BIG DATA LDN. 21-22 SEPTEMBER 2022 OLYMPIA · LONDON

The Enterprise Guide to Building a Data Mesh

Introducing

Spec Mesh

Specification-Driven Data Mesh

Sion Smith CTO - oso.sh

Neil Avery CTO - liquidlabs.com

About us

Sion Smith

CTO, OSO

15 years consulting experience solving complex problems with various cloud and programming technologies

Neil Avery

CTO, Liquidlabs

Distributed systems, previously Confluent, Luxoft, startups and others

Emerging Technology

CONFLUENT

CONSULTING & SI PARTNER

PREFERRED

Agenda

Current State of Play

- The Gartner Hype Cycle
- Foundations of Data Mesh
- Evolution of central nervous system

The Spec Mesh Way

- Domain mapping out of the box
- Example specification
- Data Mesh lifecycle management

Developer Tooling

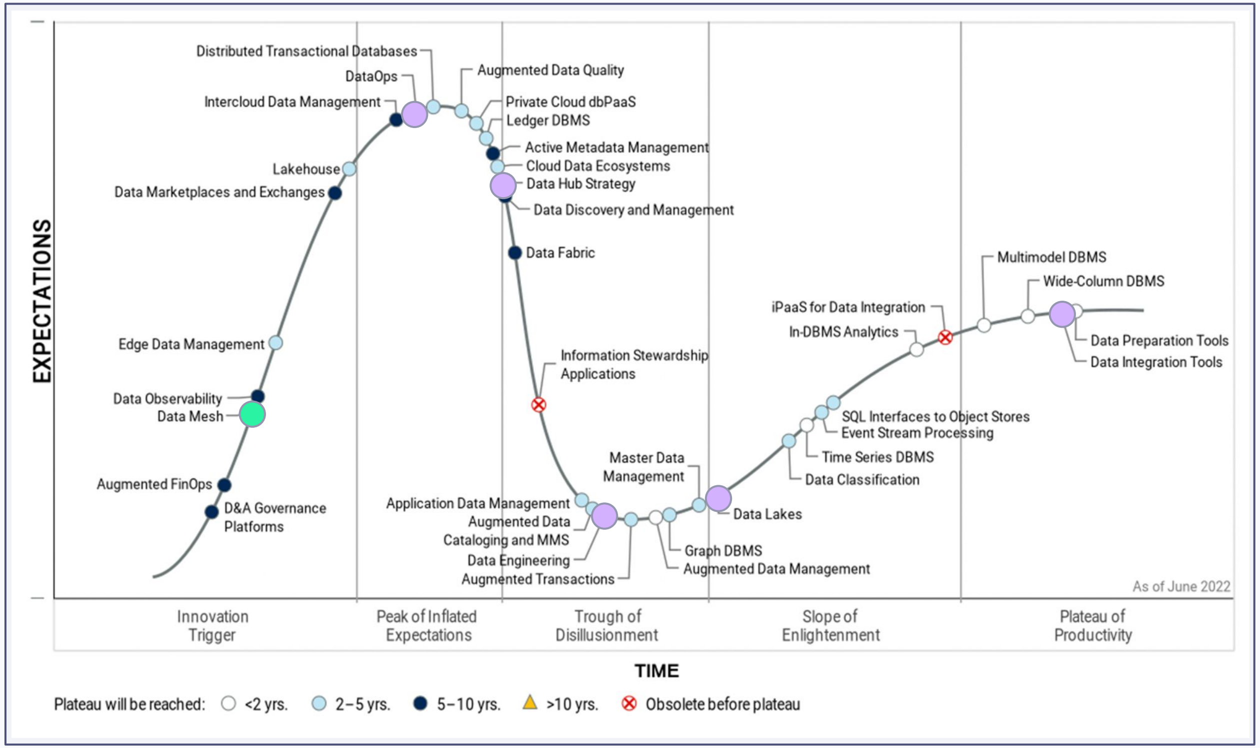
- Features of Spec Mesh
- Screenshots
- Developer roadmap

Our Data Mesh Journey

State of Play

Is the Hype Real?

Gartner Hype Cycle for Data Management - 2022

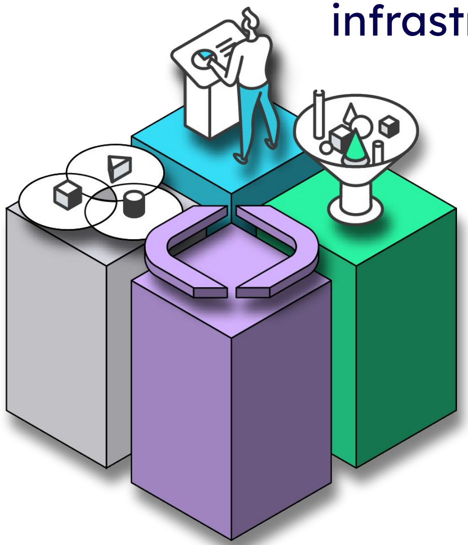


Gartner, Hype Cycle for Data Management, 2022, Donald Feinberg, Philip Russom, Nina Showell, 30 June 2022
https://www.denodo.com/en/document/analyst-report/2022-gartner-hype-cycle-data-management

Four pillars of Data Mesh

Domain-oriented decentralised **data ownership** & architecture

Self-serve data infrastructure as a platform



Data as a **Product**

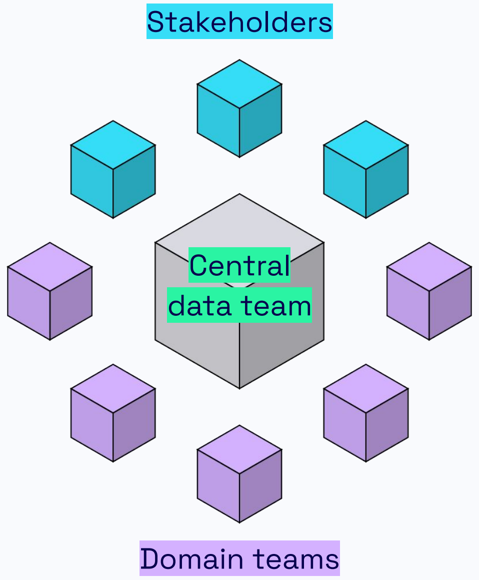
Federated computational **Governance**

Is Data Mesh *really* new?

