# Proposal: Integrating EdgeX Foundry and Project Alvarium

Trevor Conn (trevor.conn@dell.com)
Distinguished Engineer
26-Jul-2023

DELL Technologies

# Agenda

- Integration Goals
- Alvarium Quick Overview
- Integration Scope

# Integration Goals

- Alvarium is nearing its 2 year anniversary and we'd like to show an integrated demo with complimentary projects in LF-Edge
- Adoption has been somewhat difficult b/c we hear a lot of "That's a really good idea!" but then it seems hard for teams to understand where to start.
- Our hope is that a concrete example via integration will make the perceived value of Alvarium more real.
- In addition, the act of integrating into other platforms will provide Alvarium with feedback on our design in order to mature the project.
- At this stage we're not proposing Alvarium become an official, long-term integration with EdgeX. Although if the community likes what it sees, we'd be happy to collaborate on that.

# Alvarium Quick Overview

DELL Technologies

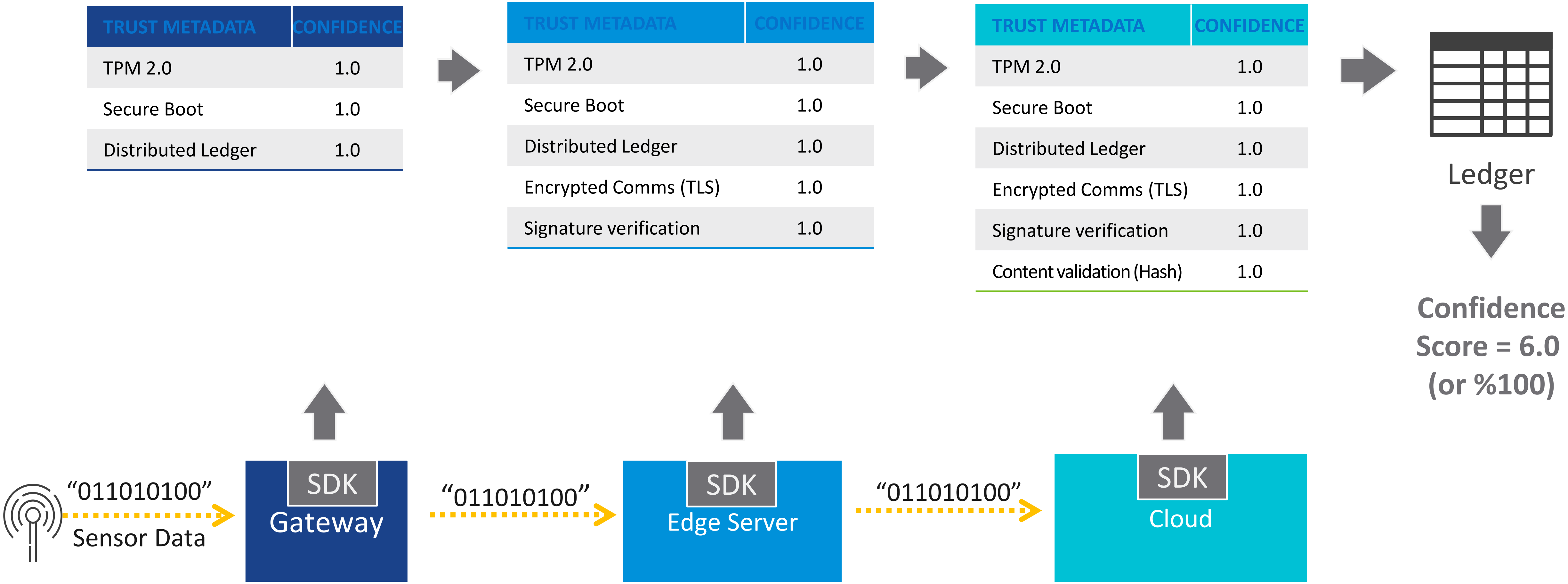
# Executive Summary

A description of the Alvarium concept and its relevance for modern use cases.

