

**OLFAI & DATA**



Egeria Webinar Program

# EGERIA AND OPEN LINEAGE

Ljupcho Palashevski, ING  
Egeria Maintainer

Mandy Chessell CBE FREng  
Egeria Open Source Project Lead


# Egeria's webinar program

Date	time	Title	Description	Presenter
<b>8th November 2021</b>	15:00 UTC	<b>Open lineage</b>	This session will describe the purpose of lineage, what type of information needs to be collected and how this information is managed and used in an enterprise with Egeria.  Zoom Conference <a href="https://zoom.us/j/523629111">https://zoom.us/j/523629111</a>	<b>Ljupcho Palashevski and Mandy Chessell</b>
<b>6th December 2021</b>	15:00 UTC	<b>What next after you have built a catalog. Part 1: the journey</b>	Journey from manual cataloging, to using automated integration and templating. Metadata discovery and stewardship will be covered here also as well as metadata deduplication.	<b>Mandy Chessell</b>
<b>10th January 2022</b>	15:00 UTC	<b>Kubernetes operators and Egeria</b>	This session will cover how easy it is to run Egeria in Kubernetes and how the Egeria Kubernetes operator can be used to manage Egeria in a Kubernetes environment.	<b>Nigel Jones</b>
<b>7th February 2022</b>	15:00 UTC	<b>Time Travelling with Egeria</b>	Every wanted to know what the state of your metadata was at some specific time in the past? This session will introduce the Crux open metadata repository that supports these historical metadata queries.	<b>Chris Grote</b>
<b>7th March 2022</b>	15:00 UTC	<b>How to build a repository connector</b>	Every wanted to build an OMRS repository connector? This session will take you through what the considerations are and you need to do. It will show how to create the simplest "Hello World" connector.	<b>Chris Grote</b>

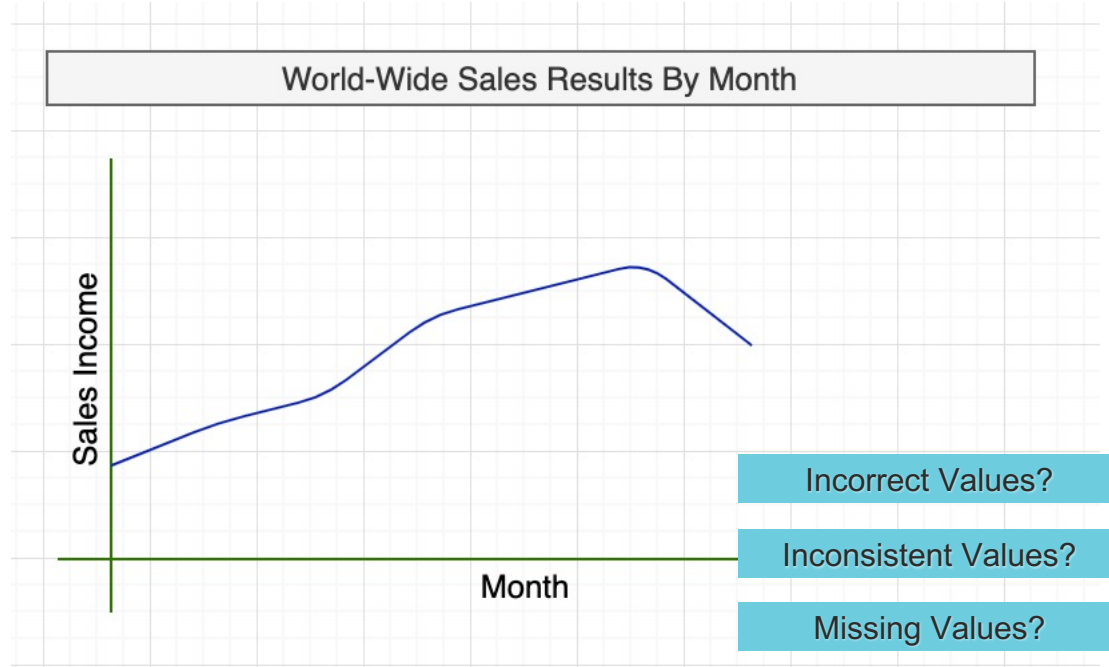
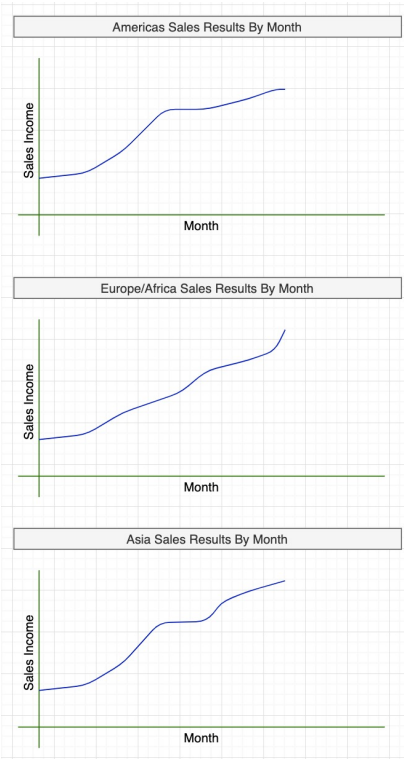
# Today's Agenda

- What is lineage?
- Lineage Architecture
  - Lineage Capture
  - Lineage Stewardship
  - Lineage Preservation and Use
- Demo

# What is Lineage?



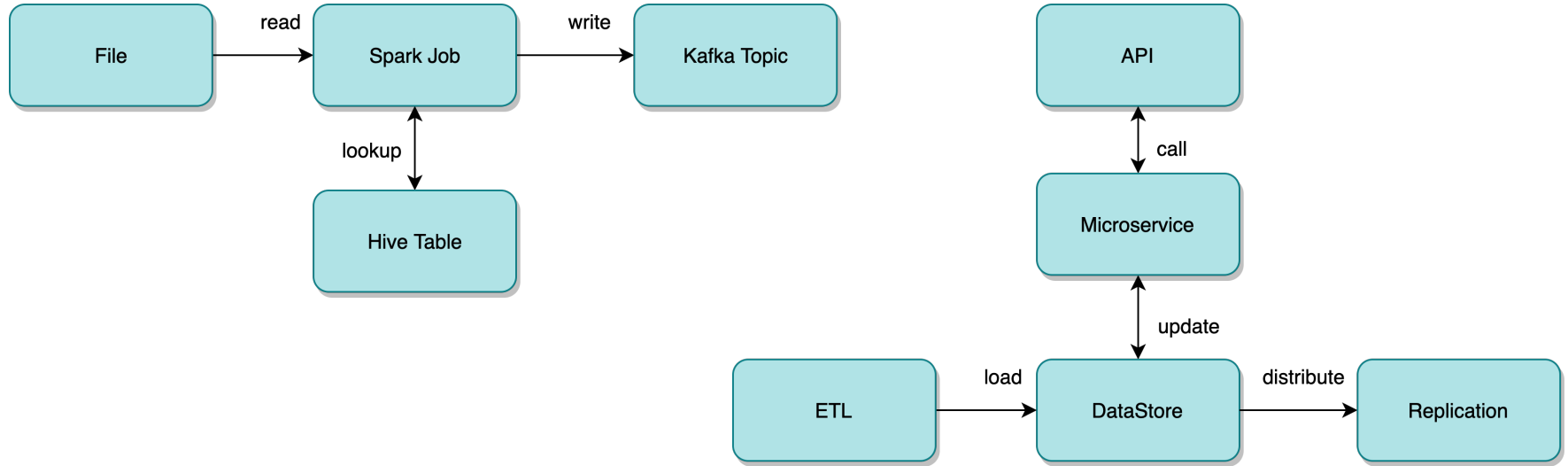
# What if the data you are using reveals unexpected results?



# What is lineage?

- Lineage shows how data flows from its origins to its various destinations. This includes details of the processing along the way. It is used to understand:
  - whether the data used in reports and analytics models has come from the correct sources and has passed through the correct processing (known as *traceability of data*).
  - what would be the impact on downstream processing and consumers if something was changed (known as *impact analysis*).
  - whether the operational processes that implement the data flows are executing correctly (known as *governance by expectation*).

