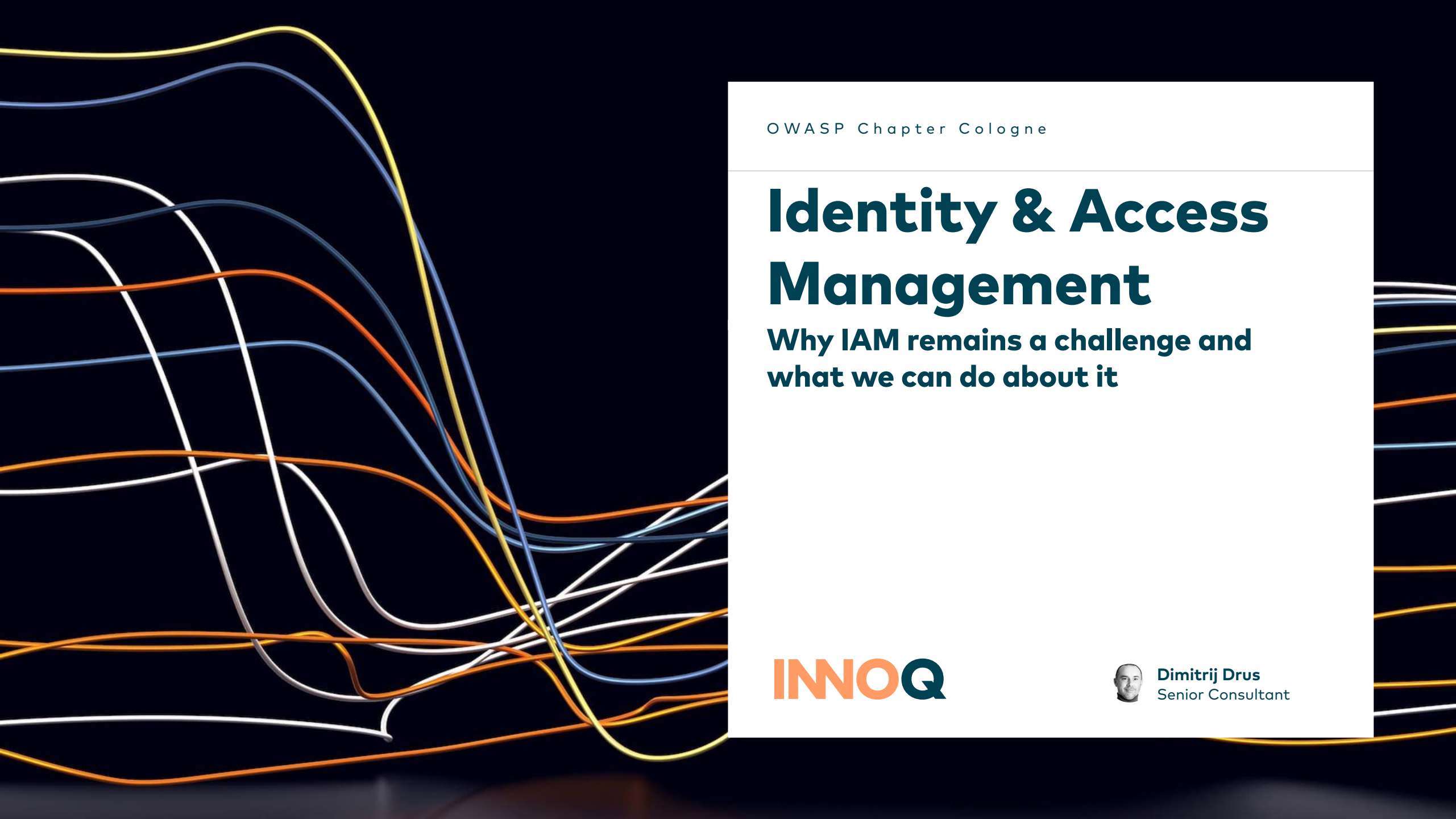


OWASP Chapter Cologne

Identity & Access Management

Why IAM remains a challenge and what we can do about it

INNOQ

Dimitrij Drus
Senior Consultant

We're focusing on features first. Security comes later.

