Project Pandora

How can we produce and extend the digital resources of cultural heritage as Linked Open Data?

PANDORA Fundamentals

RDF Persistence

How do we read and write RDF on the web effectively?

Linked Data Platform

Linked Data Platform (LDP) defines a set of rules for HTTP operations on web resources, some based on RDF, to provide an architecture for read-write Linked Data on the web.

[2] https://www.w3.org/TR/ldp/

Problem:

- How can clients and servers exchange Linked Data resources with a shared set of rules that define expectations?

Use cases for LDP

- ➔ As an institution, I need to publish large volumes of digital resources and metadata.
- ➔ As an institution, I need to provide discovery for digital resources as collections.
- ➔ As an institution, I require **Scalability, Durability, Modularity, Flexibility** for all systems.
- ➔ As an institution, I need to depend on **Web Standards**
- ➔ As an institution, I need **External Integrations**

LDP provides a standard for Linked Open Data Exchange

What some institutions do today...

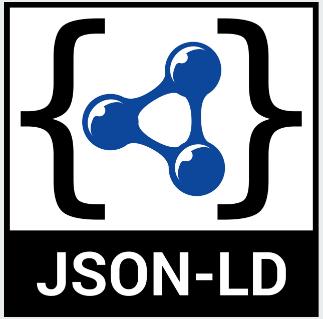
Initiatives towards interoperability

International Image Interoperability Framework

The IIIF provides a common set of APIs to allow institutions to share and publish their digital resources.

Problem:

- the “interoperability” of these APIs is currently realized by clients that can juxtapose, *but do not semantically relate* disparately served resources!



A Linked Data Format

- JSON-LD is a JSON-based format to serialize Linked Data.
- It is the format used for IIIF API serializations.
- A JSON-LD serialization can be produced from RDF.
- A JSON-LD serialization can be deserialized to RDF.

Problem:

- Current IIIF clients only understand JSON-LD as JSON, and typical IIIF implementations do not serialize JSON-LD from RDF.

an abstract syntax

RDF graphs can be serialized by Jena RIOT in 25 different formats

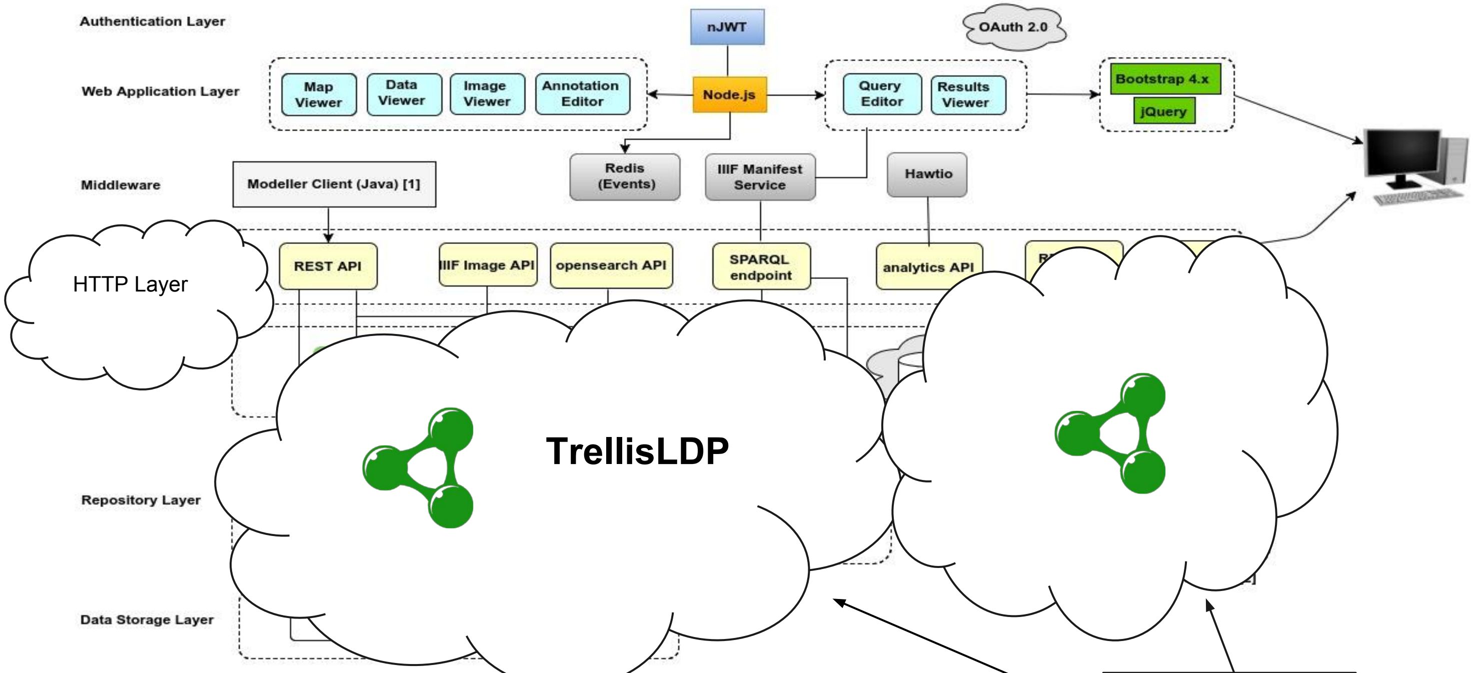
- RDF is not a serialization format.
- Data persisted as RDF provides optimal flexibility, portability, and extensibility.
- A JSON-LD formatted document can also be read and queried as a GRAPH if deserialized to RDF.
- Institutions who are already producing JSON-LD documents are providing the basis for future linked data implementations of them.