- Modern applications are extensively distributed
- Data is no longer a fixed asset stored in a silo.
- Data traverses the network and can be transformed along the way.
- Create metadata that attests verifiable authority at the origin of data
- Create metadata describing how data was handled as it traverses the eco-system
- Metadata is created at “trust insertion points”
- A measure of trust is calculated at each insertion point and can be weighted
- Trust can be rolled up into an overall confidence score for a piece of data
- A trust score may be used to govern system behavior or alert operators to an attack

# Example: Data annotated in traversal

The Alvarium code base is a lightweight SDK that annotates data streams (e.g., sensor data) with trust metadata and confidence scores, forming a Data Confidence Fabric (DCF)



| TRUST METADATA | CONFIDENCE |
|---|---|
| TPM 2.0 | 1.0 |
| Secure Boot | 1.0 |
| Distributed Ledger | 1.0 |

| TRUST METADATA | CONFIDENCE |
|---|---|
| TPM 2.0 | 1.0 |
| Secure Boot | 1.0 |
| Distributed Ledger | 1.0 |
| Encrypted Comms (TLS) | 1.0 |
| Signature verification | 1.0 |

| TRUST METADATA | CONFIDENCE |
|---|---|
| TPM 2.0 | 1.0 |
| Secure Boot | 1.0 |
| Distributed Ledger | 1.0 |
| Encrypted Comms (TLS) | 1.0 |
| Signature verification | 1.0 |
| Content validation (Hash) | 1.0 |

Ledger

**Confidence Score = 6.0 (or %100)**

# Alvarium SDK Interface

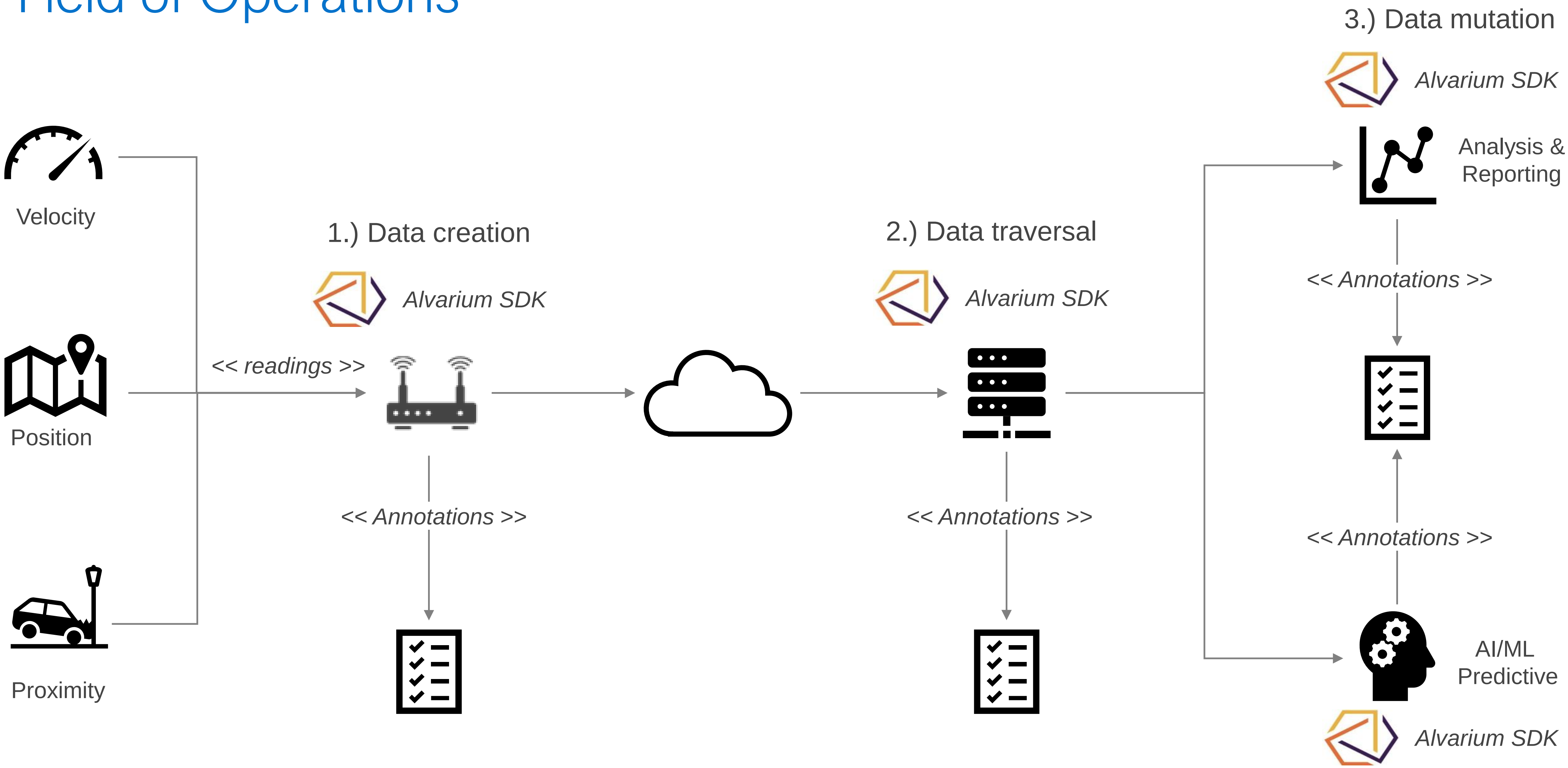
Refer here to Github documentation
https://github.com/project-alvarium/alvarium-sdk-go

