2019-01 Architectural Fly-in Summary

- Purpose
- Participants
- Goals
- Logistics
- Topics
    - Ingest
    - UI
    - Decoupling VIVO
- Architectural concerns and questions
- Retrospective
    - Before
    - During

Purpose

The VIVO architectural fly-in was focused on bringing an architecturally-minded team together who individually represent distinct VIVO stakeholder constituencies for the purpose of developing architectural approaches required to address the direction of the project. The primary goal of the two-day face-to-face meeting was to assess and document a plan for improving the VIVO application architecture towards enabling and realizing the technical efforts defined in the "Statement on VIVO's Product Direction for 2019".

Participants

1. Huda Khan
2. Andrew Woods
3. Brian Lowe
4. Justin Littman
5. Ralph O'Flinn
6. Benjamin Gross
7. Jim Blake
8. Richard Outten
9. Alex Viggio

Leading up to the face-to-face meeting, the team convened six conference calls over two months with the goals of establishing a common understanding of:

- the purpose of the effort as well as
- the perspectives held by each team member.

In addition to collecting requirements, assessing existing features, sharing documentation resources, and drilling into areas of exploration, each team member provided hypothetical architectural diagrams which informed the face-to-face discussions.

Goals

At the beginning of the fly-in, the team defined and agreed on the following goals:

1. Translating the Statement of VIVO's Product Direction for 2019 into an actionable architecture
2. Engaging and incorporating technical efforts from parts of the VIVO community that have emerged adjacent to the core application
3. Evolving VIVO into a more modular, cloud-ready application. This includes defining modules that can be deployed as separate services (containerized) as well as enabling the replacement of those modules with standard cloud services (AWS Neptune, Azure Cosmos DB, etc)
4. Ensuring continuity for the community in order to facilitate incremental, manageable change.

Logistics

Great effort was put towards making sure that the discussions were open and supportive, and that there was space for the voices of all team members. Ground rules were established around:

- maintaining topic-focused discussions
- time-boxing sessions
- establishing goals for each session, and
- reviewing actions/takeaways from each session

The fly-in took place over the course of two days, starting at 9am and concluding at 5:30pm, followed by team dinners.

Topics

Ingest

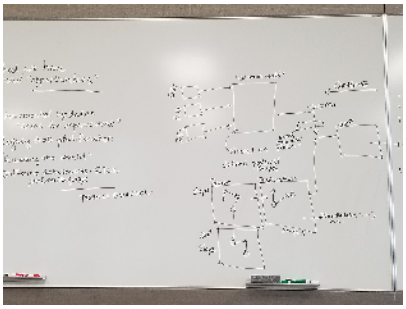
Requirements

1. Ingest must support varied and currently unknown data sources
2. Ingest must be scalable: the use of different backend triplestores or datastores must be allowed to support site-specific scaling requirements
3. Ingest must support both JSON and RDF (note: JSON content must have a known mapping to RDF model)
4. Content must be validated before being ingested. Example validation mechanisms: SHACL, ShEx, JSON Schema
5. Ingest must expect models/shapes that are expected and able to be validated (note: initial models can be derived from what is in use in Freemarker UI)
6. Ingest tooling must support two modes of operation: hands-free (automated) and curated
7. Ingest tooling must support curation of data prior to ingest: disambiguation and reconciliation of entities
8. Ingest will be entity-centric vs triple-centric. Example entities: Person, Grant, Publication, Authorship
9. Ingest tooling must not require the use of a specific programming language
10. Ingest must support realtime, incremental updates

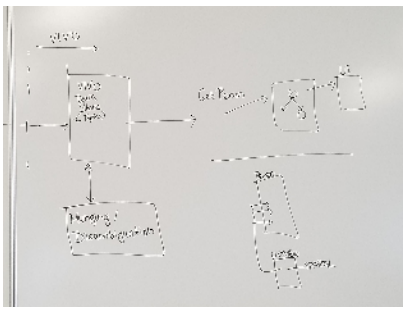
Out of scope

1. Extraction of data from data sources
2. Transformation of data from data sources

Below: DS1-3: Data sources, read in and validated based on “shapes” (or patterns or sets of triples representing entities), with entity resolution/URI creation as required, leading to a set of triples that can be read into VIVO. The arrow on the right side of the picture continues to VIVO in the next picture.



