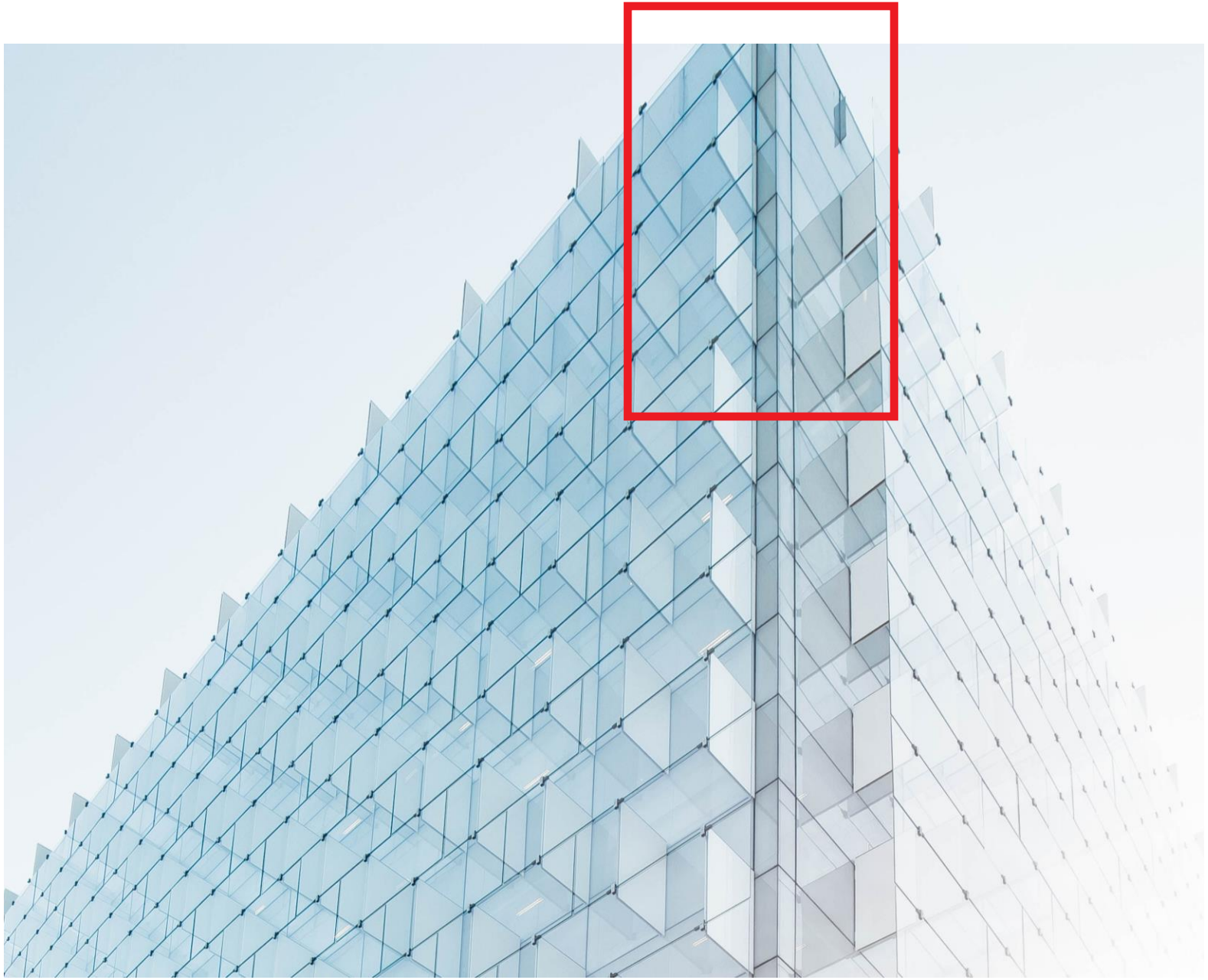
SEMAKU

VERSION CONTROL FOR LINKED DATA

PLDN-KVK Webinar



January 20, 2021

OUR SERVICES

We support our clients reaching operational excellence through a balanced service portfolio.

CUSTOM APPLICATION DEVELOPMENT

Flexible and scalable applications using (semantic) web technologies.

