ECLIPSE DATASPACE CONNECTOR

BASICS – SETUP - DEMONSTRATION

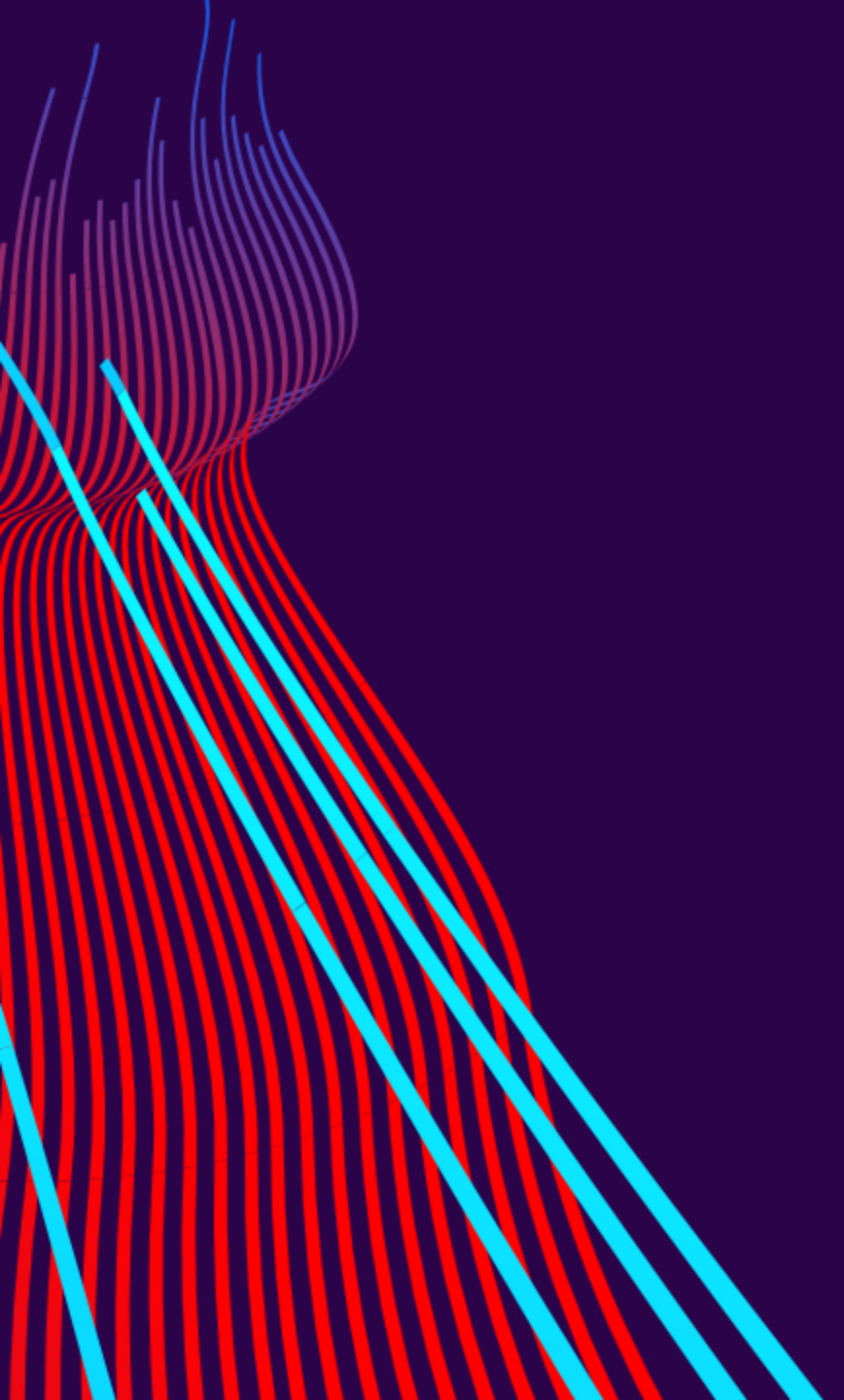
Maximilian Peresunchak

Seminar "StayInSync" – 16.05.2025

StuPro - Universität Stuttgart

SO MANY QUESTIONS

- What is the problem that the EDC is supposed to solve?
- What is the EDC even?
- What are the core principles and modules of the EDC?
- How do we set up a local testing environment?
- How do we exchange data between participants?
- Security?!



