TDR Step-by-step Demo

March 29, 2024

◉ Introduction

Welcome to the hands-on portion of the Terra Data Repository (TDR) workshop!

In this portion of the workshop, we will be creating datasets, ingesting data, and setting that data up to be shareable within TDR.

Agenda:

1. Creating a dataset
2. Ingesting data
3. Creating assets

The lead instructor will walk you through each one of these steps while sharing their own screen, and send you to breakout rooms to try the steps yourself throughout the process.

If you have any questions or hit any obstacles as you are going through these steps, please ask one of the TAs for help either via direct message in zoom, or by posting to the zoom chat if you think others may be experiencing a similar issue.

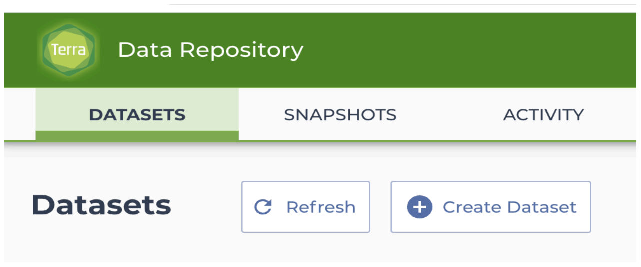
① Step 1: Creating a dataset

When storing data in TDR, the first step is dataset creation, at which point the dataset structure is defined. As you've now learned, a TDR dataset's structure includes a number of concepts (tables, relationships, assets, etc) that help to organize complex tabular data, and these are all defined by the schema of the dataset. While not all of these concepts are required to create a dataset, you do need a schema in place that defines tables and columns before you can ingest any data.

In this first step, we'll create a dataset that is empty, but that has several of these structural aspects defined, so that the data we ingest in the next step lands into the tables we've already organized. Note that we will include in our initial schema some relationships between the tables, but not assets. In practice both or neither of these can be included at this stage, and you can always add as many assets and relationships as you like later on.

In practice, all of this will be done using the JSON format, a standard format for representing structured data. This format is organized into what are called JSON objects. The full text for all the JSON objects you'll need for this demo are provided in this document, so all you'll have to do is paste them into the appropriate fields at certain steps.

Dataset creation is currently done through the TDR user interface. The first step is to log in to the Terra Data Repository using the appropriate credentials. Once you've done this, make sure you're in the Datasets tab by clicking on **Datasets** near the top left of your screen.



You can now see near the top of your screen the button labeled **Create Dataset**. Clicking this button will take you to the dataset creation screen, which has two steps.

◎ Step 1.1 - Provide Dataset Info

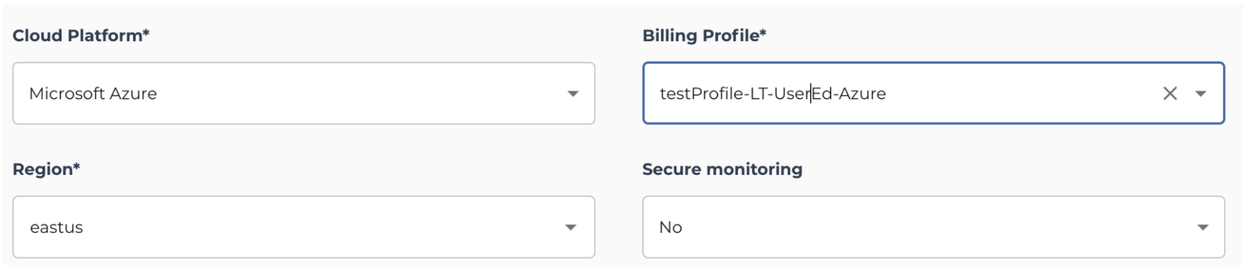
On the first screen in the two-step process, you'll provide basic information for your Dataset. Make sure you choose a unique name for your dataset. Below are some guidelines for naming TDR Datasets:

- The name must be globally unique (cannot have the same name as any other Dataset within TDR, even if it's tied to a different Billing Project).
- Dataset name can contain letters (upper or lowercase), numbers, and underscores. It cannot *begin* with an underscore, and it cannot include *spaces*).
- Suggested naming format: dataset-name > date _> initials

Once you've given your dataset a name and added any descriptive text you want, make the required selections in the dropdown menus near the bottom of the screen as follows:

- Select "**Microsoft Azure**" under **Cloud Platform**

- Select “**eastus**” under “**Region**”
- Select the one option you have available under “**Billing Profile**”



Cloud Platform*: Microsoft Azure

Billing Profile*: testProfile-LT-UserEd-Azure

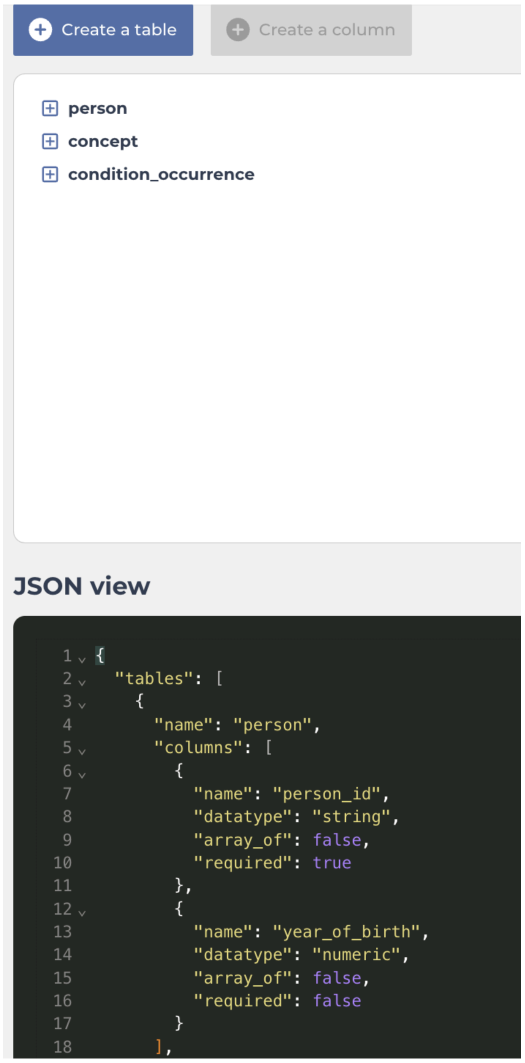
Region*: eastus

Secure monitoring: No

◎ Step 1.2 - Setting up the Schema

In the second step, we add our schema. This can either be done through the UI or it can be done by simply inputting the schema in JSON format into the field at the bottom of the screen. The two methods are essentially the same, and if you start playing with the buttons to add tables and columns, you’ll see those generated as JSONs in the lower field.

Working in the other direction (writing the JSON from scratch) can be convenient if you already have a schema JSON that you’re working with, which we do. Go ahead and paste the text (the *italicized* code) below into the field at the bottom of this screen. If you’ve done this correctly (being careful not to lose brackets or indentation/white spaces), you should see the visual representation of the tables at the top of the screen.