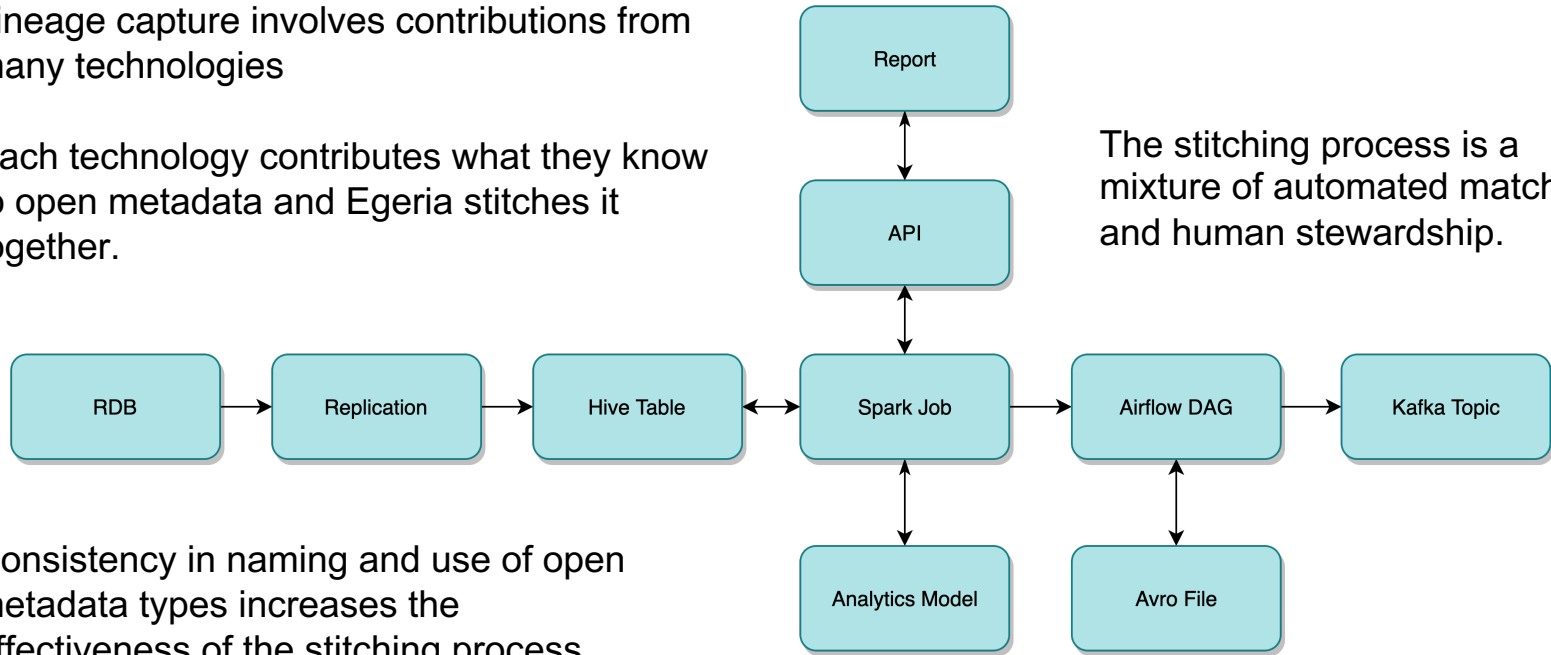
# Examples of processes



# The lineage graph emerges ...

Lineage capture involves contributions from many technologies

Each technology contributes what they know to open metadata and Egeria stitches it together.

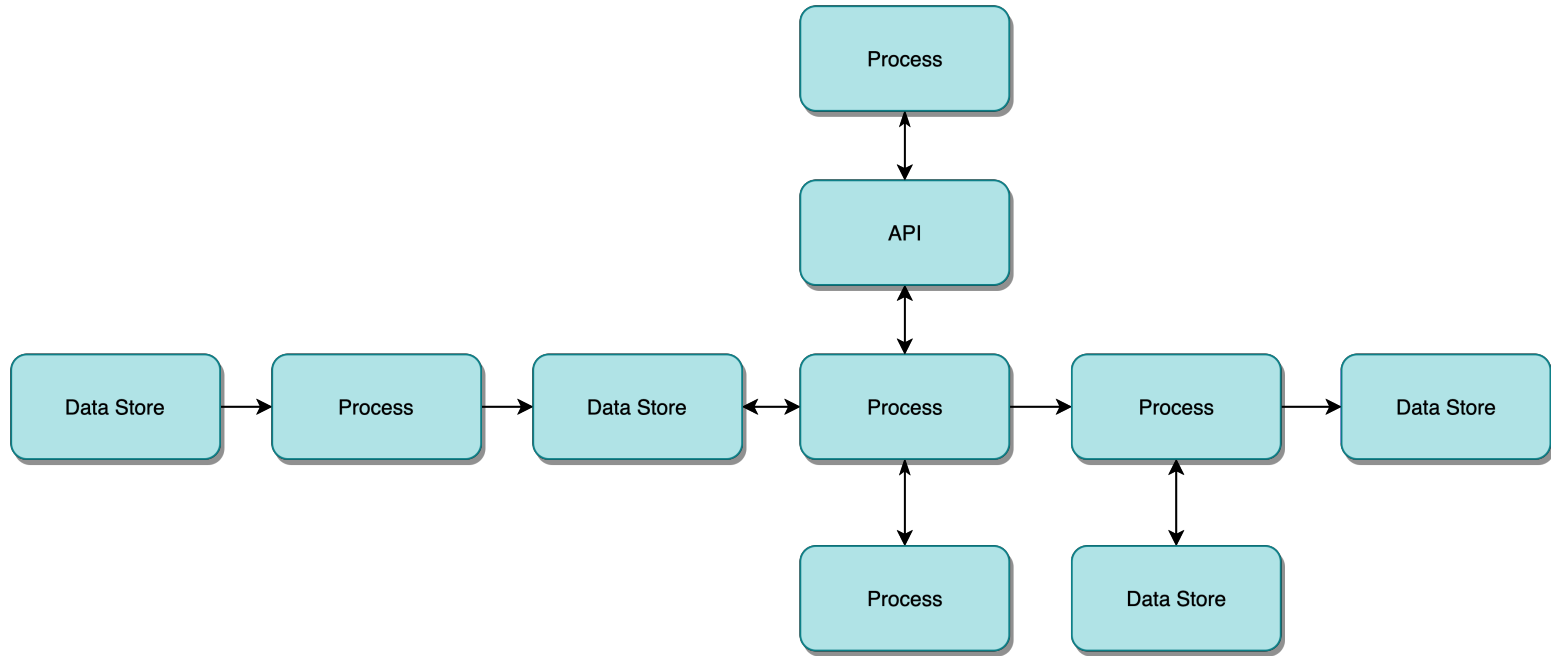


The stitching process is a mixture of automated matching and human stewardship.

Consistency in naming and use of open metadata types increases the effectiveness of the stitching process.



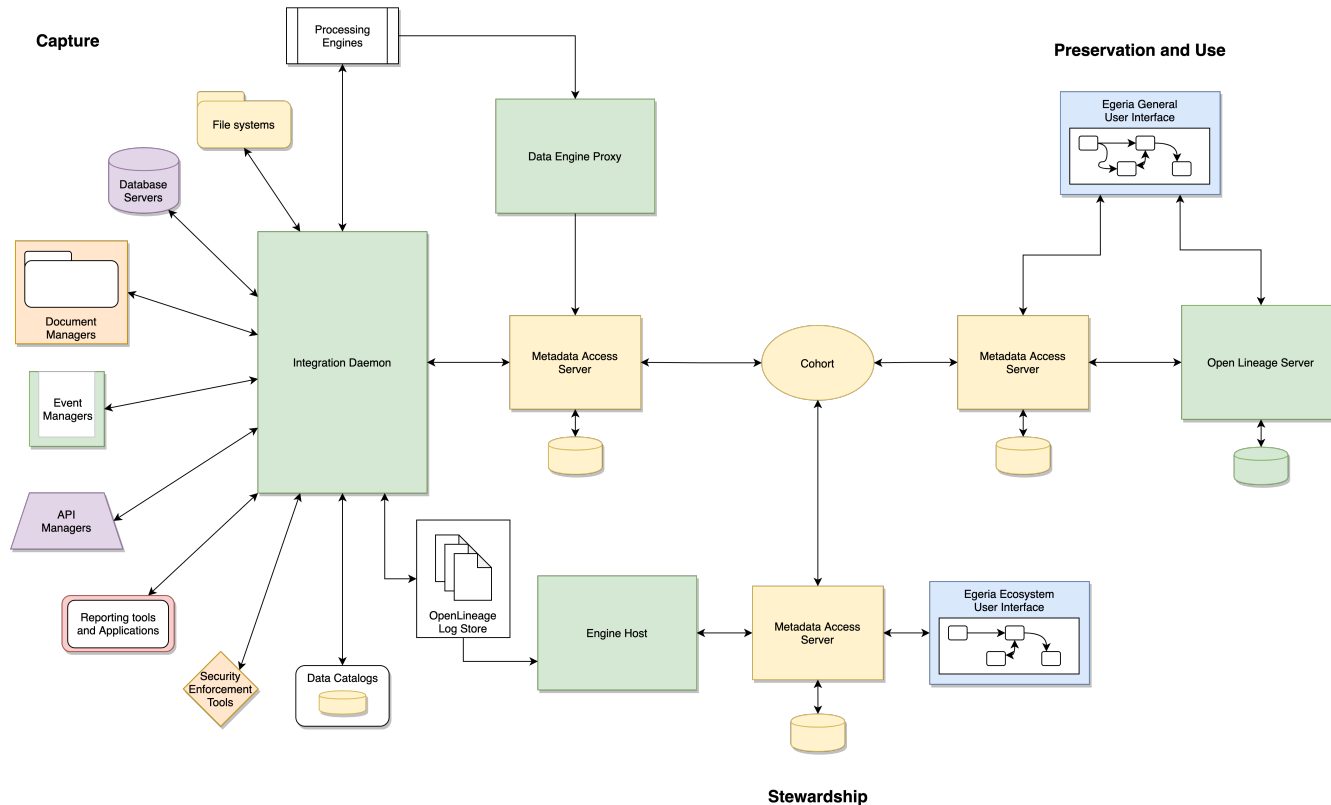
# The abstract lineage graph



# Lineage Architecture

The background of the slide is a dark, futuristic digital environment. It features a perspective view of a long, narrow corridor or tunnel. The walls and floor are composed of numerous vertical and horizontal lines of varying lengths and thicknesses, all glowing with a vibrant cyan or light blue light. The lines create a sense of depth and movement, as if data is flowing through a complex network. The overall aesthetic is clean, modern, and high-tech.

