On TAP:

Module 1: Data Ingest

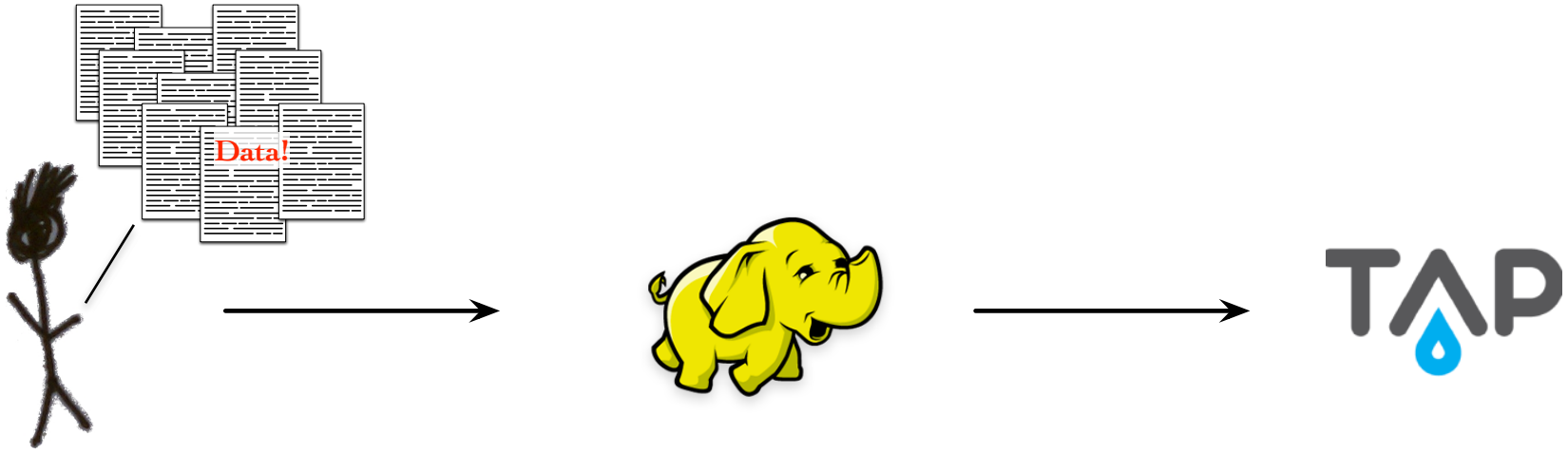
Kyle H. Ambert, PhD

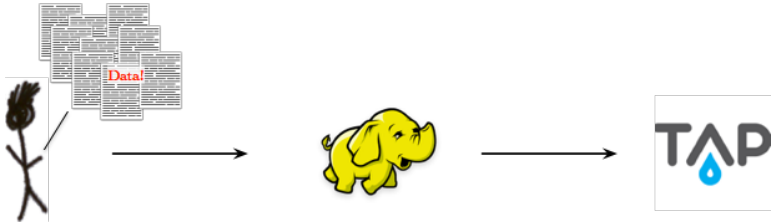
Intel Big Data Solutions, Datacenter Group

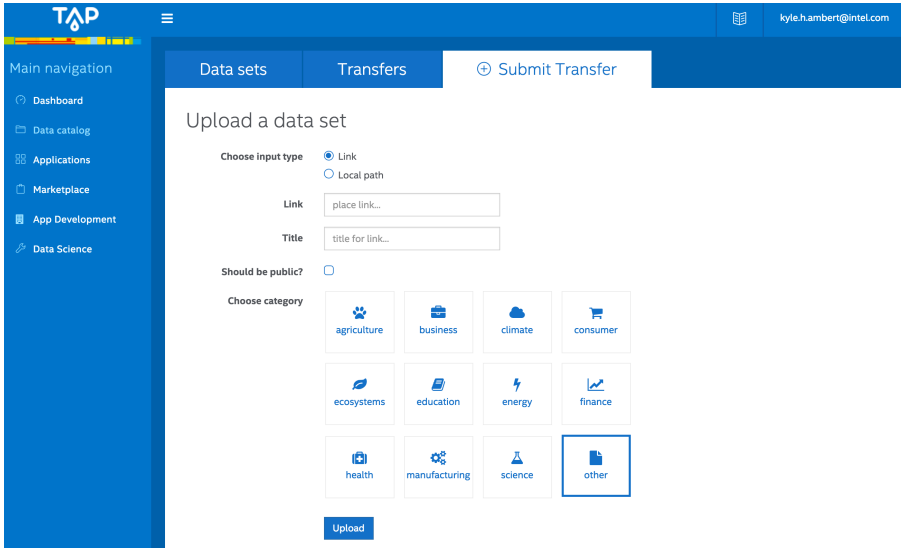
Ingest:

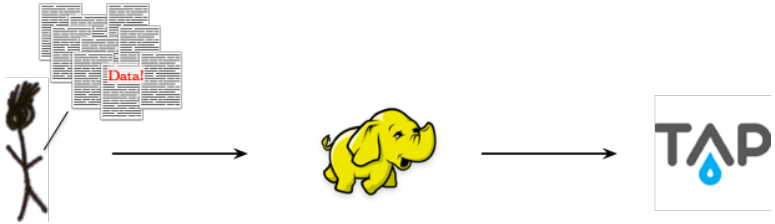
- Getting data into TAP
- Getting data into atk
- Moving from raw to the beginning of structure
- File types:
  - csv
  - xml
  - json