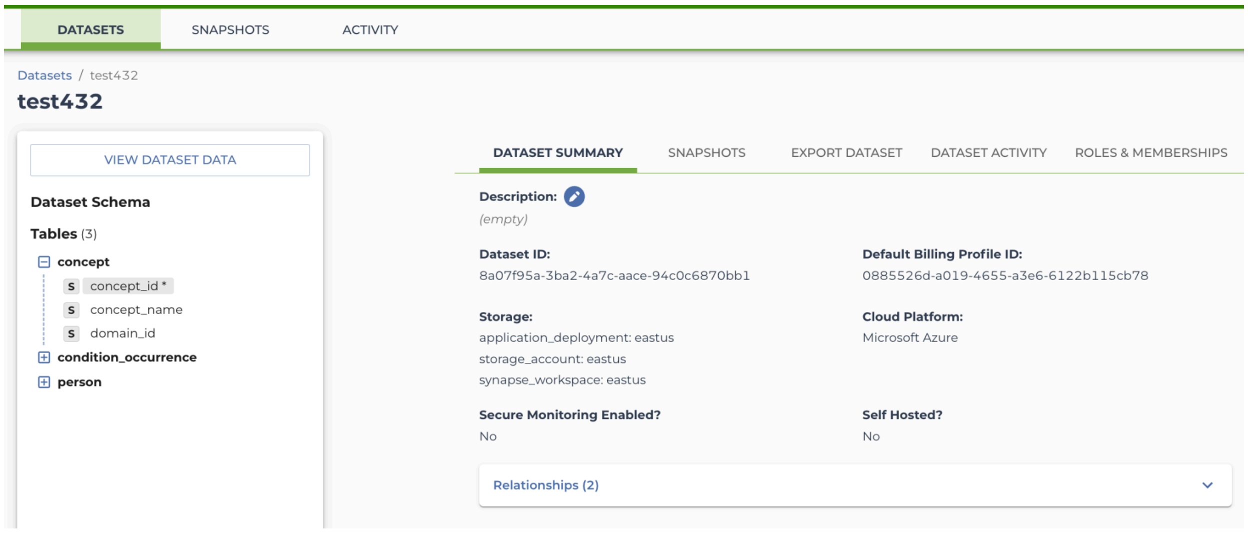


```
{
  "tables": [
    {
      "name": "person",
      "columns": [
        {
          "name": "person_id",
          "datatype": "string",
          "array_of": false,
          "required": true
        },
        {
          "name": "year_of_birth",
          "datatype": "numeric",
          "array_of": false,
          "required": false
        }
      ],
```

Schema JSON

```
{
  "tables": [
    {
      "name": "person",
      "columns": [
        {
          "name": "person_id",
          "datatype": "string",
          "array_of": false,
          "required": true
        },
        {
          "name": "year_of_birth",
          "datatype": "numeric",
          "array_of": false,
          "required": false
        }
      ],
      "primaryKey": [
        "person_id"
      ]
    },
    {
      "name": "concept",
      "columns": [
        {
          "name": "concept_id",
          "datatype": "string",
          "array_of": false,
          "required": true
        },
        {
          "name": "concept_name",
          "datatype": "string",
          "array_of": false,
          "required": false
        },
        {
          "name": "domain_id",
          "datatype": "string",
          "array_of": false,
          "required": false
        }
      ],
      "primaryKey": [
        "concept_id"
      ]
    },
    {
      "name": "condition_occurrence",
      "columns": [
        {
          "name": "condition_occurrence_id",
          "datatype": "string",
          "array_of": false,
          "required": true
        },
        {
          "name": "person_id",
          "datatype": "string",
          "array_of": false,
          "required": false
        },
        {
          "name": "condition_concept_id",
          "datatype": "string",
          "array_of": false,
          "required": false
        }
      ],
      "primaryKey": [
        "condition_occurrence_id"
      ]
    }
  ],
  "relationships": [
    {
      "name": "fk_condition_occurrence_person",
      "from": {
        "table": "condition_occurrence",
        "column": "person_id"
      },
      "to": {
        "table": "person",
        "column": "person_id"
      }
    },
    {
      "name": "fk_condition_occurrence_condition",
      "from": {
        "table": "condition_occurrence",
        "column": "condition_concept_id"
      },
      "to": {
        "table": "concept",
        "column": "concept_id"
      }
    }
  ]
}
```

If you've done this correctly, you should be able to click **Submit**, and then see a modal telling you your dataset is being created. After a minute or so, it will be replaced by a modal telling you your dataset was successfully created. You can now click **GO TO DATASET DETAILS PAGE** and confirm that the dataset is in place!



test432

Dataset Schema

Tables (3)

- concept
  - concept_id *
  - concept_name
  - domain_id
- condition_occurrence
- person

Description:
(empty)