# Capture, Stewardship, Preservation and Use



# Lineage Capture



# Capturing two types of lineage

- Design Lineage

- Shows the paths of data flow through data sources and processes

*traceability of data*

*impact analysis*

- Operational Lineage

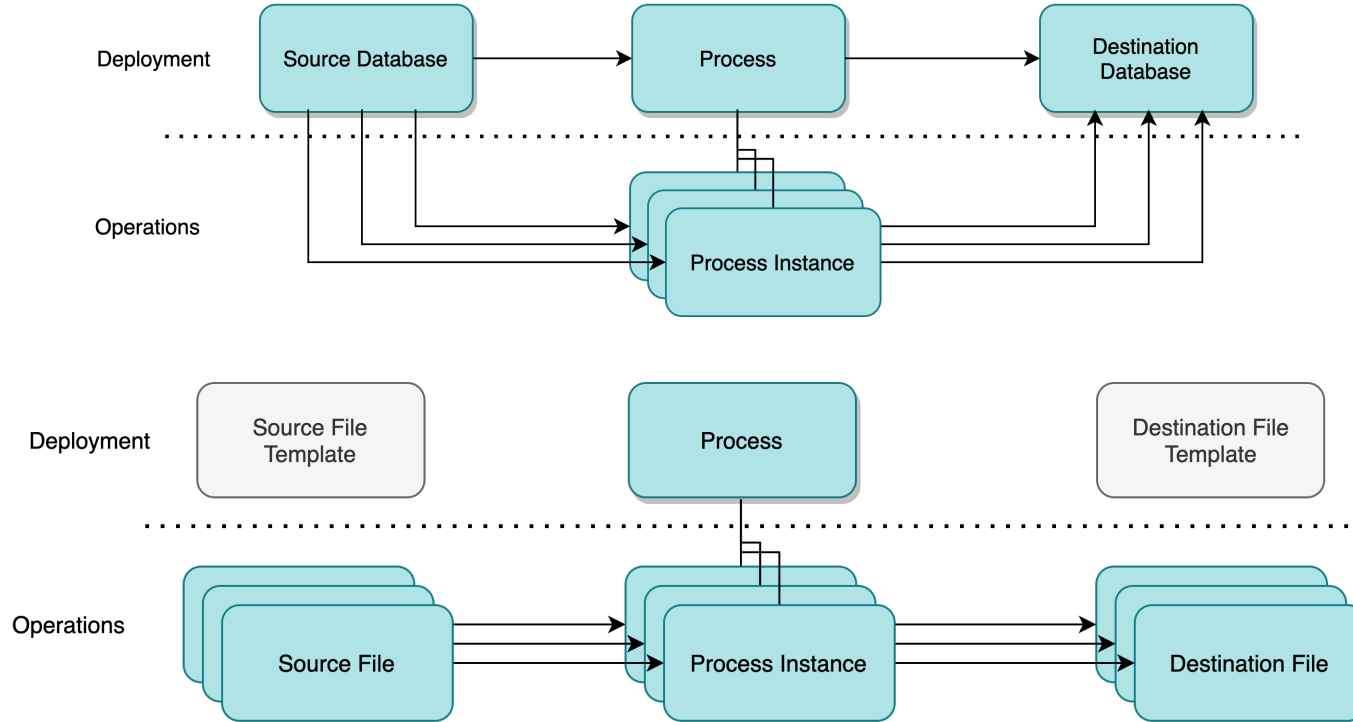
- Shows when processes ran, how much data they processed, what they discovered about the content

*governance by expectation*

# Static and Dynamic Capture

- The **static** aspect involves cataloguing all of the *resources* that are deployed into your digital landscape. This defines the data sources and processing engines and how they link together. Ideally this cataloguing is done as these resources are deployed, which may then be augmented with *automated cataloguing* of resources and *metadata discovery*. It is also possible that tools may catalogue resources under the guidance of their users and this metadata is *shared with the open metadata ecosystem*.
- The **dynamic** aspect captures information about the activity that happens day-to-day, such as the running of processes, and its effects. This could include details of the volumes of data discovered and/or processed along with any analysis of its contents.

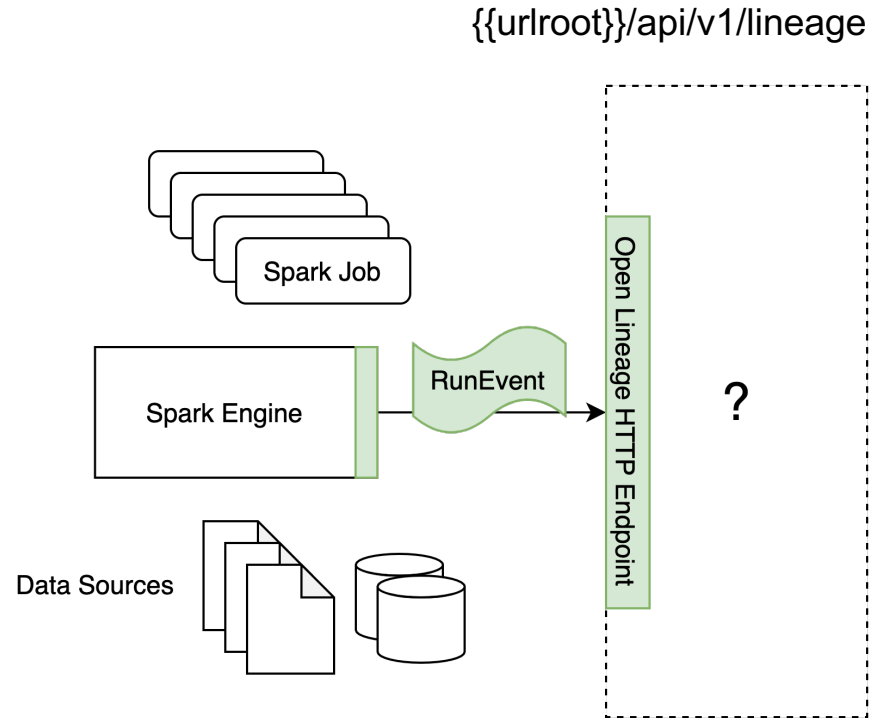
# Comparison of lineage capture for different technologies



# The OpenLineage Standard

(<https://openlineage.io/>)

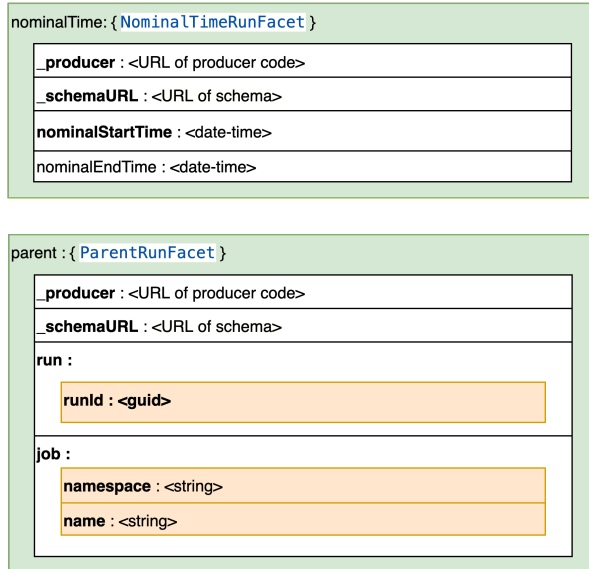
- Provides a standard payload and API URL for dynamic lineage capture



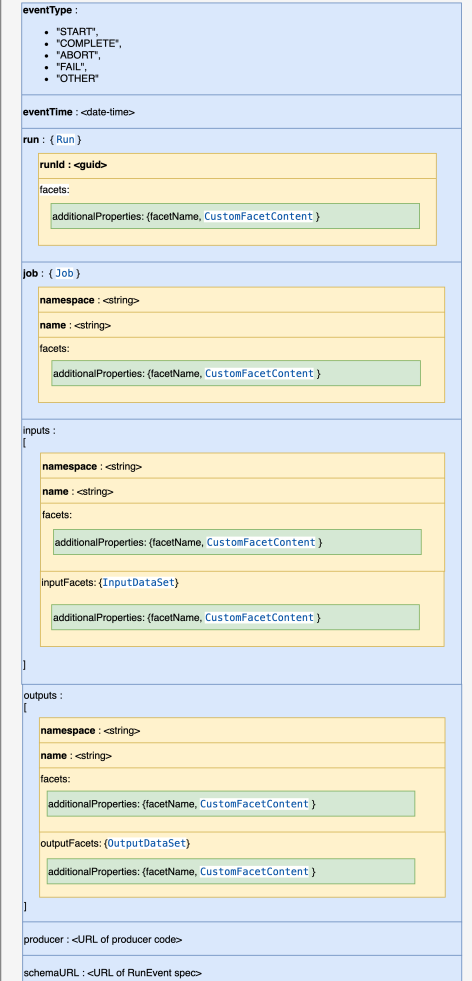


# OpenLineage events

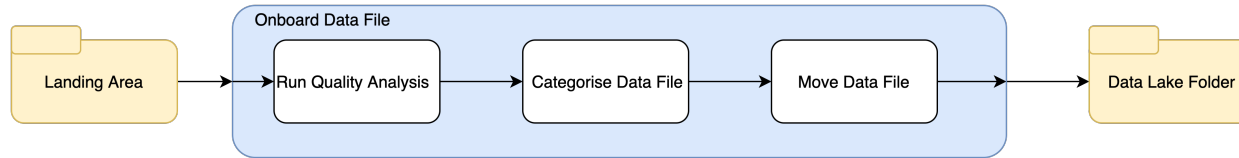
Run Facets



RunEvent



# Example process and its events



```
eventType="START", eventTime=<date-time>, runId=1, job="Onboard Data File", inputDataSource="Landing Area"
```

```
eventType="START", eventTime=<date-time>, runId=2, parentRunId=1, job="Run Quality Analysis", inputDataSource="Landing Area"
```

```
eventType="OTHER", eventTime=<date-time>, runId=2, parentRunId=1, job="Run Quality Analysis", dataQualityMetrics={...}
```

```
eventType="COMPLETE", eventTime=<date-time>, runId=2, parentRunId=1, job="Run Quality Analysis", inputDataSource="Landing Area"
```

```
eventType="START", eventTime=<date-time>, runId=3, parentRunId=1, job="Categorise Data File", inputDataSource="Landing Area"
```

```
eventType="COMPLETE", eventTime=<date-time>, runId=3, parentRunId=1, job="Categorise Data File", inputDataSource="Landing Area"
```