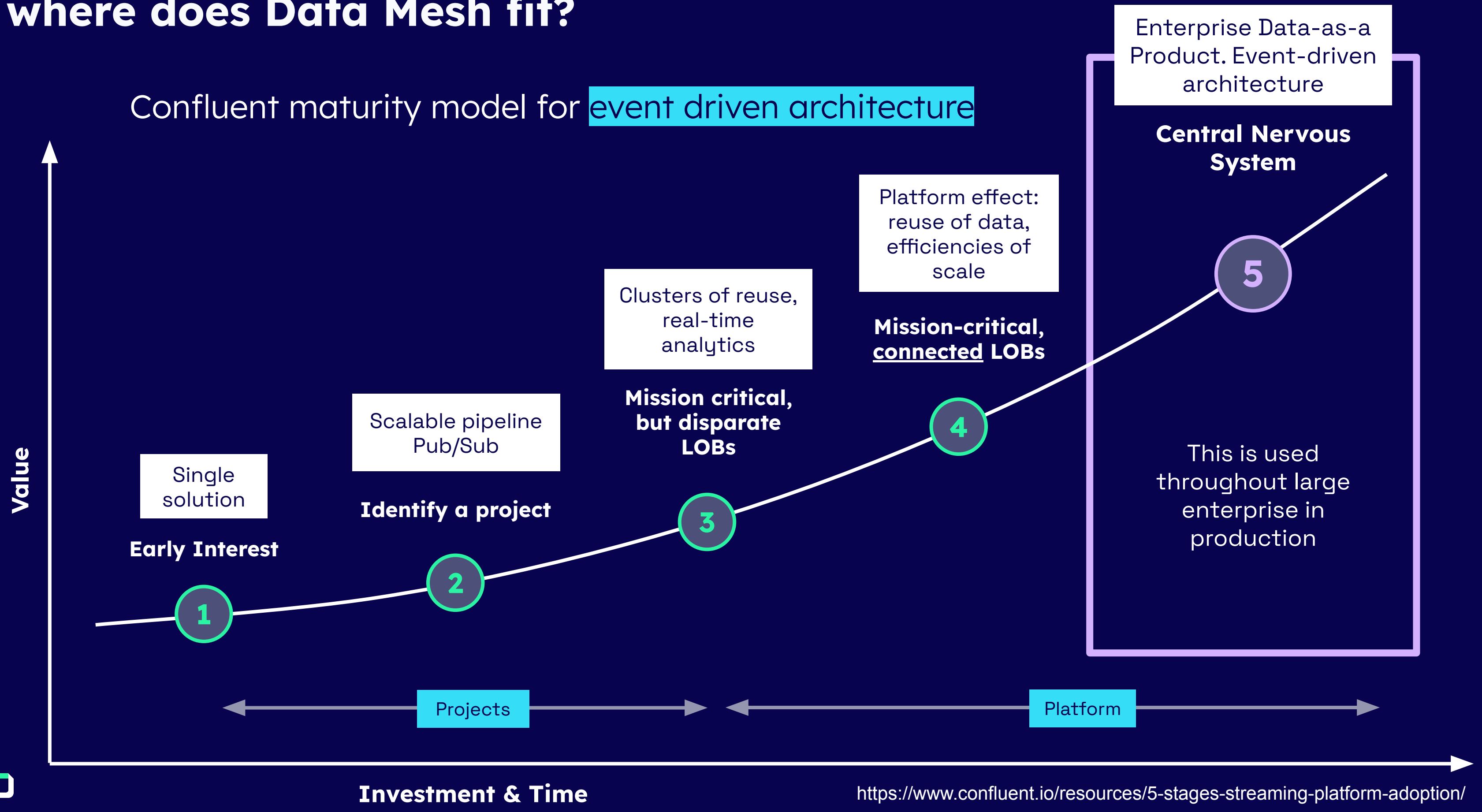
- Data mesh **incremental evolution** of style of architecture we have been building for several years for **event streaming**
- A mature data streaming system adopts a **central nervous system**
- Can we build a data mesh around event streaming principles?
- A central nervous system models topics with a domain structure and **federated computational governance**

Introducing:

- An agreement / contract for data mesh using a specification

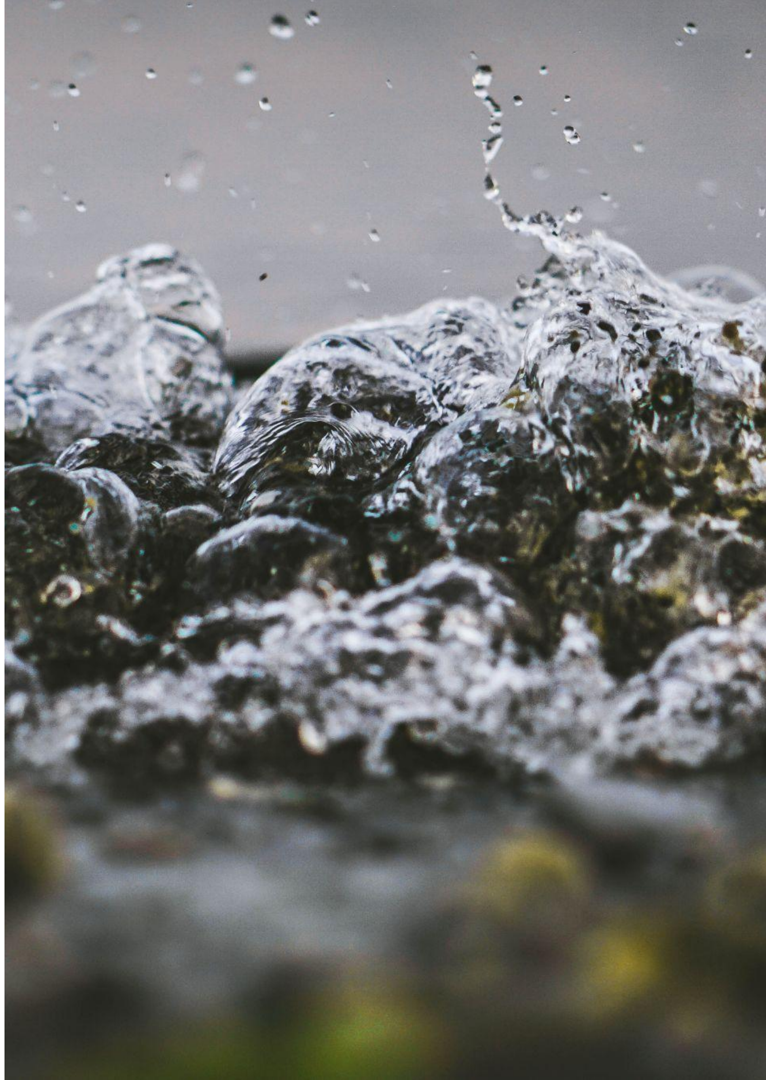


So where does Data Mesh fit?


