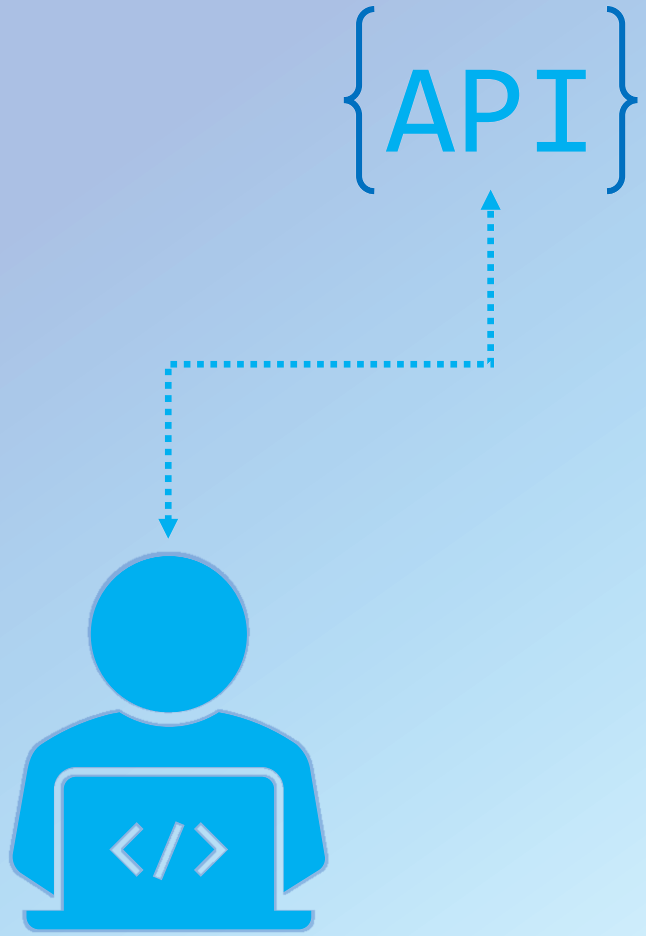
Introduction to the ASpace API

Valerie Addonizio

Atlas Systems



ArchivesSpace™ Powered by ATLAS SYSTEMS

This is Only the Beginning

Why is this a presentation and not a "workshop?"

- I've taught API workshops; they are brutal
- Not suited to foundational concepts
- Better off as a Part 2 to this Part 1
- This is a journey
  - No, really, this might take you awhile
- I genuinely think what I'm about to show you is the best I can do for an introduction

This is Only the Beginning

5 years ago

Years of frustration and failure

Excellent colleagues

LORA!
ERIC!
AUSTIN!

15+ weeks of Python classes

Python for Everybody (1-4)
Dr. Charles Severance
U Michigan via Coursera

Copying others

(seriously, I would be nowhere without these people)

Learning by teaching

"There's an API for That!"
with Lora Woodford
2016-2018

Unanticipated career shift

"And you may ask yourself, well
How did I get here?"

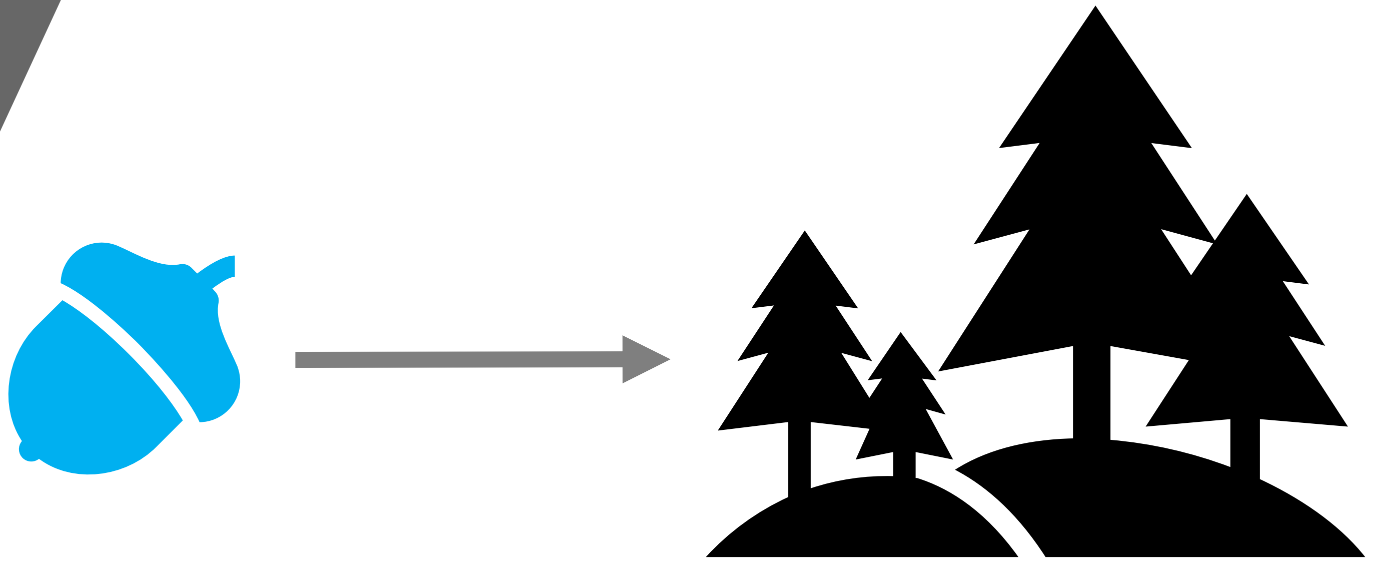
Just cannot do this in three hours (or eight hours) (or 20 hours)

This is Only the Beginning

What I can give you

- This recording, which I suggest you watch again if you end up pursuing this
- My API Playbook, a guide for your next steps
  - Includes API Client instructions (easy; start there)
- The scripts I demonstrate today
- Both the API Playbook and the scripts will be available here

Basics



Basics

- What’s an API?
- Same data, different view
- How do I access it?
- Why use it?
  - Enables work at scale
  - Enables applications to communicate directly

What’s an API?

- Stands for Application Program Interface
- There are many types
  - ASpace has a RESTful web API
- You use APIs constantly
- They aren’t specifically meant for this

This = a human interacting with an application. They are meant for *applications* to interact with *applications*.

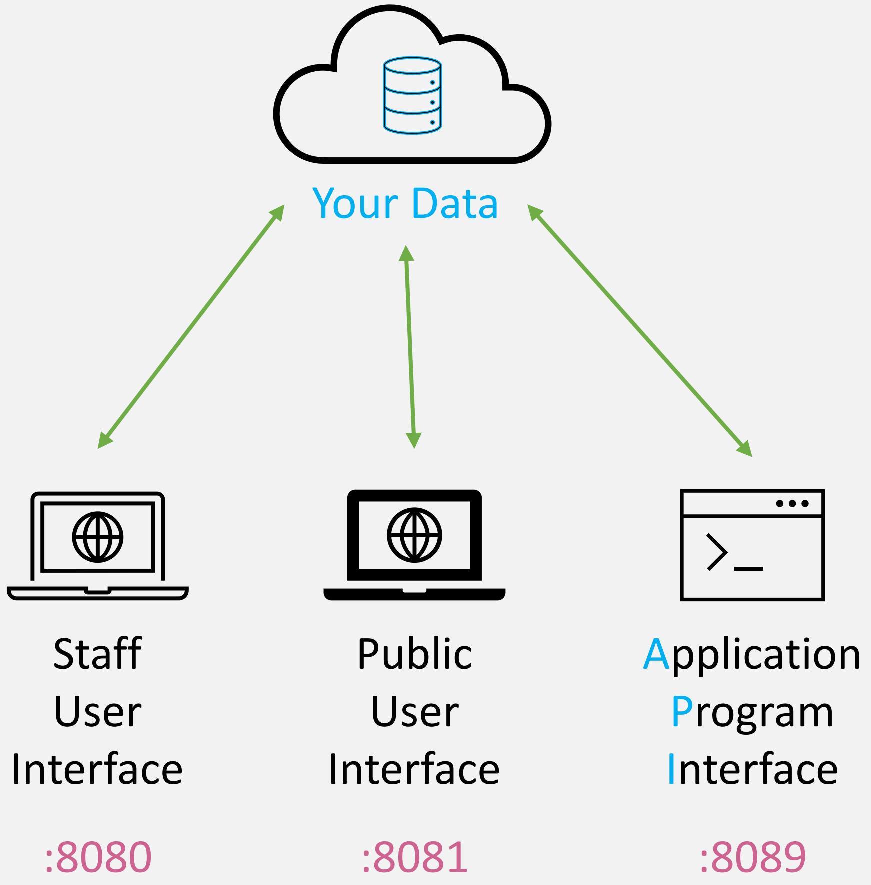
What’s an API?

- It does nothing by itself
- Think of it like an open microphone
- It does nothing until the moment someone speaks
- Otherwise it sits silently, waiting for input

Basics

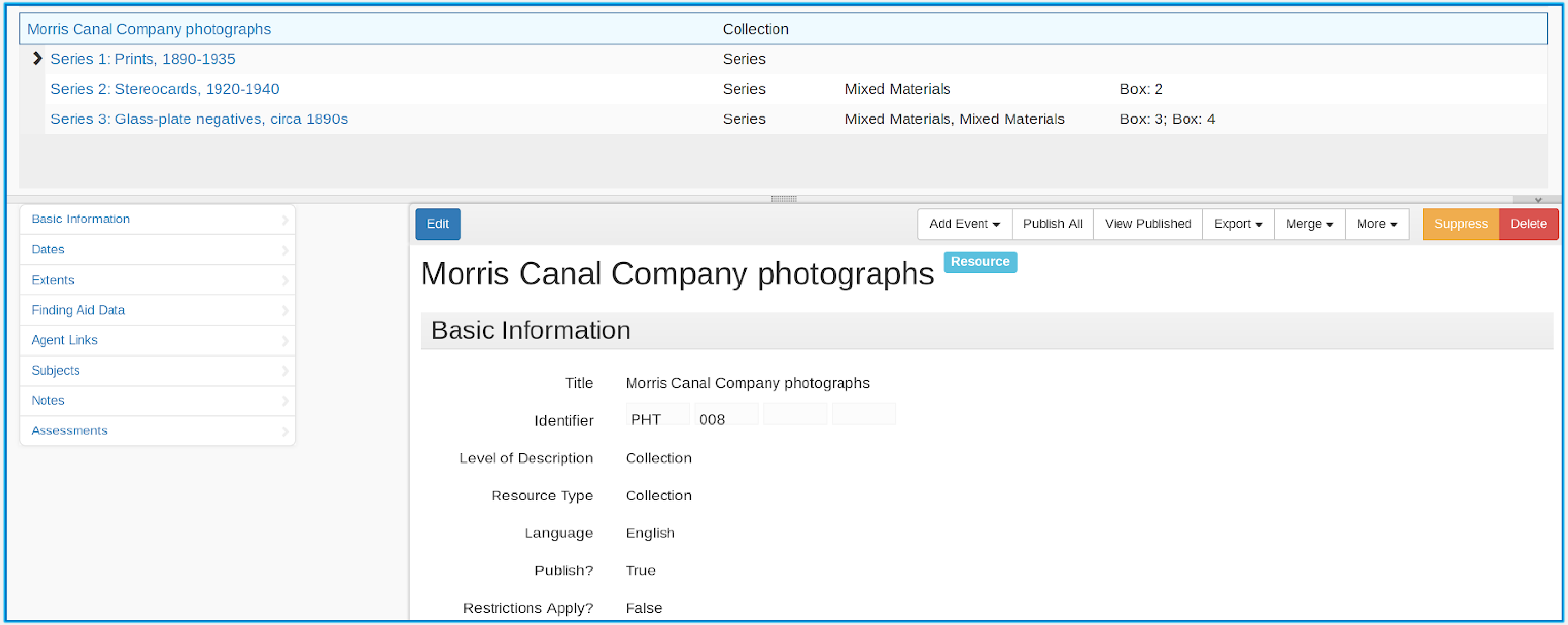
- What’s an API?
- Same data, different view
- How do I access it?
- Why use it?
  - Enables work at scale
  - Enables applications to communicate directly

Just another way to interact with your data



Same data, different view

Staff interface:



Morris Canal Company photographs	Collection		
Series 1: Prints, 1890-1935	Series		
Series 2: Stereocards, 1920-1940	Series	Mixed Materials	Box: 2
Series 3: Glass-plate negatives, circa 1890s	Series	Mixed Materials, Mixed Materials	Box: 3; Box: 4