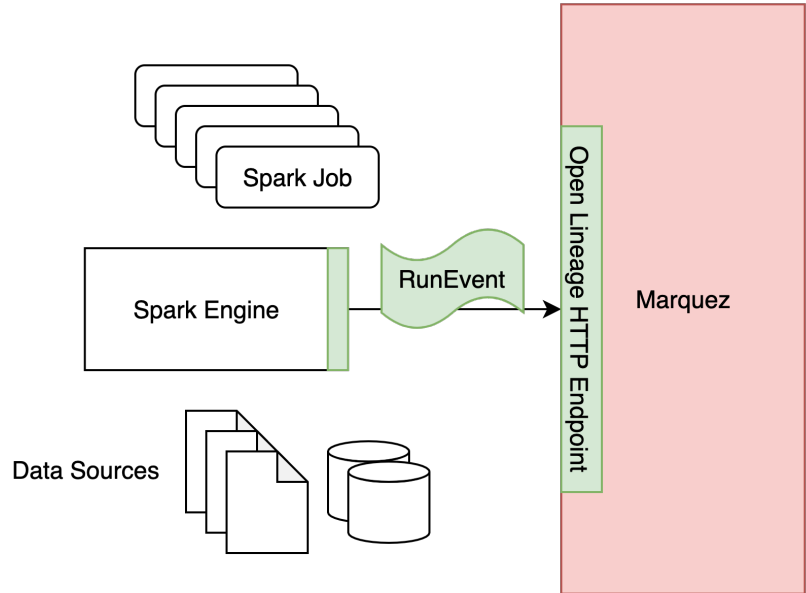
```
eventType="START", eventTime=<date-time>, runId=4, parentRunId=1, job="Move Data File", inputDataSource="Landing Area", outputDataSource="Data Lake Folder"
```

```
eventType="COMPLETE", eventTime=<date-time>, runId=4, parentRunId=1, job="Move Data File", inputDataSource="Landing Area", outputDataSource="Data Lake Folder"
```

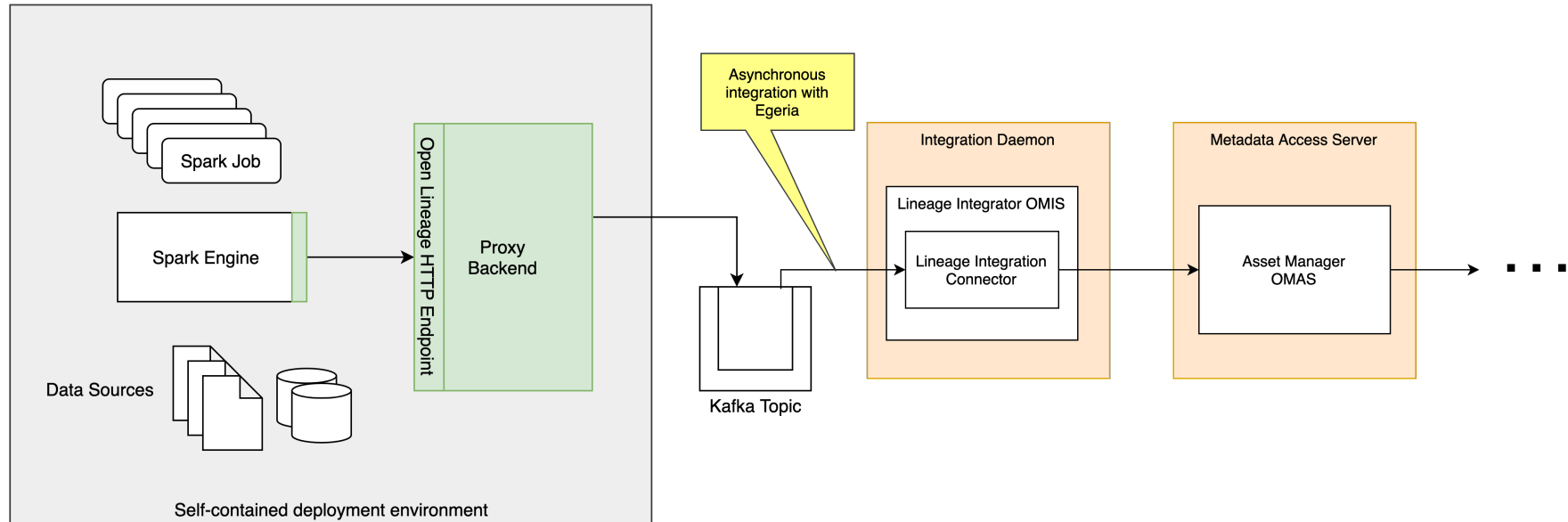
```
eventType="COMPLETE", eventTime=<date-time>, runId=1, job="Onboard Data File", inputDataSource="Landing Area", outputDataSource="Data Lake Folder"
```

# OpenLineage runtimes

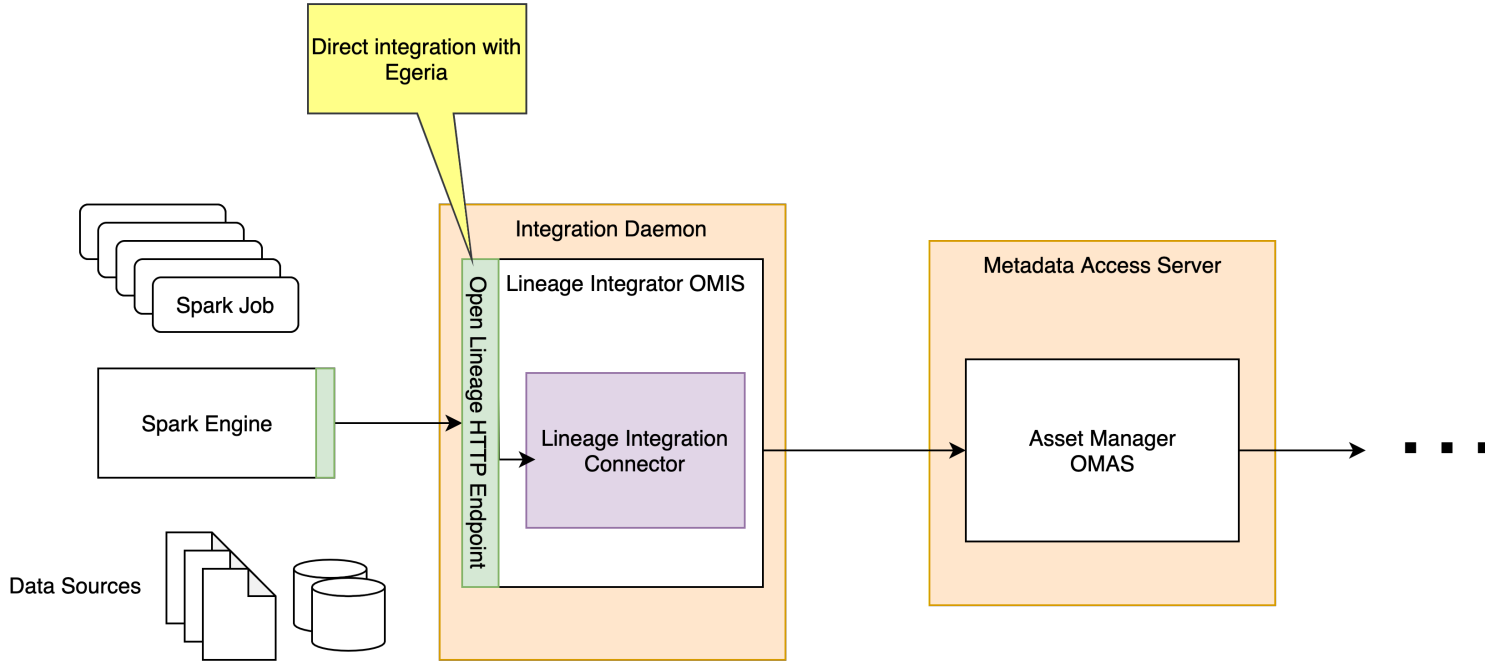
- Marquez (<https://marquezproject.github.io/marquez/>) is the reference implementation



