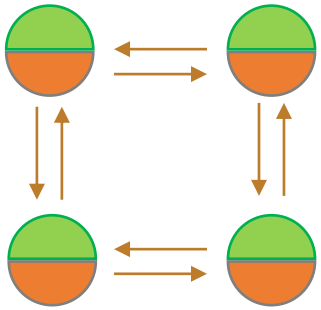
Runtime evaluation: The Policy Engine

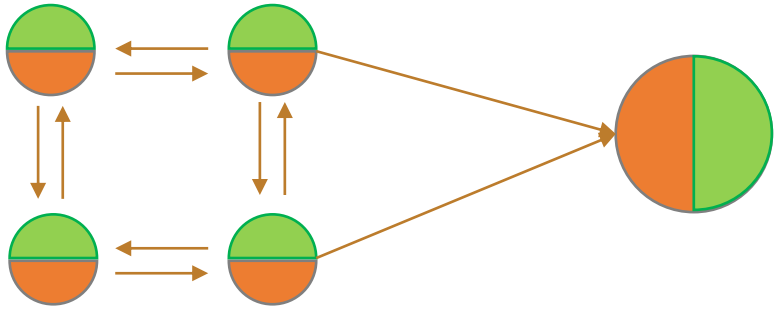
- Extensible evaluation engine
- Can support vertical enforcement
- Parses policy syntax into an internal Policy Model AST
- Evaluators and transformers to enforce policy
  - Evaluators can make policy decisions, e.g. is a connector authorized
  - Transformers can create and deploy policy to different levels
    - OPA, storage, etc.
  - Contributed as runtime extensions