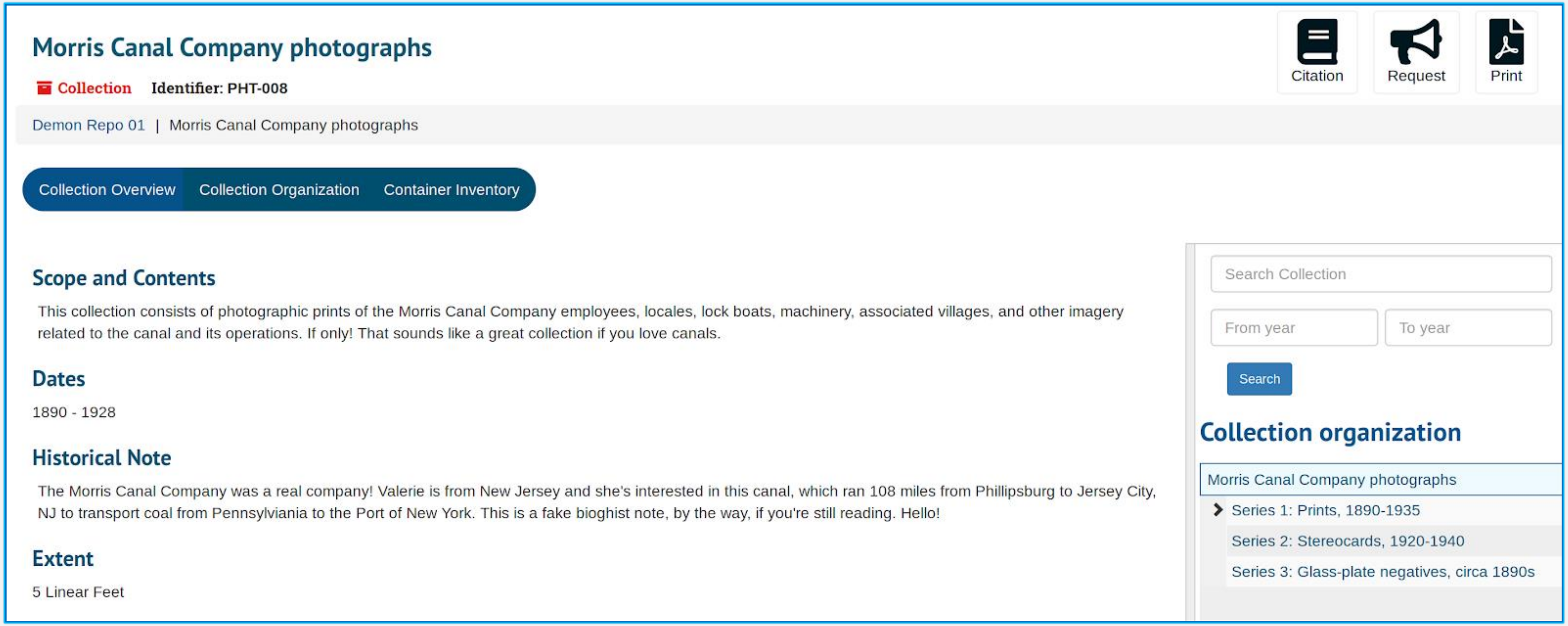
Morris Canal Company photographs

Basic Information

Title	Morris Canal Company photographs
Identifier	PHT 008
Level of Description	Collection
Resource Type	Collection
Language	English
Publish?	True
Restrictions Apply?	False

Same data, different view

Public interface:



Morris Canal Company photographs

Collection Identifier: PHT-008

Demon Repo 01 | Morris Canal Company photographs

Scope and Contents

This collection consists of photographic prints of the Morris Canal Company employees, locales, lock boats, machinery, associated villages, and other imagery related to the canal and its operations. If only! That sounds like a great collection if you love canals.

Dates

1890 - 1928

Historical Note

The Morris Canal Company was a real company! Valerie is from New Jersey and she's interested in this canal, which ran 108 miles from Phillipsburg to Jersey City, NJ to transport coal from Pennsylviania to the Port of New York. This is a fake bioghist note, by the way, if you're still reading. Hello!

Extent

5 Linear Feet

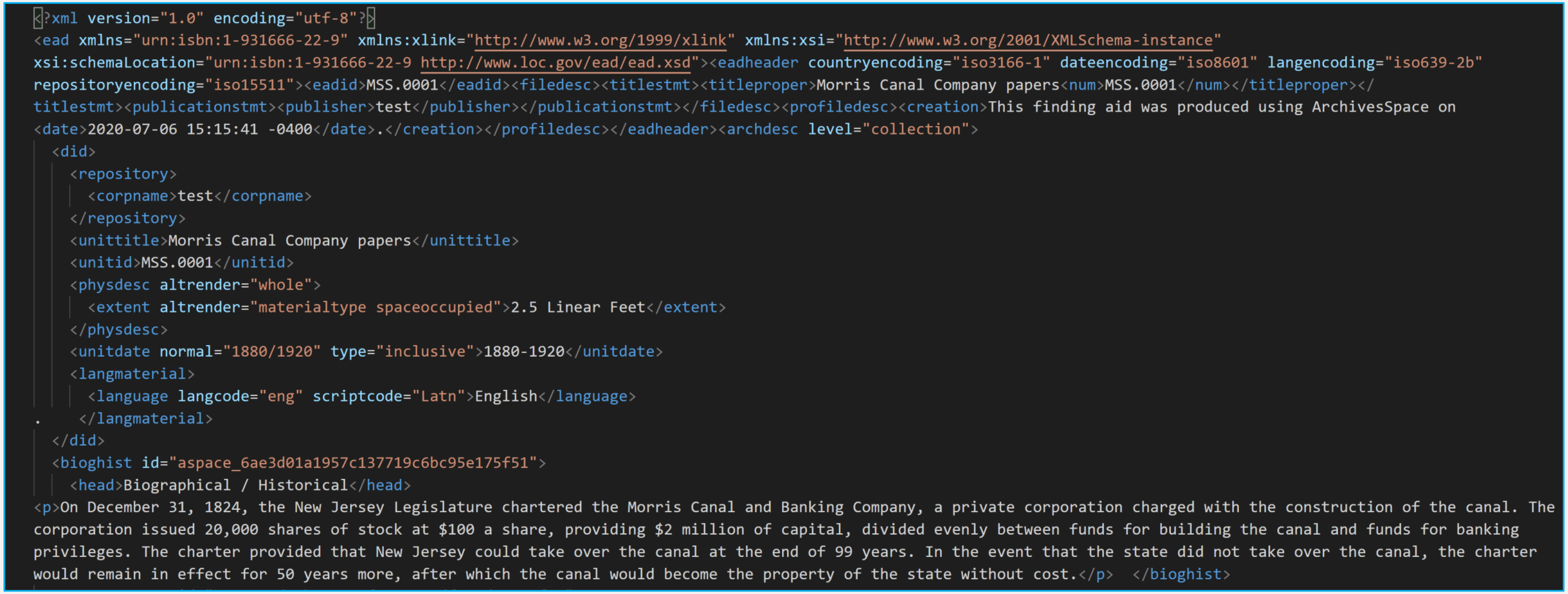
Collection organization

Morris Canal Company photographs

- Series 1: Prints, 1890-1935
- Series 2: Stereocards, 1920-1940
- Series 3: Glass-plate negatives, circa 1890s

Same data, different view

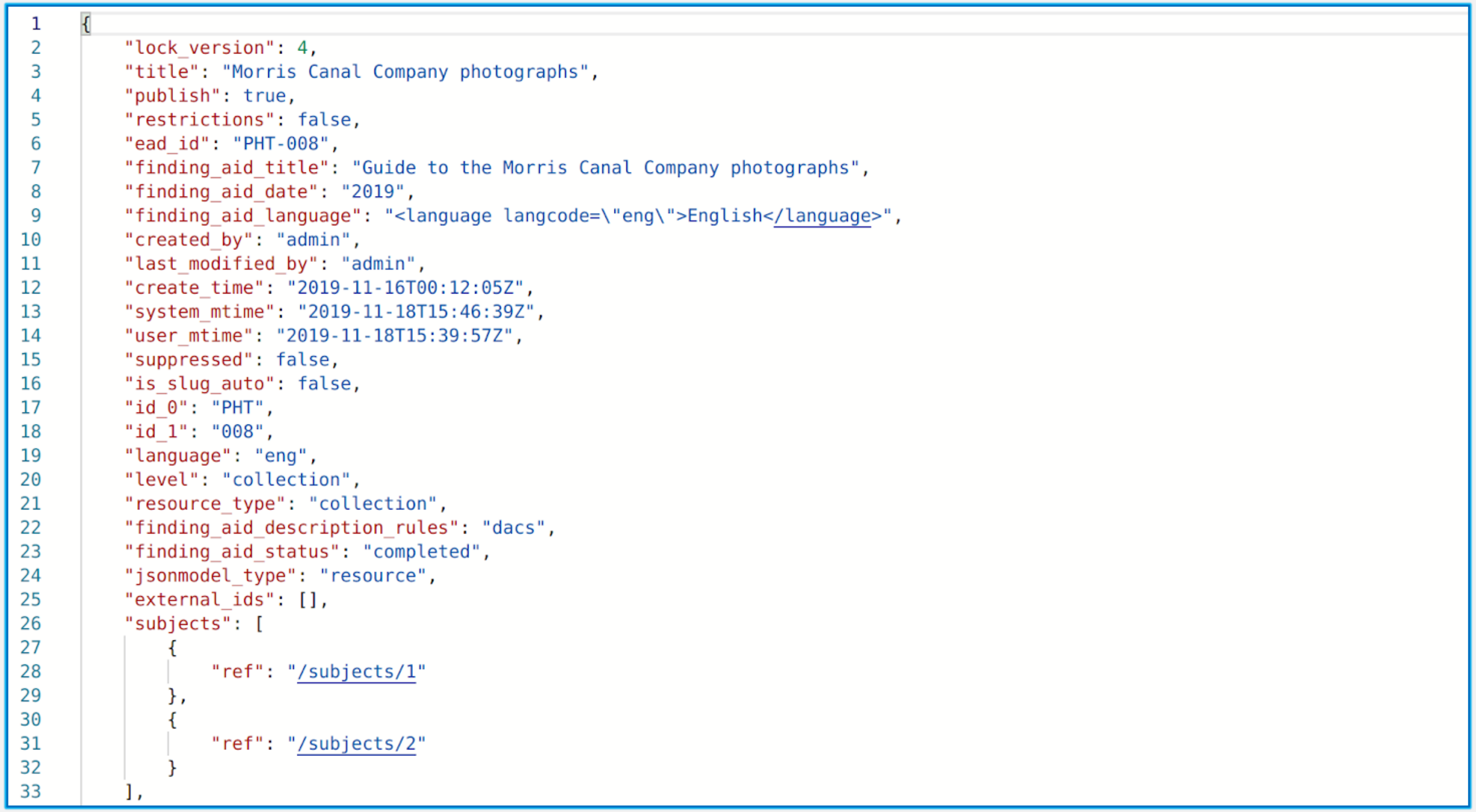
XML:



```
<?xml version="1.0" encoding="utf-8"?>
<ead xmlns="urn:isbn:1-931666-22-9" xmlns:xlink="http://www.w3.org/1999/xlink" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="urn:isbn:1-931666-22-9 http://www.loc.gov/ead/ead.xsd"><eadheader countryencoding="iso3166-1" dateencoding="iso8601" langencoding="iso639-2b"
repositoryencoding="iso15511"><eadid>MSS.0001</eadid><filedesc><titlestmt><titleproper>Morris Canal Company papers<num>MSS.0001</num></titleproper></
titlestmt><publicationstmt><publisher>test</publisher></publicationstmt></filedesc><profiledesc><creation>This finding aid was produced using ArchivesSpace on
<date>2020-07-06 15:15:41 -0400</date>.</creation></profiledesc></eadheader><archdesc level="collection">
  <did>
    <repository>
      <corpname>test</corpname>
    </repository>
    <unittitle>Morris Canal Company papers</unittitle>
    <unitid>MSS.0001</unitid>
    <physdesc altrender="whole">
      <extent altrender="materialtype spaceoccupied">2.5 Linear Feet</extent>
    </physdesc>
    <unitdate normal="1880/1920" type="inclusive">1880-1920</unitdate>
    <langmaterial>
      <language langcode="eng" scriptcode="Latn">English</language>
.     </langmaterial>
  </did>
  <bioghist id="aspace_6ae3d01a1957c137719c6bc95e175f51">
    <head>Biographical / Historical</head>
<p>On December 31, 1824, the New Jersey Legislature chartered the Morris Canal and Banking Company, a private corporation charged with the construction of the canal. The
corporation issued 20,000 shares of stock at $100 a share, providing $2 million of capital, divided evenly between funds for building the canal and funds for banking
privileges. The charter provided that New Jersey could take over the canal at the end of 99 years. In the event that the state did not take over the canal, the charter
would remain in effect for 50 years more, after which the canal would become the property of the state without cost.</p>  </bioghist>
```

Same data, different view

JSON record, through the API:



```
{
    "lock_version": 4,
    "title": "Morris Canal Company photographs",
    "publish": true,
    "restrictions": false,
    "ead_id": "PHT-008",
    "finding_aid_title": "Guide to the Morris Canal Company photographs",
    "finding_aid_date": "2019",
    "finding_aid_language": "<language langcode=\"eng\">English</language>",
    "created_by": "admin",
    "last_modified_by": "admin",
    "create_time": "2019-11-16T00:12:05Z",
    "system_mtime": "2019-11-18T15:46:39Z",
    "user_mtime": "2019-11-18T15:39:57Z",
    "suppressed": false,
    "is_slug_auto": false,
    "id_0": "PHT",
    "id_1": "008",
    "language": "eng",
    "level": "collection",
    "resource_type": "collection",
    "finding_aid_description_rules": "dacs",
    "finding_aid_status": "completed",
    "jsonmodel_type": "resource",
    "external_ids": [],
    "subjects": [
        {
            "ref": "/subjects/1"
        },
        {
            "ref": "/subjects/2"
        }
    ],
```

Basics

- What’s an API?
- Same data, different view
- How do I access it?
- Why use it?
  - Enables work at scale
  - Enables applications to communicate directly

How do I access it?