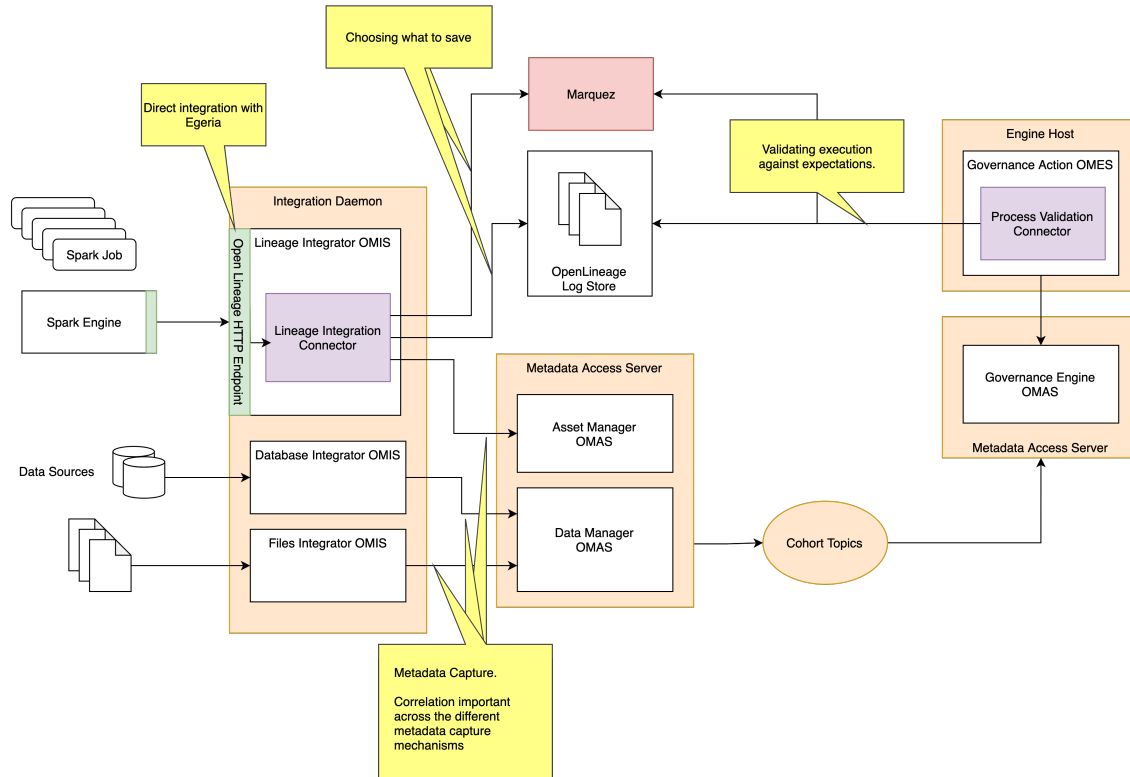
# OpenLineage runtimes – the proxy backend



# OpenLineage runtimes – direct integration with Egeria



# Egeria's OpenLineage support

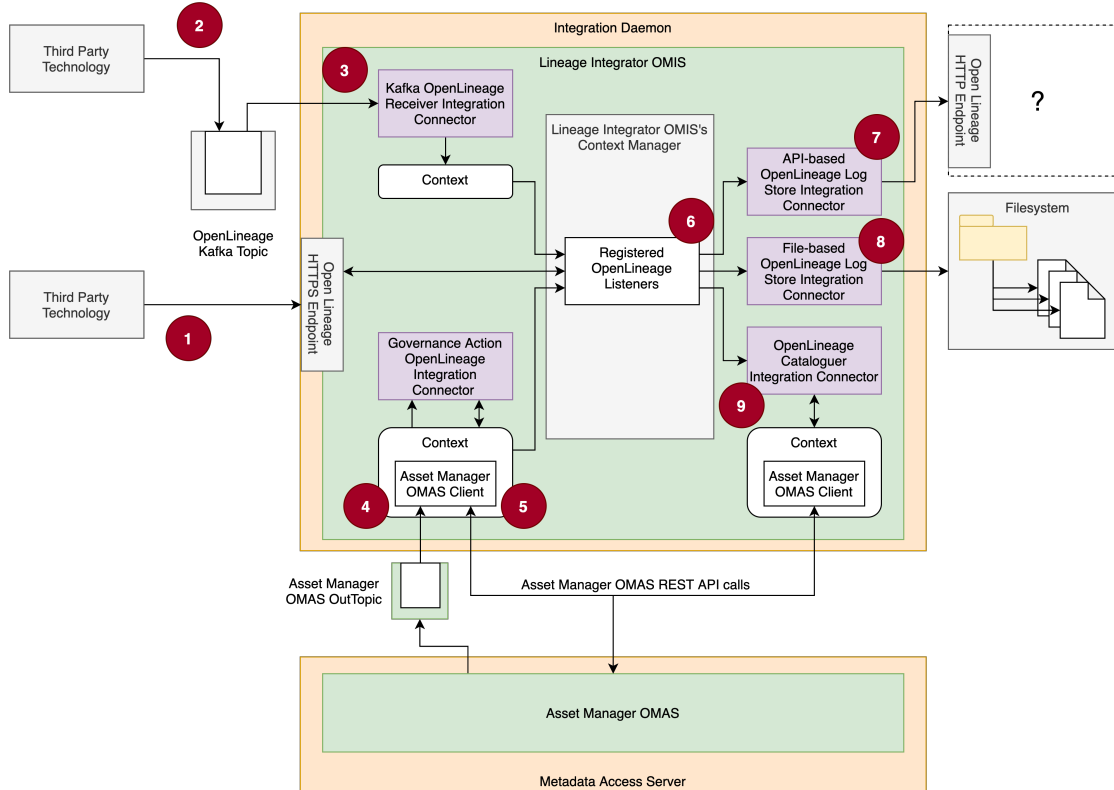


# OpenLineage Log Store

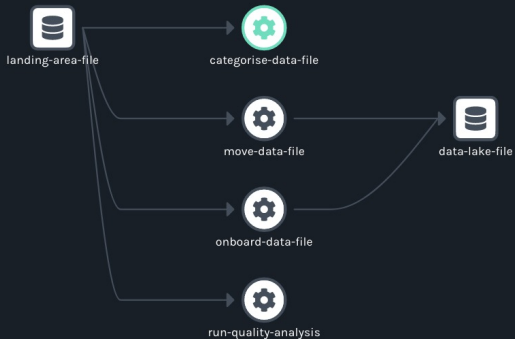
- Auditing
- Analysis

```
openlineage.log
├── governance-action-process:clinical-trials:drop-foot:weekly-measurements:onboarding
│   ├── provision-weekly-measurements-governance-action-type
│   │   ├── a382470e-0e3f-4ddd-8a46-dea28c74bcfe-2021-11-6-15-58-23:812000000-START.json
│   │   ├── a382470e-0e3f-4ddd-8a46-dea28c74bcfe-2021-11-6-15-58-41:907000000-COMLETE.json
│   │   ├── b978fa80-e28c-4891-908d-63e84b7c830a-2021-11-6-14-15-45:536000000-START.json
│   │   ├── b978fa80-e28c-4891-908d-63e84b7c830a-2021-11-6-15-52-59:459000000-COMLETE.json
│   │   ├── dbfa1b9b-df84-47d1-bea2-c5a821b5ab89-2021-11-6-15-57-58:574000000-START.json
│   │   ├── dbfa1b9b-df84-47d1-bea2-c5a821b5ab89-2021-11-6-15-58-39:100000000-COMLETE.json
│   │   ├── f29dd651-a191-422a-bc7c-0afd6b341b71-2021-11-6-15-57-10:220000000-START.json
│   │   └── f29dd651-a191-422a-bc7c-0afd6b341b71-2021-11-6-15-58-22:709000000-COMLETE.json
│   └── InitiateGovernanceAction
│       ├── AssetGovernance:copy-file
│       └── AssetGovernance:delete-file
│           ├── Oed14a4c-87f-47d9-911c-9c400938bcb0-2021-11-6-16-16-44:199000000-START.json
│           ├── Oed14a4c-87f-47d9-911c-9c400938bcb0-2021-11-6-16-16-53:920000000-COMLETE.json
│           ├── 2a2b86ed-8f6d-40b1-8bf8-974d16c220e8-2021-11-6-16-13-26:486000000-START.json
│           ├── 2a2b86ed-8f6d-40b1-8bf8-974d16c220e8-2021-11-6-16-15-58:649000000-COMLETE.json
│           ├── 5094732d-e180-4c01-ad2b-da68f1466971-2021-11-6-14-2-43:556000000-START.json
│           ├── a782402f-3639-40a1-9d99-cb3c6730d29e-2021-11-6-16-17-0:220000000-START.json
│           ├── a782402f-3639-40a1-9d99-cb3c6730d29e-2021-11-6-16-17-15:336000000-COMLETE.json
│           └── AssetGovernance:nested-in-folder
│               └── myNamespace
│                   └── categorise-data-file
│                       ├── ecea439e-228c-4264-82d9-4a82576d5002-2021-11-6-13-52-8:208000000-START.json
│                       ├── ecea439e-228c-4264-82d9-4a82576d5002-2021-11-6-13-52-13:431000000-COMLETE.json
│                       ├── ecea439e-228c-4264-82d9-4a82576d5003-2021-11-6-13-54-21:758000000-START.json
│                       └── ecea439e-228c-4264-82d9-4a82576d5003-2021-11-6-13-54-25:173000000-COMLETE.json
│                           └── move-data-file
│                               ├── d4736e42-125d-436f-97ce-34b11940d002-2021-11-6-13-52-16:694000000-START.json
│                               ├── d4736e42-125d-436f-97ce-34b11940d002-2021-11-6-13-52-19:887000000-COMLETE.json
│                               ├── d4736e42-125d-436f-97ce-34b11940d003-2021-11-6-13-54-28:550000000-START.json
│                               └── d4736e42-125d-436f-97ce-34b11940d003-2021-11-6-13-54-32:301000000-COMLETE.json
│                                   └── onboard-data-file
│                                       ├── 304e5f23-4667-4d26-9499-2f30d8e17002-2021-11-6-13-47-47:897000000-START.json
│                                       ├── 304e5f23-4667-4d26-9499-2f30d8e17002-2021-11-6-13-52-23:381000000-COMLETE.json
│                                       ├── 304e5f23-4667-4d26-9499-2f30d8e17003-2021-11-6-13-54-8:521000000-START.json
│                                       └── 304e5f23-4667-4d26-9499-2f30d8e17003-2021-11-6-13-54-35:365000000-COMLETE.json
│                                           └── run-quality-analysis
│                                               ├── 4cb8a46b-6271-4791-925c-9ae7123d1002-2021-11-6-13-47-51:250000000-START.json
│                                               ├── 4cb8a46b-6271-4791-925c-9ae7123d1002-2021-11-6-13-47-54:213000000-OTHER.json
│                                               ├── 4cb8a46b-6271-4791-925c-9ae7123d1002-2021-11-6-13-52-4:462000000-COMLETE.json
│                                               ├── 4cb8a46b-6271-4791-925c-9ae7123d1003-2021-11-6-13-54-11:799000000-START.json
│                                               ├── 4cb8a46b-6271-4791-925c-9ae7123d1003-2021-11-6-13-54-15:340000000-OTHER.json
│                                               └── 4cb8a46b-6271-4791-925c-9ae7123d1003-2021-11-6-13-54-18:764000000-COMLETE.json
└── [{"eventTime": "2021-11-06T13:47:54.213Z",
    "eventType": "OTHER",
    "inputs": {
      "facets": {
        "dataQualityAssertions": {
          "producer": "https://www.postman.com/",
          "schemaURL": "#/definitions/DataQualityAssertionsDatasetFacet",
          "assertions": [
            {
              "column": "id",
              "assertion": "not_null",
              "success": true
            },
            {
              "column": "second_id",
              "assertion": "not_null",
              "success": true
            },
            {
              "column": "id",
              "assertion": "unique",
              "success": true
            },
            {
              "column": "second_id",
              "assertion": "unique",
              "success": true
            }
          ]
        }
      }
    },
    "name": "landing-area-file",
    "namespace": "myNamespace"
  }],
  "job": {
    "facets": {
      "name": "run-quality-analysis",
      "namespace": "myNamespace"
    }
  },
  "outputs": [],
  "producer": "https://www.postman.com/",
  "run": {
    "facets": {
      "parent": {
        "producer": "https://www.postman.com/",
        "schemaURL": "https://raw.githubusercontent.com/OpenLineage/OpenLineage/main/spec/OpenLineage.json#/definitions/ParentRunFacet",
        "job": {
          "name": "onboard-data-file",
          "namespace": "myNamespace"
        },
        "run": {
          "runId": "304e5f23-4667-4d26-9499-2f30d8e17002"
        }
      }
    },
    "runId": "4cb8a46b-6271-4791-925c-9ae7123d1002"
  }
}
```