Dataset ID:
8a07f95a-3ba2-4a7c-aace-94c0c6870bb1

Default Billing Profile ID:
0885526d-a019-4655-a3e6-6122b115cb78

Storage:
application_deployment: eastus
storage_account: eastus
synapse_workspace: eastus

Cloud Platform:
Microsoft Azure

Secure Monitoring Enabled?
No

Self Hosted?
No

Relationships (2)

② Step 2: Ingesting Data

Now that you have your empty dataset ready to accept data, you can ingest data into it at whatever pace suits you. Note that the panel on the left side of your dataset details page shows your schema (specifically, your tables and the columns each table contains) and doesn't indicate whether there is actually data in these columns.

As mentioned earlier in the training, ingesting data is not currently done directly through the user interface. Instead of fields and buttons in the UI, the command to ingest data is submitted via API. There are numerous strategies and options for data ingestion into TDR, all of which are outlined in this article.

In order to keep things simple and instructive, the strategy we'll use to populate our small tables will be to use a JSON object that contains a nested array of JSON objects, each corresponding to a row of data we'd like to ingest. It's useful to recall here that JSON objects contain key:value pairs. Below is an example of a JSON object corresponding to one row of data. The data are represented by the *value* to the right of the colon. The *key* to the left of the colon specifies which attribute (aka which column) a given value belongs to.

```
{ "person_id": 1, "year_of_birth": 1963 }
```

Notice that a single JSON object can contain any number of key:value pairs, and their order with respect to each other doesn't matter. If we want to represent a whole table's worth of rows, that table should also be a JSON object, and nested within that JSON will be some key:value pairs where the values are themselves complete JSON objects. The JSON below represents all of the data we'll be ingesting into the **Person** table.

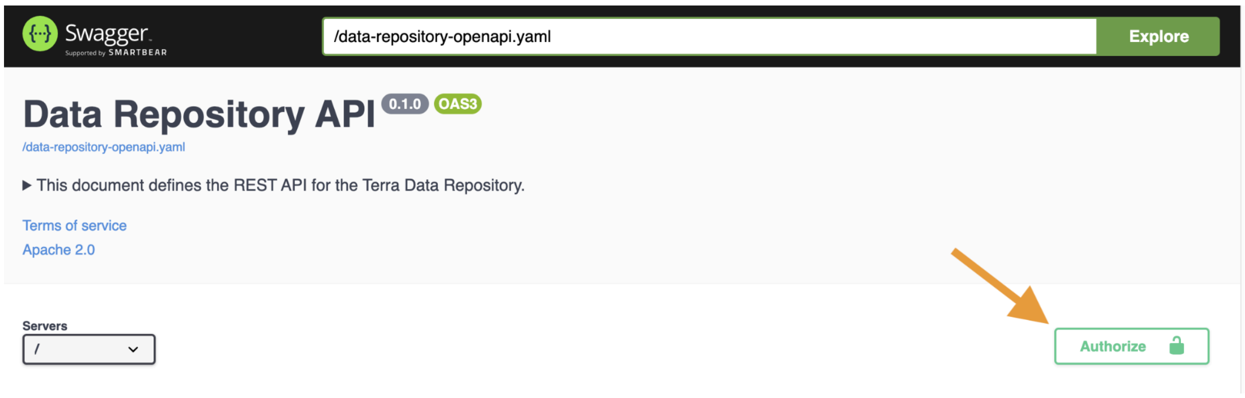
```
{
  "table": "person",
  "records": [
    { "person_id": 1, "year_of_birth": 1963 },
    { "person_id": 2, "year_of_birth": 1973 },
    { "person_id": 3, "year_of_birth": 1951 },
    { "person_id": 4, "year_of_birth": 1950 },
    { "person_id": 5, "year_of_birth": 1959 },
    { "person_id": 6, "year_of_birth": 2010 },
    { "person_id": 7, "year_of_birth": 1922 },
    { "person_id": 8, "year_of_birth": 2014 },
    { "person_id": 9, "year_of_birth": 1996 },
    { "person_id": 10, "year_of_birth": 1993 }
  ],
  "format": "array"
}
```

This JSON contains three key:value pairs at the top level. The "table": "person" pair tells TDR to which table this data should be sent, the "records": [array] pair contains the data themselves, and the “format”: “array” pair just lets TDR know what to expect as the value paired with the “records” key. Note that the pairs are separated from other pairs by commas, as are JSON objects.

