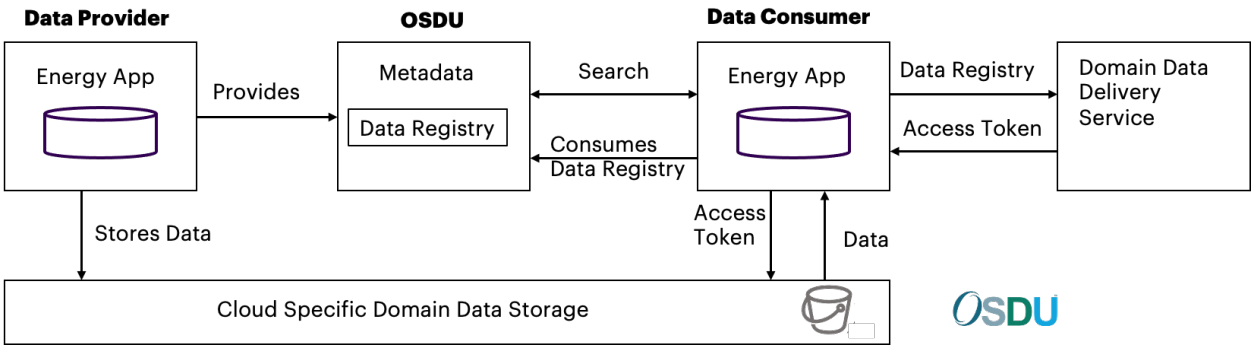
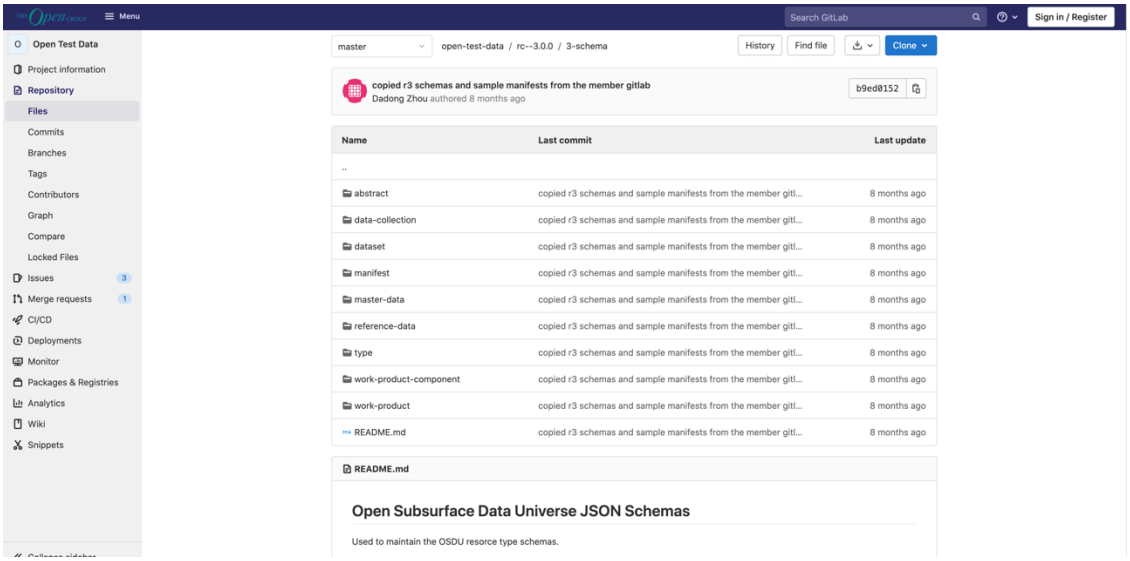
About Open Subsurface Data Universe (OSDU):
The OSDU platform is a set of energy related services and workflows orchestrated by an underlying cloud platform and deployed into the cloud environment of choice by infrastructure templates and deployment strategies provided by the Cloud Service Provider (e.g., AWS or Azure). The OSDU also provides standard schemas and data types for upstream data with the intention of extending to other, newer energy data types.



Building an Ontology for Subsurface Energy Data:
Data in the OSDU platform must follow a predefined JSON standard that exists in the OSDU schema files. These schema files explain all the elements of different group types in OSDU such as, Master Data, Reference Data, File, Work Product Component, and Work Product.