- We will be tackling this throughout the presentation
- But we'll start with some basics about access:
  - You access the API via a URL. Your IT department or hosting provider should have the address
    - http://sandbox.archivesspace.org/api/
  - Aspace comes with an API out of the box, but your IT department or hosting provider may have to enable it
  - Local installs (if you run a blank AS on your computer) have it on by default via localhost:8089
  - You log into the API via a local Aspace user account

Basics

- What’s an API?
- Same data, different view
- How do I access it?
- Why use it?
  - Enables work at scale
  - Enables applications to communicate directly

Basics

- What's an API?
- Same data, different view
- How do I access it?
- Why use it?
  - Enables work at scale
  - Enables applications to communicate directly

Enabling work at scale

- For the first time ever I’m going to pause on these words
- Scale isn’t in question; the API lets you work on any scale
- But what work do I mean? What work do YOU mean?
- You registered for this workshop because you know the API is a thing
- But thinking about what you will ultimately accomplish is important as you start this journey

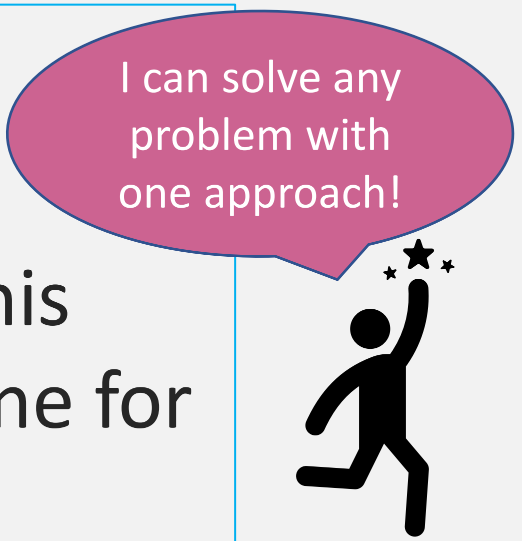
Enabling work at scale

- This presentation has always been based on my real life
- That life used to be primarily data cleanup
    - Changing existing data by improving migrated and legacy data

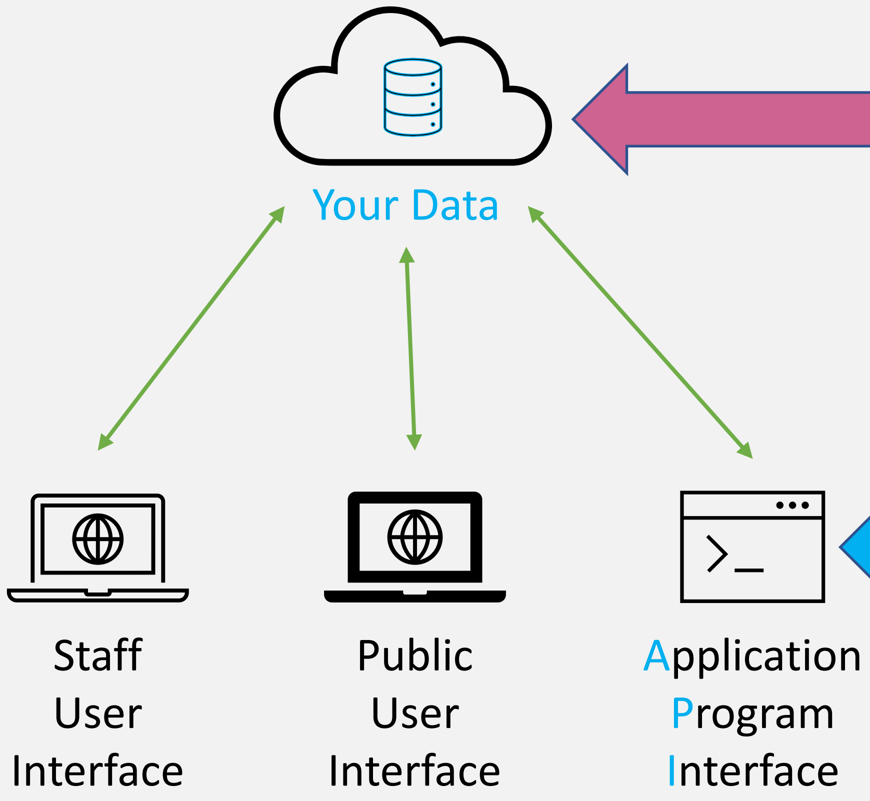
- I also do data ingest
    - Creating new records and new links with new information (or old information used in new ways)

Enables work at scale

Enabling work at scale

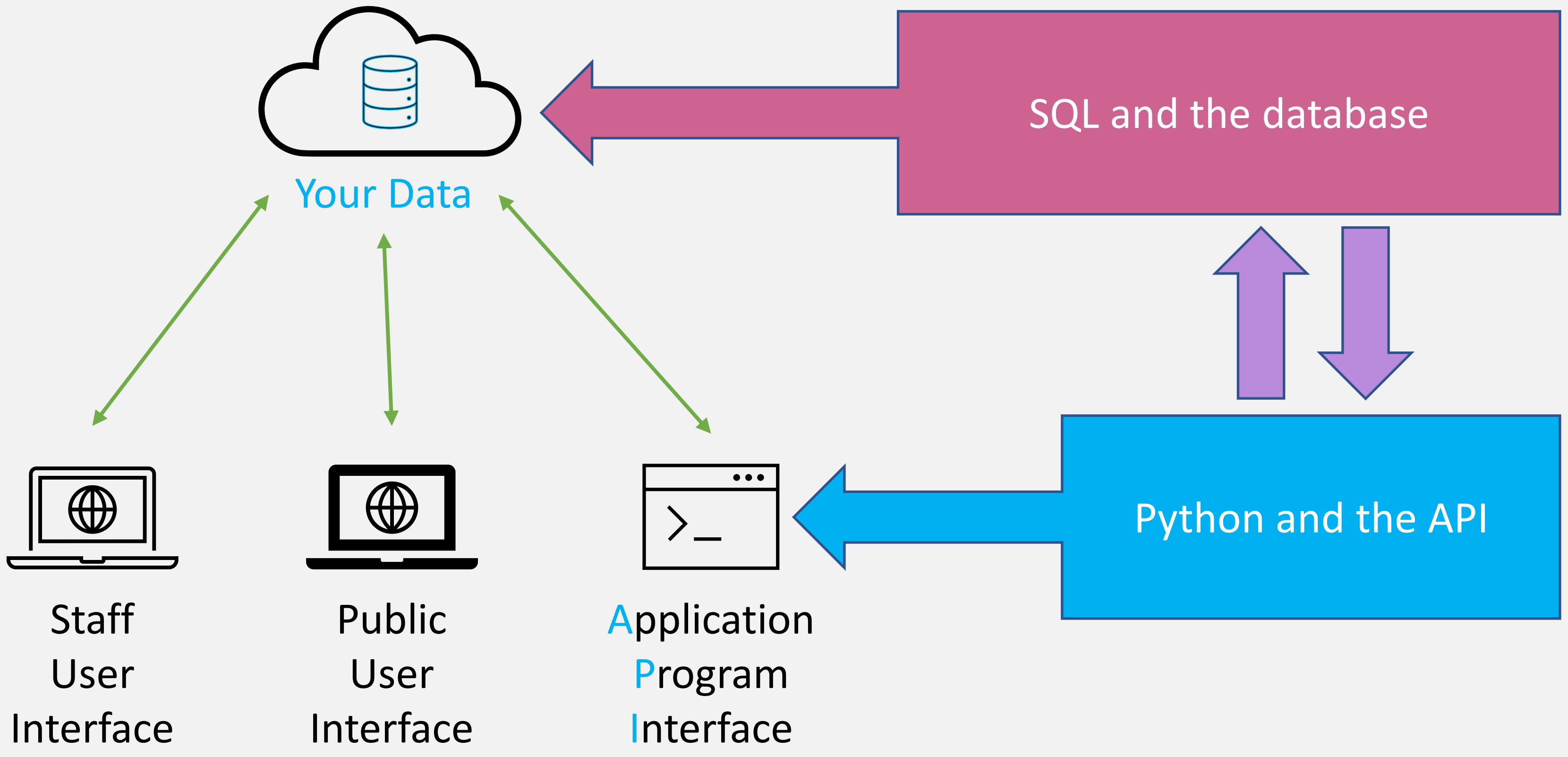


- I mention my work because that’s how I designed this workshop and how I approached everything I’ve done for the last four years. There’s assumptions in this.
- When I started teaching the API, it was how I tackled all my problems. One hammer for all tasks.
- Four years later, I now choose between two hammers: Python + API and SQL + database



Now I work directly in the database >50% of the time

I used to work here exclusively



Now I deploy:

- Python + API
- SQL + database
- Python + (SQL + database) + API

Enables work at scale

The API enables different types of "work at scale"

- Just browse the ArchivesSpace Awesome List
- Noah Huffman and Tracy Jackson (Duke University) use the API to create Trello cards for project management
- Corey Schmidt (University of Georgia) uses the API for batch exporting
- Kevin Schlottmann and David W. Hodges (Columbia University) use two APIs for their reporting and updating

Enables work at scale

What do you want to do at scale?

Do you want to change data?

Delete?

Create?

Connect?

Report?

Import?

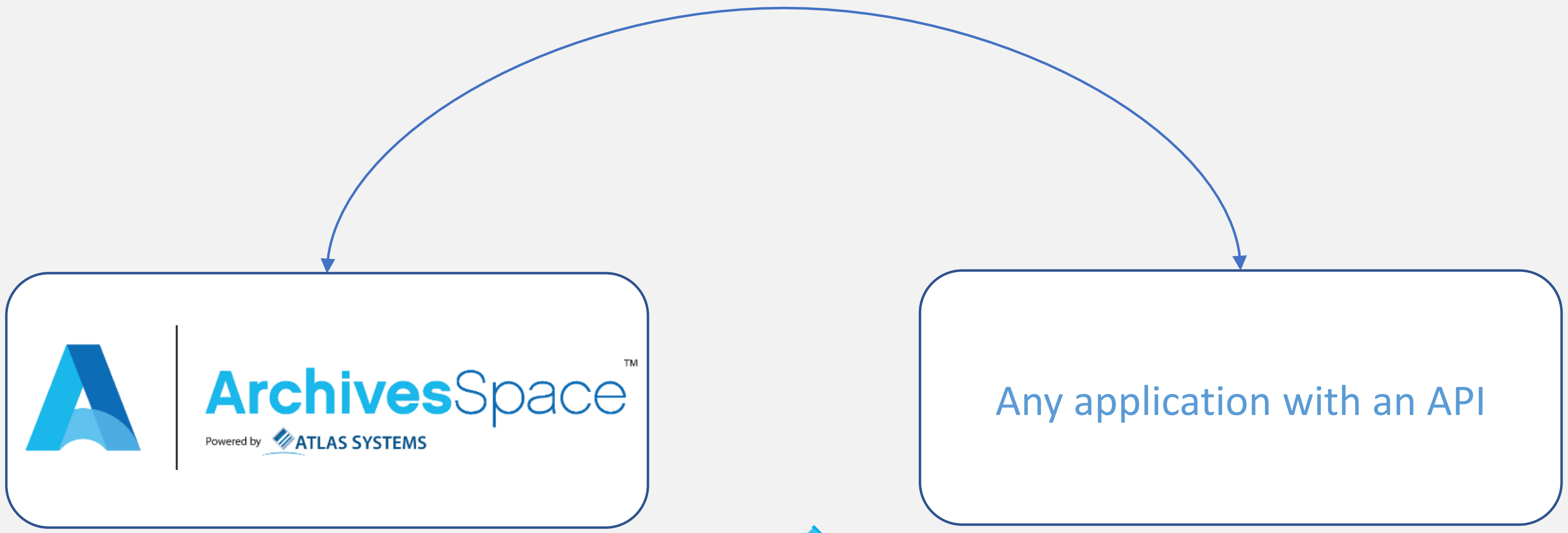
Export?

You might now know yet. That’s totally fine.

Basics

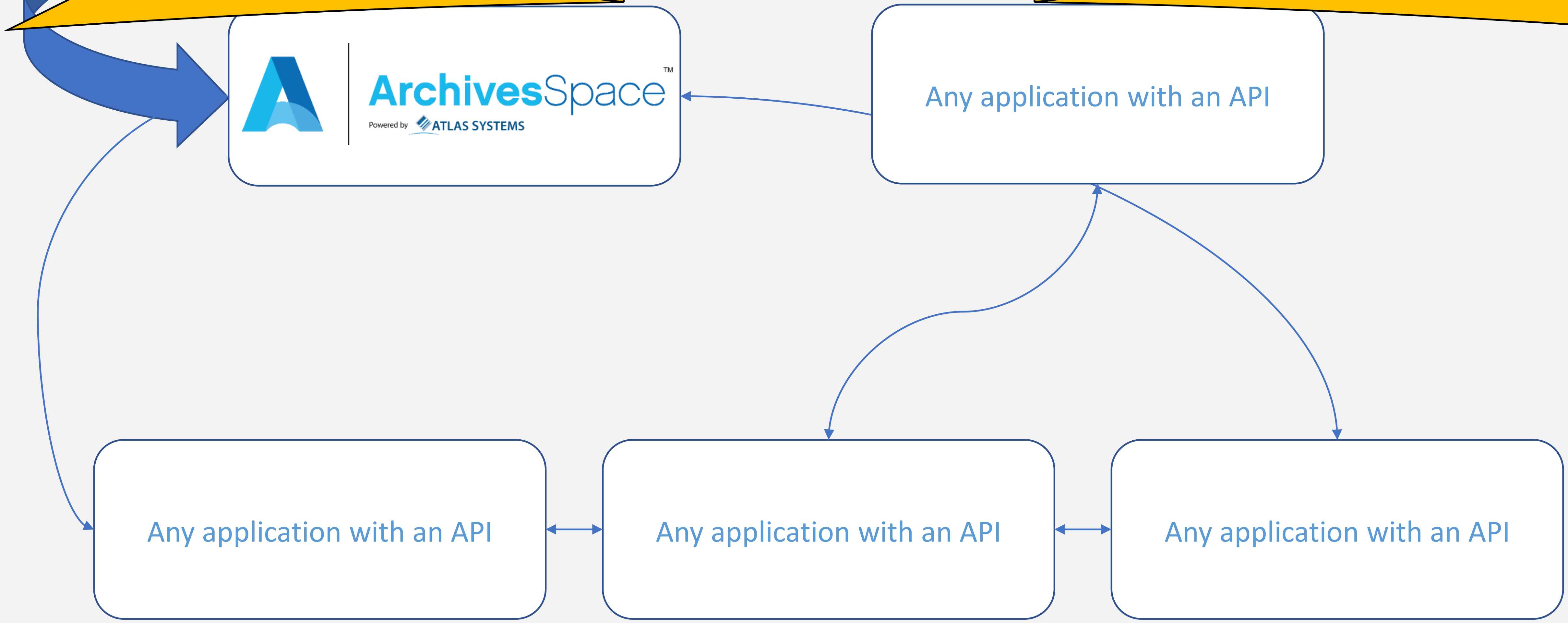
- What's an API?
- Same data, different view
- How do I access it?
- Why use it?
  - Enables work at scale
  - Enables applications to communicate directly

This is the true function of APIs, we're just capitalizing



This is already happening with the LCNAF plugin in ASpace

The Dream!