Problem:

- How can producers create abstract data natively and output expected data shapes from abstract source data?

PANDORA Design

PANDORA Architecture v.0.3.0



Revisions (v.0.4.0)

[1] Used for binary input and LDP container/metadata modelling in Fedora

[2] annotation storage applications include a client API for webapp client data access and data input/editing capability.
-- These apps store "descriptive annotations" that are HTTP addressable and are primarily referenced from an "annotation list" associated with a digital object resource in Fedora

Trellis LDP

http://www.trellisldp.org/

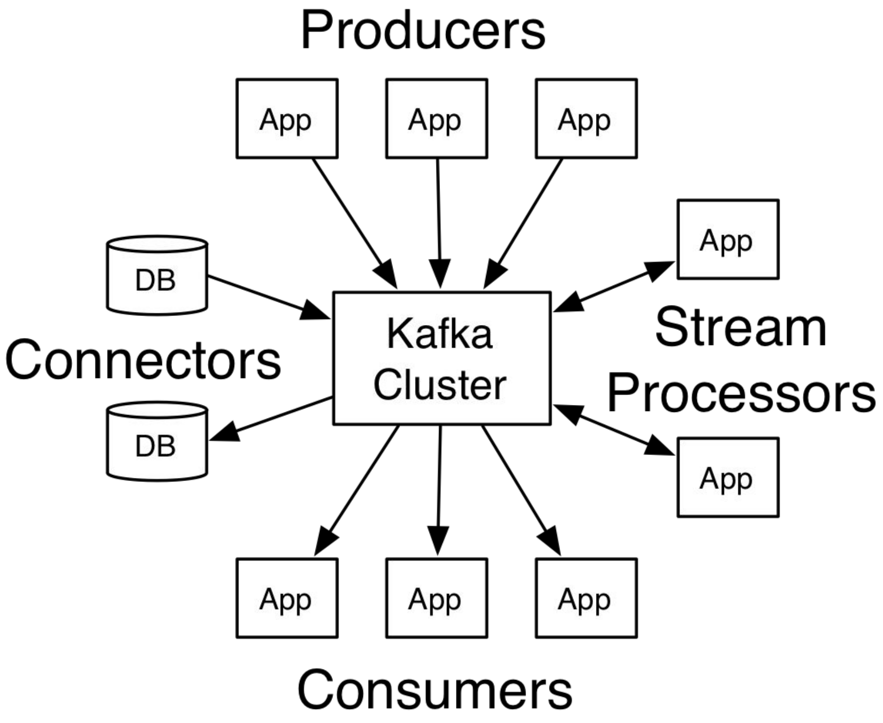
Trellis is a linked data server that can be used for storing large volumes of content, particularly in cases where that content is stored for years and decades.