CONTENT & PUBLICATION MANAGEMENT

Content syndication tools and processes using best of breed technologies.

DATA SCIENCE & DATA INTEGRATION

Source systems and data hub integration using big data and linked data technologies.

PROJECT SUPPORT & MANAGEMENT

Business analyst, project & program management, Scrum and DevOps consultants.

WHAT IS VERSION CONTROL?

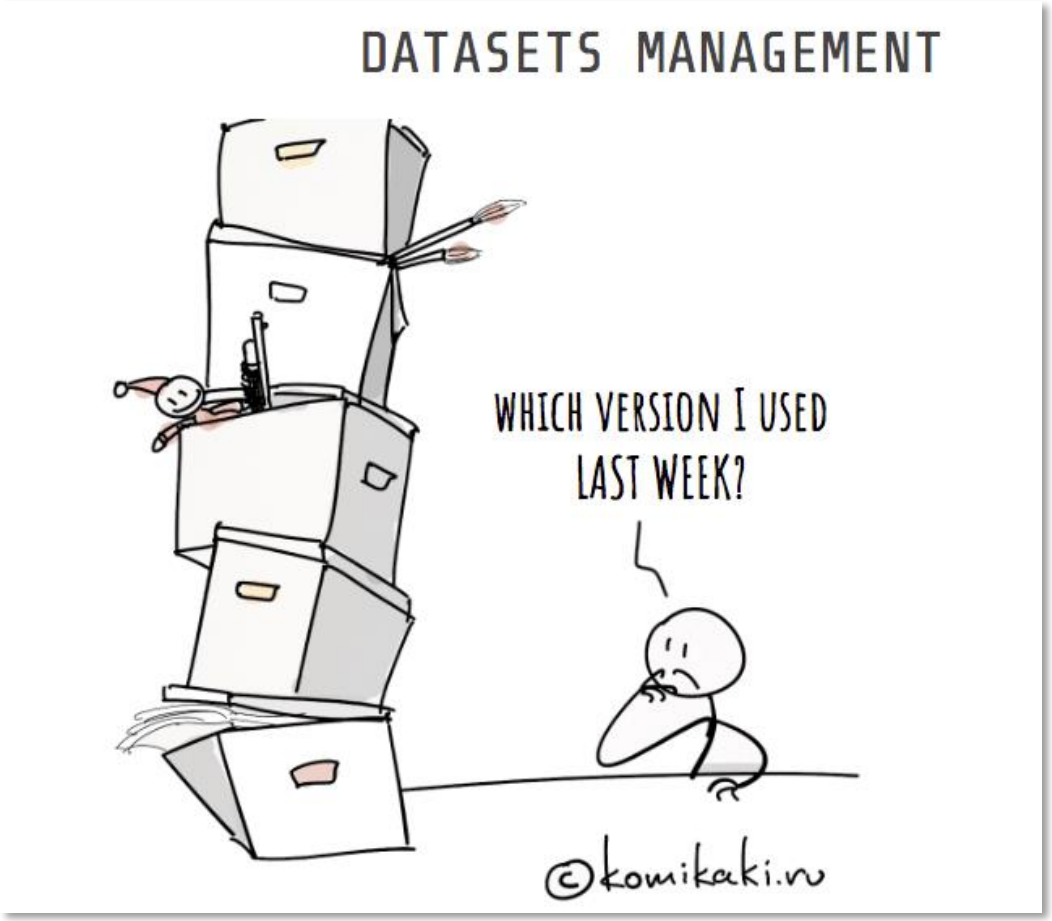
In software engineering, ***version control*** (also known as ***revision control***, ***source control***, or ***source code management***) is a class of systems responsible for managing changes to computer programs, documents, large web sites, or other collections of information. – Wikipedia

Examples include:

- *Code*: Git, Mercurial, SVN
- *Documents*: SharePoint, Alfresco
- *Web*: Drupal, Magnolia

HOW DOES IT APPLY TO (LINKED) DATA?