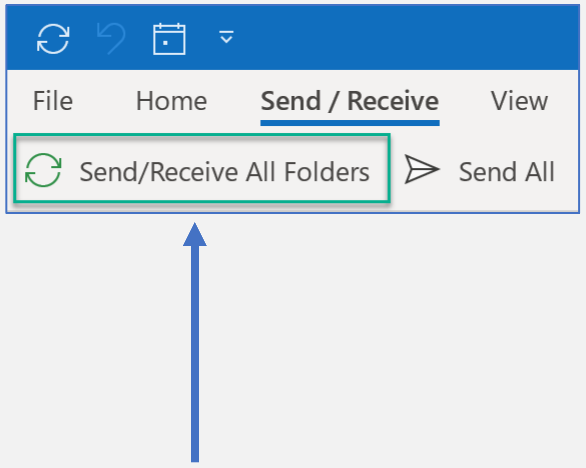

ArchivesSpace
Powered by ATLAS SYSTEMS
Any application with an API
Any application with an API
Any application with an API
Any application with an API

Everything is a Conversation

The Request-Response Cycle

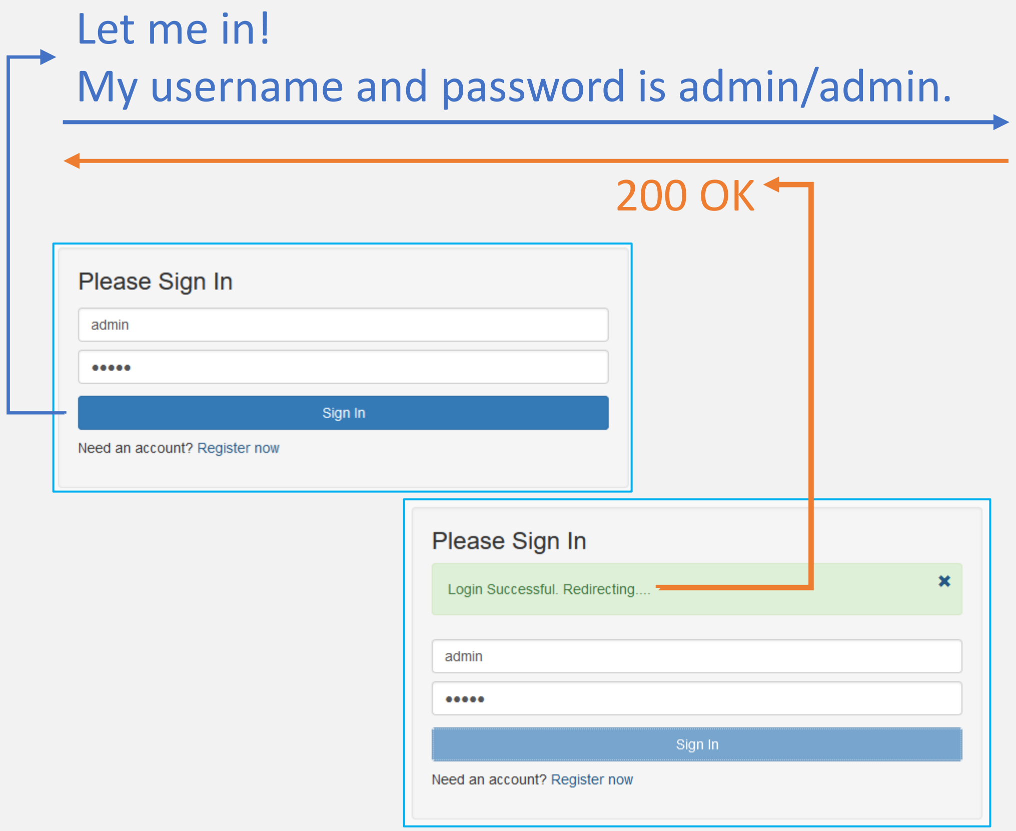
The Request-Response Cycle

Where are you?
At home like everybody else.
Did you get that thing I sent you?



Click button, make stuff happen!

The Request-Response Cycle



Please Sign In

admin

•••••

Sign In

Need an account? Register now

Please Sign In

Login Successful. Redirecting....

admin

•••••

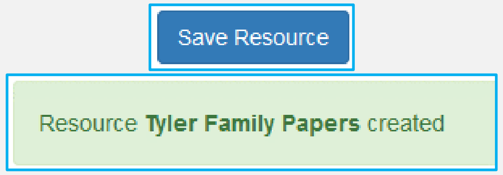
Sign In

Need an account? Register now

The Request-Response Cycle

Let me in!
My username and password is admin/admin.

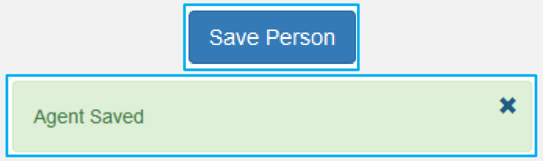
Okay, I'll let you in.


Save Resource

Resource **Tyler Family Papers** created

Create this new Resource record for me.

Okay, created.


Save Person

Agent Saved ✖

Save this edited Agent record for me.

Okay, saved.

Let me delete this.

Nope.

Your permissions persist.

Everything is a conversation

That's so deep.

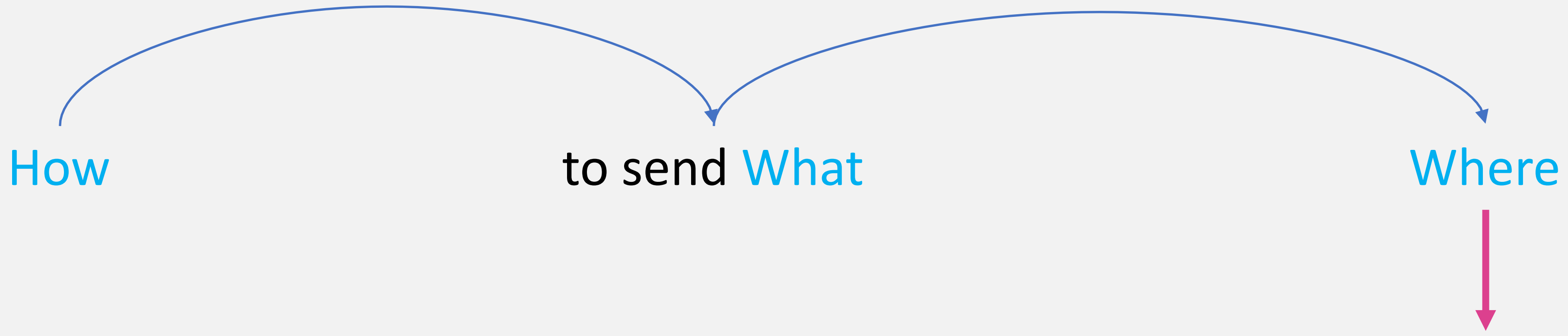
The Request-Response Cycle

- You don’t have to think about any of this when you’re in the interface
- But now I want you to realize that you are participating in a conversation with ASpace
- Cultivate request/response mindfulness

The Request-Response Cycle

- Now that you know you’re doing it, we will learn how to do it through the API
- To converse with the API, you need to know how to send what to where

The Request-Response Cycle

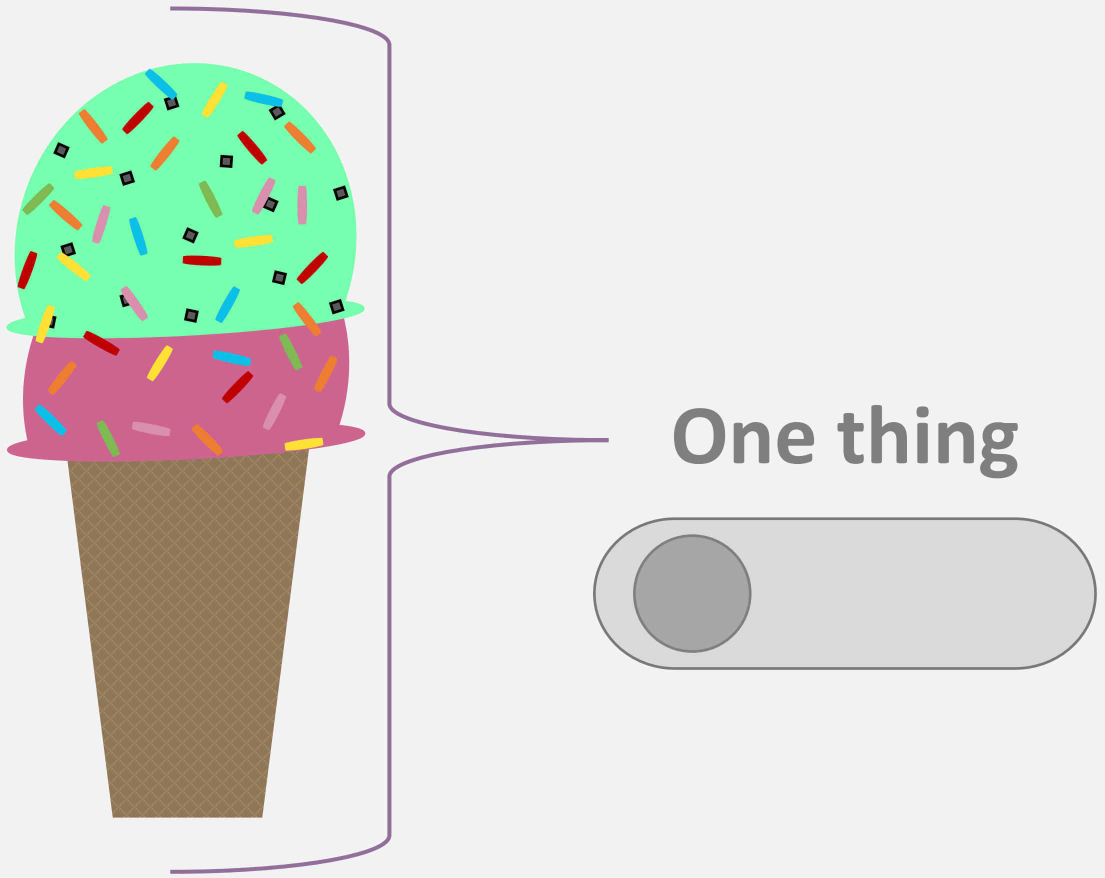


The where part is what we’re going to tackle next

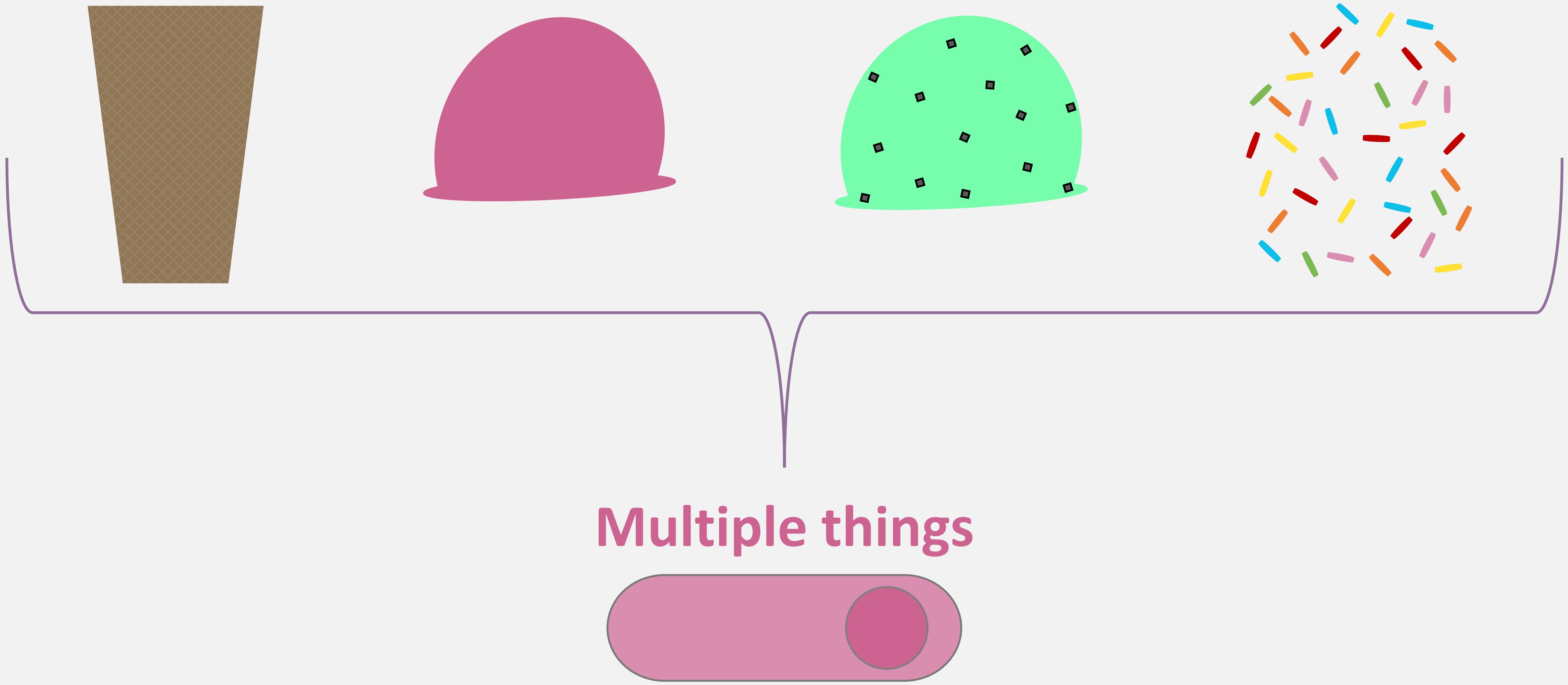
Flipping the Ice Cream Switch

Reframe how you think about archival data

Atomization



Atomization



Atomization

```
    The Morris Canal
  Company Photographs

Series I: Prints

Box 1:

Folder 1............ .....1907
Folder 2.......... ........1908
Folder 3............... . . .1909
Folder 4............ . .....1910
Folder 5............ . .....1911
Folder 6.......................1912
Folder 7............... . ......1913
```