WHAT ARE THE PROBLEMS?

Data Silos

Usually data is stored in separated and privately owned data silos of companies

Traditional APIs

Defined end-to-end where every single new connection needs to be exposed.

Trust

As soon as data is shared, you have no control over the usage of that data and who will be able to see it

Control

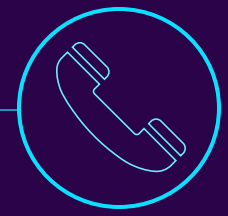
It is not possible to set up use policies and visibility polices. As soon as an API is exposed, there is no way to track visibility and usage

REAL-LIFE SCENARIO

slow

slower

very slow



Defect is spotted

Lots of customer service reports are coming in that one and the same component is failing in a specific car model. (Let's say it is an airbag trigger system)

Find the defect

We need to figure out, which supplier is responsible for the defect and thus need to reconstruct the entire supply chain

Call everyone

We figured out the supplier, but they have even more suppliers behind them. E-Mails, Calls, delays and all the data is scattered

Initiate Callback

As soon as we found out, which cars need to be called back for maintenance, we might call back too many or not enough

WHAT WE NEED

data sovereignty

The data belongs to us, so it should stay that way after sharing it with others

Trust and Enforcement

We want to have an eco system of authorized actors with enforcable agreements

Interoperability

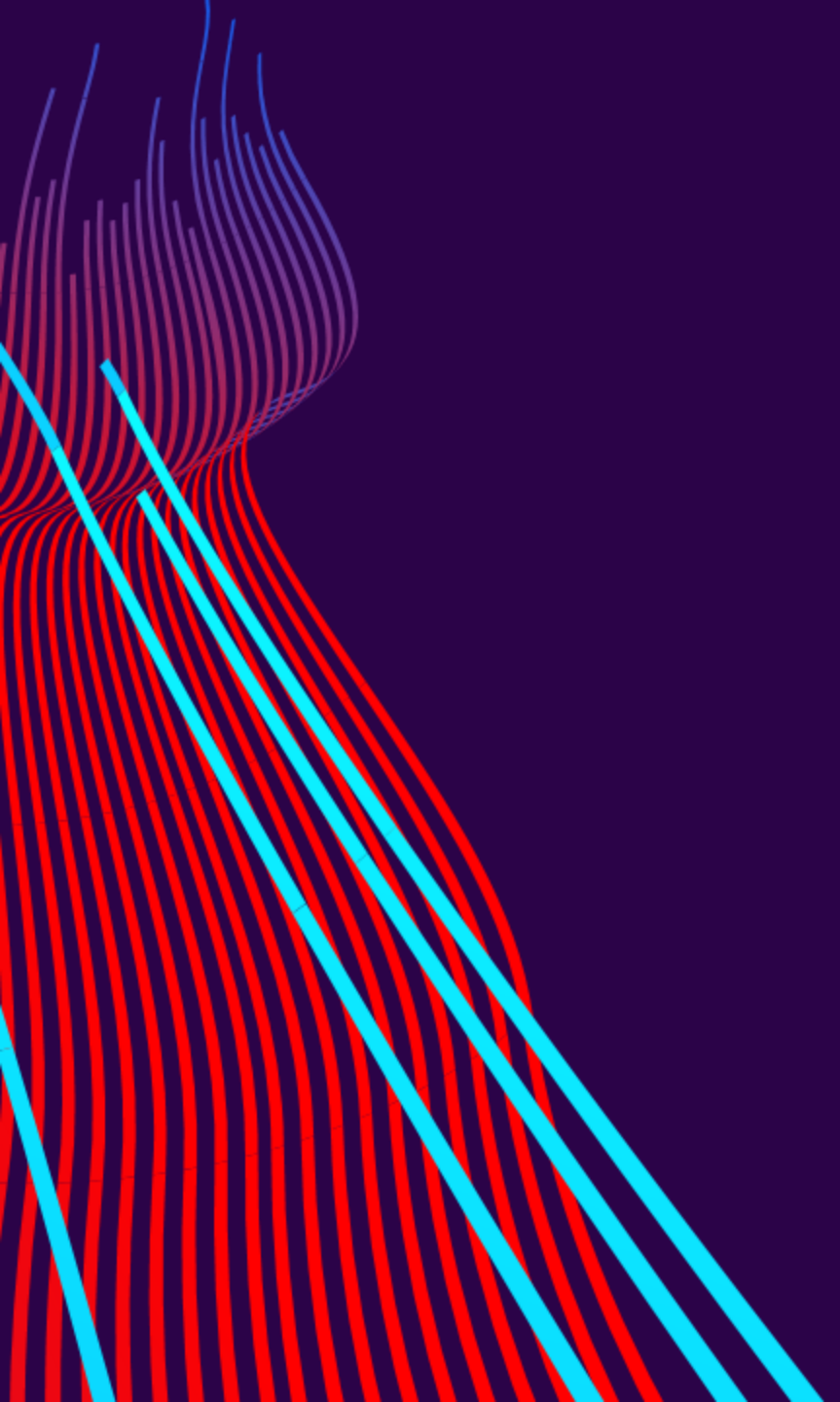
Every participant should be speaking the same language. We need standards.