# Egeria's OpenLineage support – more detail







LATEST RUN

RUN HISTORY

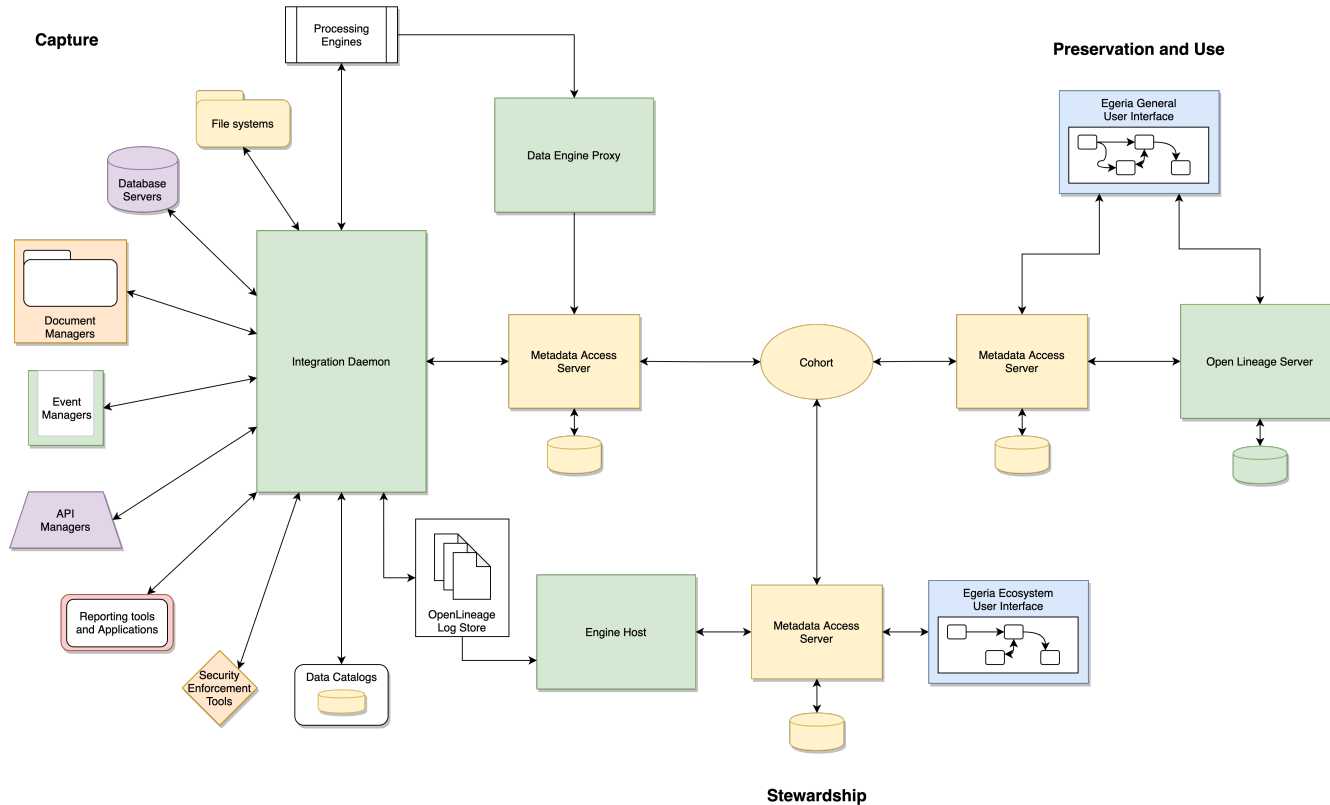
LOCATION



categorise-data-file

ID	State	Created At	Started At	Ended At	Duration
ecea439e-228c-4264-82d9-4a82576d5003	COMPLETED	Nov 06, 2021 01:54pm	Nov 06, 2021 01:54pm	Nov 06, 2021 01:54pm	0m 03s
ecea439e-228c-4264-82d9-4a82576d5002	COMPLETED	Nov 06, 2021 01:52pm	Nov 06, 2021 01:52pm	Nov 06, 2021 01:52pm	0m 05s
ecea439e-228c-4264-82d9-4a82576d5001	COMPLETED	Nov 06, 2021 01:47pm	Nov 06, 2021 01:47pm	Nov 06, 2021 01:47pm	0m 03s

# Capture, Stewardship, Preservation and Use

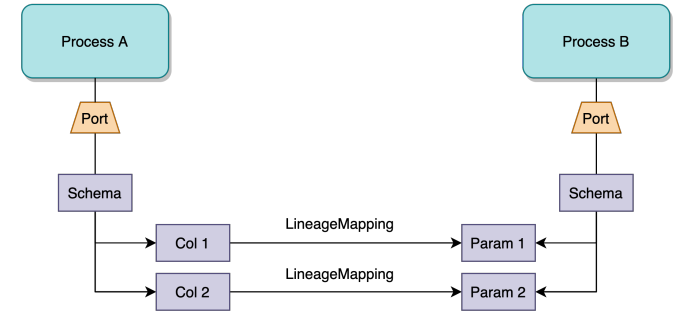
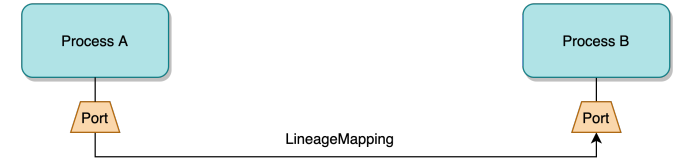
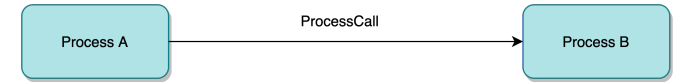