“a finding aid”

```
<archdesc level="collection">
 <did>
  <repository>
   <corpname>test</corpname>
  </repository>
  <unittitle>Morris Canal Company papers</unittitle>
  <unitid>MSS.0001</unitid>
  <physdesc altrender="whole">
   <extent altrender="materialtype spaceoccupied">2.5 Linear Feet</
extent>
  </physdesc>
  <unitdate normal="1880/1920" type="inclusive">1880-
1920</unitdate>
  <langmaterial>
   <language langcode="eng" scriptcode="Latn">English</language>
  </langmaterial>
 </did>
```

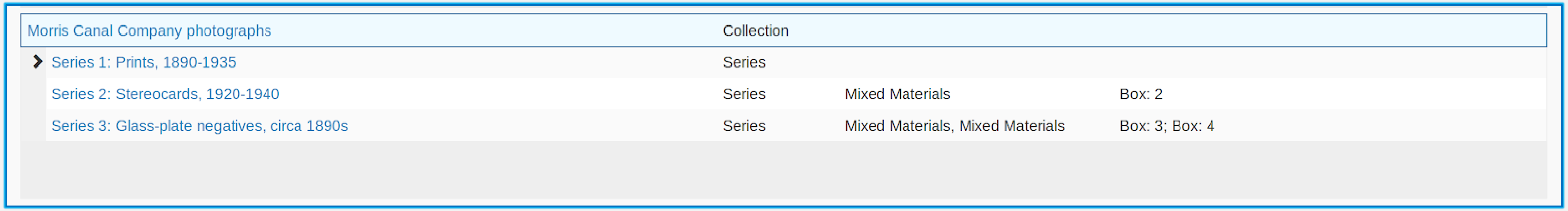
Paper

<EAD>

One thing

Atomization

“a finding aid”



Morris Canal Company photographs	Collection		
Series 1: Prints, 1890-1935	Series		
Series 2: Stereocards, 1920-1940	Series	Mixed Materials	Box: 2
Series 3: Glass-plate negatives, circa 1890s	Series	Mixed Materials, Mixed Materials	Box: 3; Box: 4

One thing

Atomization

“a finding aid”

1 resource record

7 subjects

15 digital objects

3 agents

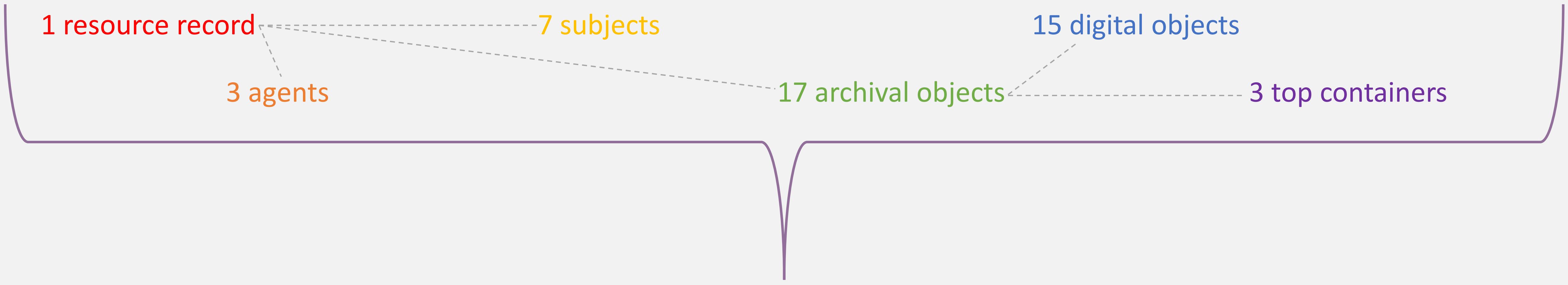
17 archival objects

3 top containers

Multiple things

Atomization

“a finding aid”



Multiple things

Atomization

Evolution of the Finding Aid

```
     The Morris Canal
   Company Photographs

Series I: Prints

Box 1:

Folder 1………….….1907
Folder 2……….…….1908
Folder 3……………...1909
Folder 4………….….1910
Folder 5………….….1911
Folder 6……………………1912
Folder 7…………….……1913
```

```
<archdesc level="collection">
 <did>
  <repository>
   <corpname>test</corpname>
  </repository>
  <unittitle>Morris Canal Company papers</unittitle>
  <unitid>MSS.0001</unitid>
  <physdesc altrender="whole">
   <extent altrender="materialtype spaceoccupied">2.5 Linear Feet</
extent>
  </physdesc>
  <unitdate normal="1880/1920" type="inclusive">1880-
1920</unitdate>
  <langmaterial>
   <language langcode="eng" scriptcode="Latn">English</language>
  </langmaterial>
 </did>
```

1 resource record

3 agents

7 subjects

17 archival objects

15 digital objects

3 top containers

Typed/printed
Bound
On a shelf

Encoded
Rendered as PDF or HTML
On the web

Atomized and stored in tables in a SQL database
Links between records maintain their context
Display and export options are limitless

The Request-Response Cycle

$$\begin{tikzcd}\text{How} \arrow[r, bend left] & \text{to send What} \arrow[r, bend left] & \text{Where}\end{tikzcd}$$

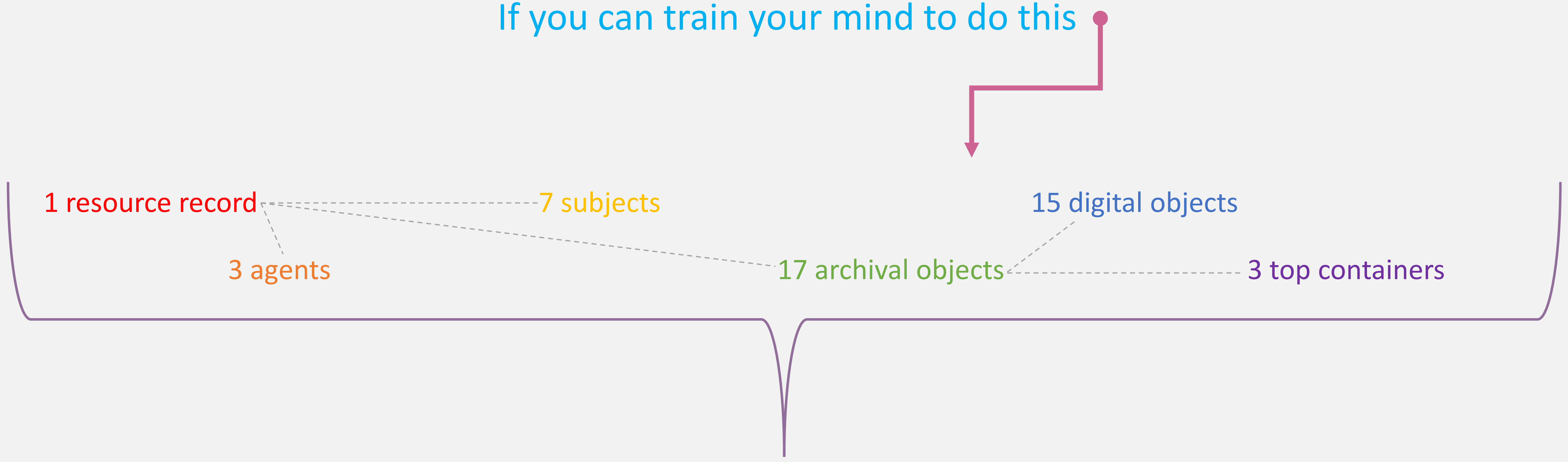
Flipping this switch is a precursor to the where

Endpoints

Unique URLs that represent an object or collection of objects

Endpoints

If you can train your mind to do this



Multiple things

Endpoints

Then your next step is this

/resources/1

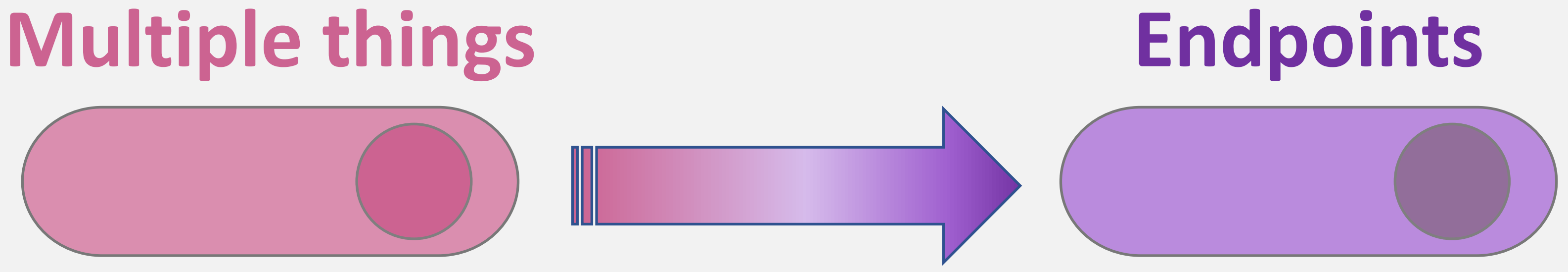
/agents/1
/agents/2
/agents/3

/subjects/1
/subjects/2
/subjects/3
/subjects/4
/subjects/5

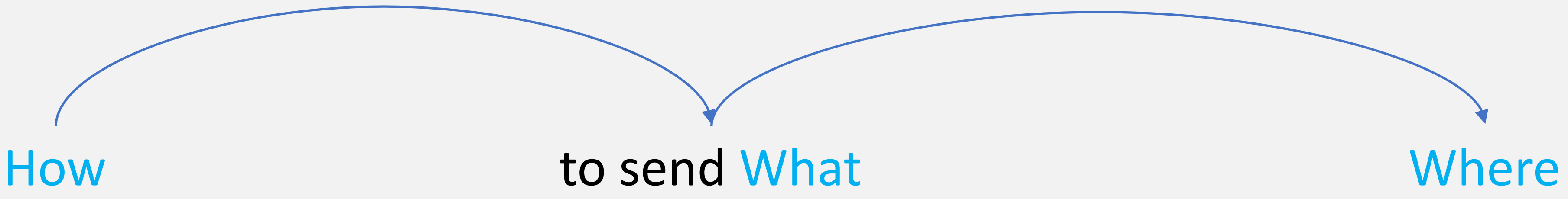
/archival_objects/1
/archival_objects/2
/archival_objects/3
/archival_objects/4
/archival_objects/5
/archival_objects/6
/archival_objects/7

/digital_objects/1
/digital_objects/2
/digital_objects/3
/digital_objects/4
/digital_objects/5

/top_container/1
/top_container/2
/top_container/3



Endpoints



/archival_objects/1
/archival_objects/2
/archival_objects/3
/archival_objects/4
/archival_objects/5
/archival_objects/6
/archival_objects/7

Endpoints are the where

Endpoints

Unique URLs that represent an object or collection of objects

http://localhost:8089/repositories/*:repo_id*/resources/*:id*

http://localhost:8089 /repositories/2 /resources/101

Protocol | Host (domain name) | Port

Repository inside ASpace

Resource inside that repository

In my institution's instance of ASpace

Inside the 2nd repository

The 101st resource

Endpoints

Unique URLs that represent an object or collection of objects

http://localhost:8089/repositories/*:repo_id*/resources/*:id*

http://localhost:8089/repositories/*:repo_id*/accessions/*:id*

http://localhost:8089/repositories/*:repo_id*/top_containers/*:id*

http://localhost:8089/agents/people

http://localhost:8089/locations

http://localhost:8089/subjects

Endpoints

Unique URLs that represent an object or collection of objects

http://localhost:8089/repositories/*:repo_id*/resources?*all_ids=true*

http://localhost:8089/repositories/*:repo_id*/resources/:id/tree

http://localhost:8089/extent_calculator

http://localhost:8089/config/enumeration_values

http://localhost:8089 */:repo_id* /jobs

Endpoints

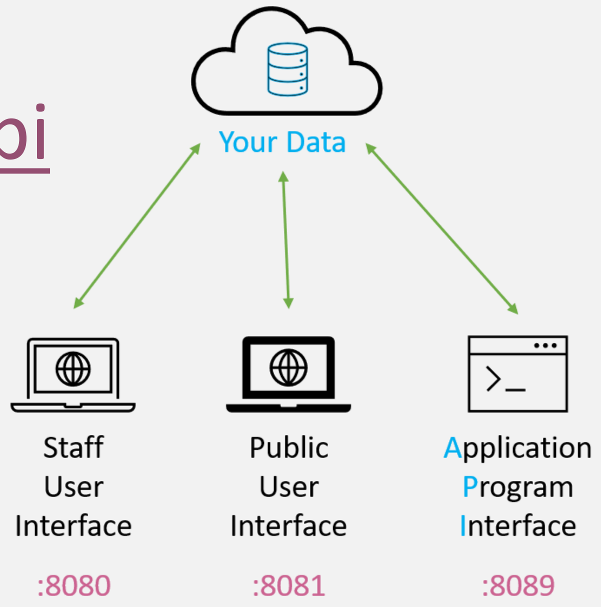
Pro-tip on a convention:

In documentation you will see the API referred to as localhost:8089

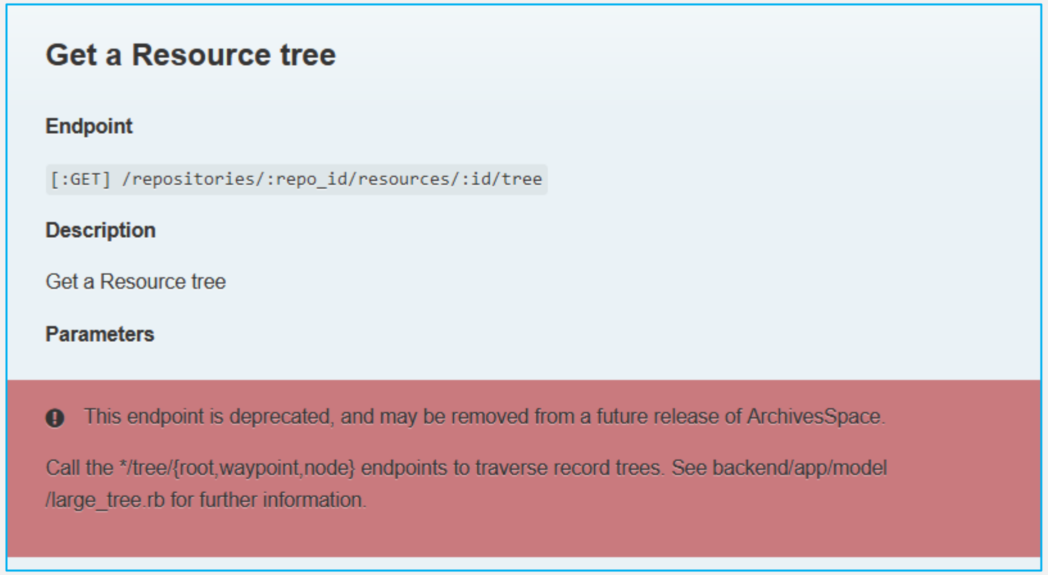
This is a placeholder for “insert your API URL here”

Here’s a real one: http://sandbox.archivesspace.org/api

What happens when I navigate there?