Confluent maturity model for event driven architecture
Value
Investment & Time
Single solution
Early Interest
1
Scalable pipeline Pub/Sub
Identify a project
2
Clusters of reuse, real-time analytics
Mission critical, but disparate LOBs
3
Platform effect: reuse of data, efficiencies of scale
Mission-critical, connected LOBs
4
Enterprise Data-as-a Product. Event-driven architecture
Central Nervous System
5
This is used throughout large enterprise in production
Projects
Platform

https://www.confluent.io/resources/5-stages-streaming-platform-adoption/

Data Mesh should not boil the ocean



Introducing SpecMesh

Specification-Driven Data Mesh

CNS Patterns applied to Data Mesh

1. Events and storage comprise data platform fundamentals required to build almost anything
2. Events and storage **already exist** and that won't change
3. Organise data resources **hierarchically**
4. Enforcing a **domain model** is need to control complexity and cope with scale
5. Orgs have a **data quality** function
6. Orgs have a **data catalogue** function
7. **Registry models** are required to model data (and support data evolution etc)
8. Most organisations suffer from **inconsistent** mechanisms for data-as-an-api

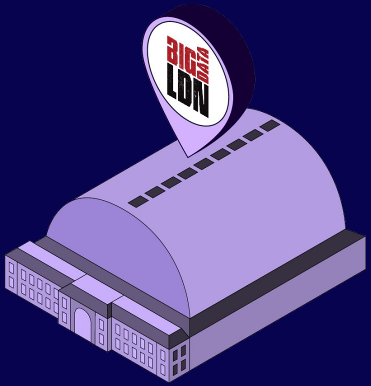
Supporting the pillars of Data Mesh

Features	Domain ownership	Self-serve	Data as a Product	Federated Computational Governance
Spec driven (async-api spec)	✔		✔	
SDLC plugins (unit-test, integration-test)		✔		
SDLC plugins (provision - terraform)		✔	✔	✔
3rd party data catalogue	✔		✔	✔

Big Data London is our data product

Services
BIG DATA LDN
AIRPORT
Travel
Retail
Hammersmith
Accomodation

Data Model



/ london / borough / venue / event
/ retail /
/ transport /
/ accommodation /
/ services /

/ london / hammersmith / olympia / bigdatalondon / public / attendee

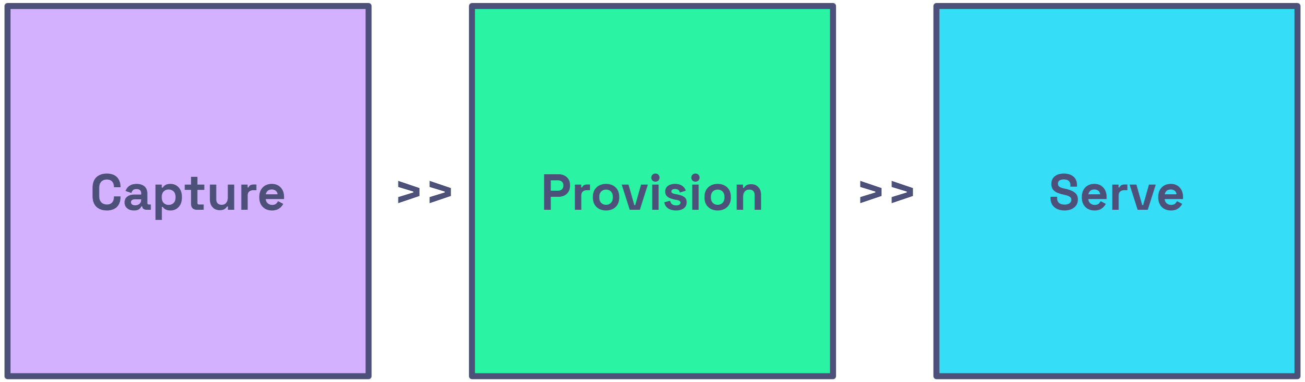
/ london / hammersmith / transport / public / tube
/ london / heathrow / transport / public / airport

/ london / hammersmith / olympia / bigdatalondon / vendor / terra / public / visitor
/ london / hammersmith / olympia / bigdatalondon / retailer / subway / public / purchase

Spec Mesh

```
asyncapi: '2.4.0'
id: 'urn:london:hammersmith:olympia:bigdatalondon'
Info:
 title: BigDataLondon API
 version: '1.0.0'
 description: Simple model of BigDataLondon as a Data Product

Servers:
 test:

  url: test.mykafkacluster.org:8092
  protocol: kafka-secure

channels:

 public/attendee:
  Publish:
    summary: Humans arriving
   Message:
     name: Human
      "tags": [
       "name": "human",
       "name": "big data london"]
     Payload:
      type: object
      Properties:
        Id:
         type: integer
         minimum: 0
         description: Id of the human
        Age:
         type: integer
         description: Age of the human
```