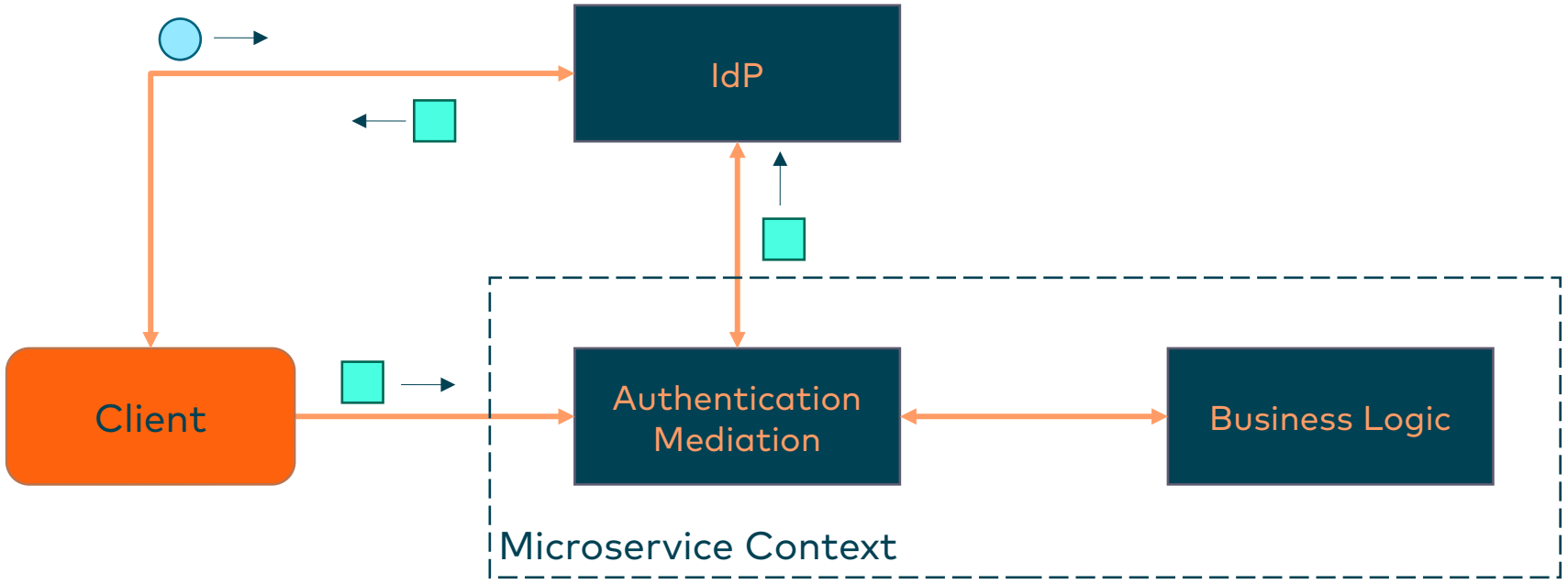
We added some security. What else before go-live?

We did care about security, but ...