Trellis is an HTTP/1.1 server designed to comply with the following specifications:

- W3C LDP Server
- W3C Activity Streams
- RFC 7089 (HTTP Framework for Time-Based Access to Resource States -- Memento)
- RFC 3230 (Instance Digests in HTTP)
- Solid WebAC (Authorization and Access Control)

Why Trellis is better than existing solutions

Scalability:	Trellis has been built on components that already support horizontal scalability: **Kafka, Zookeeper and Beam**. While it is possible to run Trellis on a single machine, scaling out across a cluster is well-defined and supported.
Durability:	Trellis makes it possible to retrieve the state of any resource at any arbitrary point in time.
Flexibility:	You can use any RDF vocabulary; you can store binaries of any type.
Extensibility:	Any time a resource is created, modified or deleted, a notification is made available, making it easy to use an integration framework to connect Trellis to external applications.
Modularity	The overall code base for Trellis is small, and it is divided up into even smaller modules that can be maintained independently.



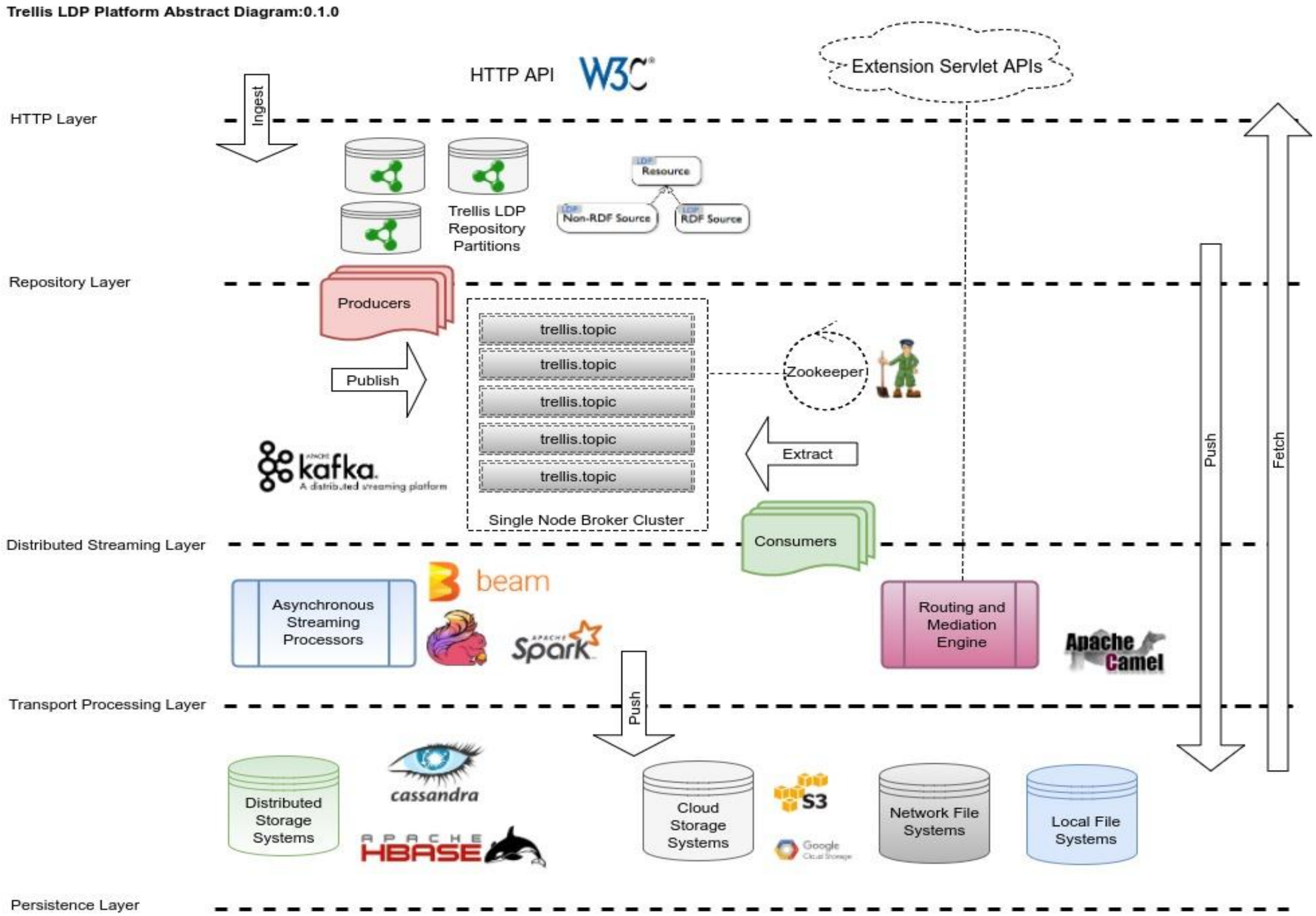
What is Kafka good for?