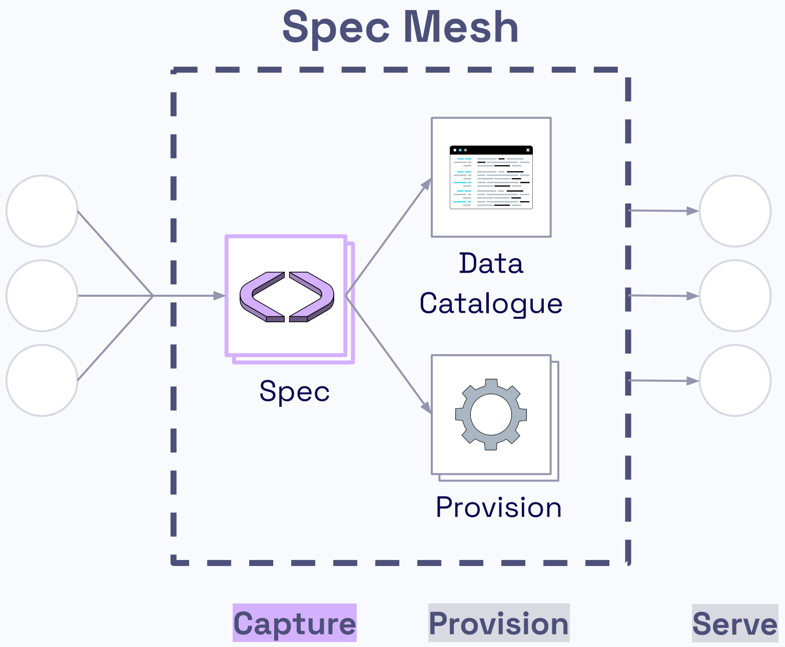
Data Mesh as Code: Spec Mesh



What is Spec Mesh?

Capture

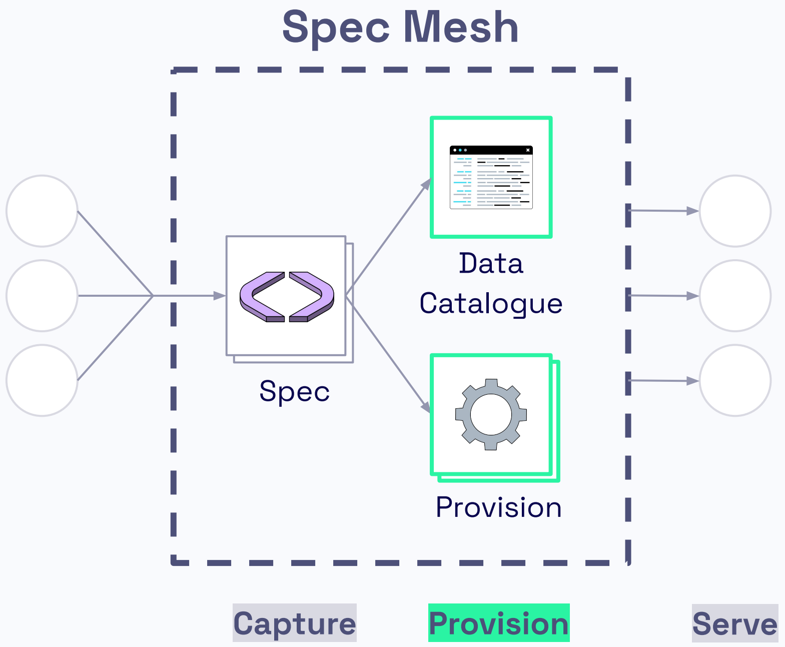
- Create repo from git template
- Async-api spec
- Apply the domain model/structure
- Specify data event streams & storage
- Model the data entities (schemas)
- Apply tags to data entities
- Permission data as public/protected/private
- Write tests and build your product



What is Spec Mesh?

Provision

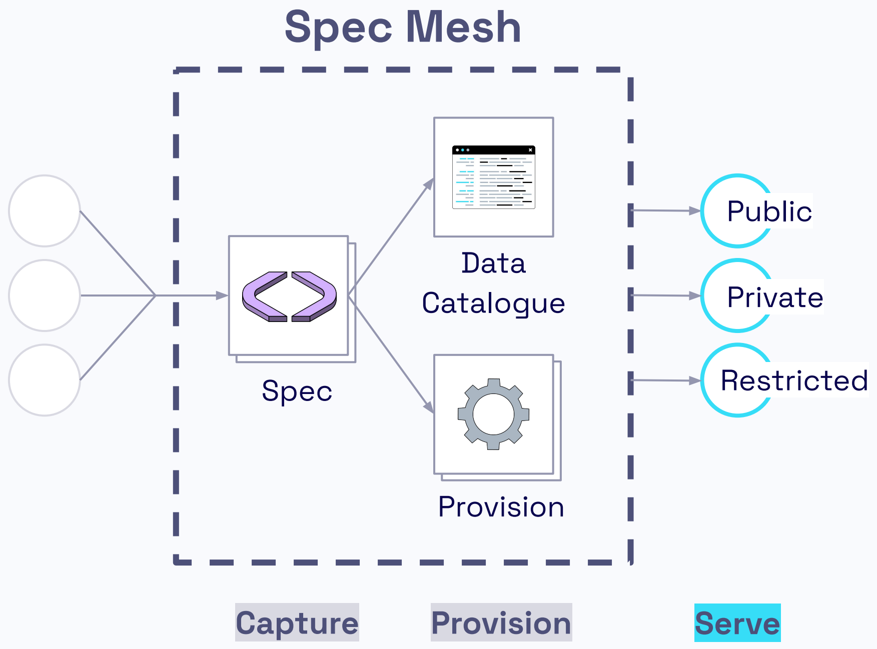
- Pipeline driven provisioning of data product based on Specification
- Includes domain structure, governance and permissions
- Data catalogue automatically updated (including tags)
- Data schema's published to registry