Data on TAP

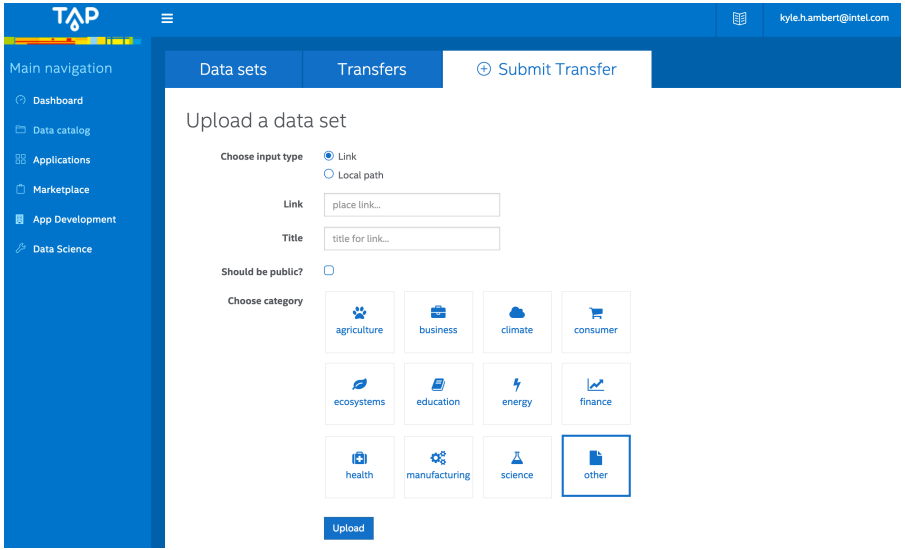


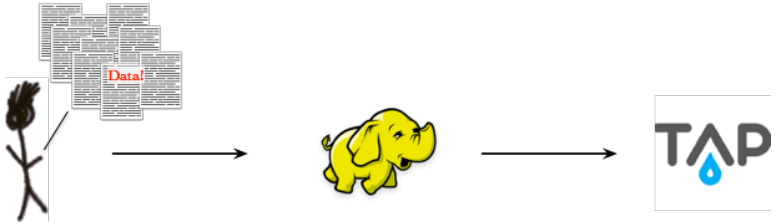

Data!
TAP


TAP
kyle.h.ambert@intel.com
Main navigation
Dashboard
Data catalog
Applications
Marketplace
App Development
Data Science
Data sets
Transfers
Submit Transfer
Upload a data set
Choose input type
Link
Local path
Link
place link...
Title
title for link...
Should be public?
Choose category
agriculture
business
climate
consumer
ecosystems
education
energy
finance
health
manufacturing
science
other
Upload