Usage Policies

We want to be able to define our policies to see and use our data.

THE SOLUTION - EDC

Open-Source Framework written by the Eclipse Foundation for the Sovereign and interoperable data exchange



WHAT WE GET FROM THE EDC

Decentralized

There is no central data storage or authority. All participants run their own connectors in the data space.

Extensible

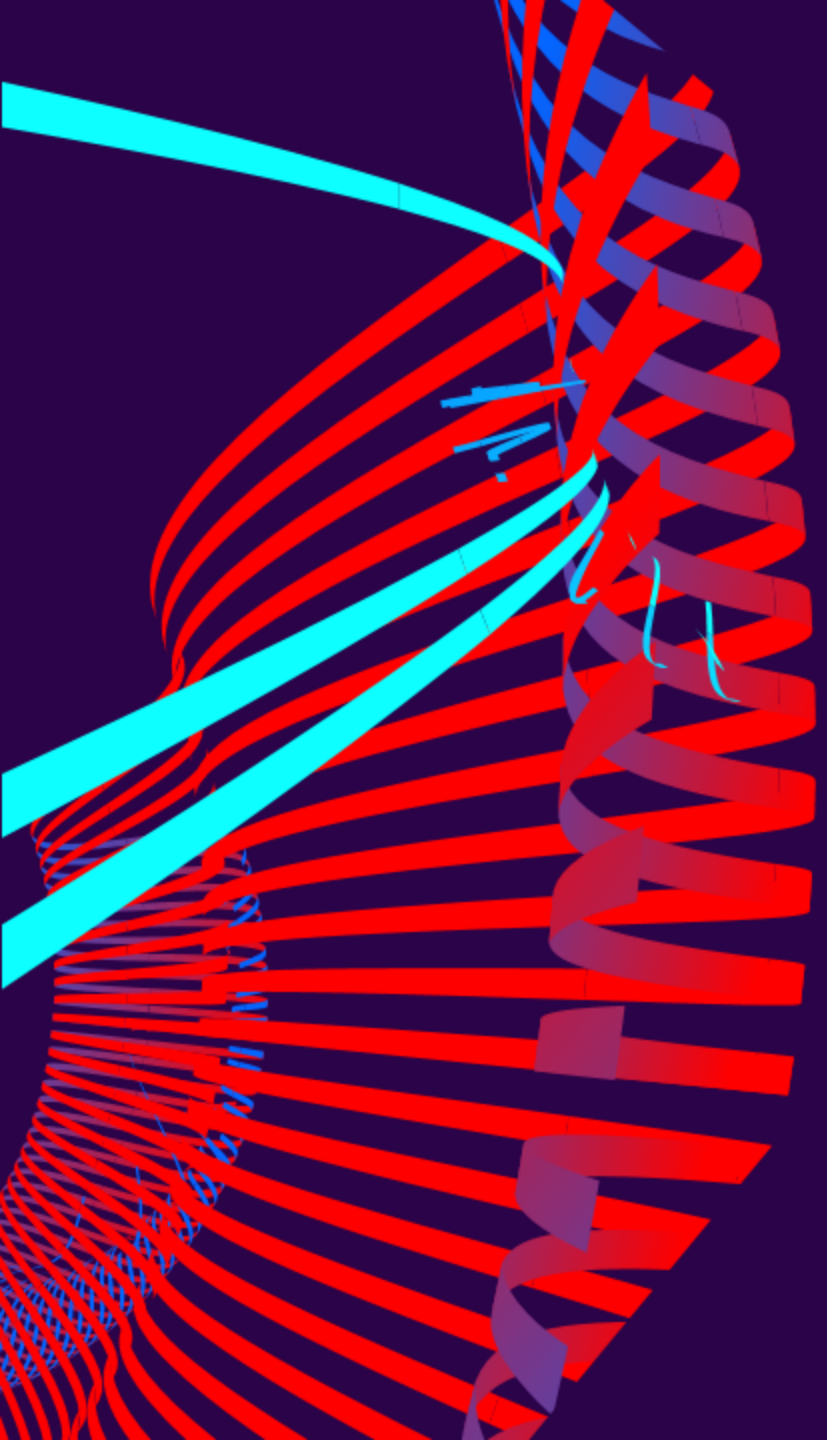
Modular architecture allows for personalized extensions like IdentityManagement or TransferTypes