1. Create() -- Used to register creation of new data with the SDK. Passes data through the SDK instance's list of annotators.
2. Mutate() -- Used to register mutation of existing data with the SDK. Passes data through the SDK instance's list of annotators.
3. Transit() -- Used to annotate data that is neither originated or modified but simply handed from one application to another.
4. Publish() -- Used to annotate data that is neither originated or modified but **before** being handed to another application.
5. BootstrapHandler() -- SDK instance method. Ensures clean shutdown of the SDK and associated resources.

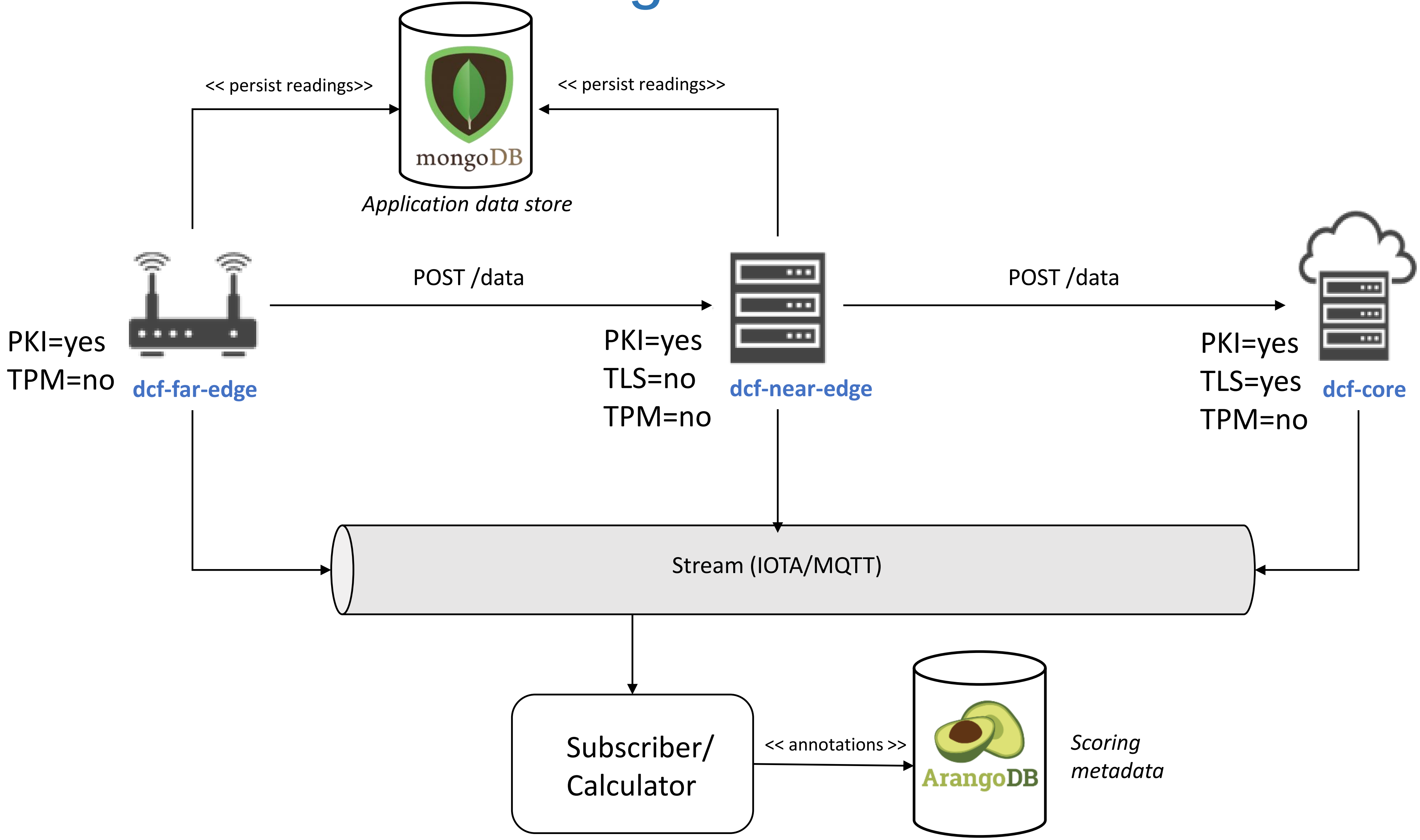
## Alvarium annotation schema

```
// Annotation represents an individual criterion of evaluation in regard to a piece of data
type Annotation struct {
        Id          ulid.ULID      `json:"id,omitempty"`        // Id should probably be a ULID -- uniquely identifies the annotation itself
        Key         string         `json:"key,omitempty"`       // Key is the hash value of the data being annotated
        Hash        HashType       `json:"hash,omitempty"`      // Hash identifies which algorithm was used to construct the hash
        Host        string         `json:"host,omitempty"`      // Host is the hostname of the node making the annotation
        Kind        AnnotationType `json:"kind,omitempty"`      // Kind indicates what kind of annotation this is
        Signature   string         `json:"signature,omitempty"` // Signature contains the signature of the party making the annotation
        IsSatisfied bool           `json:"isSatisfied"`         // IsSatisfied indicates whether the criteria defining the annotation were fulfilled
        Timestamp   time.Time      `json:"timestamp,omitempty"` // Timestamp indicates when the annotation was created
}
```