◎ Step 2.1 - Authenticating with the API

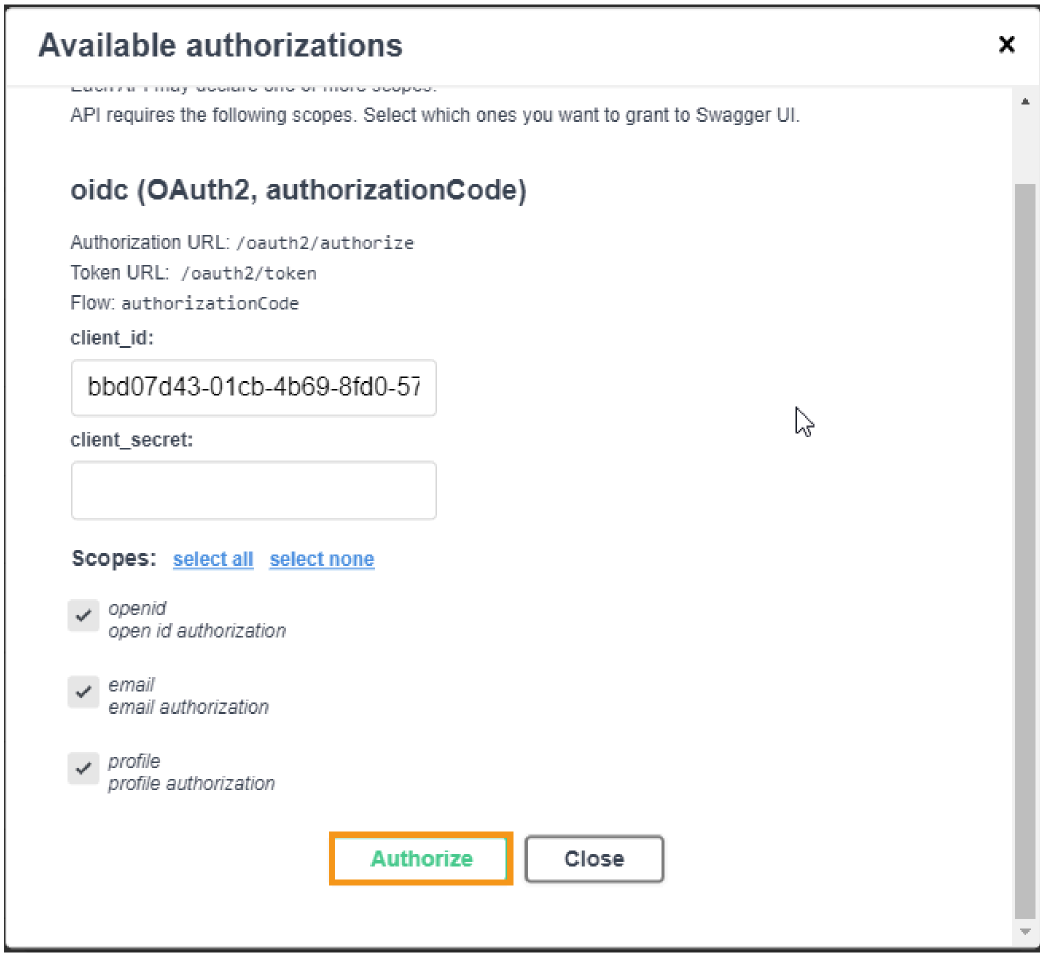
So we have a data ingest JSON ready, and we need to submit it via API. Navigate to the following URL: https://data.terra.bio/swagger-ui.html#/datasets/ingestDataset

Before you can execute the command at that URL, you’ll need to activate your ability to do so with a few simple clicks. First, you’ll need to authenticate your identity one more time. Do this by clicking **Authorize** near the top right of your screen.