Customizable

"I only want participants with a CO2-footprint of less than X to see my data"

Enforcably Rule-Based

Data access is controlled by machine readable rulesets automatically



HOW ABOUT ARCHITECTURE?

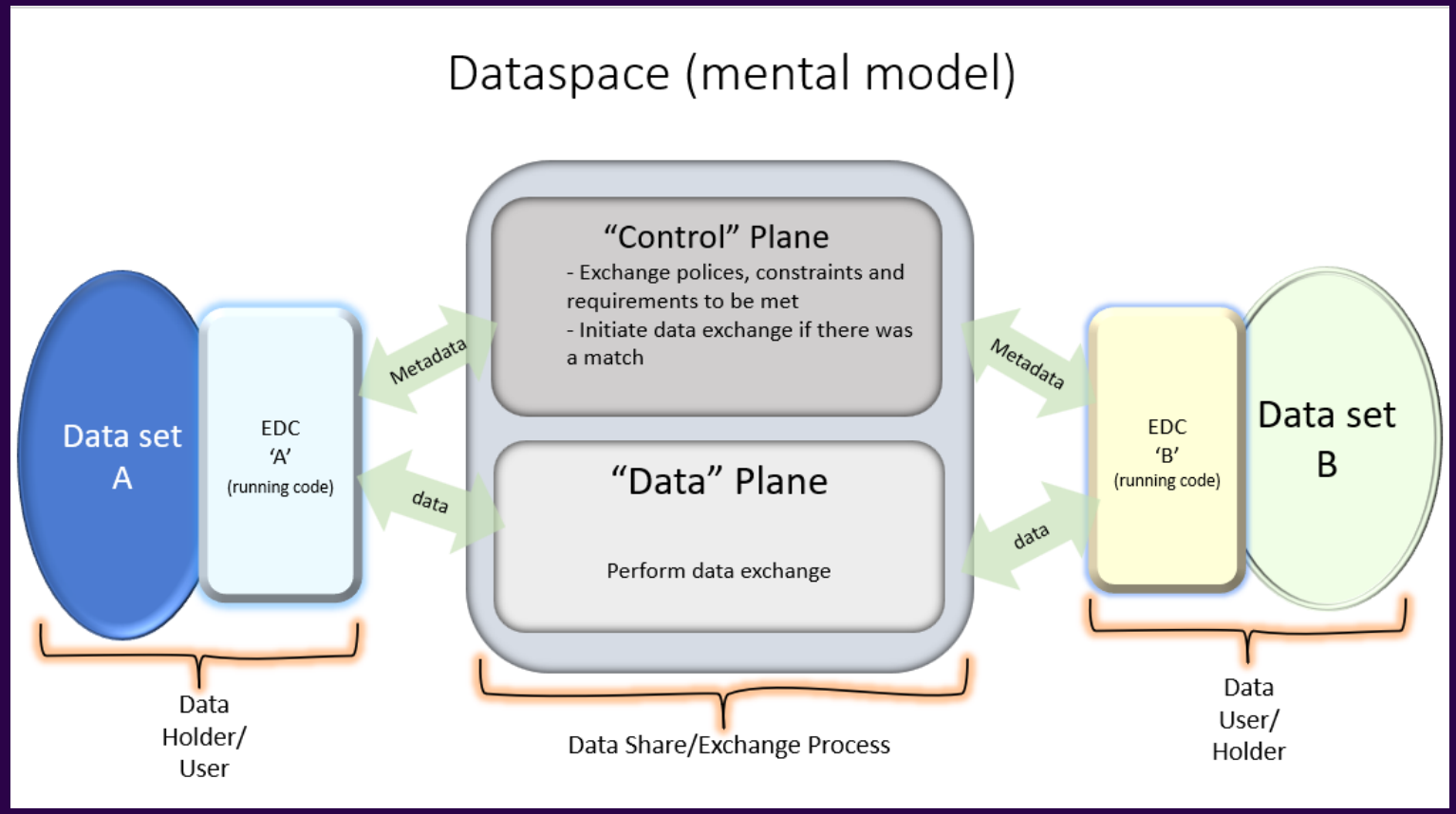
CONTROL PLANE

- Acts as the "brain" of the Connector
- Communicates through metadata for offers, negotiations and policy management

DATA PLANE

- Acts as the "muscles" of the Connector
- Securely handles data transfers between a provider and consumer connector after the connection has been authorized by the Control Plane

MORE ARCHITECTURE



HOW DO WE SPEAK EDC? - I

ASSET

Describes metadata about real data (think of an inventory list that tells you, where to find some object in your warehouse)

POLICY

Rules written in ODRL (Open Digital Rights Language) for Access and Usage Policies

JSON-LD

JavaScript Object Notation for Linked Data

Provides "namespaces" to JSON with additional tools.

WHAT IS JSON-LD AND ODRL?

Some JSON –LD Facts

Standardized Format from W3C.

Data is not just raw Values but are now decorated with semantics and embedded into a context.

@Context

Defines a mapping between the simple JSON-Keys ("name", "type") and the complete IRIs from established vocabularies

Some ODRL Facts

A language to describe usage policies and requirements for digital Content

Policies are serialized as JSON-LD to strictly define rule components

```
{
  "@context": "http://www.w3.org/ns/odrl.jsonld", // reference to standard ODRl JSON-LD context doc
  "@type": "Offer", // policy document is defined as an offer
  "uid": "urn:offer:001",
  "target": "urn:asset:product_lifecycle_data.csv", // defines the target that the policy is made for.
  "assigner": "did:web:provider.com", // who issues the policy?
  "permission": [{
    "action": "use",
    "constraint": [{
      "leftOperand": "purpose",
      "operator": "eq",
      "rightOperand": "quality_analysis"
    }]
  }]
}
```

HOW DO WE SPEAK EDC? - II

Contract Definition

Provider links Assets with Policies

These basically define, who is able to see and start negotiating for what kind of data

Contract Offer

Specific data offer that is visible inside the Federated Catalog to the Consumer who fulfills policies for viewing

Contract Agreement

Finalized digital handshake after a succesful negotiation for an asset between a Provider and a Consumer