# Field of Operations

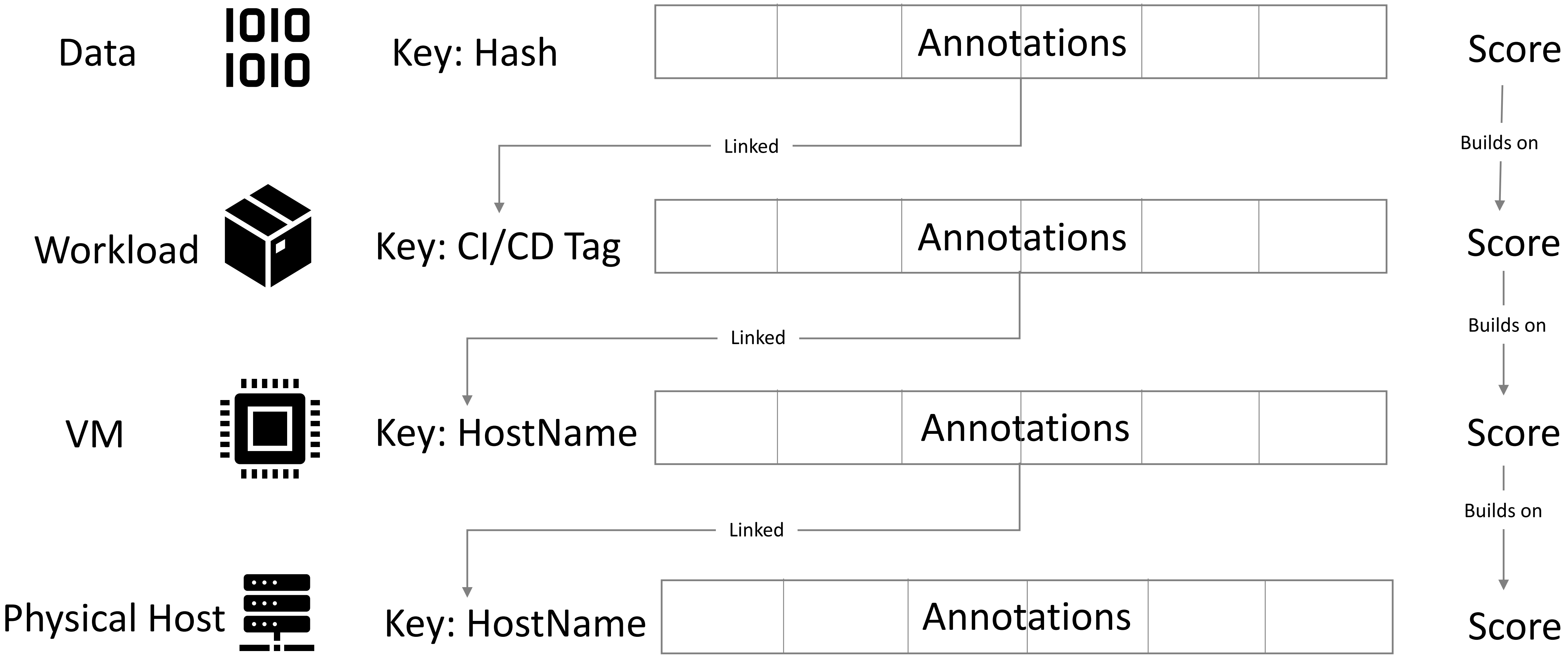


# Application Data vs. Scoring



# Integration Scope

# Data Confidence Graph (Conceptual Diagram)



Data

Key: Hash

Annotations

Score

Linked

Builds on

Workload

Key: CI/CD Tag

Annotations

Score

Linked

Builds on

VM

Key: HostName

Annotations

Score

Linked

Builds on

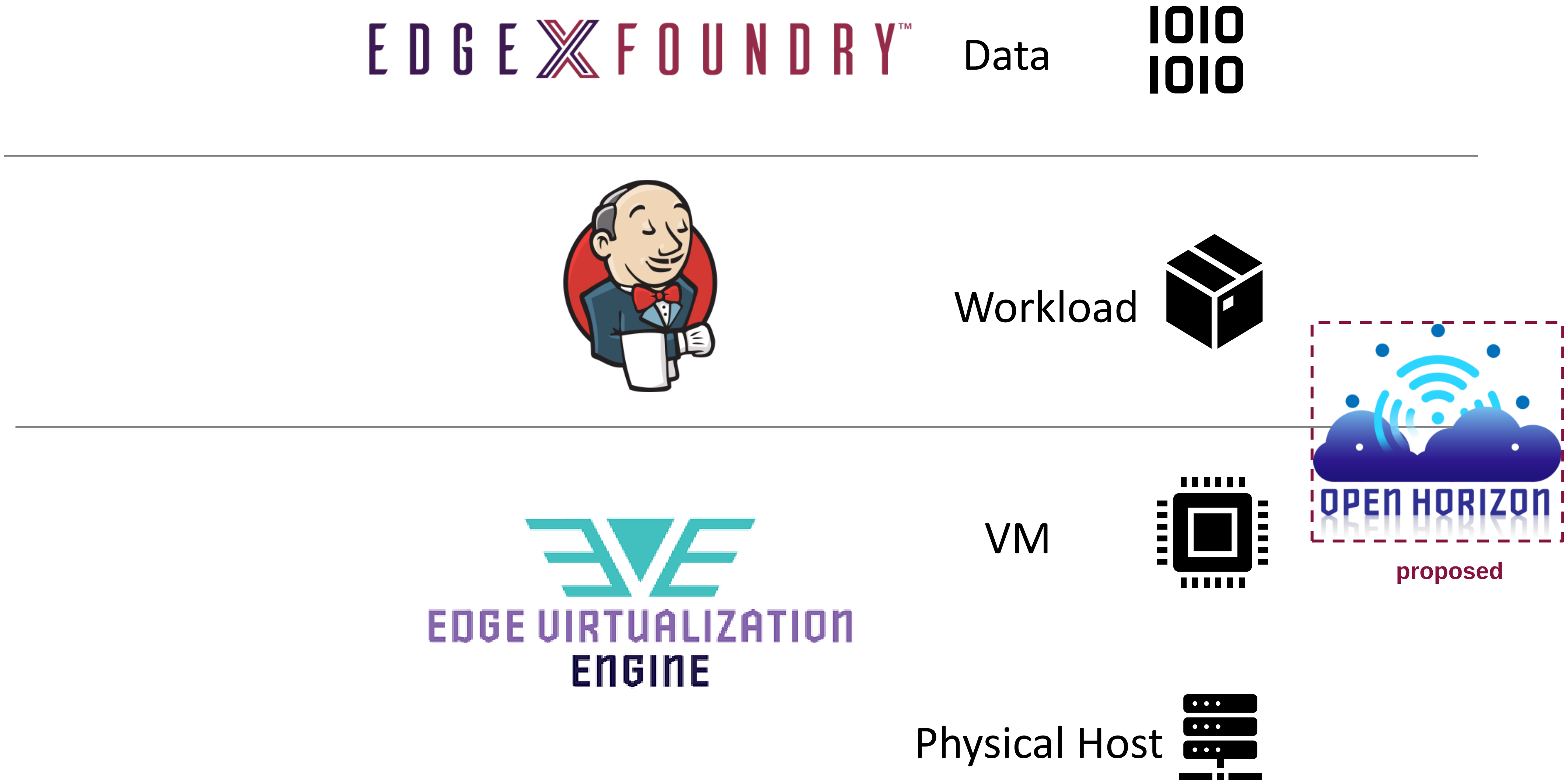
Physical Host

Key: HostName

Annotations

Score

# Data Confidence Graph Toolchain



Data

Workload

VM

Physical Host

proposed

# Data Confidence Graph Annotations

- Data Signature
- TLS