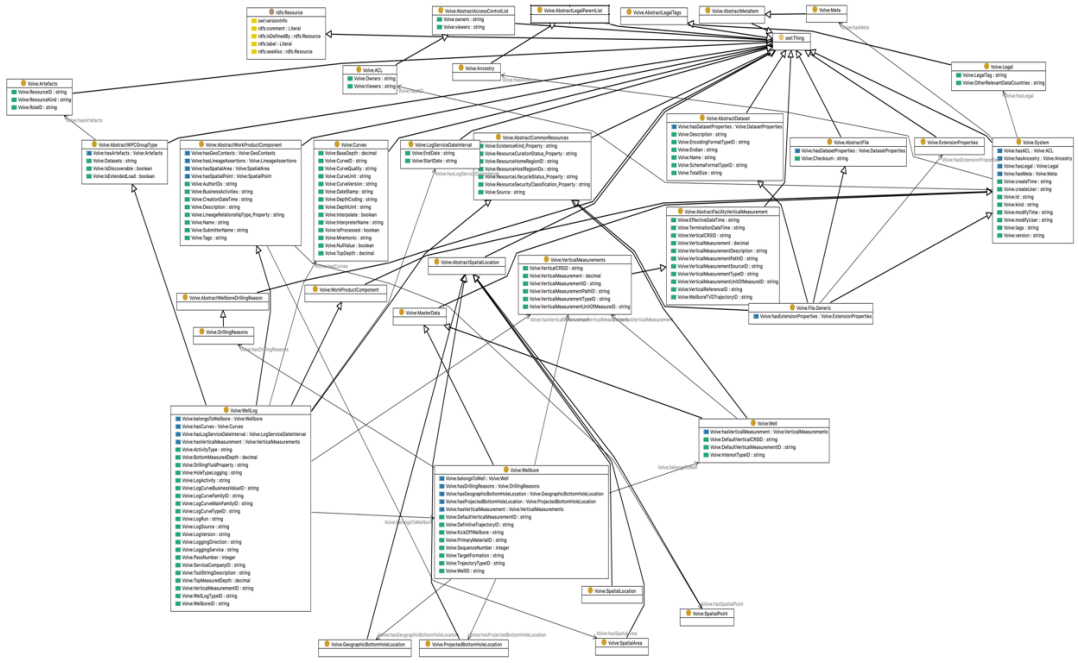


In order to make an OWL-based ontology for the subsurface energy data which is integratable in knowledge graphs and semantical graph databases, we have to follow below rules for translating the OSDU schema files.

1. We start modeling our ontology based on defining classes for each of the Group Types and their JSON schema files. For example, we define a class osdu:AbstractCommonResources for AbstractCommonResources.1.0.0.json file.

2. The majority of OSDU schema files contain similar properties such as, id, kind, version, meta, legal, and tags. Therefore, we make a System class and add all these properties as System's datatype properties. Then each class corresponding to a schema file that has this set of properties, inherit from the System class. The only exception is "acl" which we define it as a subclass of AbstractAccessControlList and we connect it to System through a "hasACL" object property.
3. If the schema file has "required" attribute, the entities listed as required data need to have an "owl:minCardinality" with value 1. For example, kind data type property and ACL class must take a minimum cardinality of 1 because they are among the required entities.
4. Any class, object property, and datatype property can also include an "rdfs:comment" with the value in the associated "description" field of the schema entity.
5. Every schema file has a "data" key which is used to describe the properties specific to its class. We do not need to create anything for "data" itself in our ontology, but we need to take care of its nested entities. The "allOf" nested entity of "data" may contain an entity "$ref". The referenced schema file as the value of "$ref", should be the super class of the class associated with this schema file. The following rules apply to the attributes of "properties" nested entity of "data".
6. If an attribute has a "type" of string or integer, a datatype property should be created in the ontology and named as the attribute. The datatype property has the class corresponding to the JSON schema file as its domain and the value of "type" as its range.
7. The created datatype property can also include "xsd:pattern" with the value associated with the "pattern" entity of the attribute.
8. When a datatype property name is already occupied by a class in our ontology, we need to add a "_property" at the end of the datatype property.
9. If the attribute has a "type" of array but does not have a "$ref" nested entity under its "items", then we only need to create a datatype property which has a range value similar as the value of the sub-"type" entity. The domain is also the class associated with the schema file.
10. If the attribute has a "type" of array and has a "$ref" nested entity, we need to define a new class for the attribute which is the subclass of the class referenced in the value of "$ref" entity. In addition, an object property needs to be created. The name of the object property starts with a "has" and continues with the attribute's name. The domain of the object property is the class associated with the schema file and the range is class associated with the attribute.
11. In case of existence of a "x-osdu-relationship" entity associated with the attribute which is now a datatype property in our ontology, we need to check the value of its nested entities "GroupType" and "EntityType". There must be a class in our ontology with the name of the value of "EntityType" (e.g. WellboreTrajectoryType) which is the subclass of the class with the name of the value of "GroupType" (e.g. Reference Data). We need to add an "owl:allValuesFrom" for our datatype property with the subclass's name as its value.

Below figure is a snapshot of part of OSDU ontology:



General Classes in OSDU Ontology:

- Abstract
  Defining the parent properties of each class
  Alias names, owner, viewer, facility info of a well, spatial locations of a well or organization
- System
  Metadata about each class
  Create/modify time/user, version, ACL, legal, and tags.
- Master Data
  High level information about subsurface energy concepts
  Drilling reasons, trajectory types, hole locations, vertical measurements
- Reference Data
  List of values against which data is validated
  Unit of measures, unit of quantity
- Work Product Component
  The data that is the result of a business activity and contains links to files including the data
  Curve quality, top depth, base depth, log version, service company, drilling fluid property, log source, log run, business activity.
- Dataset
  The path to the actual data file/s and the file/s metadata