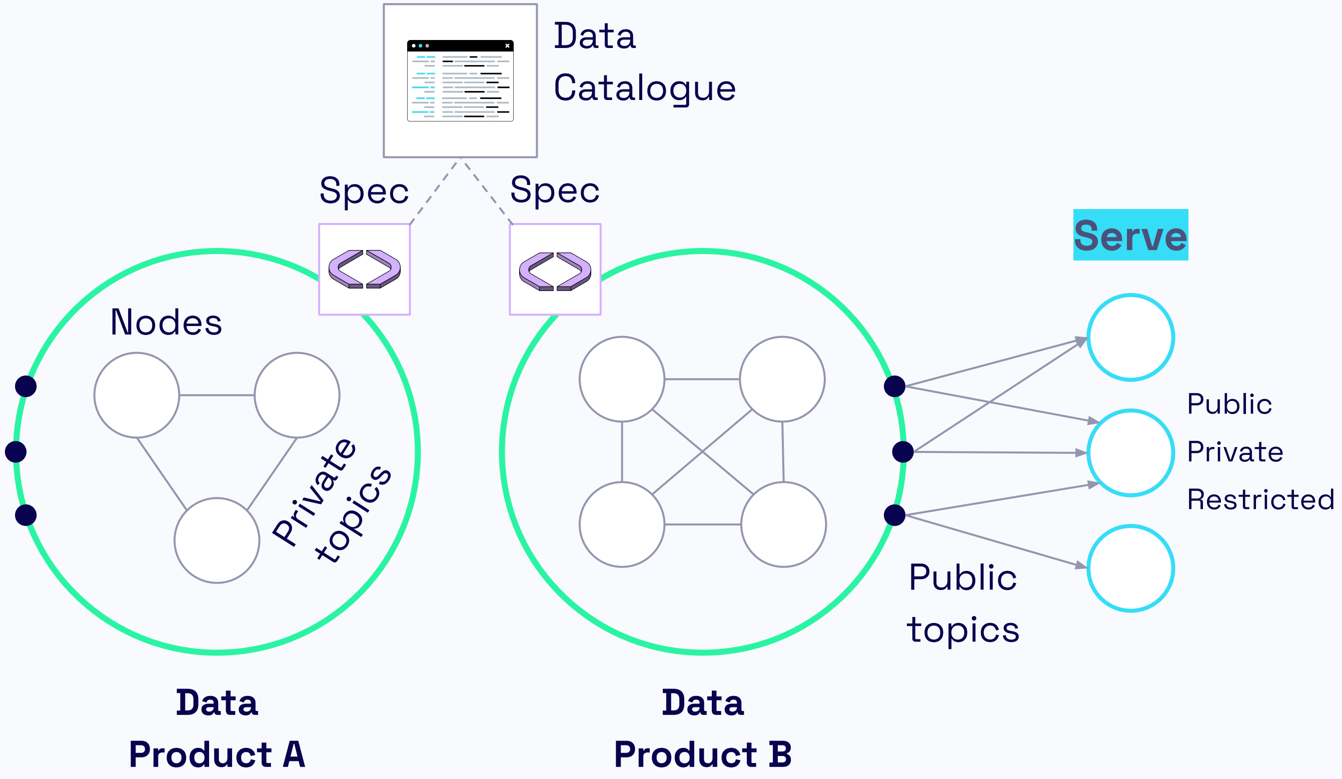
What is Spec Mesh?

Serve

- Governance integrated via underlying Data Platform. I.e. Kafka ACLs
- Data catalogue supports discovery (via tags), and access requests
- Data resources are structured using public, private and protected scope



Spec Mesh conceptual



Developer Tooling

Why use Spec Mesh?

Developer focused tooling:. This is not some platform

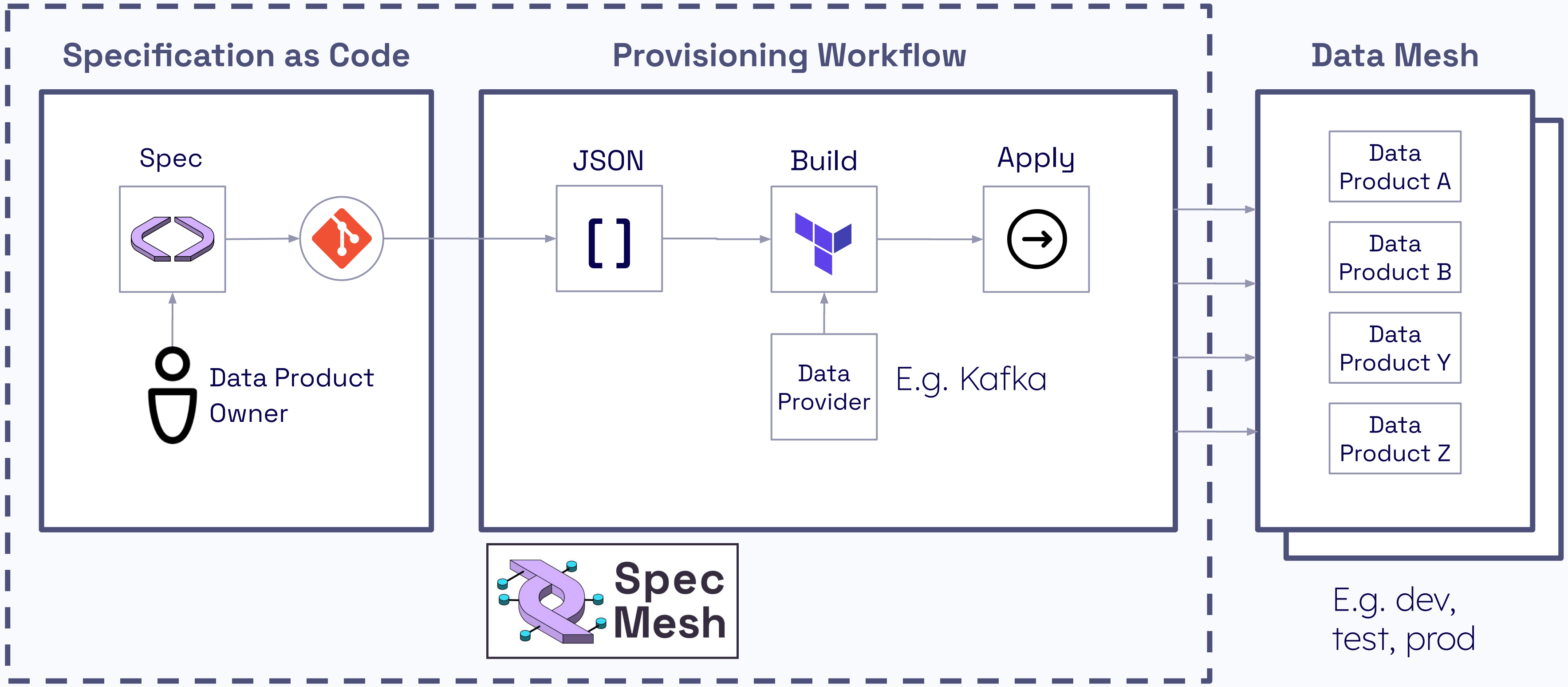
Abstract away complexities: Leverage layers of abstraction to create repeatable patterns