- Track changes
- Process control
- Compliance
- Recovery/reversion
- Analytics



RETRIEVAL FUNCTIONALITY

- Version materialization
- Single-version structured queries
- Cross-version structured queries
- Delta materialization
- Single-delta structured queries
- Cross-delta structured queries

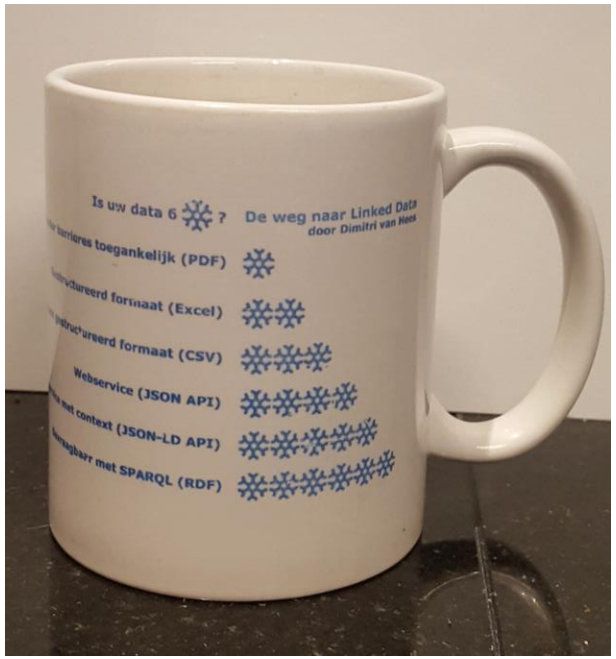
Javier D. Fernándeza, Jürgen Umbricha, Axel Polleresa, Magnus Knuth
Evaluating Query and Storage Strategies for RDF Archives
http://www.semantic-web-journal.net/system/files/swj1814.pdf

PUBLISHING VERSIONED LINKED DATA

Make historical states of entities accessible via the web as navigable hypermedia resources.