It gets used for two broad classes of application:

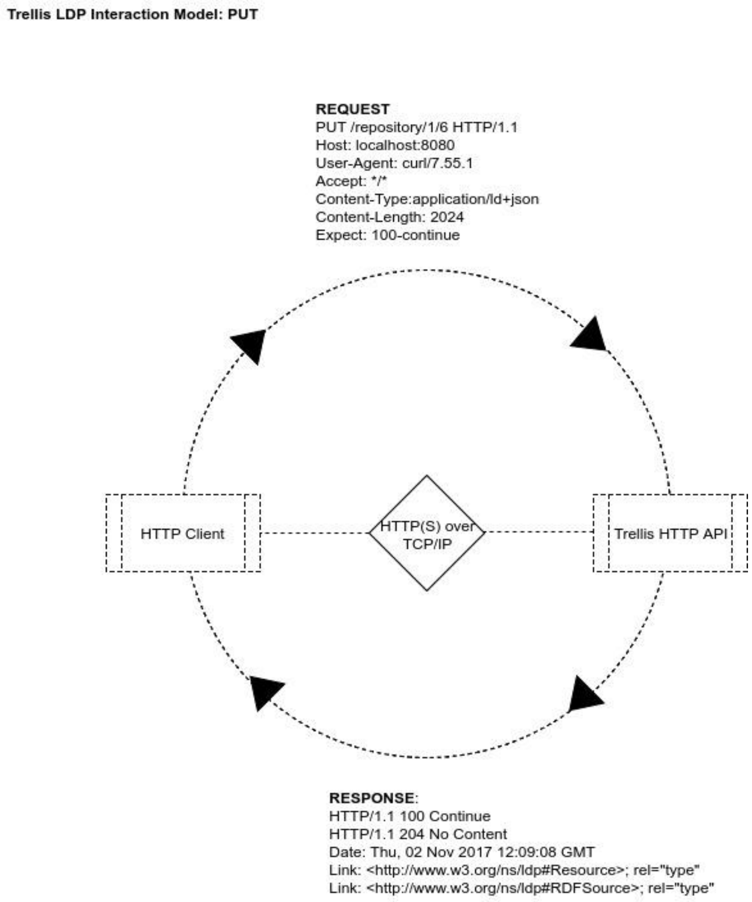
- Building real-time streaming data pipelines that reliably get data between systems or applications
- Building real-time streaming applications that transform or react to the streams of data

Kafka is run as a cluster on one or more servers.
The Kafka cluster stores streams of *records* in categories called *topics*.
Each record consists of a key, a value, and a timestamp.