▶ This document defines the REST API for the Terra Data Repository.

You’ll see a modal asking you to select authorizations. Go ahead and click **Authorize** at the bottom of the modal with just the default authorizations.


Available authorizations

Each API may declare one or more scopes.
API requires the following scopes. Select which ones you want to grant to Swagger UI.

oidc (OAuth2, authorizationCode)

Authorization URL: `/oauth2/authorize`
Token URL: `/oauth2/token`
Flow: `authorizationCode`
client_id: bbd07d43-01cb-4b69-8fd0-57
client_secret:

Scopes: select all select none

✓ openid
open id authorization

✓ email
email authorization

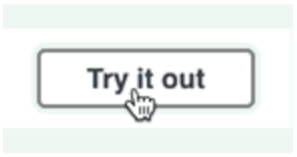
✓ profile
profile authorization

This will take you to a screen prompting you to log in with a Google or Microsoft identity. Click on “sign in with Google”. If the @terra-training.org email you were assigned for this training doesn’t appear as one of the default options, click **use another account**, and use the credentials you were given for this training.

You can read about authentication in detail in this article.

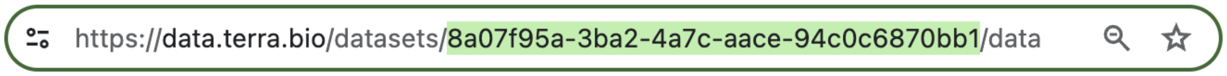
◎ Step 2.2 - Executing the API

There is one last thing you need to do before actually executing the API. You need to activate the editable JSON field by clicking **Try it out** at the right of the API we’re using.



You should now be able to edit the request body. You'll notice that the request body is one of two required fields. The other is the ID field that will tell this API where to find the dataset that is receiving the request.

Let's start with the ID field. The API identifies the datasets by their UUIDs, which are conveniently a distinct part of the URL of the datasets within the TDR site itself. If you look back at your dataset details page, you should copy the portion of the URL highlighted in this image:


https://data.terra.bio/datasets/8a07f95a-3ba2-4a7c-aace-94c0c6870bb1/data

You can also find this on the Dataset's summary page (under "Dataset ID")

Once you have copied this string, paste it into the ID field above the request body.

The request body itself contains by default a JSON object with a lot of key:value pairs we don't actually need. Our request is quite simple, and the entirety of it is covered by the JSON in the intro to ***Step 2: Ingesting Data*** of this document, so just erase the default text and paste that JSON in exactly as it is.

Here it is again:

```
{
  "table": "person",
  "records": [
    { "person_id": 1, "year_of_birth": 1963 },
    { "person_id": 2, "year_of_birth": 1973 },
    { "person_id": 3, "year_of_birth": 1951 },
    { "person_id": 4, "year_of_birth": 1950 },
    { "person_id": 5, "year_of_birth": 1959 },
    { "person_id": 6, "year_of_birth": 2010 },
    { "person_id": 7, "year_of_birth": 1922 },
    { "person_id": 8, "year_of_birth": 2014 },
    { "person_id": 9, "year_of_birth": 1996 },
    { "person_id": 10, "year_of_birth": 1993 }
  ],
  "format": "array"
}
```

This tells the API everything it needs to know about where each piece of data goes - the UUID indicates the target dataset for the request; the "table" key indicates the target table for the array; the "records" key contains an array of JSONs, each of which has a multiple key:value pairs, where the keys indicate the target columns for each piece of data.

You can now click **Execute** below the request body! Scroll down to the “response” section to see if you’ve succeeded. You should see either a **200** or **202** code. If you see one of the other codes, you can try to troubleshoot yourself by scrolling down to see what the error code means, or ask a TA for help.