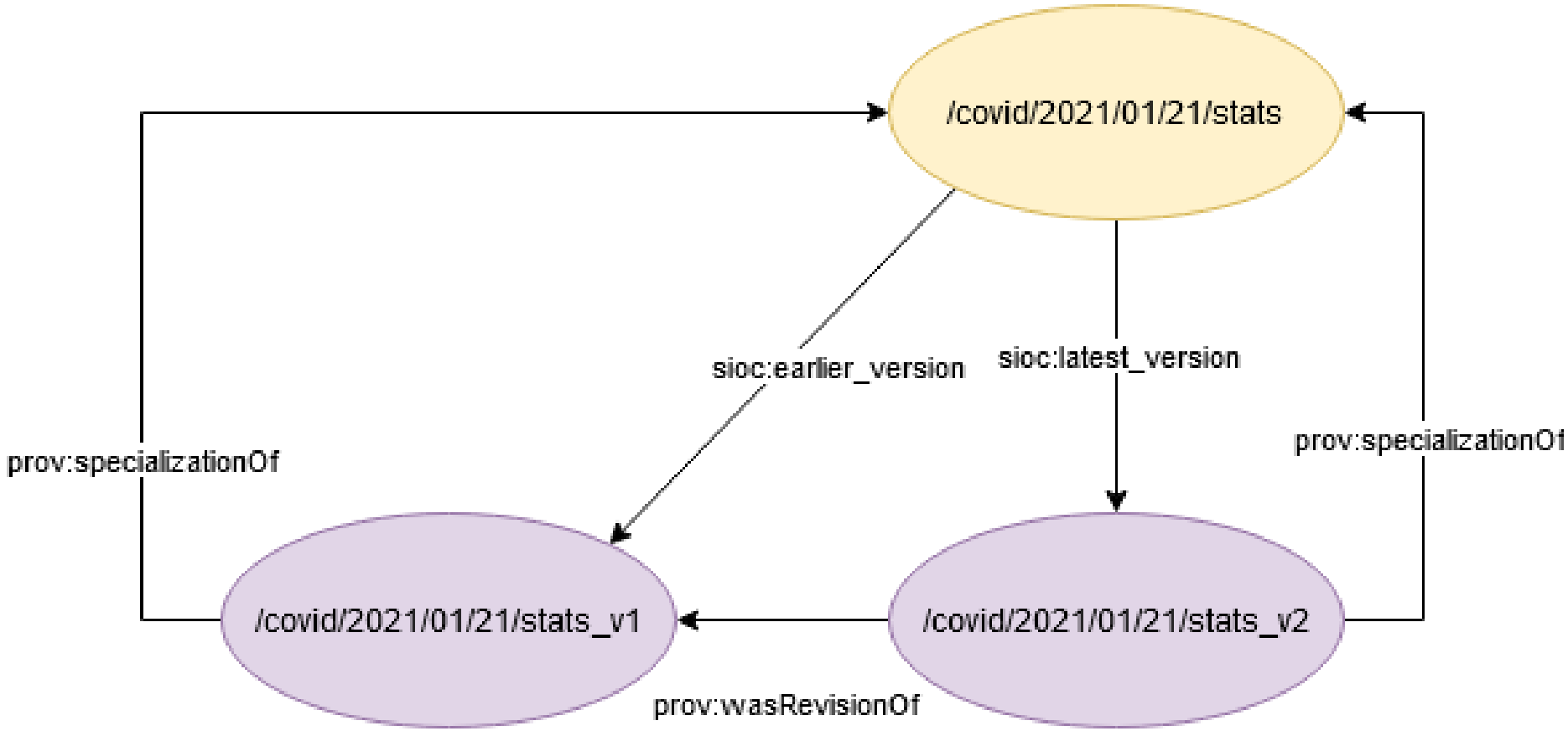
LINKED DATA RECAP

- Entities of interest should be exposed as resources on the web
- That are addressable with a stable (preferably cool) IRI
- That dereference to a representation of the resource
- Use RDF