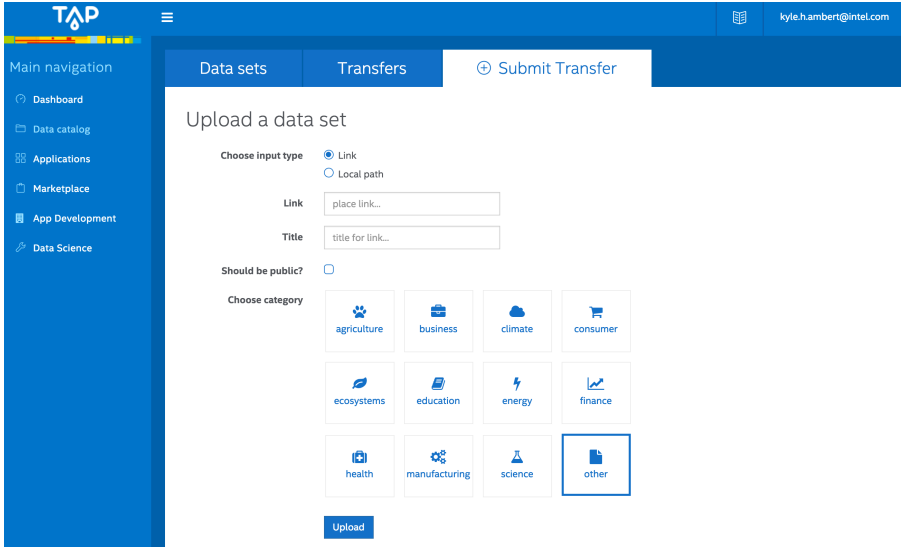
Getting data into the Toolkit

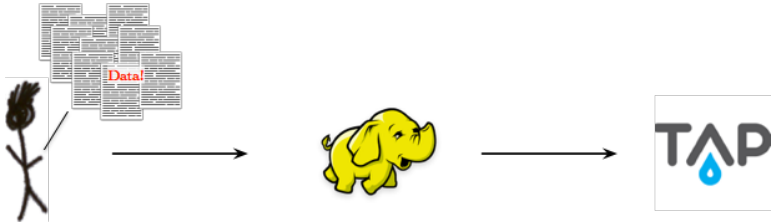




Getting data into the Toolkit

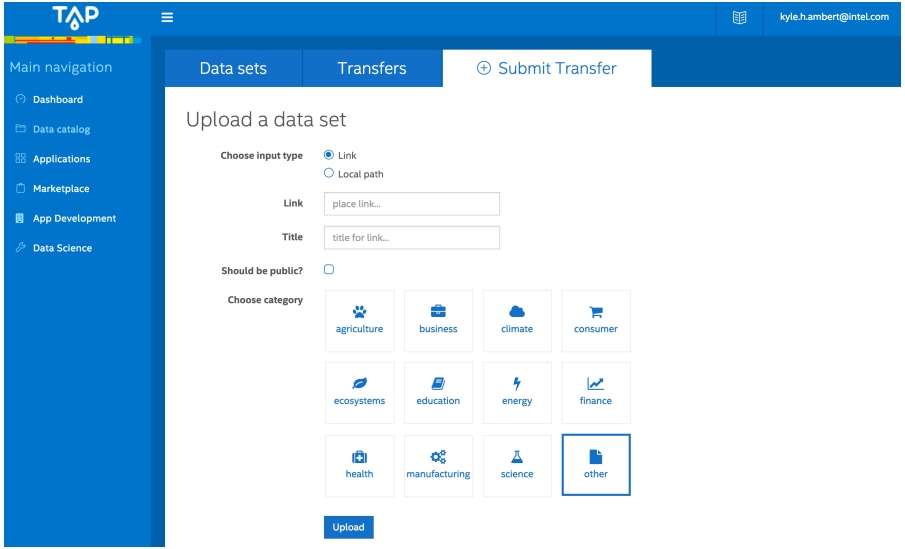
- Use the console front-end

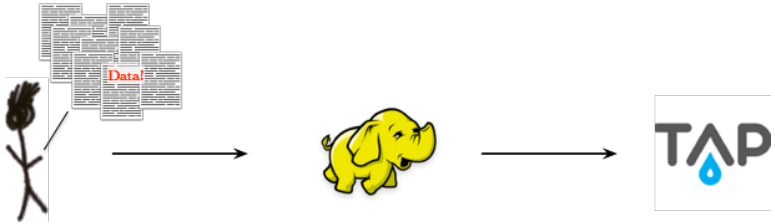




Getting data into the Toolkit

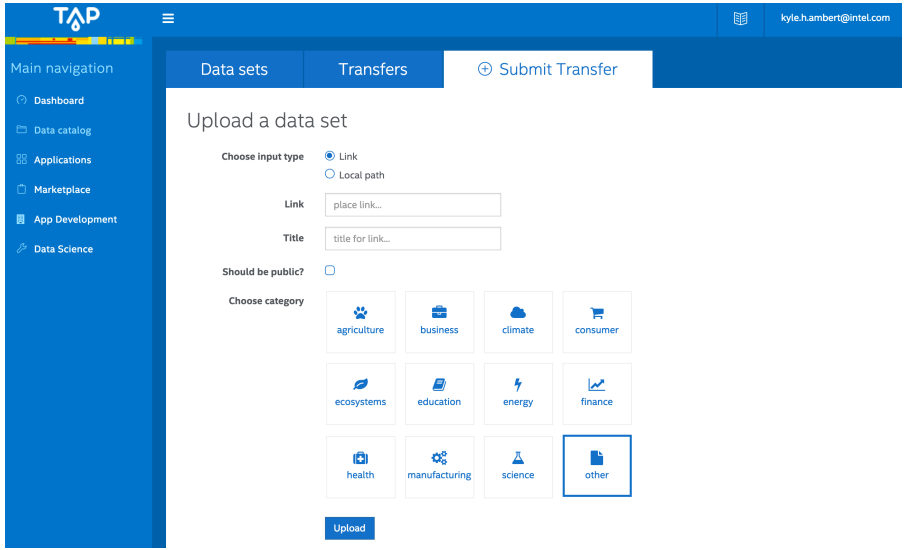
- Use the console front-end
  - Data Catalog page

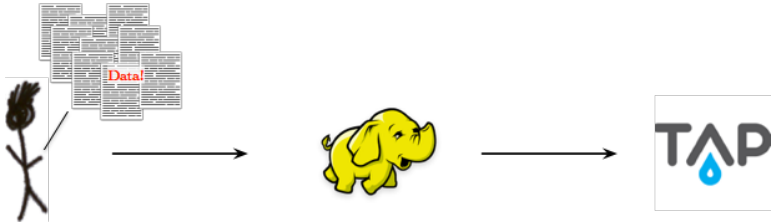




Getting data into the Toolkit

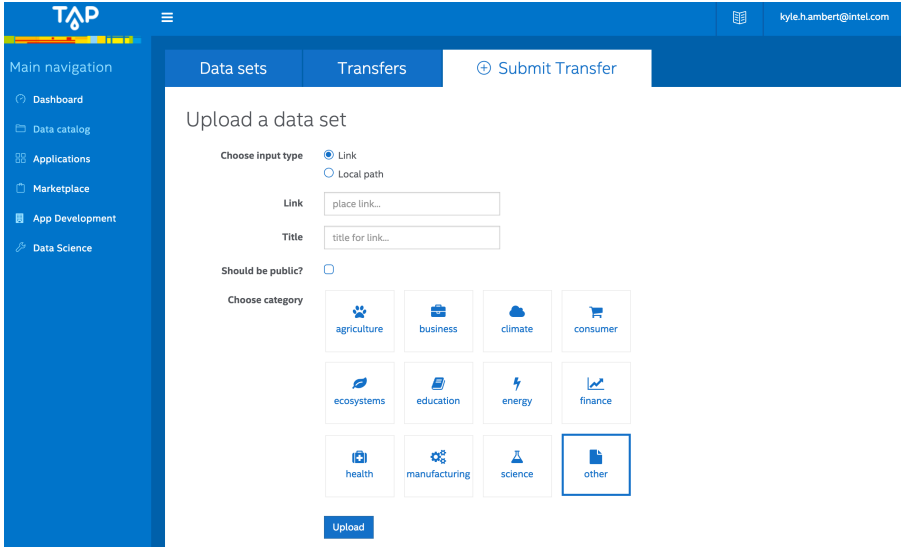
- Use the console front-end
    - Data Catalog page
- Submit transfer from local path (or remote url)