Kafka distributed streaming platform



Architecture



Trellis LDP Interaction Model: PUT

REQUEST

```
PUT /repository/1/6 HTTP/1.1
Host: localhost:8080
User-Agent: curl/7.55.1
Accept: */*
Content-Type:application/ld+json
Content-Length: 2024
Expect: 100-continue
```

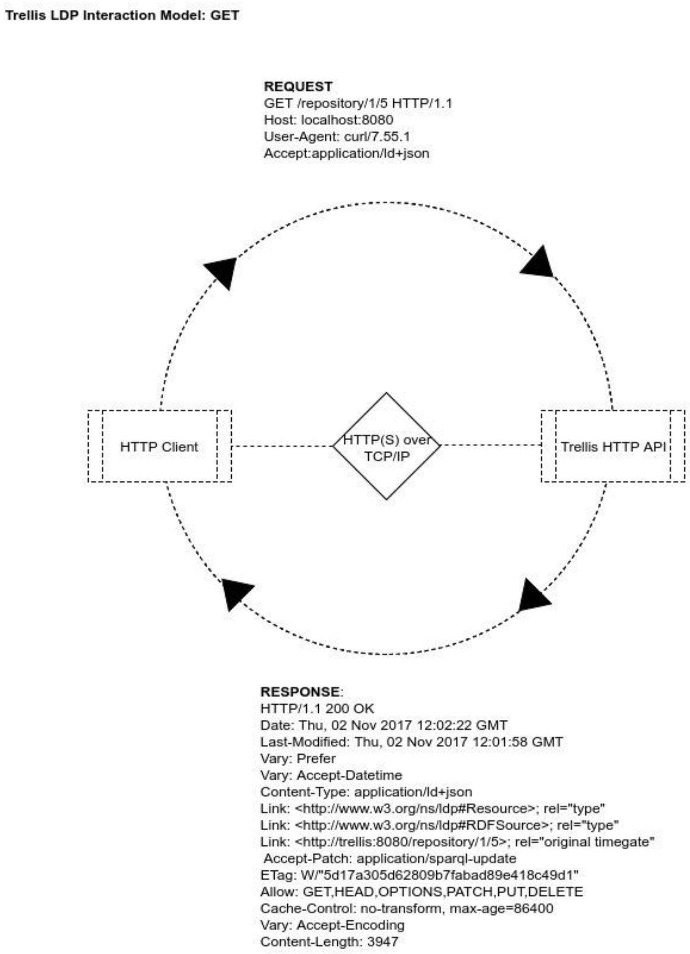
RESPONSE:

```
HTTP/1.1 100 Continue
HTTP/1.1 204 No Content
Date: Thu, 02 Nov 2017 12:09:08 GMT
Link: <http://www.w3.org/ns/ldp#Resource>; rel="type"
Link: <http://www.w3.org/ns/ldp#RDFSource>; rel="type"
```

Input Stream: JSON-LD

```
{
 "@context": "http://iiif.io/api/presentation/2/context.json",
 "@type": "oa:Annotation",
 "motivation": [
   "oa:commenting"
 ],
 "resource": [
   {
     "@type": "dctypes:Text",
     "format": "text/html",
     "chars": "<p>what a great design</p>"
   }
 ],
 "on": [
   {
     "@type": "oa:SpecificResource",
     "full":
"https://purl.stanford.edu/sz557xs2372/iiif/canvas/sz557xs2372_1"
,
     "selector": {
       "@type": "oa:Choice",
       "default": {
         "@type": "oa:FragmentSelector",
         "value": "xywh=1523,1408,2075,1357"
       },
       "item": {
         "@type": "oa:SvgSelector",
         "value": "<svg xmlns='http://www.w3.org/2000/svg'></svg>"
       }
     },
     "within": {
       "@id": "https://purl.stanford.edu/sz557xs2372/iiif/manifest",
       "@type": "sc:Manifest"
     }
   }
 ]
}
```

Interaction Model: `http://www.w3.org/ns/ldp#RDFSource;`



Trellis LDP Interaction Model: GET

REQUEST
GET /repository/1/5 HTTP/1.1
Host: localhost:8080
User-Agent: curl/7.55.1
Accept:application/ld+json

RESPONSE:
HTTP/1.1 200 OK
Date: Thu, 02 Nov 2017 12:02:22 GMT
Last-Modified: Thu, 02 Nov 2017 12:01:58 GMT
Vary: Prefer
Vary: Accept-Datetime
Content-Type: application/ld+json
Link: <http://www.w3.org/ns/ldp#Resource>; rel="type"
Link: <http://www.w3.org/ns/ldp#RDFSource>; rel="type"
Link: <http://trellis:8080/repository/1/5>; rel="original timegate"
Accept-Patch: application/sparql-update
ETag: W/"5d17a305d62809b7fabad89e418c49d1"
Allow: GET,HEAD,OPTIONS,PATCH,PUT,DELETE
Cache-Control: no-transform, max-age=86400
Vary: Accept-Encoding
Content-Length: 3947