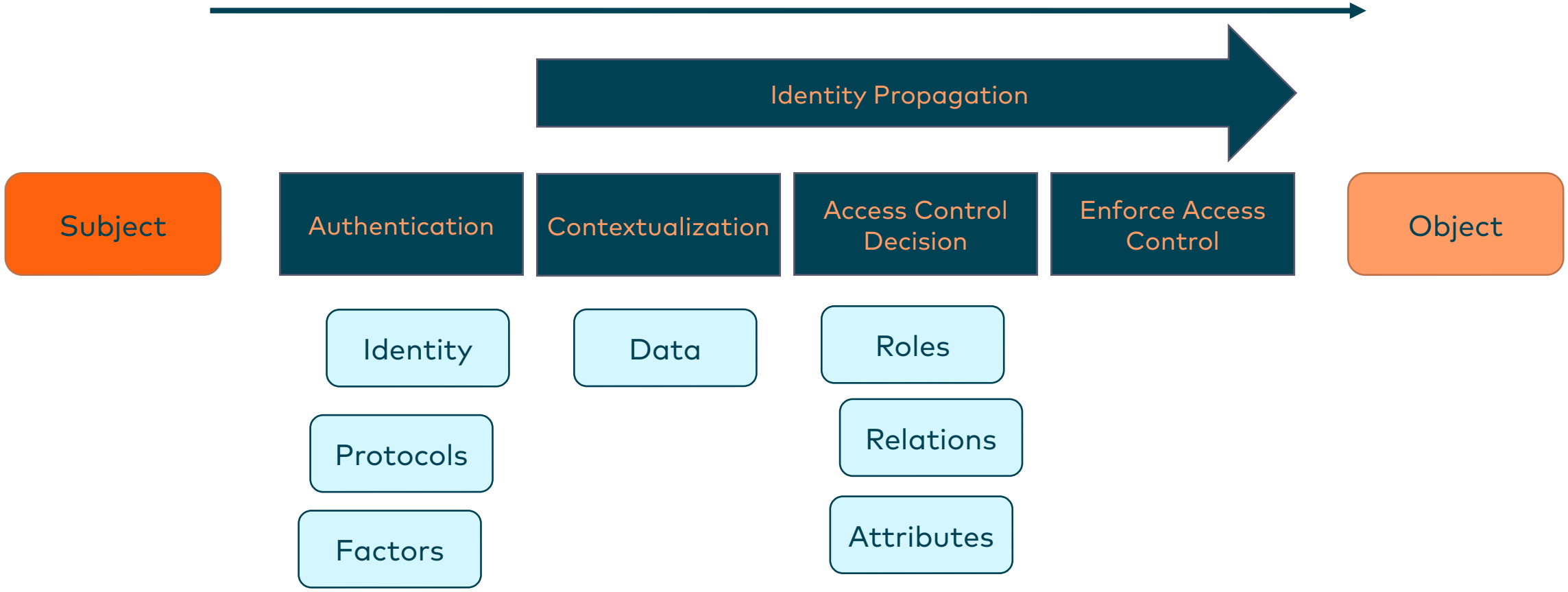

roles
scopes
access
token

Authentication First

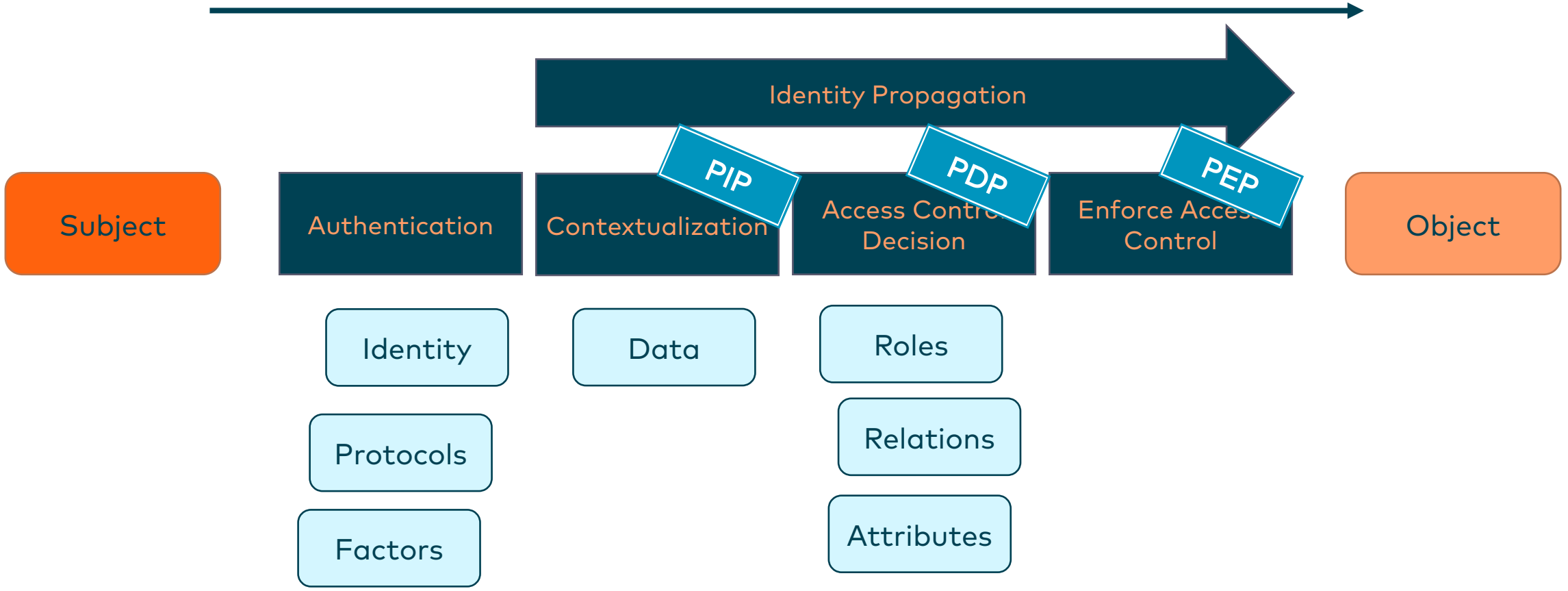
Starting with Authentication Corset



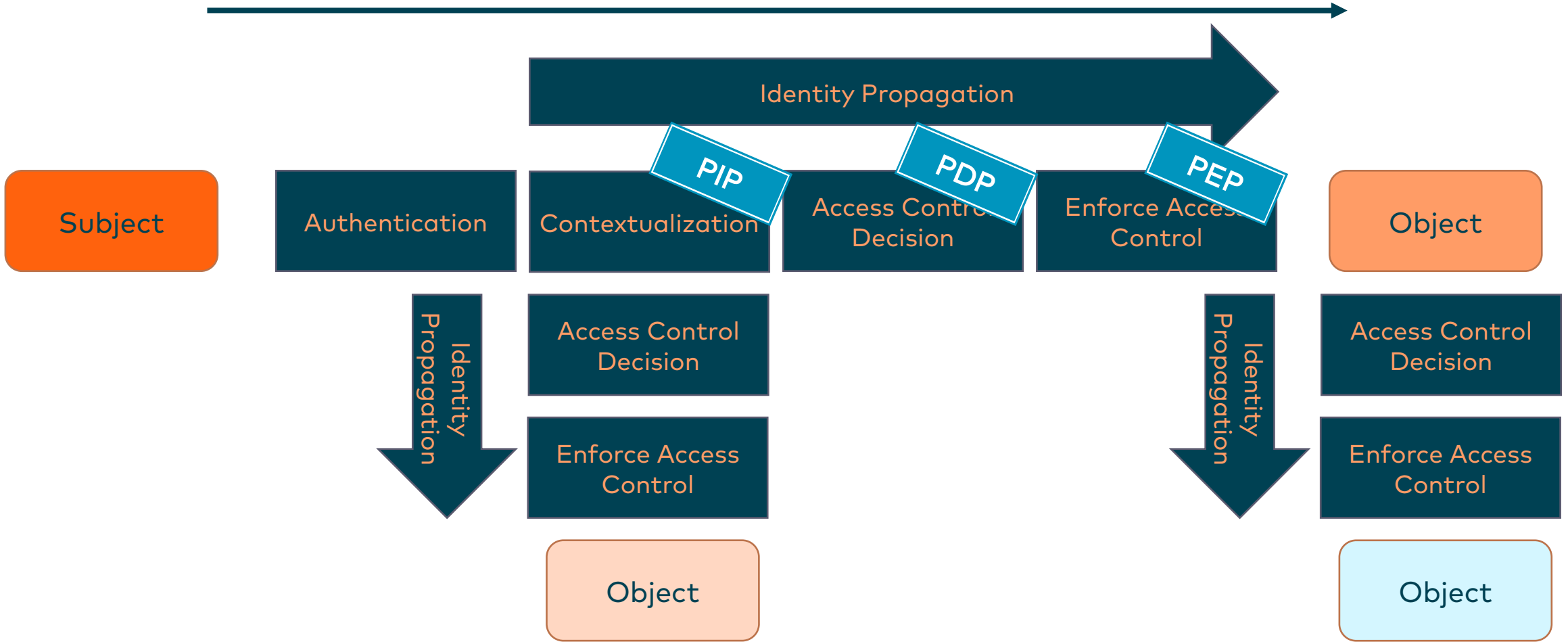