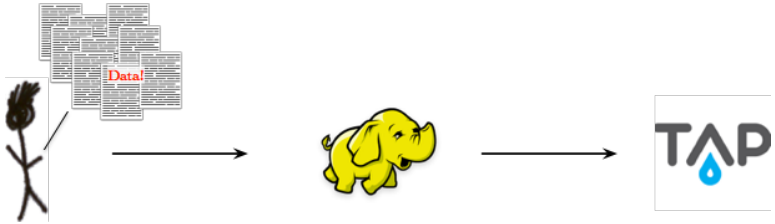




Getting data into the Toolkit

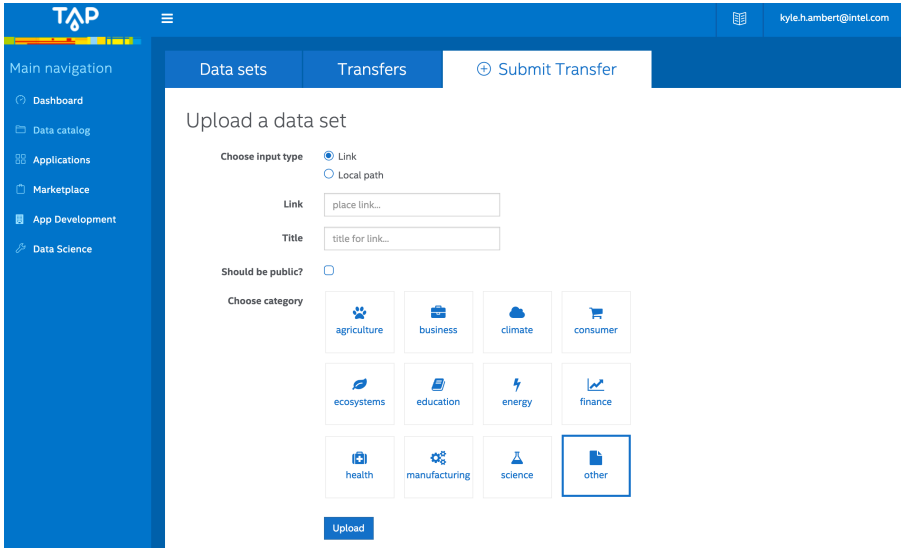
- Use the console front-end
    - Data Catalog page
- Submit transfer from local path (or remote url)
- Upload!





Getting data into the Toolkit

- Use the console front-end
    - Data Catalog page
- Submit transfer from local path (or remote url)
- Upload!
- Data can be previewed from the Data Sets tab



Input Data Manually

[**1**] Define the **Data**

- Iterative error-prone drudgery
- One-off, ad hoc models in isolation

[**2**] Define the **Schema**

- A list of tuples: `('column_name', data_type)`

[**3**] Use the `ia.UploadRows` function

Input Data Manually

[**1**] Define the **Data**

- Iterative error-prone drudgery
- One-off, ad hoc models in isolation

[**2**] Define the **Schema**

- A list of tuples: `('column_name', data_type)`

[**3**] Use the `ia.UploadRows` function

Input Data Manually

[**1**] Define the **Data**

- Iterative error-prone drudgery
- One-off, ad hoc models in isolation

[**2**] Define the **Schema**

- A list of tuples: `('column_name', data_type)`

[**3**] Use the `ia.UploadRows` function

Example:

```
animals = ia.Frame(ia.UploadRows([['puppy', 'Harrison'], ['cat', 'wahlberg']], [('animal', str), ('name', str)]))
```

Input Data Manually

[**1**] Define the **Data**

- Iterative error-prone drudgery
- One-off, ad hoc models in isolation

[**2**] Define the **Schema**

- A list of tuples: `('column_name', data_type)`

[**3**] Use the `ia.UploadRows` function

Example:

```
animals = ia.Frame(ia.UploadRows([['puppy', 'Harrison'], ['cat', 'wahlberg']], [('animal', str), ('name', str)]))
```

All columns must be the same length! All data types must be ***serializable***!

Character-separated Value Files (*.csv*)

[**1**] Define the schema

- Iterative error-prone drudgery
- One-off, ad hoc models in isolation

[**2**] ta.CsvFile

- Single-threaded, single-node processing

```
__init__(self, file_name, schema, delimiter=',', skip_header_lines=0)
```

Example:

```
import trustedanalytics as ta
ta.CsvFile("raw_data.csv", schema=[("col1", ta.int32), ("col2", ta.float32)])
```

Character-separated Value Files (*.csv*)

[**1**] Define the schema

- Iterative error-prone drudgery
- One-off, ad hoc models in isolation

[**2**] ta.CsvFile

- Single-threaded, single-node processing

```
__init__(self, file_name, schema, delimiter=',', skip_header_lines=0)
```

Example:

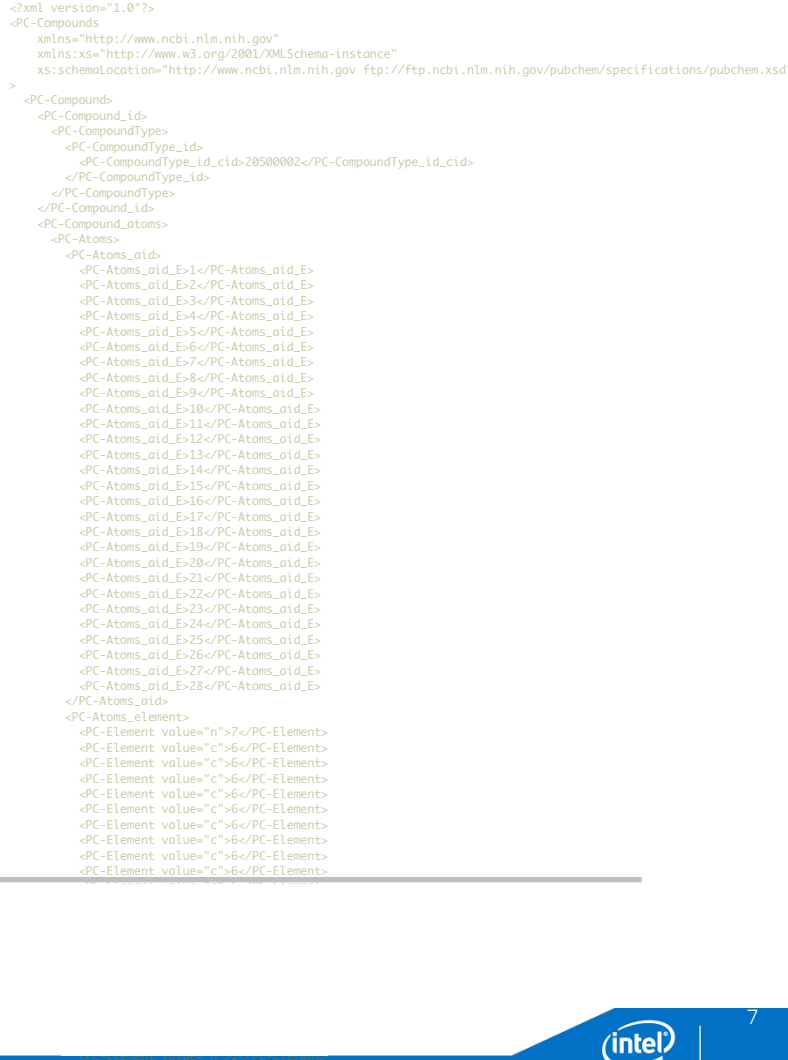
```
import trustedanalytics as ta
ta.CsvFile("raw_data.csv", schema=[("col1", ta.int32), ("col2", ta.float32)])
```

Be sure you have the correct delimiter!

XML Files

Parallel Ingest:

- Just point the toolkit to a directory containing multiple *xml* files to process them all at once.



XML Files

Parallel Ingest:

- Just point the toolkit to a directory containing multiple *xml* files to process them all at once.

Identify the tag denoting separation of entities:

```
<?xml version="1.0"?>
<PC-Compounds
    xmlns="http://www.ncbi.nlm.nih.gov"
    xmlns:xs="http://www.w3.org/2001/XMLSchema-instance"
    xs:schemaLocation="http://www.ncbi.nlm.nih.gov ftp://ftp.ncbi.nlm.nih.gov/pubchem/specifications/pubchem.xsd"
>
  <PC-Compound>
    <PC-Compound_id>
      <PC-CompoundType>
        <PC-CompoundType_id>
          <PC-CompoundType_id_cid>20500002</PC-CompoundType_id_cid>
        </PC-CompoundType_id>
      </PC-CompoundType>
    </PC-Compound_id>
    <PC-Compound_atoms>
      <PC-Atoms>
        <PC-Atoms_aid>
          <PC-Atoms_aid_E>1</PC-Atoms_aid_E>
          <PC-Atoms_aid_E>2</PC-Atoms_aid_E>
          <PC-Atoms_aid_E>3</PC-Atoms_aid_E>
          <PC-Atoms_aid_E>4</PC-Atoms_aid_E>
          <PC-Atoms_aid_E>5</PC-Atoms_aid_E>
          <PC-Atoms_aid_E>6</PC-Atoms_aid_E>
          <PC-Atoms_aid_E>7</PC-Atoms_aid_E>
          <PC-Atoms_aid_E>8</PC-Atoms_aid_E>
          <PC-Atoms_aid_E>9</PC-Atoms_aid_E>
          <PC-Atoms_aid_E>10</PC-Atoms_aid_E>
          <PC-Atoms_aid_E>11</PC-Atoms_aid_E>
          <PC-Atoms_aid_E>12</PC-Atoms_aid_E>
          <PC-Atoms_aid_E>13</PC-Atoms_aid_E>
          <PC-Atoms_aid_E>14</PC-Atoms_aid_E>
          <PC-Atoms_aid_E>15</PC-Atoms_aid_E>
          <PC-Atoms_aid_E>16</PC-Atoms_aid_E>
          <PC-Atoms_aid_E>17</PC-Atoms_aid_E>
          <PC-Atoms_aid_E>18</PC-Atoms_aid_E>
          <PC-Atoms_aid_E>19</PC-Atoms_aid_E>
          <PC-Atoms_aid_E>20</PC-Atoms_aid_E>
          <PC-Atoms_aid_E>21</PC-Atoms_aid_E>
          <PC-Atoms_aid_E>22</PC-Atoms_aid_E>
          <PC-Atoms_aid_E>23</PC-Atoms_aid_E>
          <PC-Atoms_aid_E>24</PC-Atoms_aid_E>
          <PC-Atoms_aid_E>25</PC-Atoms_aid_E>
          <PC-Atoms_aid_E>26</PC-Atoms_aid_E>
          <PC-Atoms_aid_E>27</PC-Atoms_aid_E>
          <PC-Atoms_aid_E>28</PC-Atoms_aid_E>
        </PC-Atoms_aid>
        <PC-Atoms_element>
          <PC-Element value="n">7</PC-Element>
          <PC-Element value="c">6</PC-Element>
          <PC-Element value="c">6</PC-Element>
          <PC-Element value="c">6</PC-Element>
          <PC-Element value="c">6</PC-Element>
          <PC-Element value="c">6</PC-Element>
          <PC-Element value="c">6</PC-Element>
          <PC-Element value="c">6</PC-Element>
          <PC-Element value="c">6</PC-Element>
          <PC-Element value="c">6</PC-Element>
          <PC-Element value="c">6</PC-Element>
```

