Integrating hierarchy into Skills using GraphQL

Technical Brief

Last Updated: 6/9/2021
Verified using Python 3.1 and Juypter 0.33.12

PRESALES

Table of Content

Introduction

Pluralsight Skills has introduced a nested teams experience that enables organizations to import hierarchy data into the platform. With this capability, organizations can better delegate managers to their respective team or teams they manage; giving them better visibility into their skill inventory.

Deciding the right approach

There are many factors in finding the right approach to implementing hierarchy within an organization. The organization needs to factor in the single source of truth, frequency of syncing hierarchy and complexity. Below are a couple aspects to consider when implementing the hierarchy:

- **Single Source of Truth.** Organizations have varying degrees of how the business looks at their workforce. For example, organizations might leverage their management hierarchy to reflect who manages which employees. In addition, there might be another hierarchy that reflects the organization based on functional departments or how finance reports cost centers etc. Today Pluralsight supports a single team hierarchy. (Future releases may support matrixed hierarchies) Prior to implementing nested teams within Pluralsight, it is important to agree on which hierarchy will be loaded into the platform.

- **Frequency of syncing hierarchy.** Hierarchy reflects a moment in time on how the organization is evolving. Organizations grow and restructure as they continue to evolve and the hierarchy will constantly change to reflect it. Therefore, it is important to decide the frequency in which the hierarchy needs to be loaded into the Skills platform. If the hierarchy churn is very high, consider a less frequent (ie monthly) refresh cycle so that leaders have a consistent group to measure over time.

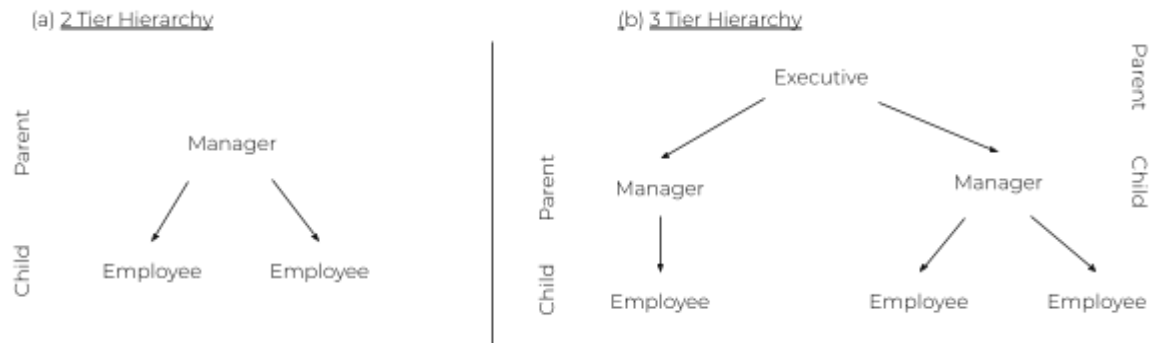
Getting Started

Prerequisites

- Experience using Python
- You will need the following prior to following along:
    - Hierarchy Script Template.zip
    - Pluralsight PaaS API Key

Overview of Schema

The approach taken in designing hierarchy is leveraging a parent-child hierarchy where each child belongs to one parent. As an example, if an organization has a two level hierarchy, manager and employee (see example a), then the manager would be considered parent level and the employee level is the child level. When the organization grows and adds a new layer such as executive, then there are two parent child hierarchies (see example b); the first is executive to manager and then manager to employee.



Pluralsight Skills has exposed a new mutation that allows organizations to define their parent-child relationship within a mutation called *addTeam*. The *addTeam* mutual requires 2 input and 1 optional input which are:

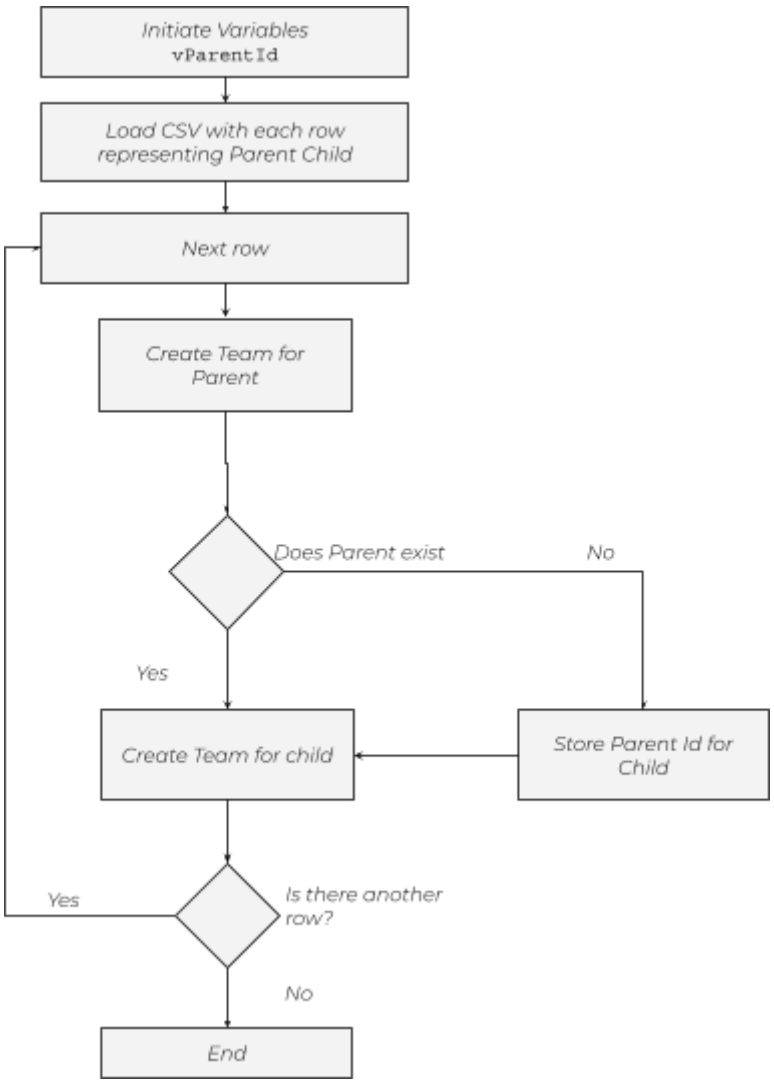
- Name - name of the team
- description : description of the team
- parentTeamId (optional): The unique identifier of a Team's parent. If not passed, it will not assign team to a parent

Basic Mutation Structure

Let's begin with a basic 2 tier hierarchy that has a manager-employee relationship as shown above.

Manager (parent)	Employee (child)
Sam	Tom
Sam	Jerry

The visual diagram below outlines how to take this 2 tier hierarchy and load it into Skills.



Let's use a basic mutation that returns the parent id of a new team:

```
query {
  mutation {
    addTeam (input: {name:"The Big Bang Theory", description:"The
Big Bang Theory"){
      id,
      name,
      description
    }
  }
}
```

Importing CSV and dynamically creating teams

To begin, you will need to import a few libraries by writing the following lines.

```
import requests
import json
import pandas as pd
import csv
import os.path
from datetime import datetime
```

Next let's load in the csv, loop through each row, define the url, query and the request information. The script below leverage the following three variables:

1. url - stores the url to Pluralsight's endpoint
2. payload - stores the GraphQL query. Please note, that since the string is wrapped in double quotes, a backslash is needed inside the quotes (\")
3. headers - is an object that contains both the content-type and authorization.

```
#Read CSV & Sort
df = pd.read_csv ('input.csv')
totalRow = df.shape[0]

#Sort parent by alphabetical order
sort_df = df.sort_values(by='Parent')

for row in sort_df.values:
     url = "https://paas-api.pluralsight.com/graphql"

     name = row[0]
     description = row[0]

      payload = r'{"query": "mutation { addTeam (input: { name:
\"%s\", description: \"%s\"}) { id, name, description } } "}' %
(name, description)

    # or even better, use json.dumps to avoid escaping altogether
    # and allow much nicer formatting of the mutation
```

```
    query = '''
      mutation {
        addTeam (input: {
          name: "%s", description: "%s"
        }) { id, name, description }
      }
    ''' % (name, description)
    payload = json.dumps({ 'query': query })

    headers = {
        'content-type': "application/json",
        'authorization': "Bearer %s" % (vApiKey)
    }

    response = requests.request("POST", url, data=payload,
headers=headers)

   print(response.text)
```

This will return a response similar to below:

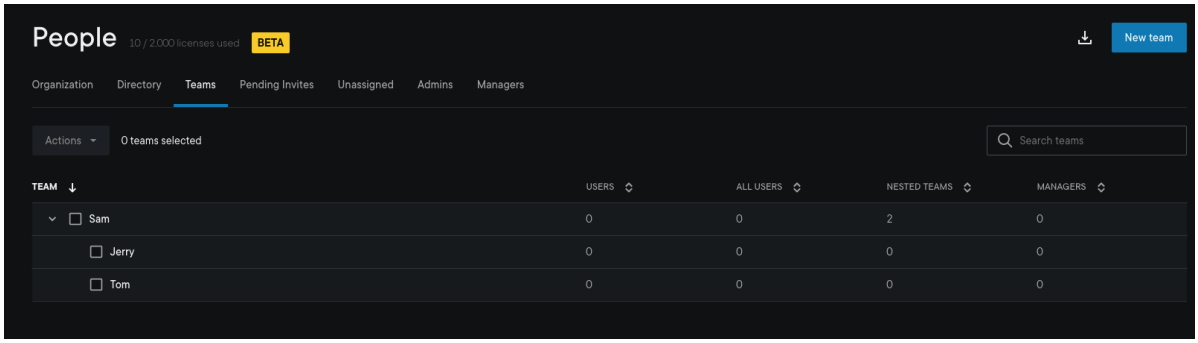
```
{"data":{"addTeam":{"id":"7590f3ee-c1a3-4624-8559-5bb00a705add","n
ame":"Tom","description":"Tom","parentTeamId":"fb5e3910-c5ba-470d-
8b31-9b9e0b0b993b"}}}
```

Finally let's output the response of the newly created team back to a new csv.

```
    with open(filename, 'a', newline='') as csvfile:
        fieldnames = ['Parent','Child','Response']
        writer = csv.DictWriter(csvfile, fieldnames=fieldnames)
        writer.writerow({'Parent':row[0], 'Child':row[1],
'Response':response.text})
```

Exploring Hierarchy within People Experience

Now that the CSV file has been loaded. You can interact with the new hierarchy in the Team section within People.



For more information please go to the help page to learn more about this new experience. https://help.pluralsight.com/help/people-experience

Conclusion

In summary, loading hierarchy data requires organization to make design decisions before implementing it. These design decisions need to address what the single source of truth is and the frequency of the data being loaded. Once these design decisions are made, Pluralsight's GraphQL provides organizations with the ability to upload their own hierarchy dynamically using custom scripting (as shown here via Python).

Resources

- GraphQL Documentation hosted on our Developer portal
  - Pluralsight Developer Portal: Home
- Playground for exploring & converting query to basic python query
  - GraphQL Playground

FAQ

Q: Got anything to quickly get started?
A: Hierarchy Script Template.zip Download. Open. Replace API token in Queries.

Q: Does the GraphQL API documentation cover data on SKILLS and FLOW plans?
A: Not at this time. It is only available for SKILLS data, though a plan that has both SKILLS and FLOW products can utilize the API to pull SKILLS data.

Q: Is there a limitation to the number of queries we create?
A: There is no limit to the number of queries. To minimize large data pulls, please use filters and pagination to create delta data pulls (and avoid all-time reports--especially for large datasets like CourseUsage, ChannelProgress, etc). This

reduces load on our servers and provides a better experience for you and all customers who are using the API endpoint.

Q: How do I request an API key?
A: This can only be done by a Pluralsight Plan Admin on the Manage Keys tab of the Developer Portal. If you do not know who your Plan Admin(s) is/are, please reach out to your dedicated CSM or support resource to identify them.

Q: My company has more than 1 Pluralsight plan. Is there a way to pull data for more than 1 plan? How do I request a multi-plan API key?
A: Yes, a multi-plan API key is what you need. To request one, please reach out to us at professionalservices@pluralsight.com. As long as each plan has an Integrations or ProServ SKU, we will be able to add it to a multi-plan API key for you.

Q: How long does it take to generate an API token?
A: After submitting a request through the Developer Portal link in the resource section, your API will be approved and ready to use immediately. If there is a delay for any reason or it does not function, please contact support@pluralsight.com.

Q: Will you create and maintain the Python script for our organization?
A: Not at this time. We may support a templated dashboard sample in the future.

Q: What data can I not see?
A: If it is not listed in the Developer Portal documentation schema, then it is not available. Keep an eye on the change log section of the webpage for updates.

Q: I do not like using GraphQL APIs or are unfamiliar with them. Can I continue to use your REST APIs?
A: You may continue to use our SKILLS REST APIs currently, however they will be decommissioned in Q2 2021. The data availability and manipulation through the REST APIs is not as complete. For example you can't get SkillIQ, RoleIQ, etc from REST APIs. We will be using GraphQL APIs as the standard for the SKILLS product. FLOW (formerly GitPrime) will continue to use REST. If you are using REST APIs currently for SKILLS data and want to know how to make the switch to GraphQL, please see the help documentation here or email us at professionalservices@pluralsight.com.