Request's Path



Request's Path



Request's Path



Resolving the Corset

Authorization Policies & Lifecycle

- **Hardcoded Policies (Coupled with Service Code)**
  PEP + PDP „logic“ is hardcoded
- **Declarative Policies (Decoupled from Service Code)**
  Policies are evaluated by a PDP and enforced in a PEP

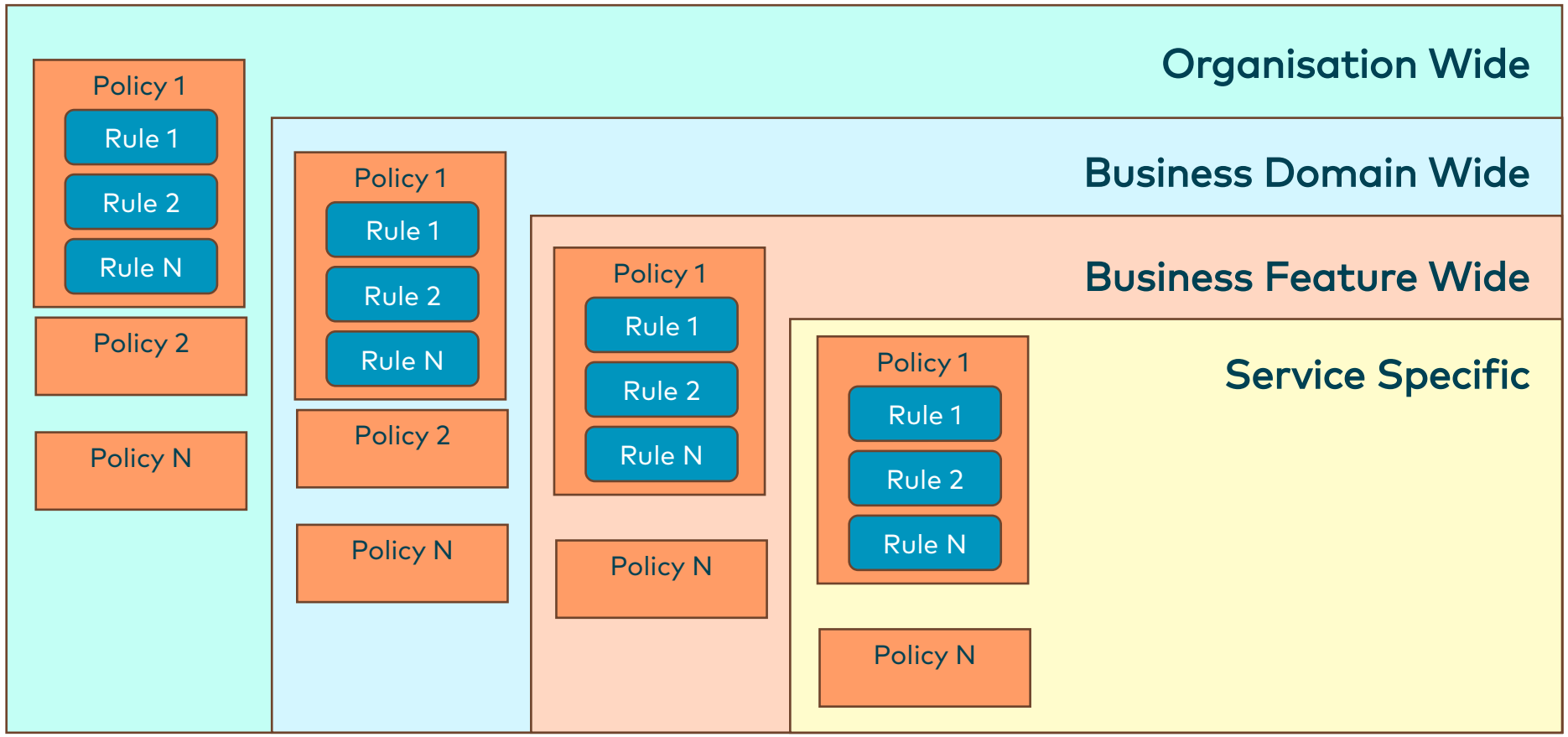
What are Authorization Policies?