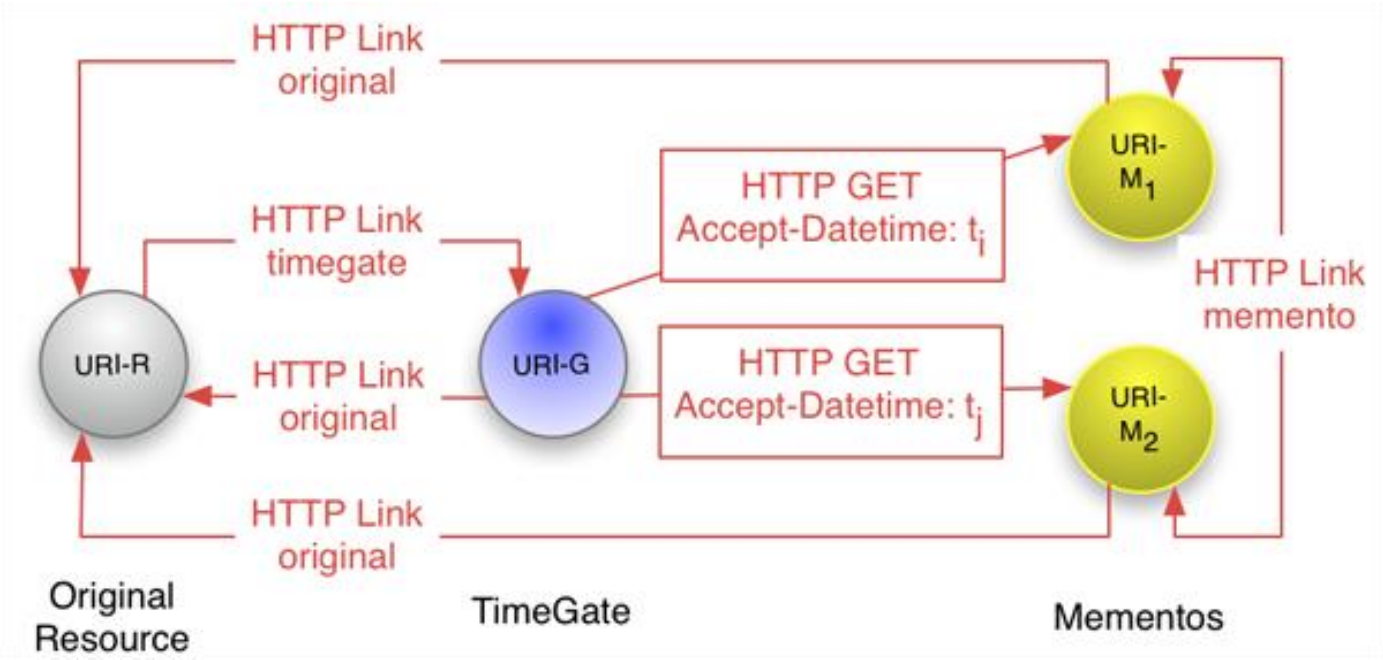
VERSIONED LINKED DATA

- Think about the entities in your dataset **and how they change over time**
  - Is history immutable or subject to change?
- **All states** of entities of interest should be exposed as resources on the web
- That are addressable with a stable (preferably cool) IRI
- That dereference to a representation the resource
- Include metadata
  - About validity period of the information
  - Link the versions to each other to allow navigation


/covid/2021/01/21/stats
sioc:earlier_version
sioc:latest_version
prov:specializationOf
prov:specializationOf
/covid/2021/01/21/stats_v1
/covid/2021/01/21/stats_v2
prov:wasRevisionOf

MEMENTO PROTOCOL

- Time-Based Access to Resource States
- Give me resource X as it was at timestamp Y