Endpoints


Get a Resource tree

Endpoint

`[:GET] /repositories/:repo_id/resources/:id/tree`

Description

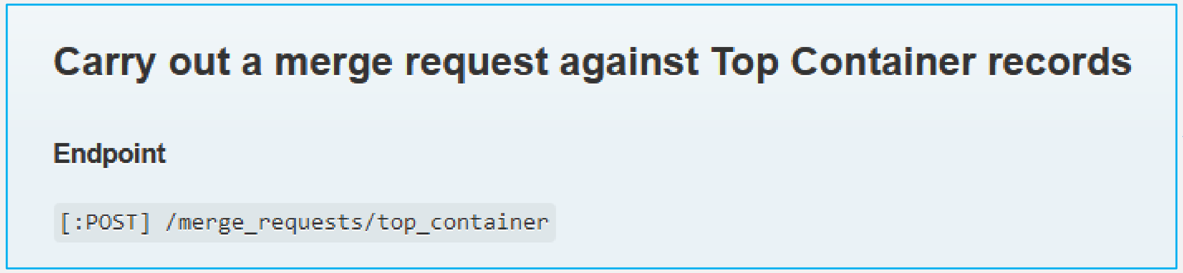
Get a Resource tree

Parameters

This endpoint is deprecated, and may be removed from a future release of ArchivesSpace.

Call the */tree/{root,waypoint,node} endpoints to traverse record trees. See backend/app/model /large_tree.rb for further information.

- Endpoints change!
- Mainly they get added
- But remember that every update to AS might bring changes to endpoints


Carry out a merge request against Top Container records

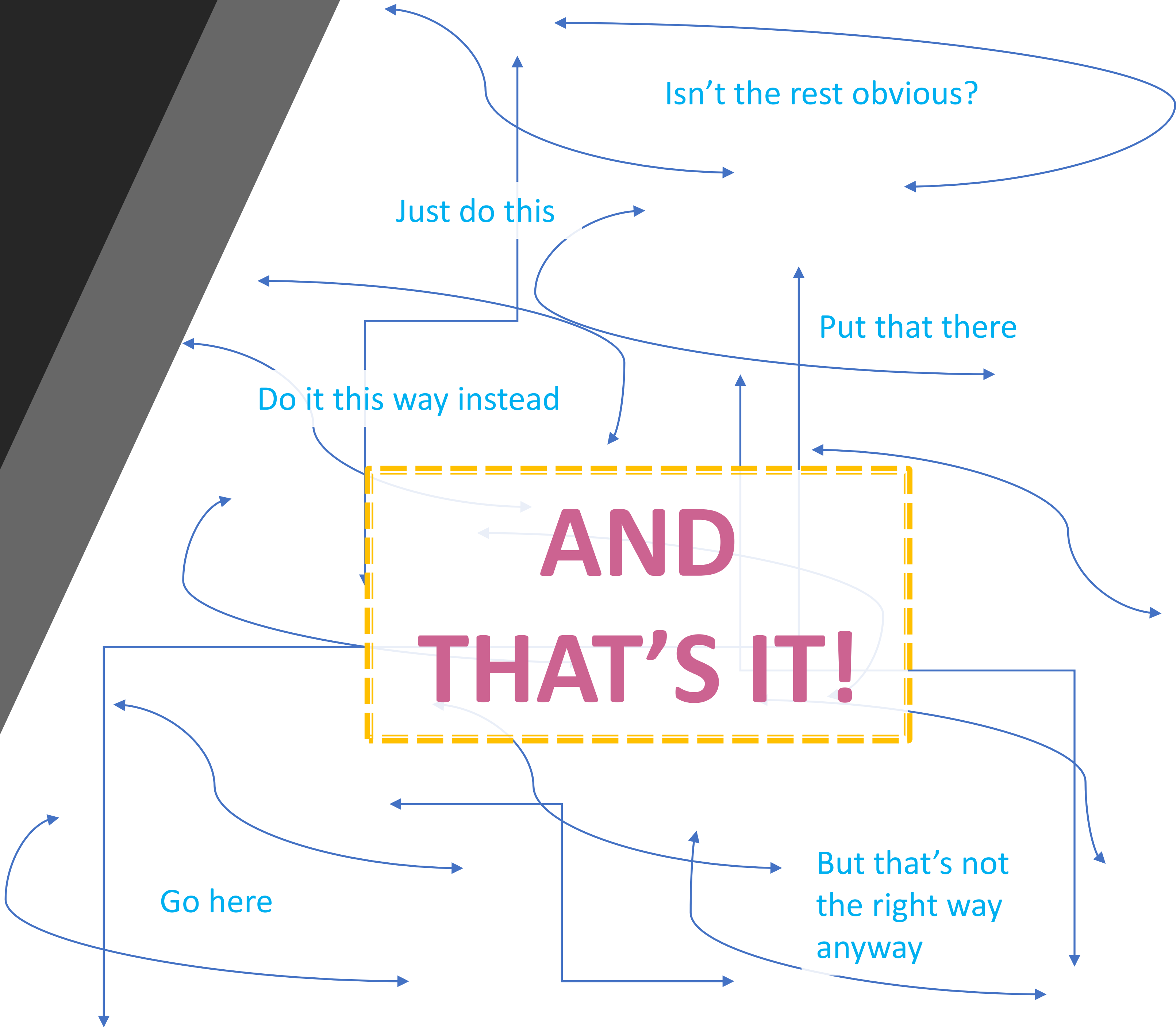
Endpoint

`[:POST] /merge_requests/top_container`

This endpoint only applies to 2.8.0+
But the documentation does not tell you that

Demos!

We're be covering ~~too much~~ a lot



Demonstrations

How → to get What → from Where

We're going to start with this

Any API client (Postman)

Scripts (Python 3)

JSON

Which is going to get us some of this

Watch these as we go

Endpoints

(we'll come back to the presentation to discuss JSON)

Demonstrations

How

Any API client (Postman)

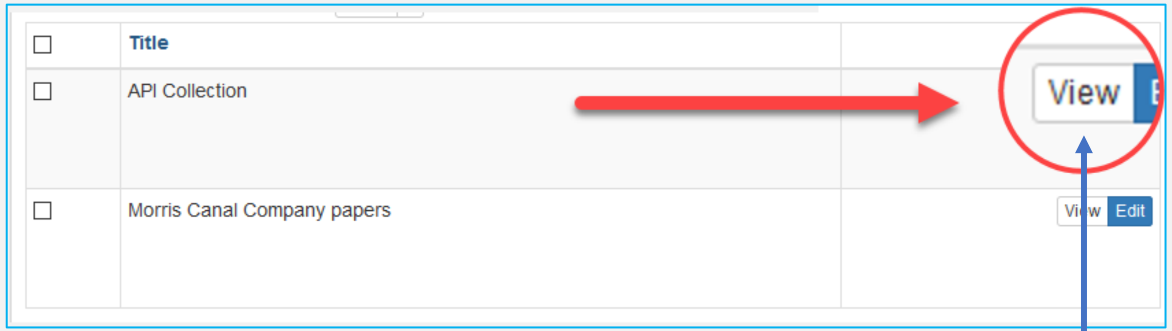
- GUI interface for working with APIs
- It's more powerful than I realize
- I use it for simple calls
- **Great for getting started**

Scripts (Python 3)

- The only practical way
- **Huge** barrier to entry for archivists
- Python3 near-universal in the AS community
- I'm using Jupyter Notebooks to show you Python scripts today

Demonstrations

How



	Title	
☐	API Collection	View
☐	Morris Canal Company papers	View Edit

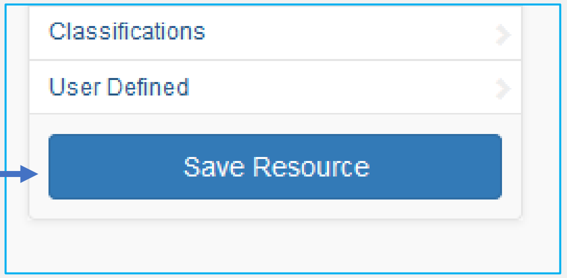
Whichever you use

Any API client (Postman)
Scripts (Python 3)

You need these

Basic commands:

- GET
- POST
- DELETE



Demonstration

What we've covered so far

- The API shows you the same data, different view
- That different view is available via a different route (endpoint)
- Working with the API is a conversation
- The first conversation is always authentication

This demo

- Introduce Postman, an API client
- Mirror two experiences:
  - Log in/authenticate
  - GET a record
  - Edit and POST a record back to ASpace

Javascript Object Notation

JSON

This is not scary.

Demonstrations

How → to get What → from Where

We're going to start with this

Any API client (Postman)

Scripts (Python 3)

JSON

Which is going to get us some of this

Watch these as we go

Endpoints

JSON

- Data interchange format
  - How a lot of APIs talk to each other
- Human-readable
- Key-value pairs
  - “Key”: “Value”
- Has arrays

JSON

<XML>

```
<unittitle>Morris Canal Company papers</unittitle>

<element></element>
```

↓

{JSON}

```
“Title”: “Morris Canal Company papers”

“key”: “value”
```

JSON

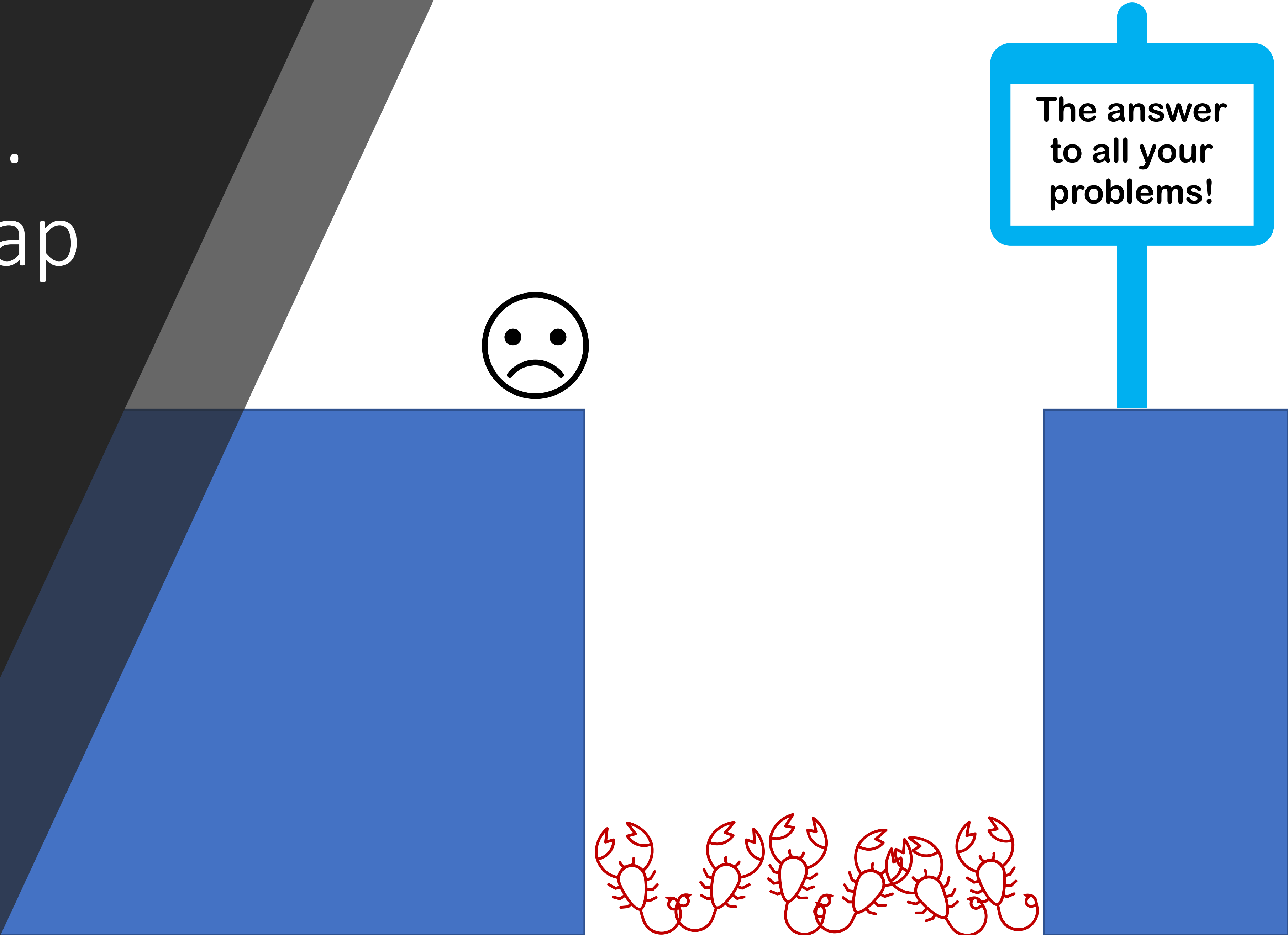
<XML>

```
<archdesc>
        <bioghist><p>The Morris Canal Banking Company was founded in 1824…</p></bioghist>
        <scopecontent><p>This collection consists of…</p></scopecontent>
</archdesc>
```