Below: The previous picture represents a view of the combine/ingest process that leads to VIVO in this picture (the munging/disambiguation box was supposed to be moved to the ingest process instead). Triples can then be requested for different entities which provide a UI. The portion on the bottom right identifies how the current VIVO system uses SPARQL queries defined in list views which together define what is expected to be displayed for specific entity types.



UI

Requirements

1. Current Freemarker UI will stay in-place for the scope of this plan (although deprecated)
2. UI must provide read access to VIVO data
3. UI must minimally be informed by the Production Evolution effort
4. UI must be based on data coming from a JSON endpoint
5. UI should render data served by a GraphQL server
6. GraphQL server should be configured with the same models used by ingest tooling (note: DocumentModifier.java may be updated to populate search index with these models)
7. UI must support accessibility
8. UI must support internationalization (i18n)
9. UI should avoid querying the triplestore when rendering

Decoupling VIVO

By decoupling VIVO, we envision a collection of independent services that interact with one another over HTTP. Each of these components provide services based on well-documented contracts/APIs to further enable the replacement of one implementation of a component with another technology. Where possible, the component contracts/APIs should be aligned with native cloud services (i.e. AWS, Azure). Finally, to ensure consistent deployment environments and to facilitate transitioning from local to cloud deployment, each of the components below should be bundled as Docker images.

Triplestore

1. Initial service abstraction is represented in the RDFService.java interface
2. Implementations to support: Fuseki, BlazeGraph, Neptune
3. Respond to SPARQL-Query
4. Ingest set of triples
5. Generate resource URIs
6. Produce list of named graphs
7. Produce serialization of single graph
8. Produce serialization of entire graph store
9. Determine if internal graph is different from a serialized graph

Search Index

1. Initial service abstractions are represented in the following interfaces: SearchIndexer.java and SearchEngine.java
2. Implementations to support: Solr, Elasticsearch, GraphQL
3. Note: The search index machinery could potentially be used to transform data for import to and use in other derivative stores

Reasoners (TBox / ABox)

1. Enable configuration to set reasoning to on-demand or to on-change (Brian L has example code)

Triple Pattern Fragments

1. Either move current implementation into its own component, or use one of the other community implementations

Asset store

The asset store is where images are currently stored. Documents could potentially be stored via the asset store as well.

1. Initial service abstraction is represented in the FileStorage.java interface

Architectural concerns and questions

1. From an architectural perspective, having a triplestore at the core of the application brings significant limitations
2. As we decouple components, we must ensure that we also decouple logic expectations between the components
3. Is the VIVO ontology undergoing a significant revision? If so, what is the nature of the impact we should expect on the VIVO application?

Retrospective

It was suggested that we hold a similar, architectural face-to-face meeting annually, with quarterly community calls to reflect on progress and pivots.

If we hold future face-to-face meetings, the following suggestions/observations should be applied before and during the meetings.

Before

1. Assign homework early in the f2f planning process for individuals to engage in deep exploration of specific topics
2. Have team create architectural diagrams early in the planning process
3. Best timing: Mid-February to Mid-March

4. Use pre-f2f calls to establish common understandings
5. Guiding principles/documents are vital, e.g. Product Direction for 2019
6. Team size should be limited ~10
7. Ideally bring in voices/participation from adjacent efforts to the VIVO problem space
8. Ideally bring in voices/participation from practitioners, those facing real-world challenges

During

1. Ensure coffee/tea/snacks are available at the venue
2. Ensure whiteboards are available at the venue
3. Ideal not having a projector/presentations
4. Ideal to be close to an airport
5. Important to have facilitation of the meeting and ground-rules
6. Ideal to have collaborative agenda-setting at the beginning of the meeting based on a set of previously discussed topics
7. Important to establish goals at the beginning of each session
8. Important to review/summarize actions/decisions at the end of each session