**REST Import API**

**Overview**

* Create or update data in bulk
  + Communities/Domains/Assets
  + Characteristics (attributes, relations, or complex relations)
* Import Methods
  + /import/json-job
  + /import/csv-job
  + /import/excel-ob
* Each method requires a file or a field
* The import will convert the file (if necessary) to a JSON format

**Import APIs**

**Comparison**

* Import API vs Core API
  + Better suited for bulk actions
  + Faster since APIs are executed in bulk
* Import APOI vs Catalog
  + Granularity of what gets imported
  + Import data Catalog doesn’t support
  + Does not allow for all Catalog Functionality
    - Profiling
    - Sampling
    - Data Classification
    - Etc.
* Note: You should only import data that you want to be managed in Collibra. Bringing too much data could transform your Collibra Platform into a Data Swamp!

**JSON File Format**

**Array of objects**

[

{command for Community 1},

{command for Community 2},

…

{command for Community N},

{command for Domain 1},

…

{command for Domain M},

{command for Asset 1},

…

{command for Asset P},

…

]

**Tips**

* When importing simultaneously import communities before domains
* If mapping between resources, import the resources first, then map them
* You can import
  + Communities
  + Domains
  + Assets
  + Relations
  + Attributes
  + Tags
  + Mappings

**Import API Resources**

**Documentation**

* <https://university.collibra.com/developer/rest/import-api-documentation>
* <instance\_URL>/docs/rest-importer/index.html

**Technical Lineage**

**Example**

* Use Collibra Data Lineage to ingest and extract technical lineage from many different sources
* What if the source is not currently supported?
* Use the import API:
  + Construct the JSON payload in accordance with the schema
  + Make a Colliibra REST Post call to /rest/2.0/import/json-job
  + Wait for the import job to finish with the Collibra REST API   
    /rest/2.0/jobs/{jobId}
  + Review imported relationships/lineage in Collibra Platform

**Scenario**

**Details**

* We are going to bring in domains, assets, and a complex relation between those assets
* In order to do this efficiently, we will use the Import API to bring in these resources simultaneously