Unit testing specifications: Java based tooling supports development lifecycle

Tooling support flexibility: Using configurable guardrails

Increased extensibility: Interoperate with existing Data Management tooling and modules

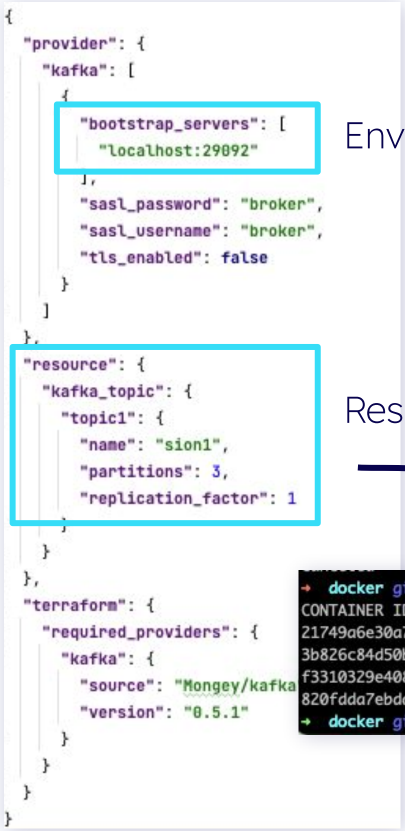
What do you get out of the box?



Developer Tooling

Demo

Specification output (JSON)

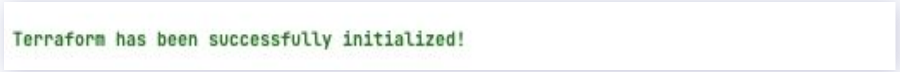


```
{
  "provider": {
    "kafka": [
      {
        "bootstrap_servers": [
          "localhost:29092"
        ],
        "sasl_password": "broker",
        "sasl_username": "broker",
        "tls_enabled": false
      }
    ]
  },
  "resource": {
    "kafka_topic": {
      "topic1": {
        "name": "sion1",
        "partitions": 3,
        "replication_factor": 1
      }
    }
  },
  "terraform": {
    "required_providers": {
      "kafka": {
        "source": "Mongey/kafka
        "version": "0.5.1"
      }
    }
  }
}
```

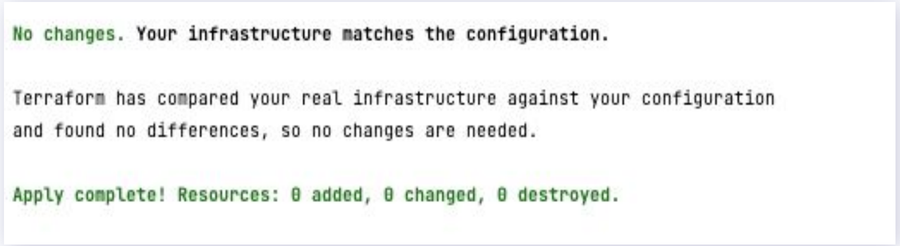
Environment

Resources

Provisioning summary

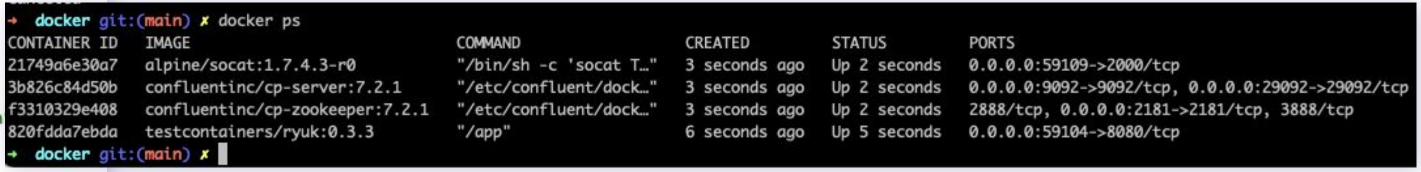


```
Terraform has been successfully initialized!
```



```
No changes. Your infrastructure matches the configuration.

Terraform has compared your real infrastructure against your configuration
and found no differences, so no changes are needed.

Apply complete! Resources: 0 added, 0 changed, 0 destroyed.
```

Kafka Infrastructure



```
→ docker git:(main) ✗ docker ps
CONTAINER ID   IMAGE                             COMMAND                  CREATED          STATUS         PORTS
21749a6e30a7   alpine/socat:1.7.4.3-r0           "/bin/sh -c 'socat T…"   3 seconds ago    Up 2 seconds   0.0.0.0:59109->2000/tcp
3b826c84d50b   confluentinc/cp-server:7.2.1      "/etc/confluent/dock…"   3 seconds ago    Up 2 seconds   0.0.0.0:9092->9092/tcp, 0.0.0.0:29092->29092/tcp
f3310329e408   confluentinc/cp-zookeeper:7.2.1   "/etc/confluent/dock…"   3 seconds ago    Up 2 seconds   2888/tcp, 0.0.0.0:2181->2181/tcp, 3888/tcp
820fdda7ebda   testcontainers/ryuk:0.3.3         "/app"                   6 seconds ago    Up 5 seconds   0.0.0.0:59104->8080/tcp
→ docker git:(main) ✗
```

Roadmap

Proof of Value - Q4 22

Developer Experience
- Local testing support (Java & Python) - Test containers via gradle plugin
- SpecMesh gradle plugin to use HCL Terraform & state mgmt

Domain model
- Domain modelling built into Spec.Id field

Build Pipeline
- Executions pushes the APISpec to apply orthogonal resource allocation

Governance
- Manual governance of ‘public’ resources through Credential creation (API Keys + ACLs)

Extensions
- Kafka support for Async API

Q1 - 2023

Extensions
- More Kafka extensions supported (incl provisioning support): Quotas, Storage & ACLs for Public, Private, Protected resources (basic governance)

Data Catalogue
- Initial integration to existing products to support discovery and basic governance

Storage
- Storage resource mapping

Governance
- Delegation to a Governance API where protected topics are specified and requested

Q2 - 2023

Data As A Product
- Data Product Open Registry; simple Data Catalogue with Discovery and tagging (Spec resources contain tags - show example)

Observability
- Topology view (derived from specs showing product/consume data flows and ownership visualisation)

Bolt-ons
- Multi-region
- Cross resource
- SMT
- DQ integration + other simple additions

Beyond

Community driven

Help us shape the roadmap by suggesting and contributing to features!