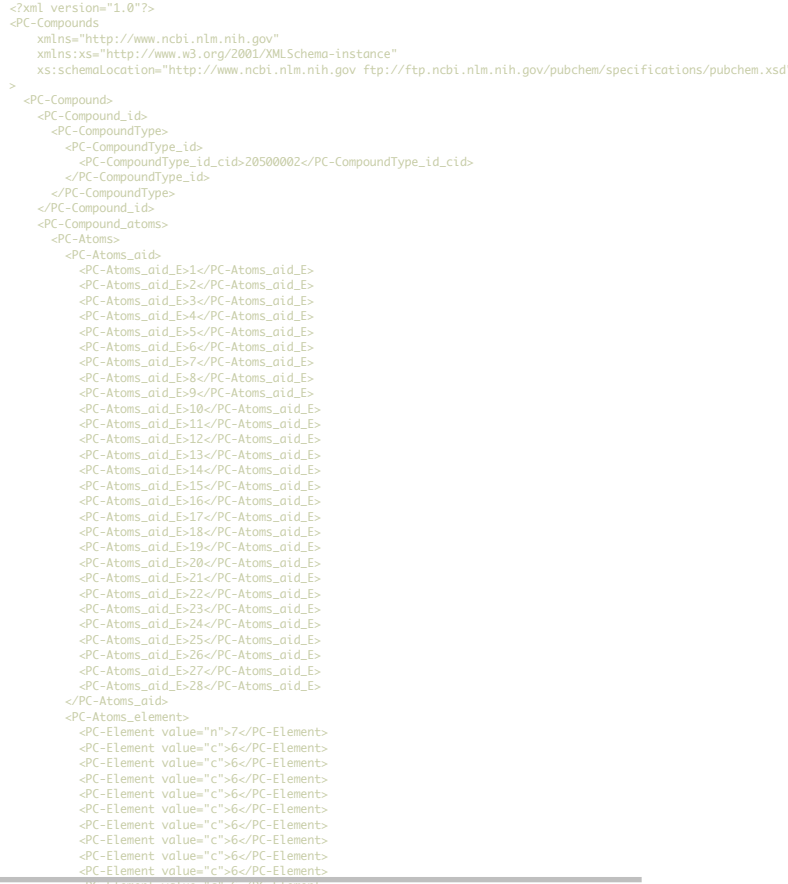
XML Files

Parallel Ingest:

- Just point the toolkit to a directory containing multiple *xml* files to process them all at once.

Identify the tag denoting separation of entities:

```
<note>
   <to>Self</to>
   <from>Self</from>
   <heading>Eagles Dig</heading>
   <body>Chip Kelly was better at UO</body>
</note>
```



XML Files

Parallel Ingest:

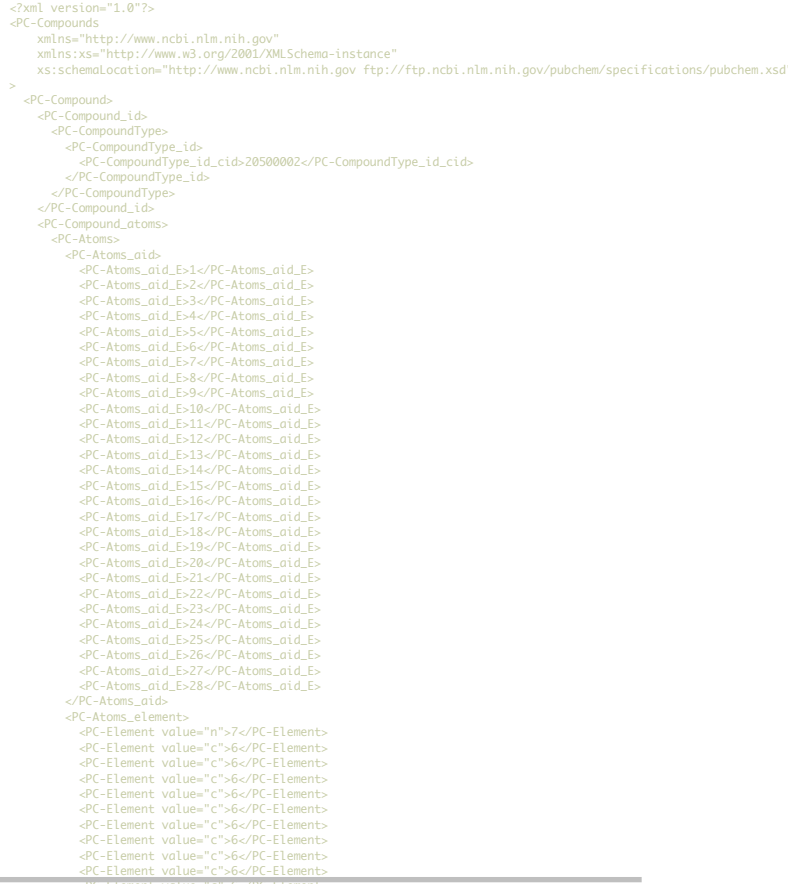
- Just point the toolkit to a directory containing multiple *xml* files to process them all at once.