**Task 1**

Encode our Collibra password in base64

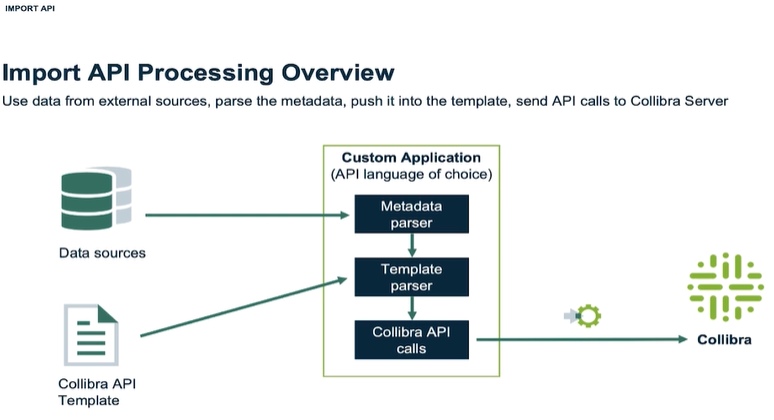
**Task 2**

Create JSON Templates

**Task 3**

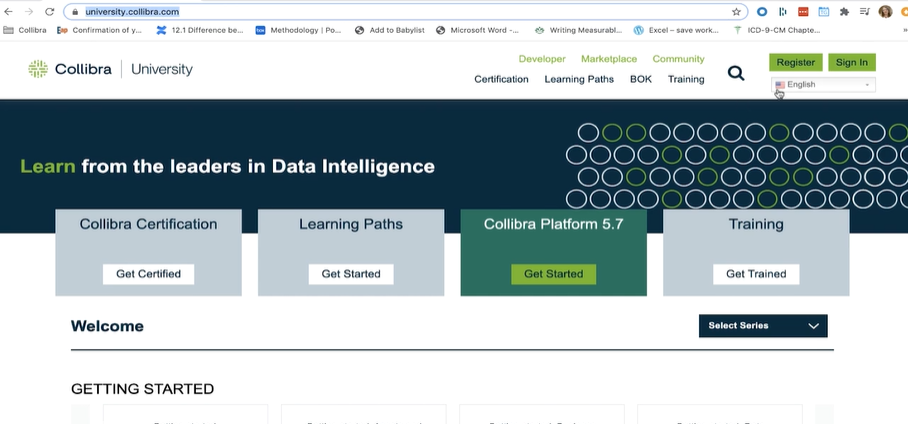
Bring in the following using Import API

* Data Asset Domain
* Mapping Domain
* 2 Data Element Assets
* 1 Mapping Specification Asset
* 1 Complex Relation between them with transformation logic

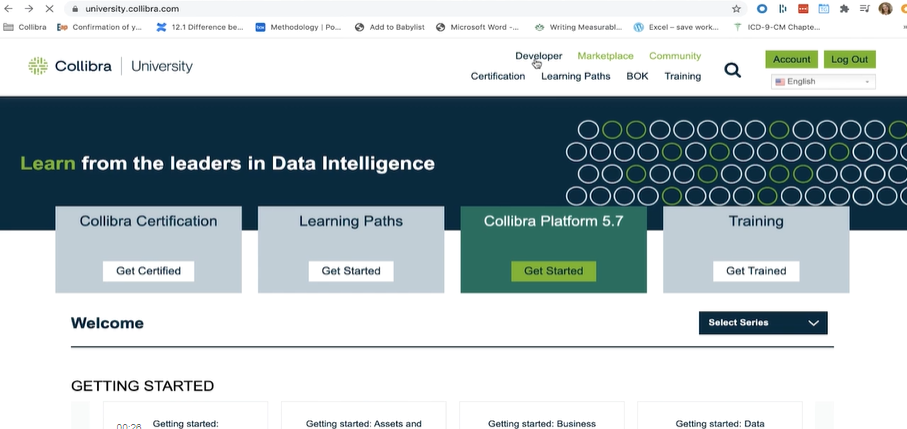


How to access – Ignore below – use [developer.collibra.com/api/rest/import-api](https://developer.collibra.com/api/rest/import-api)