Thank you

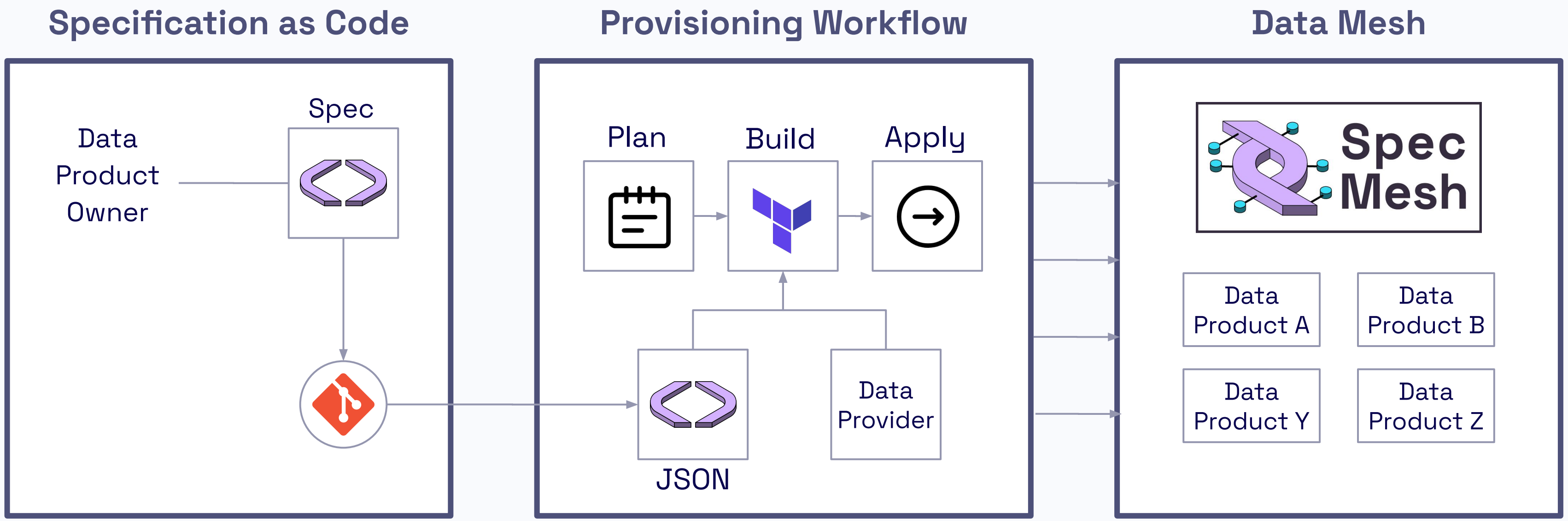
Get involved!

Scan & thumbs ups on the Github issue

github.com/specmesh

@osodevops @avery_neil

What do you get out of the box?



Roadmap - THIS HAS BEEN MAPPED TO A SLIDE

Proof of Value (0.1) (current status of Spec Mesh OS):

- Developer support for local testing (Java and Python) - i.e. TestContainers resources are provisioned using gradle plugin. The Spec Mesh Gradle plugin uses Terraform (making it very open to using existing operators, scripts, i.e. CDK)
- Domain modelling is built into the Spec.Id field - and upon provisioning via the Gradle/Terraform plugin applies correct domain knowledge to resources (see example)
- Build pipeline execution pushes the APISpec to apply orthogonal resource allocation (state mgmt via terraform) to provision pipe-line environment resources (build pipe example showing DEV, UAT and PROD)
- Manual governance of 'public' resources through Credential creation (API Keys + ACLs)
- Kafka extensions for Async API spec is being built

Roadmap after PoV (local testing and build pipeline)

1. More Kafka extensions supported (incl provisioning support): Quotas, Storage & ACLs for Public, Private, Protected resources (basic governance)
2. Storage resource mapping
3. Initial Catalogue integration to existing products to support discovery and basic governance
4. Delegation to a Governance API where protected topics are specified and requested
5. Data Product Open Registry: simple Data Catalogue with Discovery and tagging (Spec resources contain tags - show example)
6. Topology view (derived from specs showing product/consume data flows and ownership visualisation)
7. Bolt ons: multi-region, cross resource, SMT, DQ integration + other simple additions

Data Model

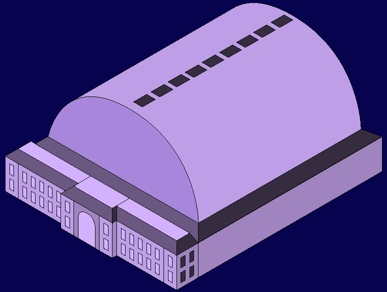
/ london / borough / venue / event

⬇

/ london / hammersmith / olympia / bigdataldn

/

/



/

Map domain model into spec

- Now that specification in my branch / git repo
- I need to build a test around this / init test my data product (include the topic/channel name)
- The unit test in this example is Big Data LDN (People coming to the event)
    - Test container example

Established patterns of Data Mesh and central nervous system

Data as a Product

Discoverability	Spec Mesh provides	
Addressability		
Trustworthy		
Self-describing semantics		
Interoperable / standards		
Security		

Build pipeline

- Leverage @IntegrationTest - gets run first and calls on the SpecMeshProsivioner (Java-> terraform) to parse the spec, build topic names and execute against target cluster location

DEMO MAYBE

- ALL IN JAVA
  - Drive TF from Java
  - Parse the spec from github location
  - Provision kafka topics from what is in the specification
-
- LETS SEE THIS BAD BOY IN ACTION

stuff

The Mission

To make Data Mesh simple, Open Source and available to all - without lockin, without complex tooling by using an approach centered around ‘specifications’, existing tools and baking in a ‘domain’ model.

<insert triangular image here>

Why we should build a mesh using spec's

Benefits

Organisation

+ Real life insights into technology problems
+ Uncover new opportunities
+ Builds trust and good working relationships
+ Brand awareness

Domain

+ Insights into emerging technologies
+ Access to technology experts
+ Technical focus
+ Professional development for staff
+ Builds team morale
+ Free lunch!