- **Business Rules**
  Specific business features
- **Regulatory Obligations**
  Independent from business feature set
- **Security Requirements**
  Independent from business feature set

Authz Policies are High Stake Assets

- **Data Breaches**
  As a result of granting access when you shouldn't
- **Business Breaker**
  If access is revoked incorrectly

Authorization Policies Width & Depth



Cool, let's add a PDP to the mix ;)

Policy-Input Data Characteristics

- **Data Locality**
  Boundaries of data relevance
- **Data Cardinality**
  Number of distinct attributes across all subjects or resources
- **Data Freshness**
  Maximum acceptable delay between an attribute value changing, and that change being reflected in authorization decisions

Policy-Input Data Distribution

- **On-Demand Data Pull**
  The PDP fetches data from PIPs at the time of policy evaluation
- **Out-of-Band Data Push**
  Data is proactively sent to the PDP in advance, and stored in memory or a local data store
- **Request Time Data Injection**
  Required data is passed directly in the request from the PEP to the PDP
- **Embedded Data**
  Data is baked directly into the PDP's configuration

Policy Ownership

- **Microservice Team**
  Policies authored and maintained by the team of a microservice
- **Domain Level**
  Policies shared across services within a business domain
- **Central (Organization Level)**
  Policies governed by a central security, compliance, or platform team