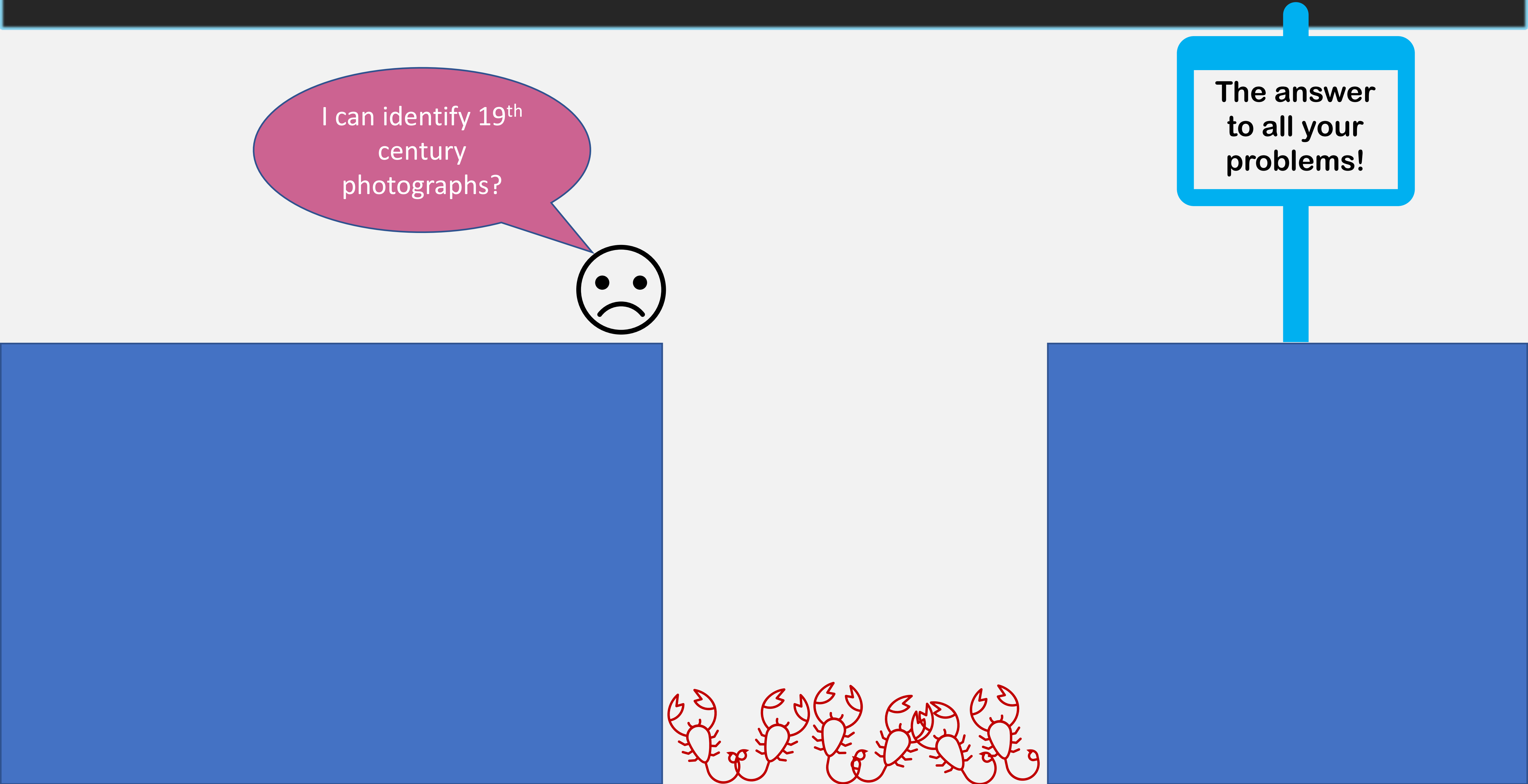
{JSON}

```
"Notes": [
        {
                "type": "bioghist",
                "content": "The Morris Canal Banking Company was founded in 1824…"
        },
        {
                "type": "scopecontent",
                "content": "This collection consists of…"
        }
]
```

And now….
The Big Leap



Big Leap


I can identify 19th century photographs?
The answer to all your problems!

Big Leap

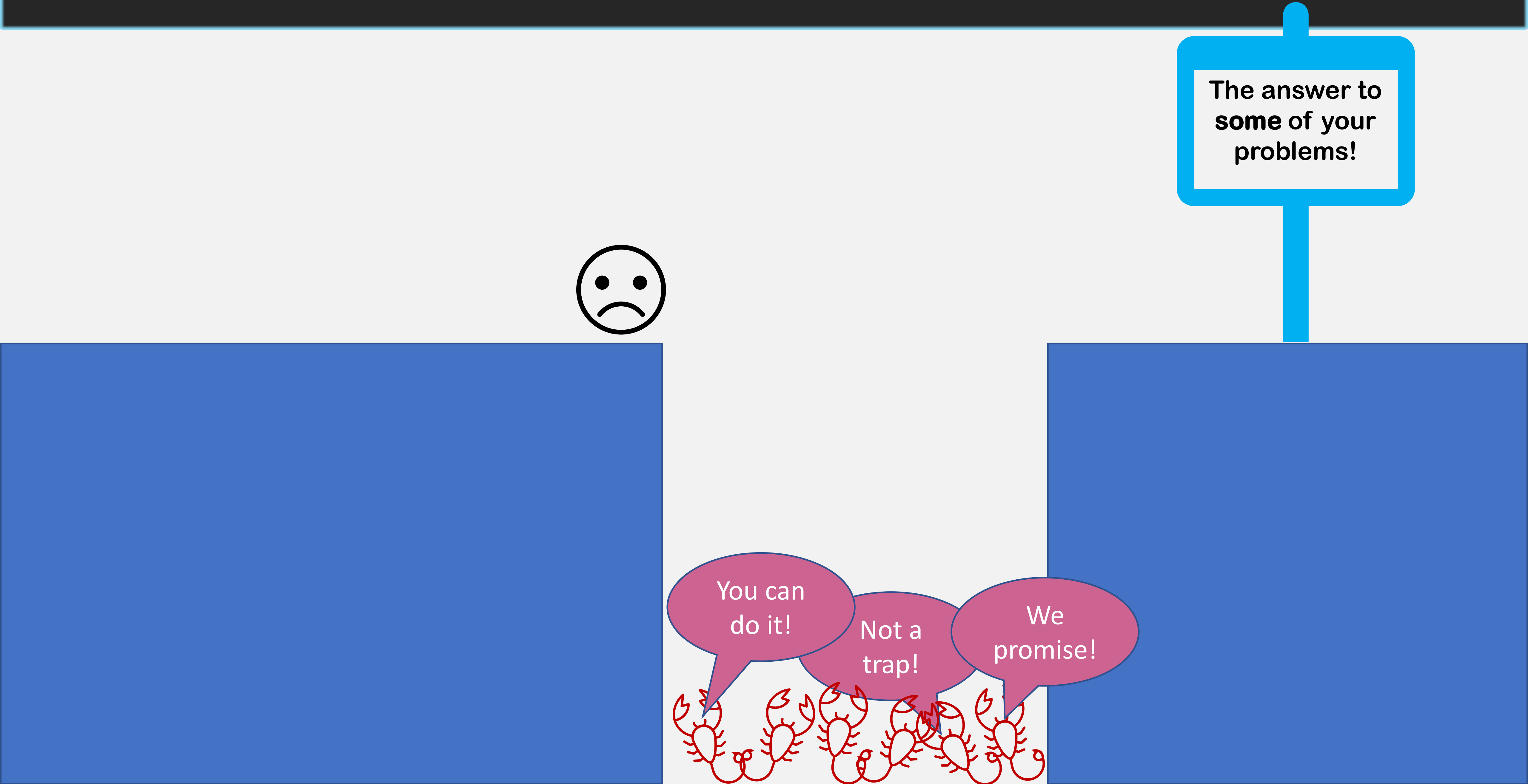
What I can't do today

- Scripting fundamentals
- Python fundamentals
- Set up a Dev environment

What I can do today

- Focus on Aspace-specific:
  - Authentication
  - Linking
  - Example-walkthrough
- Suggest your exact next steps
  - Give you my Playbook
  - Give you some scripts

Big Leap


The answer to **some** of your problems!
You can do it!
Not a trap!
We promise!

Scripting

Your biggest hurdle.

Scripting

You may have watched this presentation already knowing that…

Scripting is the only practical way to use the API

This stopped me in my tracks for about three years.

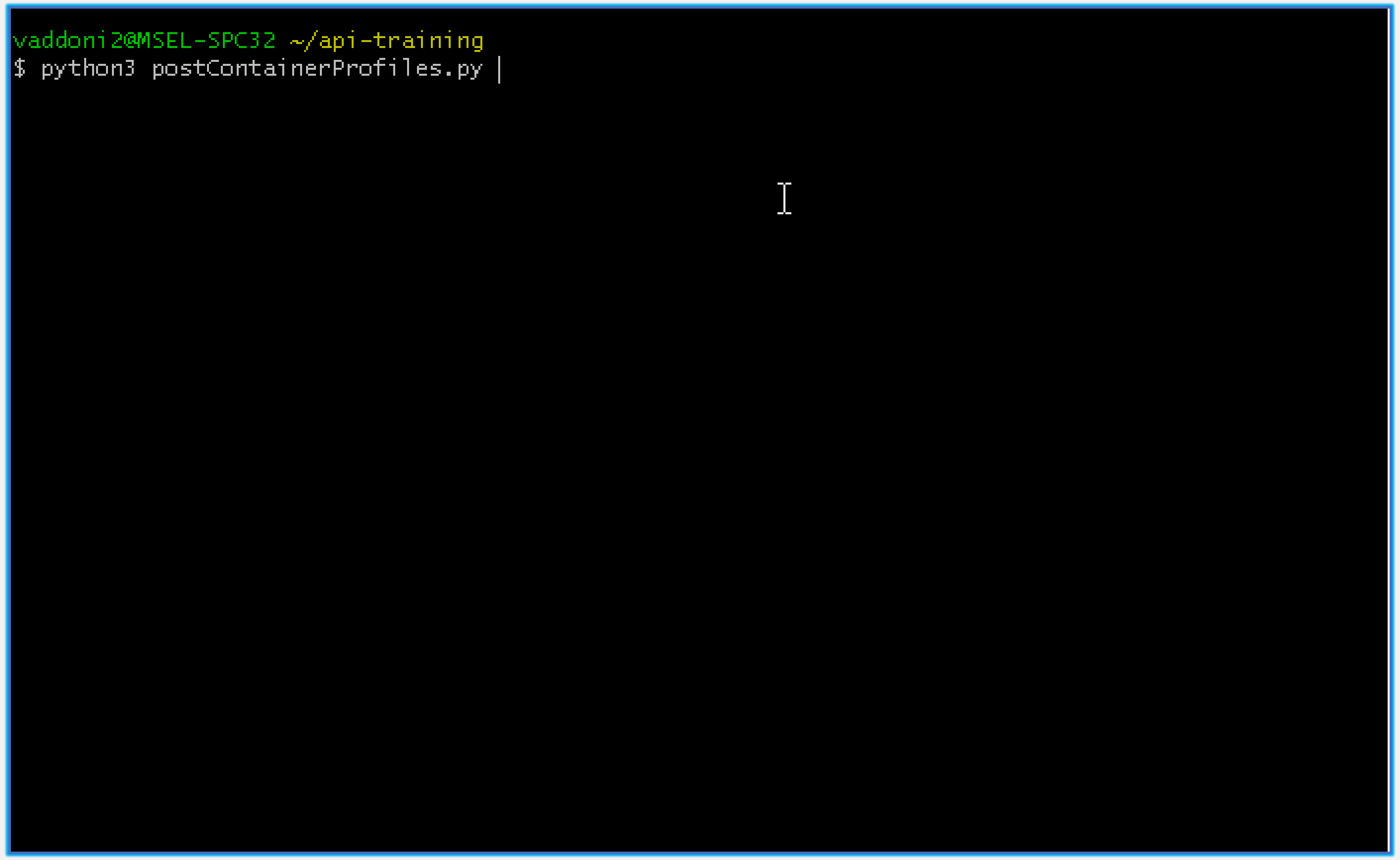
Scripting

Why do you need scripts?

- Because you can only do one thing at a time either in the interface or in the API
  - You can only hit one endpoint at a time
- We make computers do what we do, just faster
- A script can still only do one thing at a time…. But really fast.

Scripting

How fast you say?



```
vaddoni2@MSEL-SPC32 ~/api-training
$ python3 postContainerProfiles.py |
```

Scripting

Why do you need scripts?

- So the API *itself* isn't a magical thing that makes all these awesome API projects ~~*happen*~~
- It's the scripts that make things happen, and the API just makes using scripts possible
- So in the end, we all need an introduction to scripting *and* an introduction to the API

Demonstrations

How

to get What

from Where

Now we're using Python

Any API client (Postman)

Scripts (Python 3)

JSON

Which is STILL going to get us some of this

And we're still watching these as we go

Endpoints

Scripting

Scripts do what you do, just *much faster*

- Since my work focuses a lot on data cleanup and data mapping, that’s what these demos lean toward
- If you’re looking to create system integrations, I have no direct experience with that, but can probably talk my way through it

Demonstration

What we've covered so far

- The API shows you the same data, different view
- That different view is available via a different route (endpoint)
- Working with the API is a conversation
- The first conversation is always authentication

This demo

- Introduce Jupyter notebooks for demonstrating scripts
- Mirror our Postman experience:
  - Log in/authenticate
  - GET a record
  - Edit and POST a record back to ASpace

Pro-Tips

A few important reminders and FYIs

Pro-Tip #1: There are Pusheen icons in PowerPoint!!