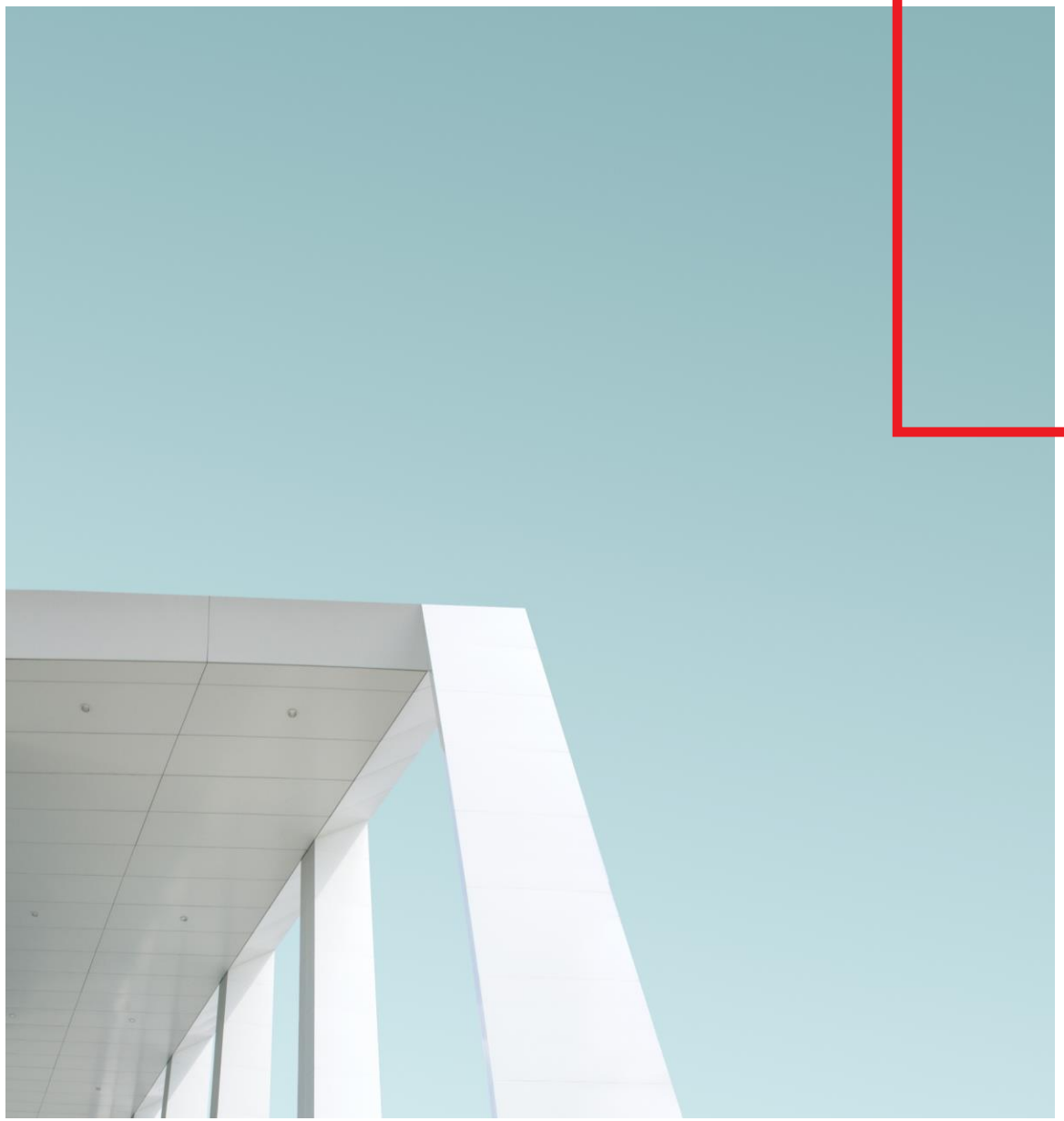


EXAMPLES FOR VERSIONED URIs

- Date/time stamped
  `http://example.com/people/2021/me-20210120`
- Hash based (possibly content addressable)
  `http://example.com/people/df9d64/me`
- Random identifier e.g. UUID
  `http://example.com/0000200b-118e-4de1-95a7-c6cb71810c8e`

OTHER IDEAS

- Publish a feed of changed resources (e.g. RSS / Atom)
- Allow consumers to subscribe to changes (e.g. WebSub)
  - Notify when state of a resource changes
- Make low-cost historic snapshots of datasets available for download/query
  - HDT archives
  - TPF endpoints



USING VERSIONED LINKED DATA

Querying with SPARQL

SYNC DATA TO RDF STORE

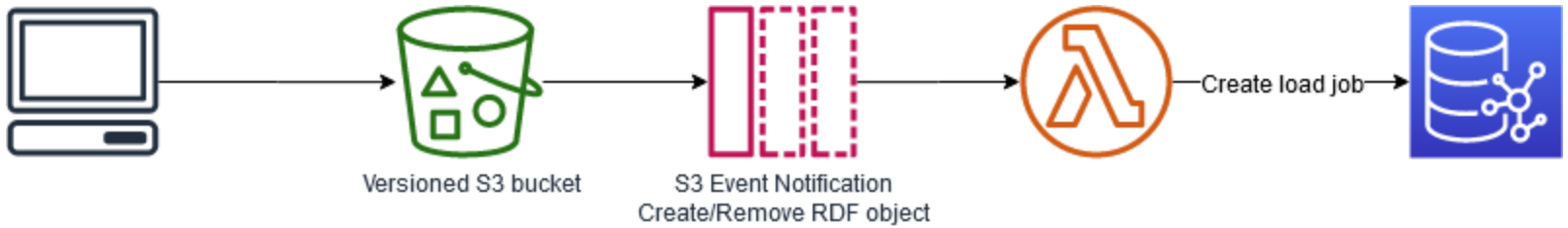
- **Use case**: RDF store is not the ‘system of record’
- Use named graph per resource pattern
- Apply changes in store using SPARQL Update or Graph Store HTTP Protocol
- Try to keep updates atomic at named graph level
- SPARQL Update
  - Use DELETE and LOAD operations in a single procedure
- SPARQL Graph Store HTTP Protocol
  - Use PUT or DELETE method

GRAPH MANAGEMENT

- Maintain snapshot of a given state of the dataset in the store
  - Reduces complexity on query side
  - Does not allows querying into the history
  - Use the union graph in case of property paths across graph boundary
- Maintain history of all historic states in the store
  - Requires mechanism to constrain the query to operate over the pertinent named graphs e.g. additional metadata statements