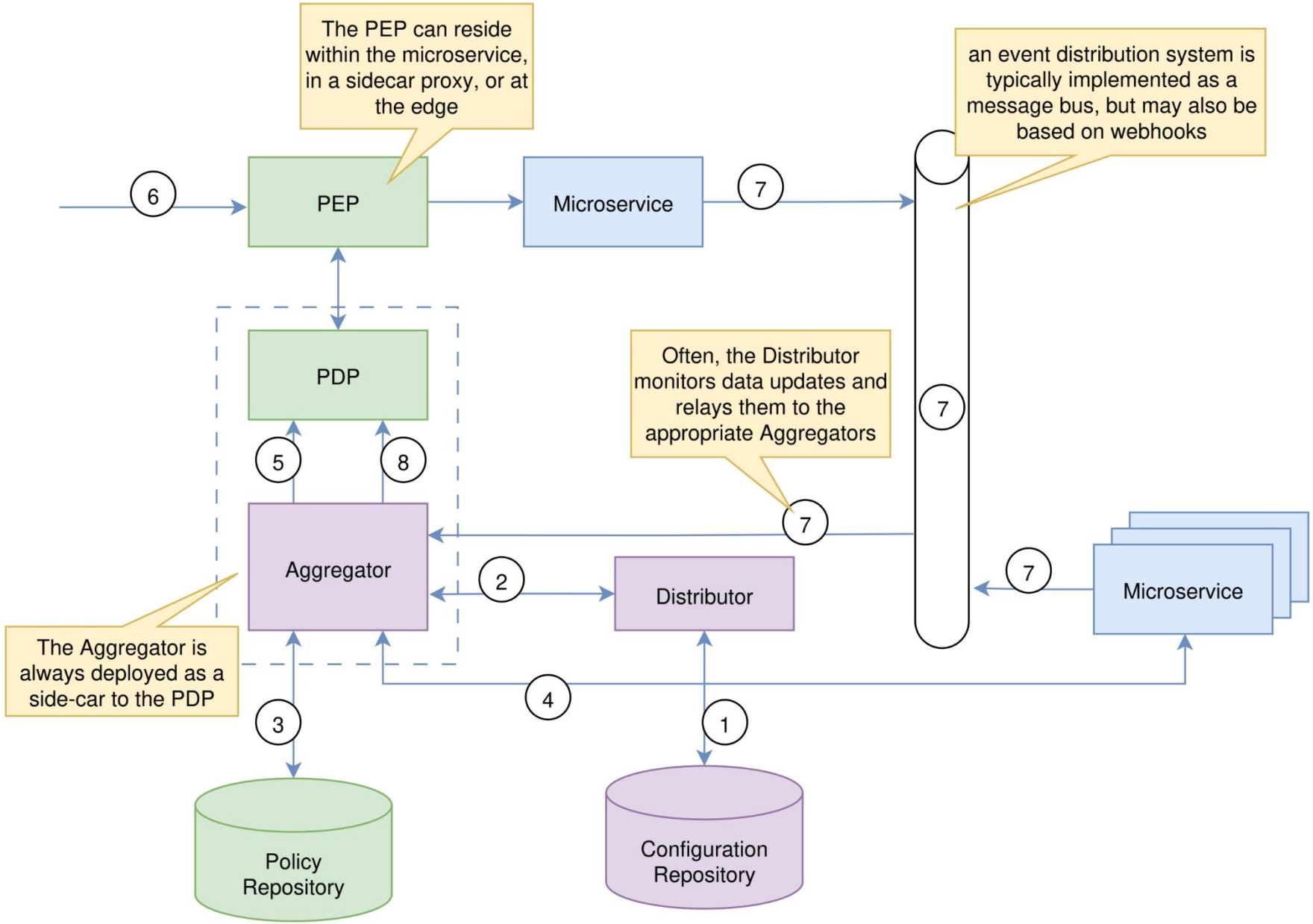
Policy Change Latency

- **Immediate**
  Policies must take effect as soon as they are changed
- **Fast**
  Policies should be applied within hours to days
- **Delayed**
  Policies can be applied on a longer timescale

Handling Policy Results

- **PDP as Filter, aka Brute-Force Lookup**
  PEP fetches data from a PIP and checks each item with the PDP for access
- **Authorized Data Set**
  PDP returns a complete set of allowed resources
- **Authorization Filter**
  PDP returns a filter expression, which the PEP applies to retrieve only authorized data.