Identify the tag denoting separation of entities:

```
<note>
   <to>Self</to>
   <from>Self</from>
   <heading>Eagles Dig</heading>
   <body>Chip Kelly was better at UO</body>
</note>
```

Example:

```
xml = ia.XmlFile(file_name='path', tag_name='tag')
frame = ia.Frame(source=xml, name='name')
```



XML Files

Parallel Ingest:

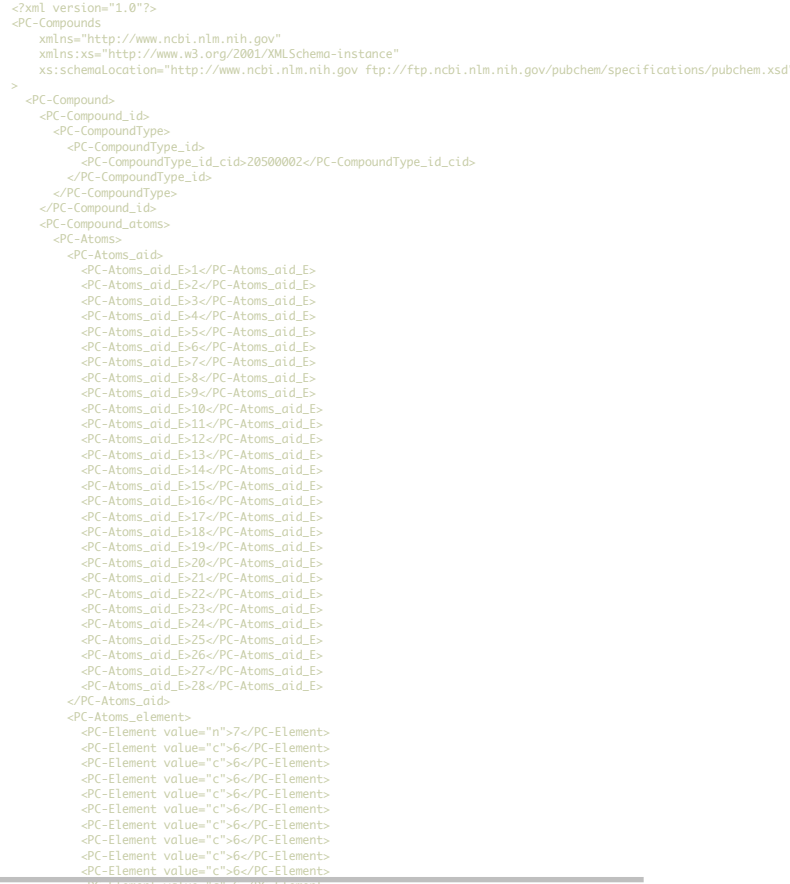
- Just point the toolkit to a directory containing multiple *xml* files to process them all at once.

Identify the tag denoting separation of entities:

```
<note>
   <to>Self</to>
   <from>Self</from>
   <heading>Eagles Dig</heading>
   <body>Chip Kelly was better at UO</body>
</note>
```

Example:

```
xml = ia.XmlFile(file_name='path', tag_name='tag')
frame = ia.Frame(source=xml, name='name')
```



Schema-naïve processing is currently a WIP. It will make everyone's life easier!

XML Fil

¡Pro Tip!

Pa gest:

Just point
multiple *x*

Identify the tag

```
<note>
   <to>Self</
   <from>Self
   <heading>E
   <body>Chip
</note>
```

Wrapping this sort of processing code in a function will be useful when working with complex xml files!

```
def parse_xml_to_frame(path, tag, name):
    """
    Helper function to convert an xml file on the hdfs into a data frame...
    """
    xml = ia.XmlFile(path, tag)

    # Check that the frame doesn't already exist. Drop it, if it does...
    if name in ia.get_frame_names():
        sys.stderr.write("Dropping existing frame named {NAME}...\n".format(NAME=name))
        ia.drop_frames(name)
    frame = ia.Frame(xml, name=name)
    return frame
```