HOW DO WE SPEAK EDC? - III

DID

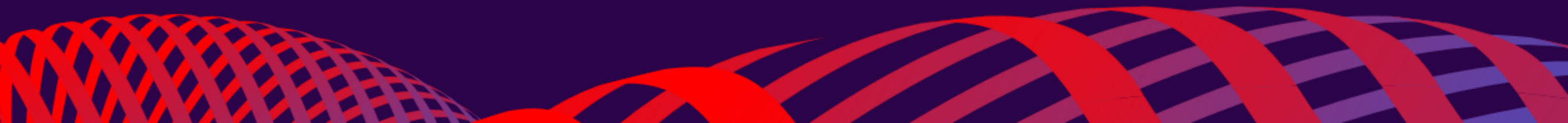
Stands for Decentralized Identifier and describes the unique digital ID of a connector

DCP

Stands for Decentralized Claims Protocol and acts as a standard for proof of identity with so called VCs

VCs

Stands for Verifiable Credentials and represents a way to display certain claims about ones identity or other possible attributes.



WHAT ARE THE CORE MODULES? - I

CATALOG SERVICE

Provider exposes Asset offers that are protected with policies inside their connector

CONTRACT NEGOTIATION SERVICE

A state machine for contract negotiations that runs pessimistically asynchronous

WHAT ARE THE CORE MODULES? - II

POLICY ENGINE

The "heart" of the connector which constantly checks ORDL-Policies with each interaction on the control plane

TRANSFER PROCESS SERVICE

The management service of the data plane that engages with the actual data transfers

WHAT ARE THE CORE MODULES? - III

IDENTITY SERVICE

Obviously manages Identities and describes the first usual extension point. In Tractus-X, this happens through DCP with DIDs/VCs.

EXTENSION POINTS

Interface that allows to hook custom business logic, different data exchange protocols and backends

THE DATA EXCHANGE WORKFLOW

1.Discovery

Consumer requests Federated Catalog of Provider

This request contains header tokens for auth

Authentication checks, if Consumer fulfils any policies for available Assets.

This filtering is policy based.

2.Negotiation

Consumer selects a specific Asset

Provider-Policy-Engine checks against Consumer Claims /VCs

On success, for a Contract Agreement

3.Transfer

Consumer requests actual data with their new agreement.

Final authorization against contract.

Safe transfer between Consumer and Provider

HOW TO SPIN UP A LOCAL MVD - I

Core Technologies

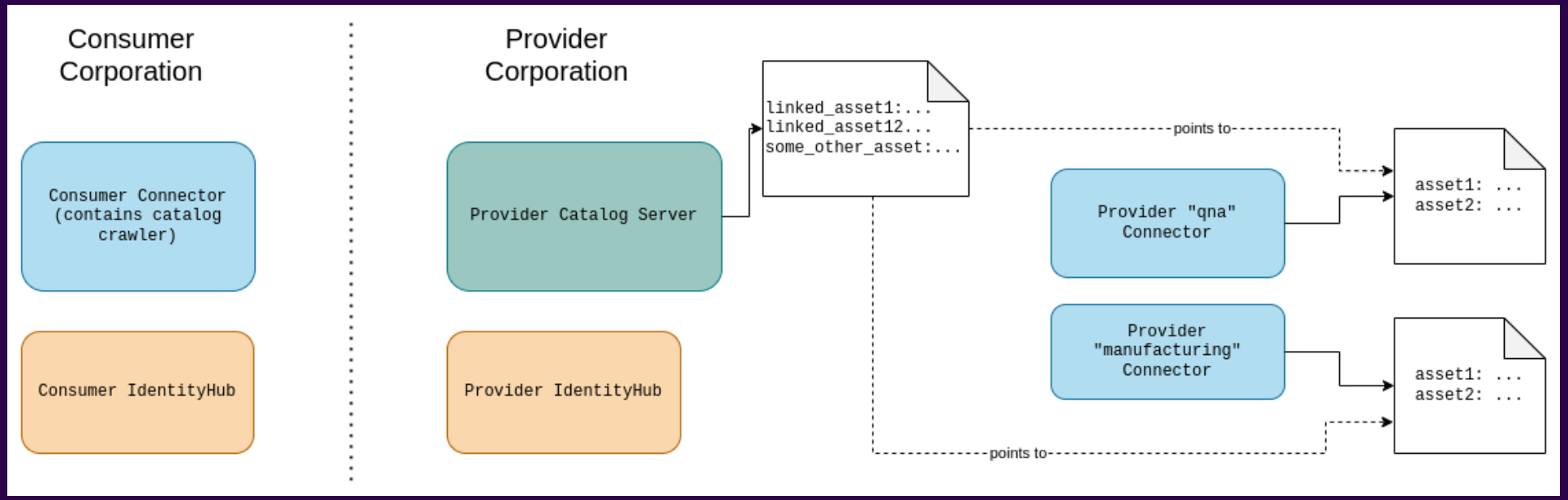
- eclipse-edc MVD github
- Decentralized Claims Protocol
- Docker
- IntelliJ IDEA (alternatively Kubernetes)
- Postman/HTTP Client