EXAMPLE DEPLOYMENT

Using AWS services: S3, Lambda functions, Neptune



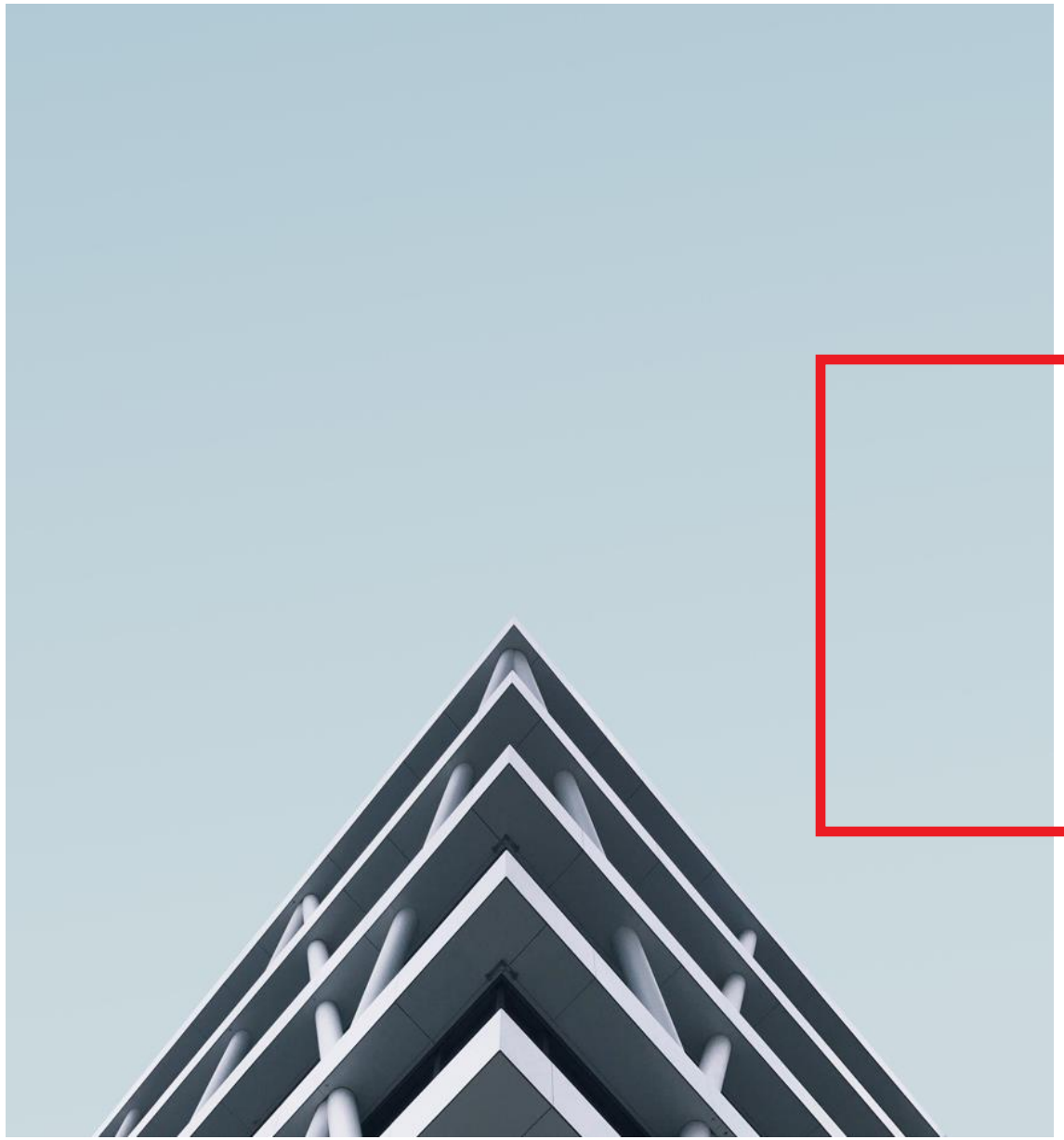
MANAGING & STORING VERSIONED LINKED DATA

EVENT SOURCED KGs

- **Use case**: RDF store is the ‘system of record’
- Capture all changes to RDF graph state as an event object
- Reconstruct state by replaying events
- Any change ‘event’ on RDF can be reduced to statements added and/or removed
- Make states and events addressable
- Allows for efficient replication of state

PATCH FORMATS FOR RDF

- Representing the change in state of the representation of a resource
  - JSON has JSON Patch [RFC 6902]
  - XML has XML Patch [RFC 5261]
  - RDF has …
- SPARQL 1.1 Graph Store HTTP Protocol suggests (informative) that SPARQL 1.1 Update can be used as a patch document
- Several vendor specific formats and implementations exist



THANK YOU!
QUESTIONS?

Visit us:

SFJ 3D Gebouw Videolab (Strijp-S)

Torenallee 20

5617 BC Eindhoven

Contact us:

info@semaku.com

SEMAKU