Example:

```
xml = ia.XmlFile(file_name='path', tag_name='tag')
frame = ia.Frame(source=xml, name='name')
```


```
<?xml version="1.0"?>
<PC-Compounds
    xmlns="http://www.ncbi.nlm.nih.gov"
    xmlns:xs="http://www.w3.org/2001/XMLSchema-instance"
    xs:schemaLocation="http://www.ncbi.nlm.nih.gov ftp://ftp.ncbi.nlm.nih.gov/pubchem/specifications/pubchem.xsd"
>
  <PC-Compound>
    <PC-Compound_id>
      <PC-CompoundType>
        <PC-CompoundType_id>
          <PC-CompoundType_id_cid>20500002</PC-CompoundType_id_cid>
        </PC-CompoundType_id>
      </PC-CompoundType>
    </PC-Compound_id>
    <PC-Compound_atoms>
      <PC-Atoms>
        <PC-Atoms_aid>
          <PC-Atoms_aid_E>1</PC-Atoms_aid_E>
          <PC-Atoms_aid_E>2</PC-Atoms_aid_E>
          <PC-Atoms_aid_E>3</PC-Atoms_aid_E>
          <PC-Atoms_aid_E>4</PC-Atoms_aid_E>
          <PC-Atoms_aid_E>5</PC-Atoms_aid_E>
          <PC-Atoms_aid_E>6</PC-Atoms_aid_E>
          <PC-Atoms_aid_E>7</PC-Atoms_aid_E>
          <PC-Atoms_aid_E>8</PC-Atoms_aid_E>
          <PC-Atoms_aid_E>9</PC-Atoms_aid_E>
          <PC-Atoms_aid_E>10</PC-Atoms_aid_E>
          <PC-Atoms_aid_E>11</PC-Atoms_aid_E>
          <PC-Atoms_aid_E>12</PC-Atoms_aid_E>
          <PC-Atoms_aid_E>13</PC-Atoms_aid_E>
          <PC-Atoms_aid_E>14</PC-Atoms_aid_E>
          <PC-Atoms_aid_E>15</PC-Atoms_aid_E>
          <PC-Atoms_aid_E>16</PC-Atoms_aid_E>
          <PC-Atoms_aid_E>17</PC-Atoms_aid_E>
          <PC-Atoms_aid_E>18</PC-Atoms_aid_E>
          <PC-Atoms_aid_E>19</PC-Atoms_aid_E>
          <PC-Atoms_aid_E>20</PC-Atoms_aid_E>
          <PC-Atoms_aid_E>21</PC-Atoms_aid_E>
          <PC-Atoms_aid_E>22</PC-Atoms_aid_E>
          <PC-Atoms_aid_E>23</PC-Atoms_aid_E>
          <PC-Atoms_aid_E>24</PC-Atoms_aid_E>
          <PC-Atoms_aid_E>25</PC-Atoms_aid_E>
          <PC-Atoms_aid_E>26</PC-Atoms_aid_E>
          <PC-Atoms_aid_E>27</PC-Atoms_aid_E>
          <PC-Atoms_aid_E>28</PC-Atoms_aid_E>
        </PC-Atoms_aid>
        <PC-Atoms_element>
          <PC-Element value="n">7</PC-Element>
          <PC-Element value="c">6</PC-Element>
          <PC-Element value="c">6</PC-Element>
          <PC-Element value="c">6</PC-Element>
          <PC-Element value="c">6</PC-Element>
          <PC-Element value="c">6</PC-Element>
          <PC-Element value="c">6</PC-Element>
          <PC-Element value="c">6</PC-Element>
          <PC-Element value="c">6</PC-Element>
          <PC-Element value="c">6</PC-Element>
          <PC-Element value="c">6</PC-Element>
```


Schema-naïve processing is currently a WIP. It will make everyone's life easier!

JSON Files

Parallel Ingest

- Just point the toolkit to a directory containing multiple *json* files to process them all at once.
- Creating a frame from *json* will result in a frame with a single column: '`data_lines`'

ia.JsonFile:

```
__init__(self, file_name)
```

Example:

```
json_file = ta.JsonFile("data/raw_data.json")
my_frame = ta.Frame(json_file)
```


```
{u'ADMITTING_CCS_LVL_1_LABELs': [u'Diseases of the respiratory system'],
 u'ADMITTING_CCS_LVL_2_LABELs': [u'Other lower respiratory disease [133.]'],
 u'ADMITTING_CCS_LVL_3_LABELs': [u'Other and unspecified lower respiratory disease'],
 u'ADMITTING_CCS_LVL_4_LABELs': [u' '],
 u'ADMITTING_DIAG_CCS_LVL_1_LABELs': [u'nan',
                                      u'Diseases of the respiratory system'],
 u'ADMITTING_DIAG_CCS_LVL_2_LABELs': [u'nan',
                                      u'Other lower respiratory disease [133.]'],
 u'ADMITTING_DIAG_CCS_LVL_3_LABELs': [u'nan',
                                      u'Other and unspecified lower respiratory disease'],
 u'ADMITTING_DIAG_CCS_LVL_4_LABELs': [u'nan', u' '],
 u'ADMITTING_DIAG_CODEs': [u'786.05'],
 u'ADMITTING_DIAG_DESCs': [u'Shortness of breath'],
 u'ADMIT_REASONs': [u'COPD EXACERBATION', u'SICK0985'],
 u'ADMIT_SERVICE_CODE': u'GER',
 u'ADMIT_SERVICE_DESCRIPTION': u'GERIATRIC MEDICINE',
 u'ADMIT_SERVICE_GROUP': u'MEDICINE',
 u'ADMIT_SOURCE': u'Emergency Room',
 u'ADMIT_SOURCE_MEDVIEW': u'EMERGENCY OP UNIT',
 u'ADMIT_TYPE': u'Emergency',
 u'ADM_DATE': {u'$date': 1557447300000},
 u'AGE': 77,
 u'ARITHMETIC_LOS': 4.2,
 u'CCS_LVL_1_LABELs': [u'Diseases of the respiratory system',
                       u'Diseases of the circulatory system',
                       u'Endocrine; nutritional; and metabolic diseases and immunity disorders'
                       u'Diseases of the blood and blood-forming organs'],
 u'CCS_LVL_2_LABELs': [u'Chronic obstructive pulmonary disease and bronchiectasis [127.]',
                       u'Diseases of the heart',
                       u'Diabetes mellitus without complication [49.]',
                       u'Gout and other crystal arthropathies [54.]',
                       u'Anemia',
                       u'Other lower respiratory disease [133.]'],
 u'CCS_LVL_3_LABELs': [u'Obstructive chronic bronchitis',
                       u'Congestive heart failure; nonhypertensive [108.]',
                       u' ',
                       u'Deficiency and other anemia [59.]',
                       u'Other and unspecified lower respiratory disease'],
```


For large, nested json, `data_lines` can get quite large, causing some operations to take time

code!

Problems:

[**1**] Create a 3*x1* frame with the following contents:

(c1) “animals”: five species of animal

[**2**] Ingest a new character-separated (csv) file (offsides.tsv)

[**3**] Create a 100 x 2 frame with the following contents:

(c1) “rn”: 100 random integers between 0-1000

(c2) “evenodd”: “EVEN” if the corresponding integer is even, otherwise “ODD”