Federated Catalog Services

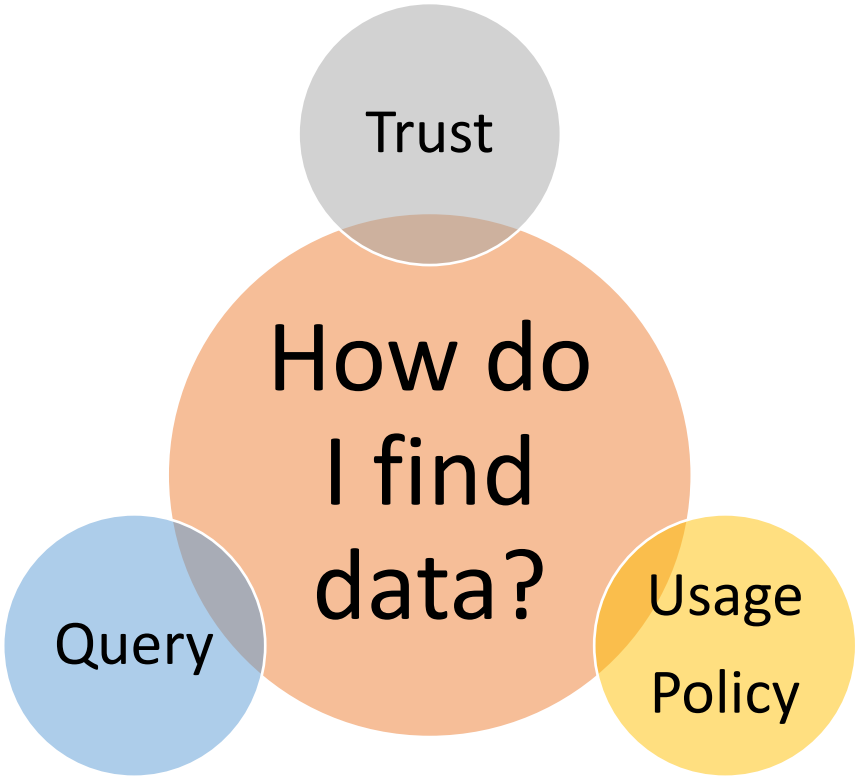
Securely Publishing Data

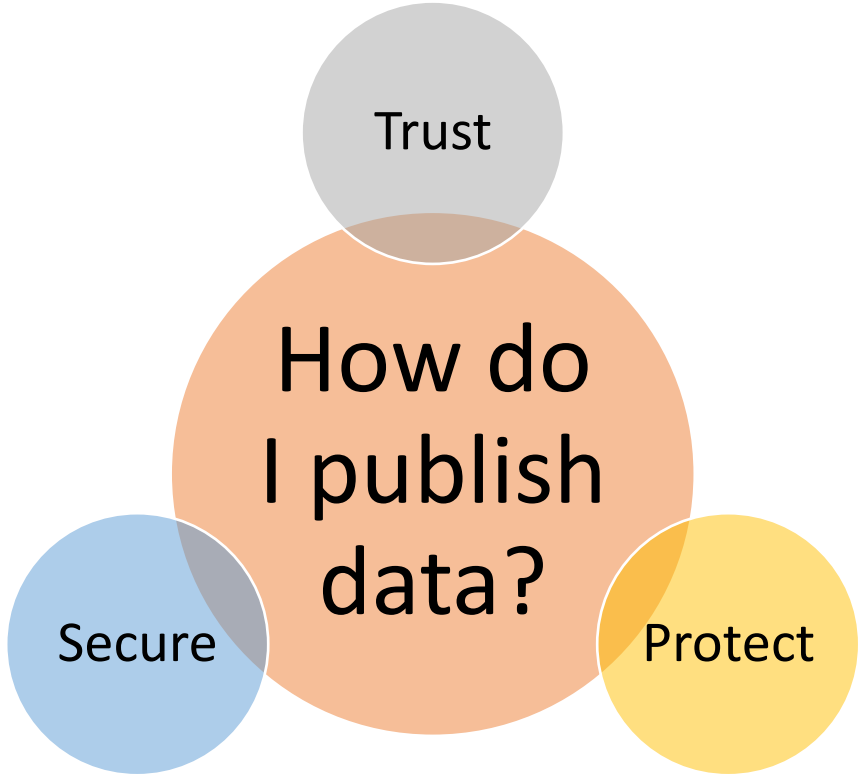




ECLIPSE
INCUBATION

Federated Catalog Services

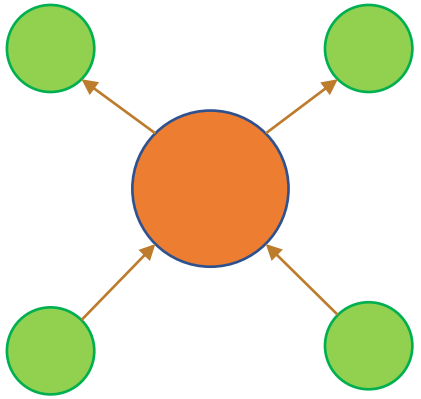




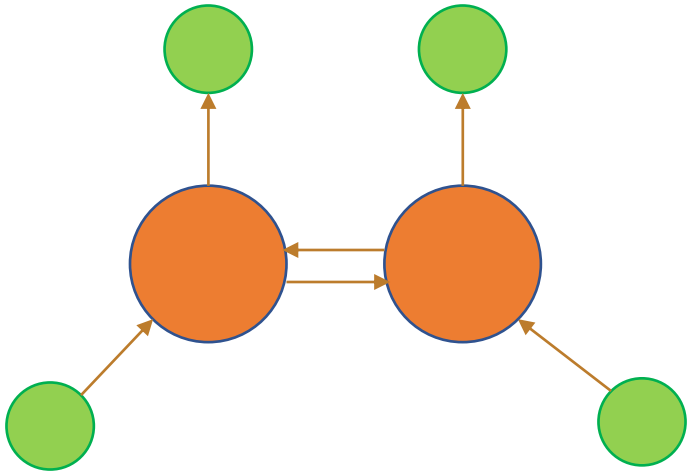
ECLIPSE INCUBATION

Fully- and semi- centralized catalog architectures

- Require a broker where participants publish their catalogs



Centralized Broker Dataspace



Semi-Centralized Dataspace

Common issues with centralized catalog architectures

- Data visibility and sovereignty
  - Is it acceptable for a third-party to have access to an organization’s data catalog?
  - Is it acceptable for an organization to rely on a third-party to advertise its data?
  - Can a third-party catalog provider properly enforce an organization's access rules?
- Reliability and scalability
  - In fully-centralized systems, what happens when the catalog is down?
  - In semi-centralized systems, how can replication-at-scale be managed?