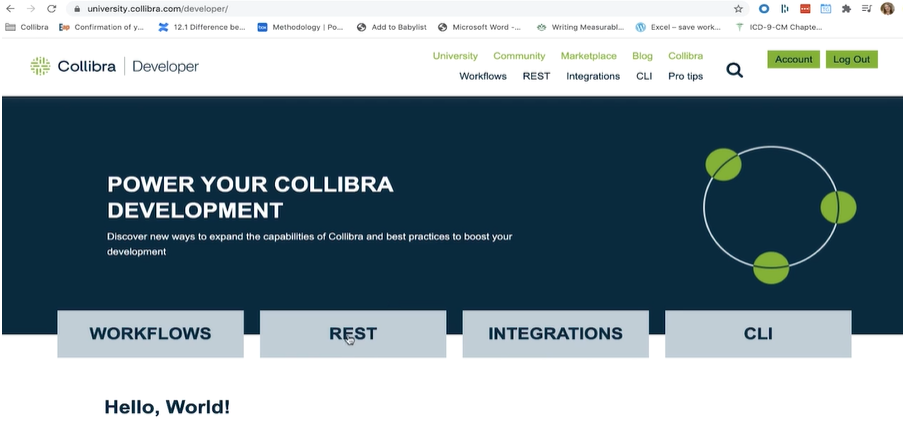
Sign in



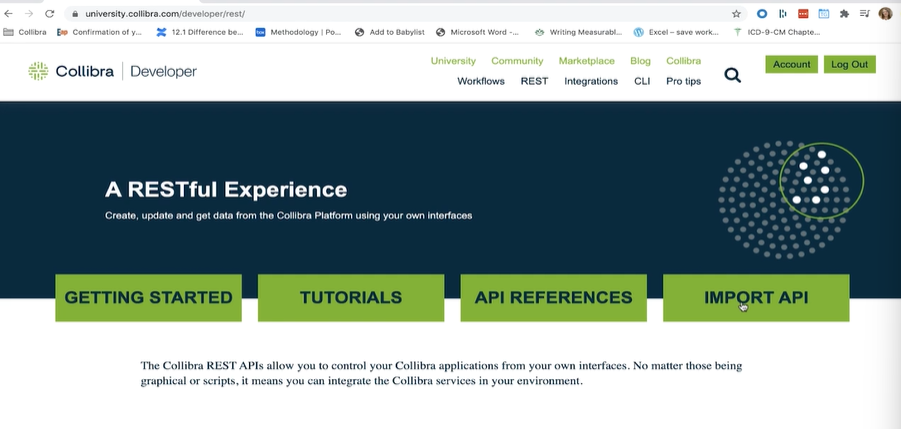
Click Developer

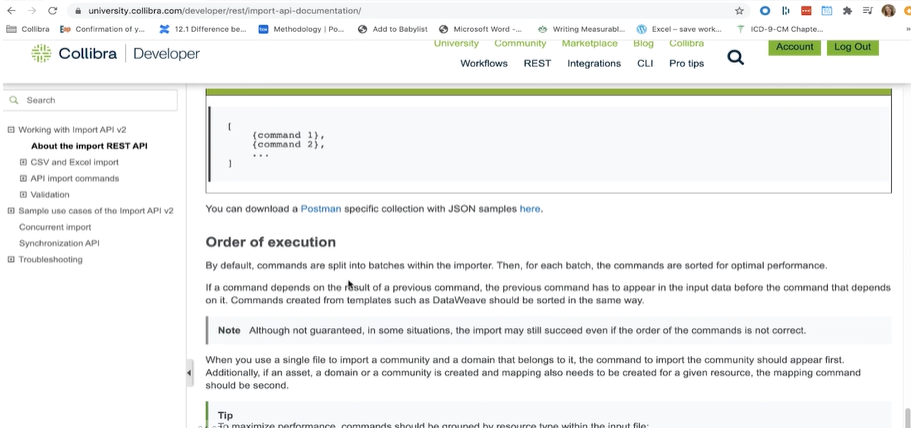


Click REST



Click Import API





**REST Call URL**

**Consists of 3 parts**

* Instance base URL (https://<your\_collibra\_url>)
* REST application path (/rest/2.0)
* End point path (/application/info)
* Example
  + https://<your\_collibra\_url>/rest/2.0/application/info

**Authentication**

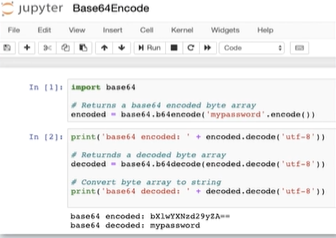
**Details**

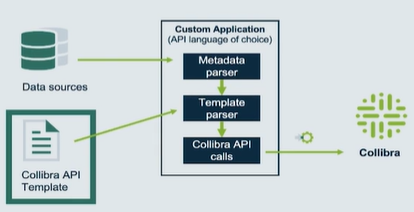
* OOTB session idle timeout is 30 minutes between API calls
* Minimally requires URL, username, and password
* We will encode our passwords using Base64

**Jupyter Notebook**

**Overview**

* Open-source web application
* Create/share live code in your language of choice
* In-browser editing for code, with automatic syntax highlighting, indentation, and tab completion/introspection
* Execute code from the browser
* Start running a notebook server from the command line “jupyter notebook”





**JSON Templates**

We want to create

* Domains
* Assets
* Complex Relations between our assets

We will need

* Domain Template
* Asset Template
* Complex Relation Template

**Complex Relation**

**Details**

Assets

* Client Name (Data Element)
* Client ID (Data Element)
* SQL Server Mapping (Mapping)
  + Has Transformation Logic, which is logic used to transform Client Name, Client ID

Field Mapping (Complex Relation)

* Source: Client Name
* Target: Client ID