Set it up

Line up with client

↓

Get it in the diary

↓

Agree the theme and key questions

↓

Arrange lunch

↓

Run the session

Build using ASyncAPI specifications

Outline

Talk Structure (30 mins) 15 slides

- we believe data mesh should not be a large complex undertaken lasting multi years costing millions
- Target audience developers
- we believe data mesh should be useable and free for everyone. ITs not about complex custom built tooling
- foundations are opinionated on a specification way
- Problem statement (Why? Most Kafka environments we work in are shit, its a graveyard of topics - no uniform way of provisioning we cannot scale)
- Why do we think data mesh has anything todo with this. These are the principles: how we are using these principles to build Kafka at scale.
- Data ownership by domain
- your domain model is defined by people in your organisation, they should be able to look at it and know where they sit and where to discovery events they need.
- data governance / data as a product
- why we think the async api matter, data as a product is represented by the api / specification for data > that allows us to model the domain and thefore the
- Modelling London, here are the specs for parts of this
- data model Fulham
- you came in from as an event today
- Here is the spec for big data ldn : this is what it looks like when we use it > we are using this to test our code
- local test > we need to get away from the graveyard >
- Domain model for London (
- private events buying a kebab
- public events: me moving from Fulham to central

Data mesh, self service ubiquitous access to data

Goal: To make Data Mesh simple, Open Source and available to all - without lockin, without complex tooling by using an approach centered around ‘specifications’, existing tools and baking in a ‘domain’ model. (see image)

Scope - running OTS infrastructure like connectors and all things, webservices, microservices or processors…. They are the problem of the client, not the Mesh itself

Problem statement (what):

- Everyone is touting data mesh and it’s getting confusing, many bespoke, many commercial solutions… it should be accessible & simple, without lockin
- We see Data Mesh as a natural, formalisation of building the central nervous system (everything via streams of topic data)
- Been building CNS for many years across some of the largest companies in the world
- CNS focuses on Event-Streams, but during implementation includes most of the guiding principles of Datamesh (ownership by domain, as a product, selfservice, governance)
- Avoid vendor lockin (doesnt need to be all encompassing and conflict with existing data infra)
- Most data-mesh solutions dont provide SDLC/development tools for test and infrastructure

Why:

- A specification based approach provides uniformity (part of the solution)
- Fixes Kafka resource mess/grave-yard
- Testing support (developer SDLC unlike most tooling)
- Repeatability -> provides guard-rails, reuse, reliable, simple and consistent approach for all
- Not just kafka, but eventually Kinesis, G PubSub, EventHubs, S3, from OnPrem to Cloud, multiple envs, clusters etc

How (Can we do this with minimal effort by leveraging existing tools and principles):

- Simple: Data Ubiquity should have 2 forms, Streams of Events (Kafka stream or others) + Storage (think S3) (because not everything fits in a stream). This simplifies everything - don't boil the ocean
- Simple: Specification based using AsyncAPI specs
- Simple and existing: Data as an API in the CNS maps into AsyncAPI Spec + Schemas
- Existing: by using the AsyncAPI Spec 'id' hierarchy we can express domain ownership (example)

Data Mesh Principle mapping

- Domain ownership is captured using the ID of the spec (example)
- Data as a product is reflected by the Spec itself, the spec is used to provision resources (example) (more later)
- Data available everywhere (discoverable) - built using existing tools such as AWS Glue Data Catalogue, Collibra and others
- Data governed everywhere - using streams and storage we build a integration to the data catalogue, and integrate access requests through the use of private, protected, public topics & storage while automating restrictive controls (i.e. ACLs in Kafka, principle mapping etc).

Worked example of the city of london → BDL

PoV (current status of Spec Mesh OS):

1. Developer support for local testing (Java and Python) - i.e. TestContainers resources are provisioned using gradle plugin. The Spec Mesh Gradle plugin uses Terraform (making it very open to using existing operators, scripts, i.e. CDK)
2. Domain modelling is built into the Spec.Id field - and upon provisioning via the Gradle/Terraform plugin applies correct domain knowledge to resources (see example)
3. Build pipeline execution pushes the APISpec to apply orthogonal resource allocation (state mgmt via terraform) to provision pipe-line environment resources (build pipe example showing DEV, UAT and PROD)
4. Manual governance of 'public' resources through Credential creation (API Keys + ACLs)
5. Kafka extensions for Async API spec is being built

Challenges! No one size fits all

There is not one tool to solve all the problems, a framework or suite of tools is needed.

+ Every company is different, there is **no prescribed solution**
+ **Scaling** your data products as more is captured
+ How to separate **signals** from **noise**?
+ **Complexity** of data is growing exponentially
+ Maintaining **data quality** whilst keeping consistent

Roadmap after PoV (local testing and build pipeline)

1. More Kafka extensions supported (incl provisioning support): Quotas, Storage & ACLs for Public, Private, Protected resources (basic governance)
2. Initial Catalogue integration to existing products to support discovery and basic governance
3. Delegation to a Governance API where protected topics are specified and requested
4. Data Product Open Registry: simple Data Catalogue with Discovery and tagging (Spec resources contain tags - show example)
5. Topology view (derived from specs showing product/consume data flows and ownership visualisation)
6. Bolt ons: multi-region, cross resource, SMT, DQ integration + other simple additions

Sion Notes

- People can work more collaboratively using a decentralised standardised approach
    - different levels of expertise of using data in each domain
- how do you mentor and empower each domain on contributing to the mesh
    - set some enterprise level standards so teams can educate themselves on things like the team structure etc.
- Decentralised doesn’t mean free for all
- Federated governance is at the domain. Example the finance team needs to know what privacy
- NOT building a data silos, its more about people and processes not the technology
    - interoperability and easy to navigate
- Not every domain knows how to build and manage a scalable API.

Pillars of Modern Data Architecture

- Scalable on demand
- Purpose-build data services
- Seamless data movement
- unified governance
- Performant and cost-effective

Sion notes

Why Spec Mesh

- Encourage data-driven agility
- Support domain-local governance through lightweight specifications
- Isolate data resources with clear contracts
- Consider data-as-a-specification which can exist in any system

Introduction

Why title

Design and Build title

Scale title

+ aaa

+ aaa

+ aaa