- Content Checksum

EDGEXFOUNDRY™

Data

1010
1010

- Source code integrity
- SBoM / Dependency Check
- Artifact Checksum

Workload

- TPM Validation

EDGE VIRTUALIZATION ENGINE

VM

OPEN HORIZON

proposed

Physical Host

# Data Confidence Graph Orchestration

- Data Signature
- TLS
- Content Checksum

EDGEXFOUNDRY™

Data

- Source code integrity
- SBoM / Dependency Check
- Artifact Checksum

Workload

- TPM Validation

EDGE VIRTUALIZATION ENGINE

VM

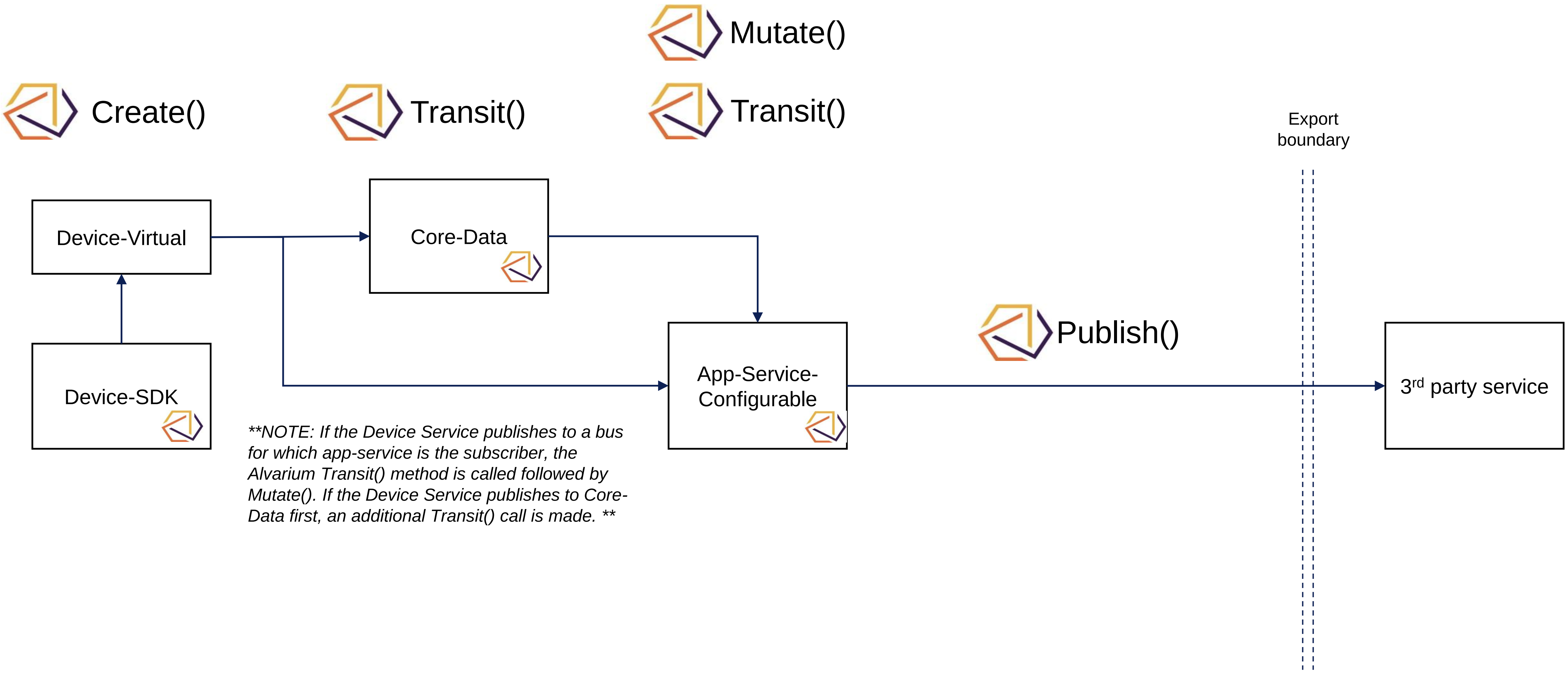
Physical Host

OPEN HORIZON

**proposed**

Open Horizon orchestrates deployment according to workload confidence threshold configured via policy

# High-Level Integration Points in EdgeX



Mutate()

Create()

Transit()

Transit()

Export boundary

Device-Virtual

Core-Data

Device-SDK

App-Service-Configurable

Publish()

3rd party service

***NOTE: If the Device Service publishes to a bus for which app-service is the subscriber, the Alvarium Transit() method is called followed by Mutate(). If the Device Service publishes to Core-Data first, an additional Transit() call is made. ***

Indicates which apps are directly integrated with Alvarium SDK

# To Be Continued…