Local Installs

- NGINX (via Docker) with port 9876 and specific mounts
- newman (for Postman coll. on cmd)
- jq (cmd JSON processor)
- Node.js required for newman and jq

EDC Runtimes

- Consumer Control Plane and Identity Hub
- Provider Control Plane and Identity Hub
- Provider Catalog Server



HOW TO SPIN UP A LOCAL MVD - II

1.Clone MVD repository

```
git clone https://github.com/eclipse-edc/MinimumViableDataspace.git
```

2.Start NGINX for Issuer DID

`docker run […]` as provided in repository with port 9876

Verifiable with:

```
curl http://localhost:9876/.well-known/did.json
```

3.Launch EDC Runtimes in IntelliJ

Open the project inside IntelliJ

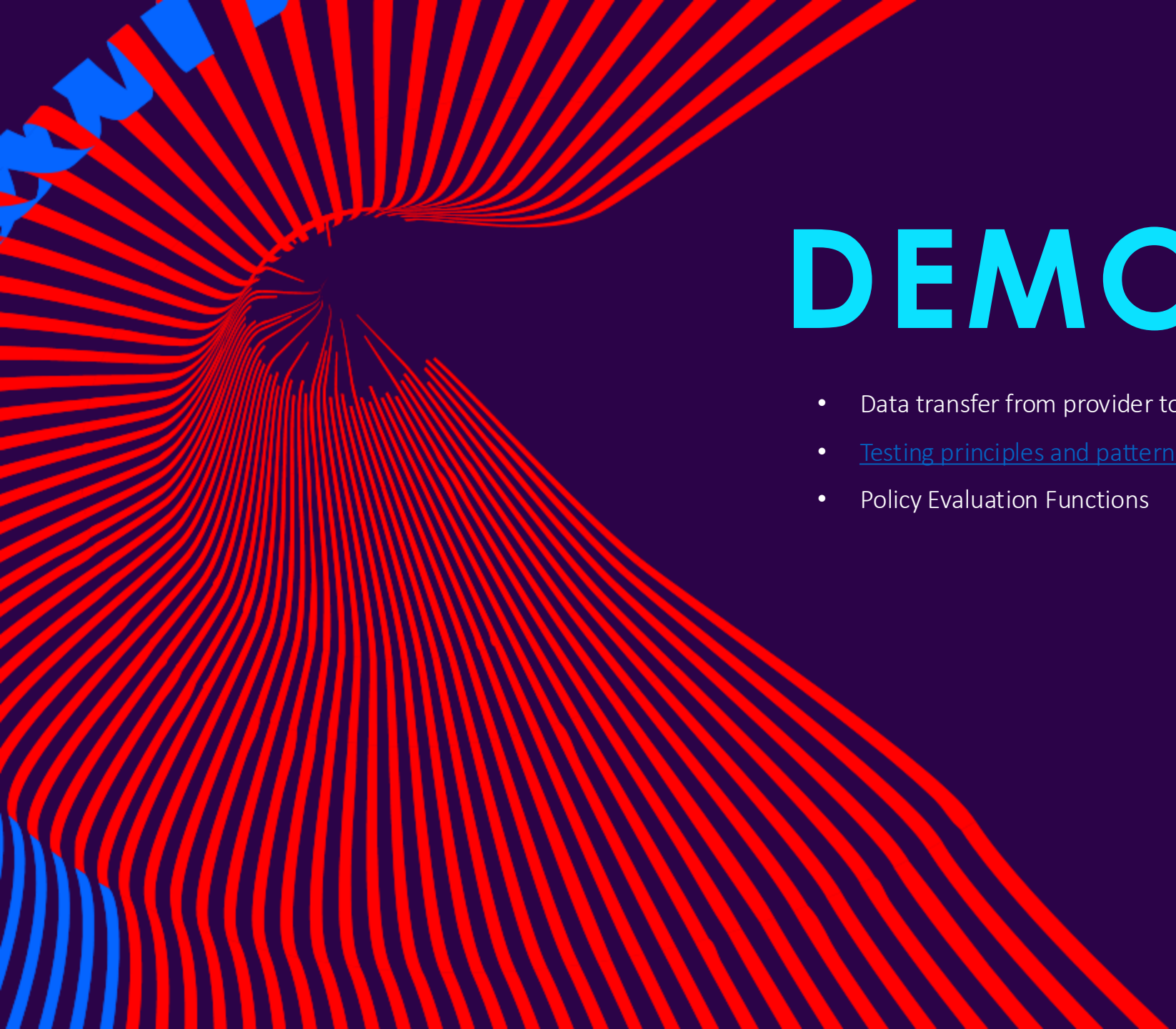
Run compound `dataspace` configuration to start edc components