# Lineage Stewardship



# Stitching

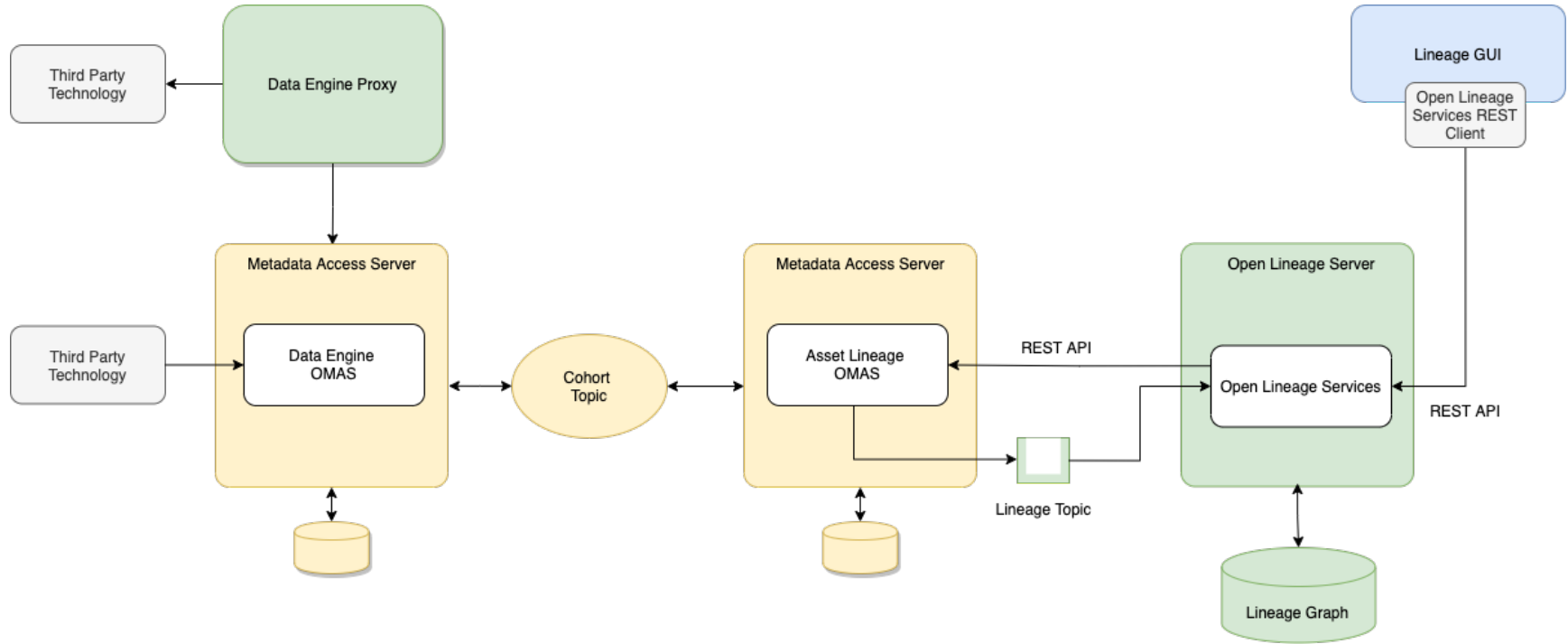
- [Data passing relationships](#) add the links to show which process called another and the style of the invocation.
  - *DataFlow* - Shows that data is passed between the two processes - typically by the processing engine that hosts them.
  - *ControlFlow* - Shows that control is passed between the two processes - typically by the processing engine that hosts them.
  - *ProcessCall* - Shows that one process makes an explicit call to another.
- [LineageMapping relationships](#) associates two elements from different assets that are equivalent.





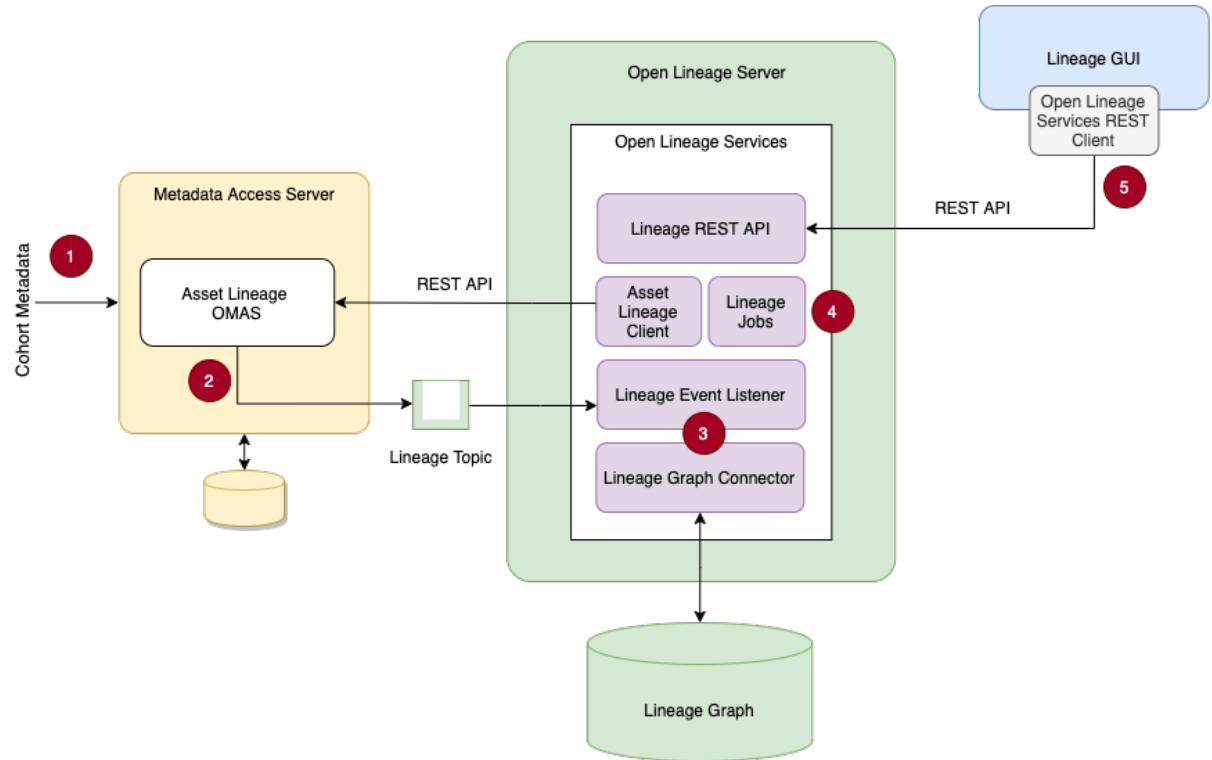
# Lineage Preservation and Use

# Lineage capture, preservation and use



# Building a lineage warehouse

1. Metadata arrives on the cohort topic.
2. Asset Lineage looks at lineage metadata, completes lineage context for assets and sends it for preservation.
3. Open Lineage Server stores lineage elements building up the lineage graph.
4. Background jobs run to optimize the querying and detect changes.
5. Apps or tools query lineage information for consolidated business views and further use.



## Exercise 1

### Capturing lineage manually

In this exercise Peter and Erin will start with minimal use-case and execute steps to create lineage manually. They are looking at simple high level transformation activity implemented using CocoETL, in-house developed ETL tool that uses python scripting language. Files from previous clinical trials are stored on server location accessible by the tool. ConvertFileToCSV is script that reads file coming out of legacy system of records and transform it to csv file structure.

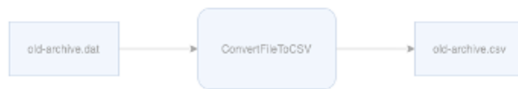


Figure 2: Simple asset lineage

For use-cases like this one, **Data Engine Access Service (OMAS)** API seems perfect match. It enables external data platforms, tools or engines to interact with Egeria and share metadata needed to construct lineage graph.

#### Check if assets are present in the catalog

At first, Erin wants to be sure upfront that the assets are not present in the catalog. She uses Egeria UI Asset Catalog search option but first she needs to log in.

**Important:** When running this lab using kubernetes deployment, make sure that you [expose the Egeria UI](#) running in the container to your local network and access it via localhost.

To access Egeria UI go to <https://localhost:8443/>

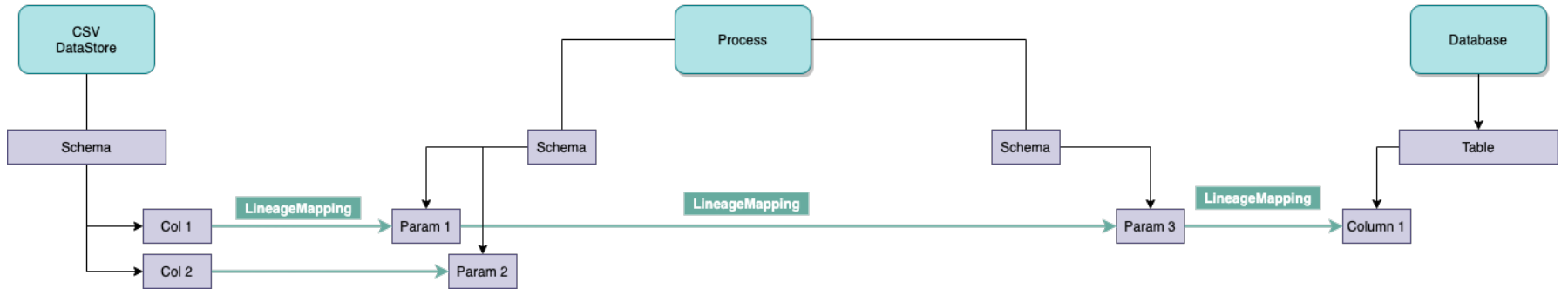
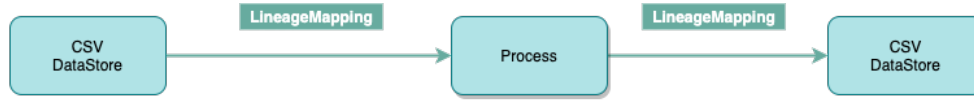
```
username: erinoverview  
password: secret
```