If you’ve successfully executed this ingest command, you should be able to confirm this by navigating to your Dataset Details page, clicking **VIEW DATASET DATA**, and selecting the **Person** table from the dropdown menu. The records we just submitted via API should now be there.

Go ahead and populate the other two tables by repeating step 2.2 two more times using the ingest request JSONs below:

JSON for “condition_occurence” Table Ingest Request

```
{
  "table": "condition_occurrence",
  "records": [
    {
      "condition_occurrence_id": "1",
      "person_id": "1",
      "condition_concept_id": "1010"
    },
    {
      "condition_occurrence_id": "2",
      "person_id": "1",
      "condition_concept_id": "2000"
    },
    {
      "condition_occurrence_id": "3",
      "person_id": "4",
      "condition_concept_id": "1000"
    },
    {
      "condition_occurrence_id": "4",
      "person_id": "8",
      "condition_concept_id": "2000"
    }
  ],
  "format": "array"
}
```

JSON for “concept” Table Ingest Request

```
{
  "table": "concept",
  "records": [
    {
      "concept_id": "1000",
      "concept_name": "Low blood pressure",
      "domain_id": "Condition"
    },
    {
      "concept_id": "1010",
      "concept_name": "Elevated blood pressure",
      "domain_id": "Condition"
    },
    {
      "concept_id": "2000",
      "concept_name": "Non-small cell lung cancer",
      "domain_id": "Condition"
    },
    {
      "concept_id": "3000",
      "concept_name": "aspirin 81mg",
      "domain_id": "Drug"
    }
  ],
  "format": "array"
}
```

---

③ Step 3: Adding Assets

Now that our dataset is in place and populated with data, there is one last missing piece we need in order for the data to become shareable: Assets!

As you learned earlier, TDR data that needs to be shared must be organized into assets - sets of columns from tables within a given dataset. In principle, this facilitates granular over data sharing, since you can grant any individual or group access to exactly the tables and columns you wish, but you can make a “full view” asset that includes all columns from all tables.

In practice, an asset is also defined by a JSON object. This can be included in the schema during dataset creation, or it can be added after the fact. Since the flexibility to create new assets for different research purposes is one of the features of TDR, we’re opting to demonstrate asset creation downstream of dataset creation and data ingest.

Asset creation is not currently available through the UI, so just like in step 2, we’ll be using API. There is a detailed article on adding assets via API in our support documentation. Let’s walk through these steps. The process is the same as before (and you may need to re-authenticate), but now you’ll need the asset creation API, found here:
https://data.terra.bio/swagger-ui.html#/datasets/addDatasetAssetSpecifications

Once you’ve authenticated, clicked **Try it out**, and added your dataset’s UUID to the ID field, paste the JSON below into the request body, and hit **Execute**. Take a moment to look at the JSON itself to understand what information is specified by an asset. The asset being created includes all of the columns from one table, some of the columns from another table, as well as the relationships between the two tables.

JSON for Asset Specification Request

```
{
  "name": "person_and_condition",
  "tables": [
    {
      "name": "person",
      "columns": [
        "person_id",
        "year_of_birth"
      ]
    },
    {
      "name": "condition_occurrence",
      "columns": [
        "person_id",
        "condition_occurrence_id"
      ]
    }
  ],
  "rootTable": "person",
  "rootColumn": "person_id",
  "follow": [
    "fk_condition_occurrence_person"
  ]
}
```

Once you successfully executed this step, you can confirm that this worked by taking the first few steps of snapshot creation. From your Dataset Details page, click **VIEW DATASET DATA**, then click on the dashed triangle in the panel on the right side of your screen. This will bring up your data filtration modal. Click **Next** at the bottom, and on the following screen you'll see a dropdown menu near the bottom prompting you to select an asset. Click on this to verify that the asset was successfully added.

◉ Wrap-up

Congratulations! You have completed this tutorial on creating TDR datasets and ingesting data. As a follow-up exercise, see if you can add a “full view” asset.