Effect on Macro Architecture

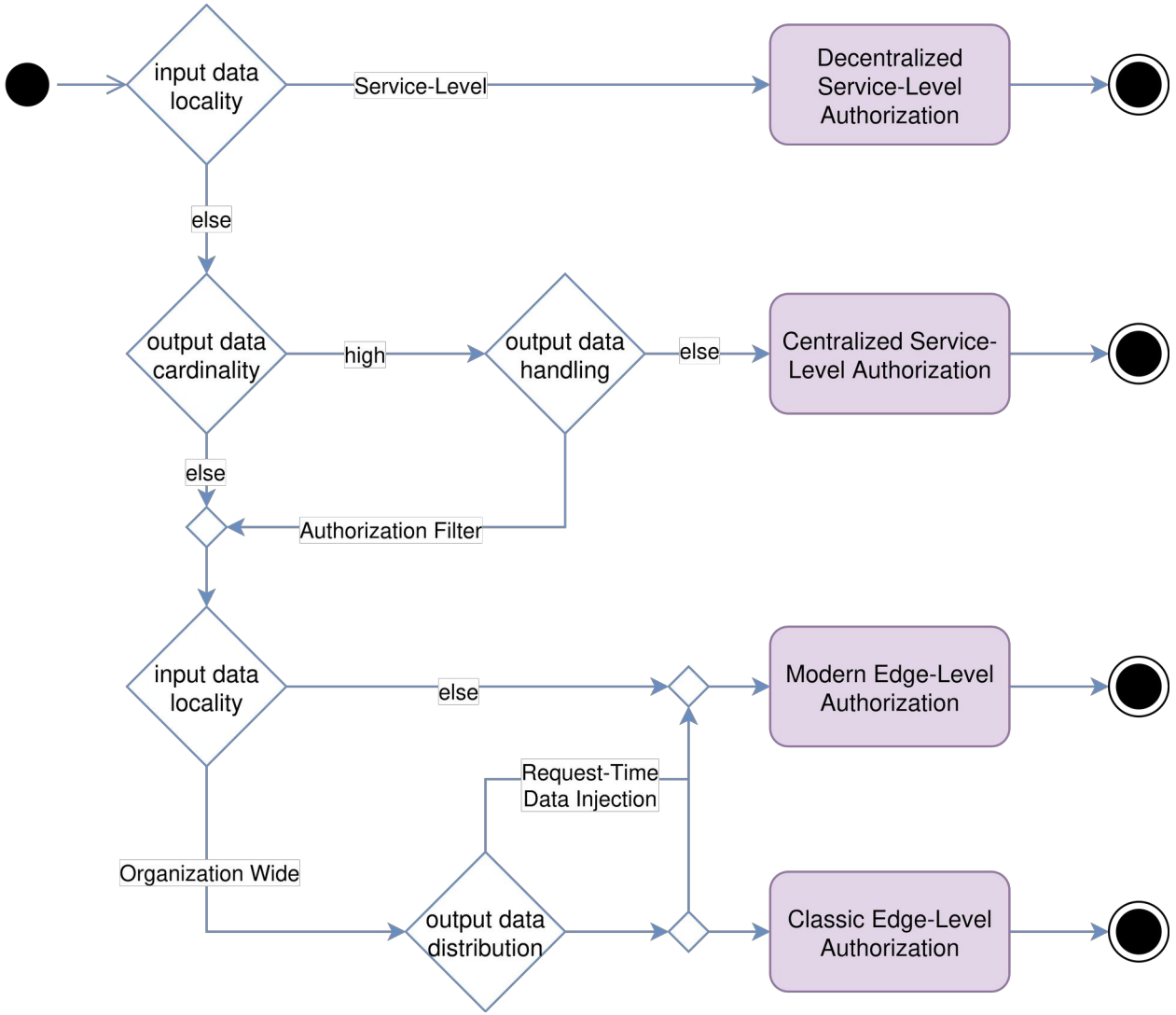

The PEP can reside within the microservice, in a sidecar proxy, or at the edge
an event distribution system is typically implemented as a message bus, but may also be based on webhooks
6
PEP
Microservice
7
PDP
Often, the Distributor monitors data updates and relays them to the appropriate Aggregators
7
5
8
7
Aggregator
2
Distributor
7
Microservice
The Aggregator is always deployed as a side-car to the PDP
4
3
1
Policy Repository
Configuration Repository

Which Authorization Pattern to use?

Authorization Patterns

- **Decentralized Service-Level Authorization**
  Everything (PEP, PDP, Policies) hardcoded in microservice code
- **Centralized Service Level Authorization**
  PDP externalized, PEP hardcoded in microservice code
- **Classic Edge-Level Authorization**
  PDP externalized, PEP hardcoded in the edge component
- **Modern Edge-Level Authorization**
  PDP externalized, PEP controlled by microservice code

Authz Patterns Recommendations



Time to think about Authentication
&
Identity Propagation

Identity Propagation Patterns

- **External Identity Propagation**
  Passing externally received authentication data/token downstream
- **Simple Service-Level Identity Forwarding**
  Passing simple IDs downstream without ability to verify it
- **Token Exchange-Based Identity Propagation**
  Trading externally received token for another, which is then used for downstream communication
- **Protocol-Agnostic Identity Propagation**
  Verification of any authentication data at the edge, creation of a new token abstracting away used protocols, which is the passed downstream

Authentication Patterns

- **Service-Level Embedded Authentication**
  Each service is responsible for identity management, incl. credentials verification
- **Service-Level Code-Mediated Authentication**
  Identity management externalized, microservice code cares about related flows
- **Service-Level Proxy-Mediated Authentication**
  Identity management externalized, a side-car proxy cares about related flows, microservice code receives the result, e.g. a token
- **Edge-Level Authentication**
  Identity management externalized, edge-proxy cares about related flows, microservice code receives the result, e.g. a token
- **Kernel-Level Authentication**
  Cares about mutual workload authentication on Layer 3


primer

Thank you

A shameless advertisement :)
Take a look at heimdall