# DEMO

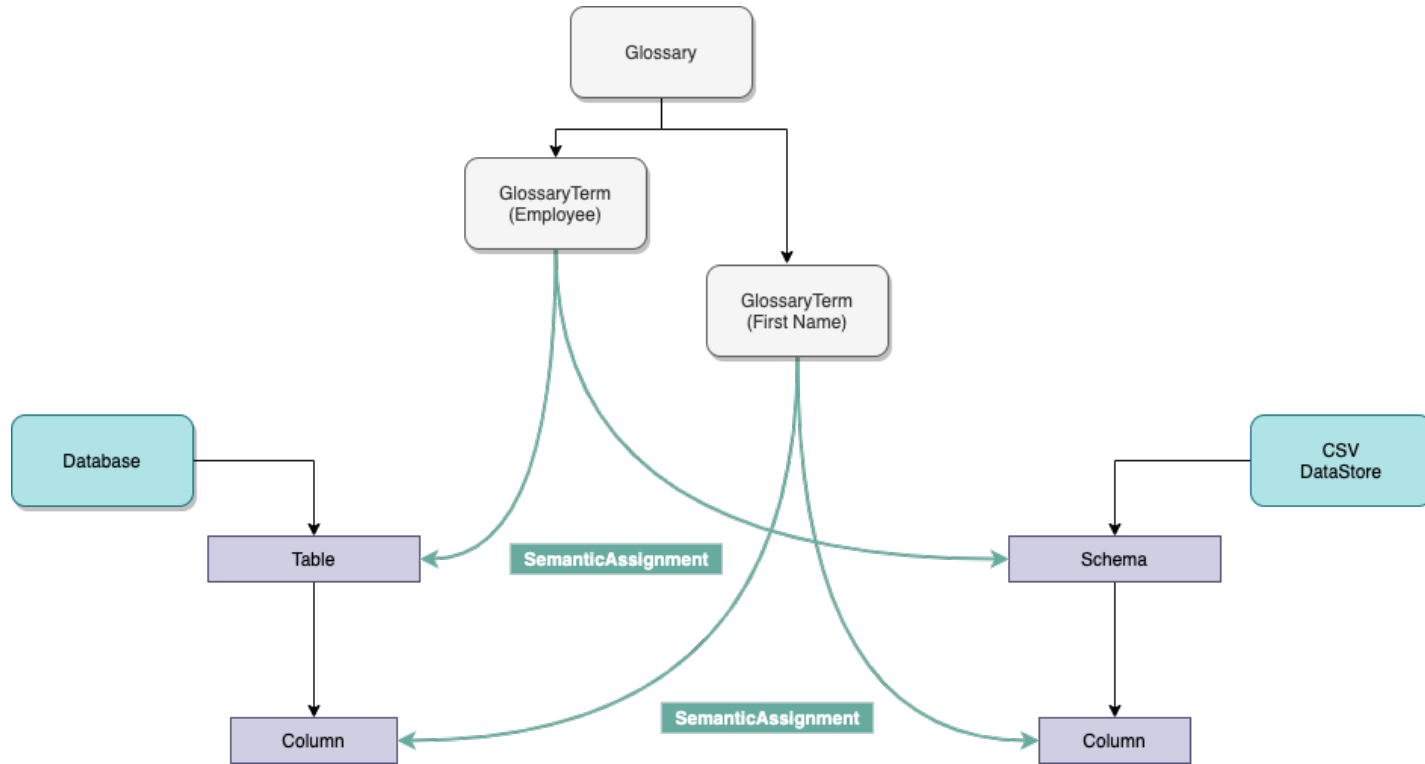




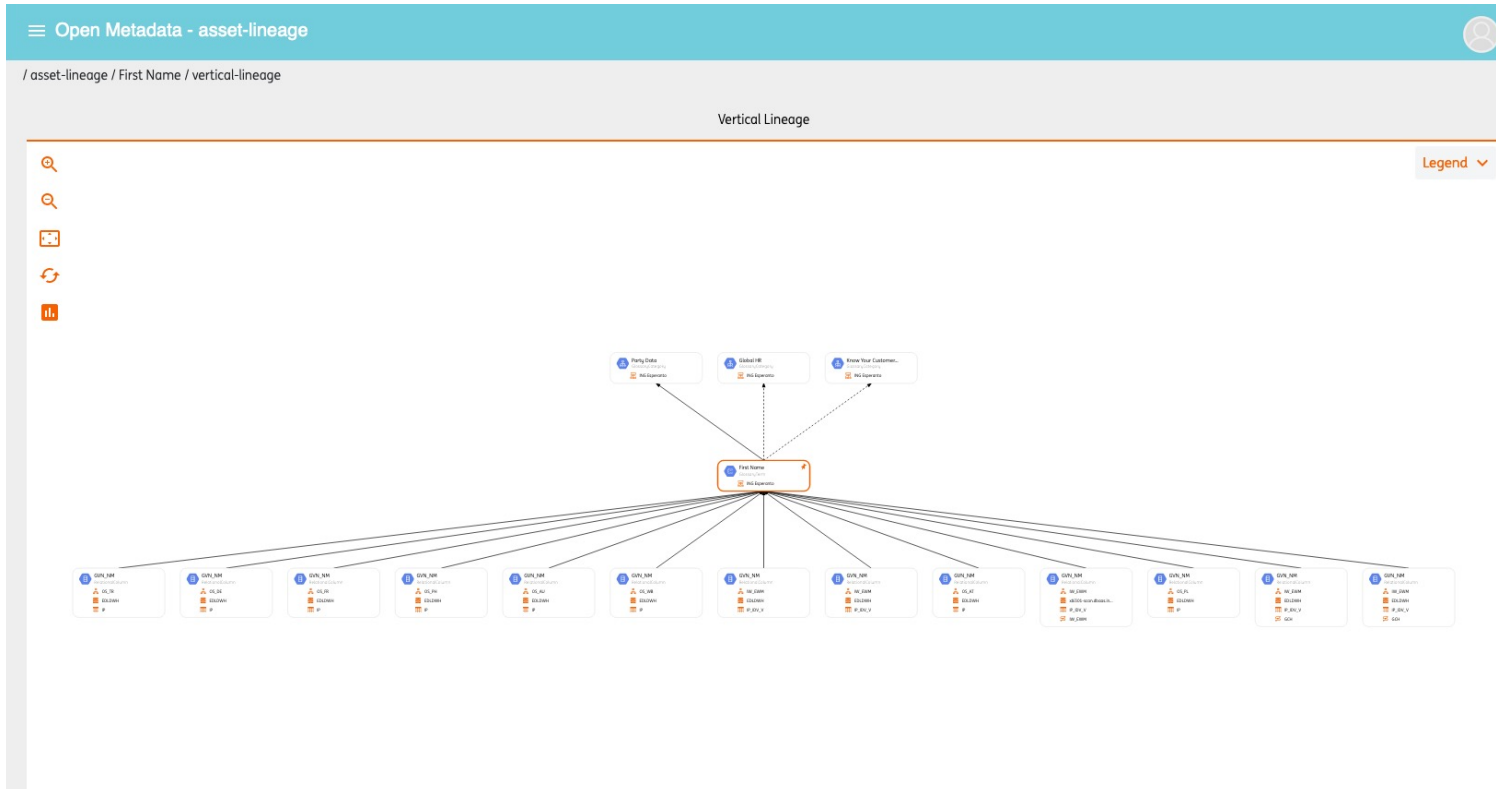
# Horizontal lineage view



# Vertical lineage view



# Vertical lineage view example



# Open forum



# THANK YOU!

<https://odpi.github.io/egeria-docs/features/lineage-management/overview/>



# Egeria's webinar program

Date	time	Title	Description	Presenter
8th November 2021	15:00 UTC	Open lineage	This session will describe the purpose of lineage, what type of information needs to be collected and how this information is managed and used in an enterprise with Egeria. Zoom Conference <a href="https://zoom.us/j/523629111">https://zoom.us/j/523629111</a>	Ljupcho Palashevski and Mandy Chessell
6th December 2021	15:00 UTC	What next after you have built a catalog. Part 1: the journey	Journey from manual cataloging, to using automated integration and templating. Metadata discovery and stewardship will be covered here also as well as metadata deduplication.	Mandy Chessell
10th January 2022	15:00 UTC	Kubernetes operators and Egeria	This session will cover how easy it is to run Egeria in Kubernetes and how the Egeria Kubernetes operator can be used to manage Egeria in a Kubernetes environment.	Nigel Jones
7th February 2022	15:00 UTC	Time Travelling with Egeria	Ever wanted to know what the state of your metadata was at some specific time in the past? This session will introduce the Crux open metadata repository that supports these historical metadata queries.	Chris Grote
7th March 2022	15:00 UTC	How to build a repository connector	Ever wanted to build an OMRS repository connector? This session will take you through what the considerations are and you need to do. It will show how to create the simplest "Hello World" connector.	Chris Grote