Federated Catalog Services

- Solves the problems of data visibility and enterprise scalability &reliability
- Federated Cache Crawler (**FCC**)
  - Crawls and caches other participant catalogs on a periodic basis
  - Data queries execute against the local cache
- Federated Cache Node (**FCN**)
  - Advertises assets to other FCCs
  - Enforces access control through ***access policy*** and ***contract usage policy***

"Only allow my partners to access this data"

Access Policy

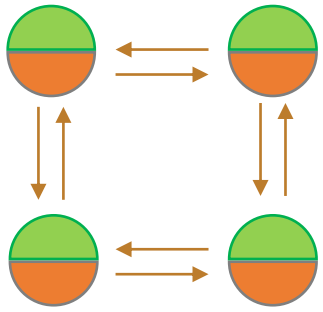
"Organizations can only store this data in Europe"

Usage Policy

ECLIPSE INCUBATION

Federated Catalog Services architecture

- Each node consists of a Federated Cache Node **(FCN)** and a Federated Cache Crawler **(FCC)**
- The FCN makes its asset catalog available to other participants
- The FCC crawls other FCN instances on a periodic basis and caches the results

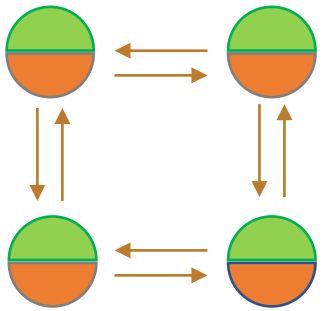


ECLIPSE
INCUBATION

The issue of data visibility

- FCC presents its identity and credentials to an FCN
- The FCN uses the same modules as the Connector to run policy access and usage control checks to filter the returned assets
  - Organizations maintain control of their asset catalogs and access control
  - Through extensibility, organizations may also implement custom access control logic

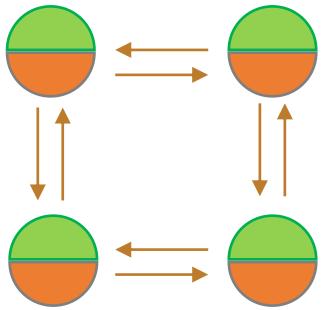
Exchange and validation of credentials at each point



ECLIPSE INCUBATION

Scalability and reliability

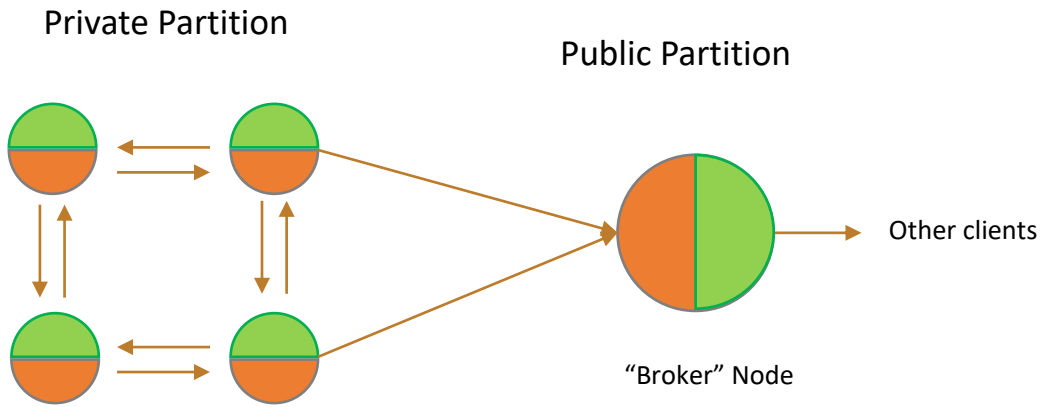
- Each FCC node caches its results
- Enables instantaneous distributed queries since asset catalogs are mirrored locally
  - Only assets the client node is entitled to view
- The dataspace becomes fault tolerant and resilient
  - If the origin FCN is down, the local cached copied will continue to work



ECLIPSE
INCUBATION

Deployments

- The FCN and FCC made be deployed in a connector process or as separate services (recommended)
- The crawler architecture is designed for peer-to-peer but can also support broker models or a hybrid combination
  - For example, a dataspace may have private data shared via a peer-to-peer partition and public data offered via a broker



Thank You