4.Seed the Dataspace

After runtimes are built and running, execute `bash seed.sh`

Requires newman and jq installed

This creates participants, DIDs and preloads assets and credentials



DEMO TIME

- Data transfer from provider to consumer - a walkthrough
- Testing principles and patterns in the EDC
- Policy Evaluation Functions

SECURITY!

ODRL POLICIES

In the Policy Engine, ODRL is constantly evaluated at every step of the asynchronous multilayered communication between connectors.

DCP & VCS

Identities and Properties (e.g. subscriptions) are verified from a trusted source. Used in headers during request in negotations.

API TOKENS

Finalized Contract agreements generate API tokens that are then used to request actual data on the Data Plane and are automatically revoked on expiry.

THANK YOU FOR LISTENING

Maximilian Peresunchak

Seminar StuPro – 16.05.2025

Universität Stuttgart

CONTENT & PICTURE SOURCES

Content Sources

https://projects.eclipse.org/projects/technology.edc [last visited 11.05.2025]

https://github.com/eclipse-edc/Connector [last visited 15.05.2025]

https://github.com/eclipse-edc/docs/blob/main/developer/handbook.md [last visited 11.05.2025]

https://github.com/eclipse-edc/docs/blob/main/developer/contributing/coding-principles.md [last visited 13.05.2025]

https://www.youtube.com/watch?v=ic-XEGzdODM [last watched 15.05.2025]

https://eclipse-edc.github.io/documentation/ ff. [last visited 14.05.2025]

https://github.com/eclipse-tractusx/tractus-x-umbrella [last visited 11.05.2025]

https://github.com/eclipse-edc/MinimumViableDataspace [Used in setup on slides, last visited 15.05.2025]

https://www.w3.org/TR/odrl-model/ ff. [last visited 15.05.2025]

Content Sources

SVG icons are from Microsoft Powerpoint 365 - student version

Slide 2 Connector illustration: https://eclipse-tractusx.github.io/assets/images/connector-kit-logo-8456c4e8c495465295c71b90a21353ba.svg

Slide 9 Architectural Overview: https://blog.doubleslash.de/wp/wp-content/uploads/2022/08/Abbildung_2_Dataspaces_mental_model.png.webp

Slide 17 Local MVD Participants from MinimumViableDataspace Github-Project \resources, source mentioned in Content Sources