Output Stream: JSON-LD (compacted)

```
{
 "@graph": [
  {
   "id": "_:b0",
   "type": "Annotation",
   "body": "_:b1",
   "target": "_:b2",
   "motivation": "commenting"
  },
  {
   "id": "_:b1",
   "type": "Text",
   "format": "text/html",
   "http://www.w3.org/2011/content#chars": "<p>what a great design</p>"
  },
  {
   "id": "_:b2",
   "type": "SpecificResource",
   "dcterms:isPartOf": {
    "id": "https://purl.stanford.edu/sz557xs2372/iiif/manifest"
   },
   "selector": "_:b4",
   "source": "https://purl.stanford.edu/sz557xs2372/iiif/canvas/sz557xs2372_1"
  },
  {
   "id": "_:b3",
   "type": "FragmentSelector",
   "value": "xywh=1523,1408,2075,1357"
  },
  {
   "id": "_:b4",
   "type": "Choice",
   "oa:default": {
    "id": "_:b3"
   },
   "oa:item": {
    "id": "_:b5"
   }
  },
  {
   "id": "_:b5",
   "type": "SvgSelector",
   "value": "<svg xmlns='http://www.w3.org/2000/svg'></svg>"
  },
  {
   "id": "https://purl.stanford.edu/sz557xs2372/iiif/manifest",
   "type": "http://iiif.io/api/presentation/2#Manifest"
  }
 ]
}
```

Interaction Model: http://www.w3.org/ns/ldp#RDFSource;

How can “smart clients” use Linked Data?

Smart Clients are built with **from RDF** processors, for example, the JSON-LD Processor API.

A Smart Client depends on HATEOAS (Hypermedia as the Engine of Application State) interaction.

A Smart Client can deserialize “generic” RDF format responses from LDP servers and can execute stream processing functions or SPARQL queries as required for the service implementation.

Stream processing is a *very powerful* Java 8 technique when applied to RDF graphs.[8]

For example, testr0012() provides validation that the subject of any triple with rdf:type “sc:sequence” is present and is an IRI:

```java
void testr0012() {
  BlankNodeOrIRI id =
        closeableFindAny(graph.stream(null, RDF.type, SC.Sequence)).map(
              Triple::getSubject).orElseThrow(() ->
              new RuntimeException(SC.Sequence.getIRIString() + " type not found"));
  assertTrue(id instanceof IRI);
}

static Optional<? extends Triple> closeableFindAny(Stream<? extends Triple> stream) {
  try (Stream<? extends Triple> s = stream) {
     return s.findAny();
  }
}
```

“Conventional” methods using the JSON structure to do “null” type checking are more rigid and can be error prone.

Next Steps

What next?

- **Trellis LDP 0.1.0** was released on 1 November, 2017.
    - It represents a new standard for Digital Object Repositories and a significant achievement by the developer, **Aaron Coburn**.
    - It provides an solid and robust foundation for the further development of massive-scale semantic web integration applications and “smart client” interactions.
- **IIIF Presentation API 3.0** (Q1 2018)
- **JSON-LD 1.1** (Q2 2018 (est.))
- **Apache RDF Commons 0.5.0** (Q1 2018)
- **Smart Clients (????)**

Thank you very much!

References

[1] Trellis LDP: http://www.trellisldp.org/

[2] LDP Specification: https://www.w3.org/TR/ldp/

[3] JSON-LD 1.1 Specification: https://json-ld.org/spec/latest/json-ld/

[4] International Image Interoperability Framework: http://iiif.io/

[5] Apache Kafka: https://kafka.apache.org/

[6] Apache Beam: https://beam.apache.org/

[7] PANDORA: https://github.com/pan-dora

[8] Apache RDF Commons: https://github.com/apache/commons-rdf

UB Leipzig

@chjohnson39

GCDH
Göttingen

@joewett