Pro-Tips for the ASpace API

Session time is the same

- If you get logged out after an hour in the staff interface, same for the API
- You can change this in the config
- Re-auth might be handled by certain libraries

You need a *local ASpace account* to access the API

- You cannot authenticate to the API using your institution’s authentication

Pro-Tips for the ASpace API

Remember that your permissions still matter

- If you can’t do it in the interface, using the endpoint for the same action will be no different
- The API documentation will not mention permissions
- [:GET] /update-feed

Pro-Tips for the ASpace API

Know your link directions

- Record linking is not bidirectional
- Test your linking assumptions before finalizing your project

Be careful of overwrites

- When posting content back to a pre-existing record, you must post everything
- Anything left out will be overwritten as blank

Some fields don’t appear if empty

- Example: If you have a top container without a barcode, there won’t be an empty barcode field in the JSON
- There just won’t be a barcode field at all
- This is another reason to start your test in the interface

Pro-Tips for the ASpace API

Never test against Production

- Do whatever it takes to NOT work directly in Production until you are ready

Always make backups

- Ask IT or your hosting provider for a backup immediately before undertaking an API project
- Alert your co-workers

Wait for the indexer

- If you make thousands of changes in a short time, AS will need to catch up
- Wait a few minutes/an hour or more if you don't see immediate results when you were sure you should
- Use Edit mode in the Staff interface for these kinds of checks (not View mode and not the PUI)

The API isn’t your only solution

Python through the API

Get all archival objects and then get each AO and check the title and if it doesn’t have a comma, continue the script but if it does have a comma trim the comma and replace the value and then post the entire record back in and check the status of the post and then move onto to the next AO.

SQL against the DB

```
UPDATE `archival_objects`
SET `title` = TRIM(',' FROM `title`)
WHERE `title` regexp ',$’;
```

The API isn’t your only solution

You probably want the API

- If you’re creating new data or links
- If you’re changing data and the change relies on archival context/ the hierarchy
  - “Add an access restrict note to any child of any child that is marked as restricted”
  - Flag any top container linked to an archival object at the series level

You can probably use SQL in the db

- If your change does not require the hierarchy
- If you want simple, custom reports
- For simple changes. What’s simple? If it only takes a few nouns and one verb to describe it:
  - “Unpublish all digital objects”
  - "Remove all trailing commas"
  - "Find all ampersands"

Now what?

This was great and all, but…

The API Playbook

After this presentation is over, you will receive my API Playbook

It is my 25-ish page recipe for getting started

It is based on my experience

Sometimes the playbook is a step-by-step guide

Sometimes it's a boat-ton of YouTube links

It's the order of events and the specificity of the playbook that makes it relevant to you

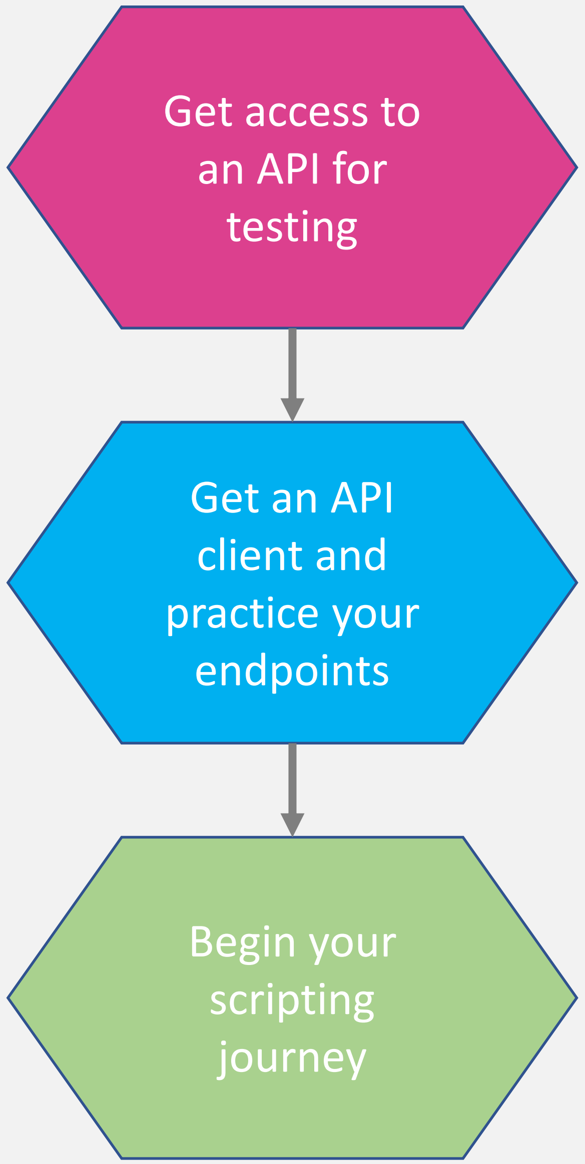
I sincerely think that following this recipe will start you on this journey

I would hand this to my 2016 self

There's NO WAY
I can do this

Go through the Stargate
and deliver this to your
younger self.

The API Playbook


Get access to an API for testing
Get an API client and practice your endpoints
Begin your scripting journey

After this presentation is over, you will receive my API Playbook

It is my 25-ish page recipe for getting started

It is based on my experience

Sometimes the playbook is a step-by-step guide

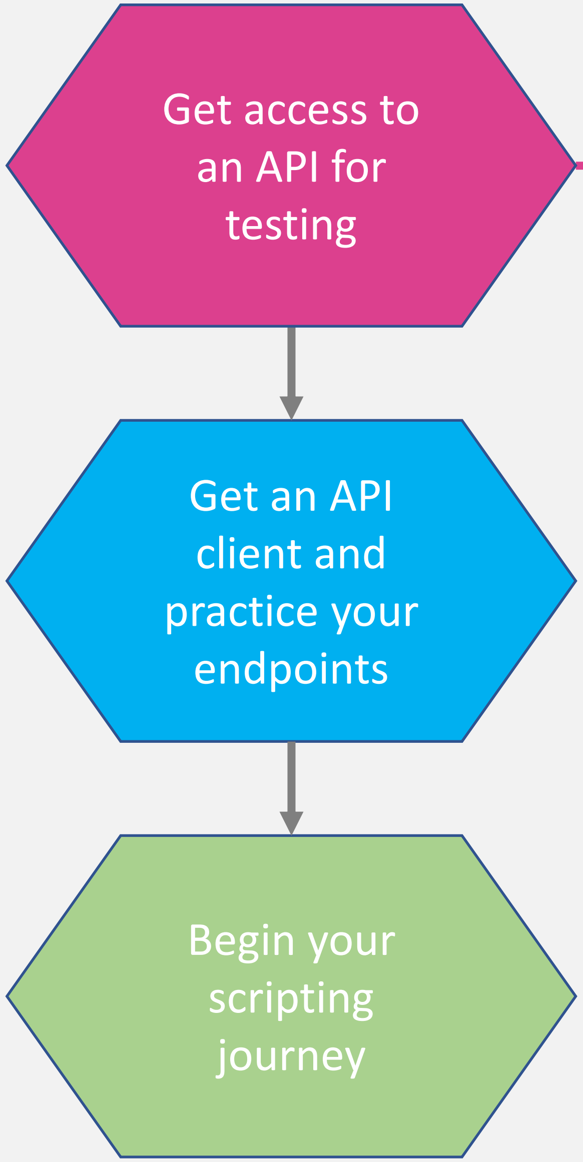
Sometimes it's a boat-ton of YouTube links

It's the order of events and the specificity of the playbook that makes it relevant to you

I sincerely think that following this recipe will start you on this journey

I would hand this to my 2016 self

The API Playbook

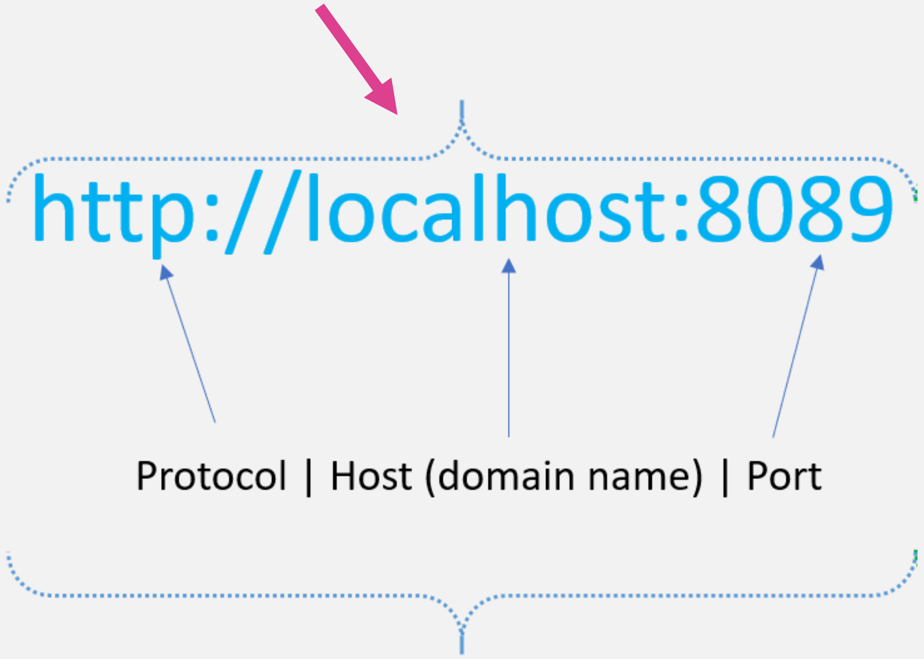

Get access to an API for testing
Get an API client and practice your endpoints
Begin your scripting journey

1. Use a copy of your Production data. Ask the following to your IT department or hosting provider:

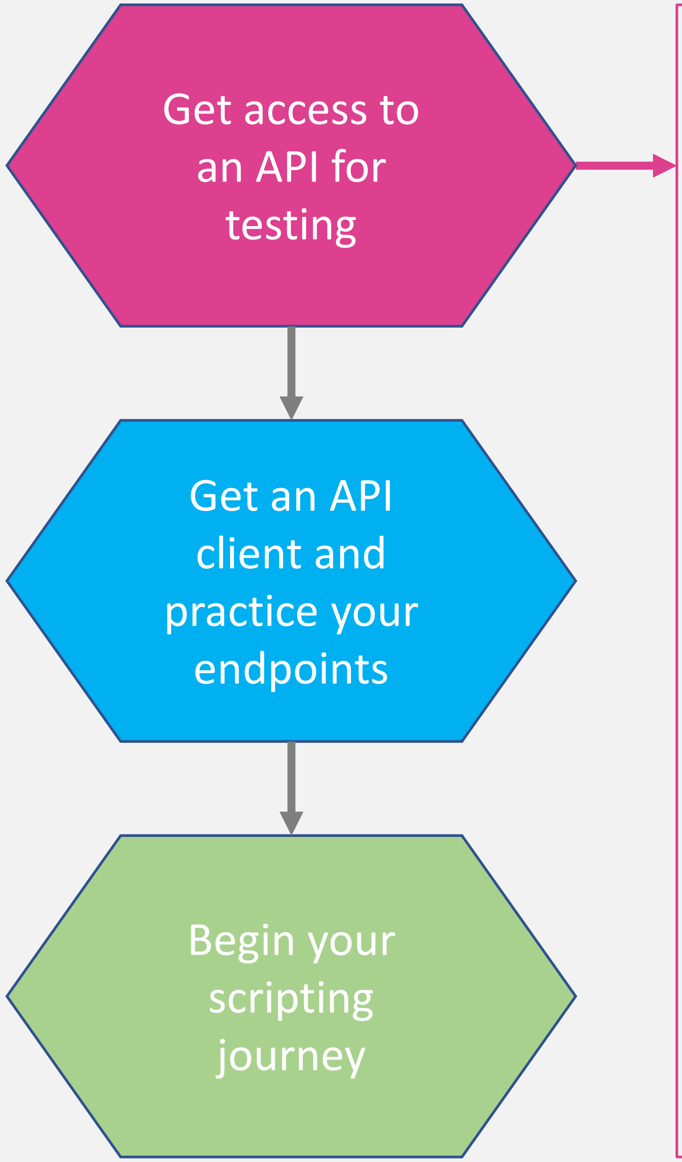
- ➢ Do/can we have a sandbox?

  i.e. a separate copy of your Production data that you can play in

  - Is the API open?
  - What's the URL?



The API Playbook

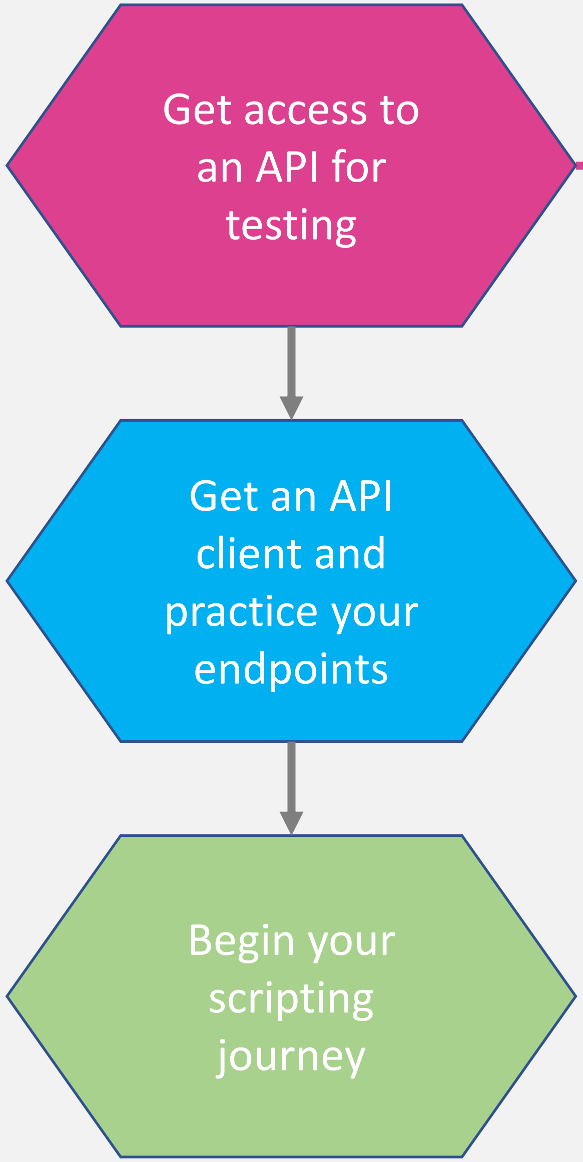

Get access to an API for testing
Get an API client and practice your endpoints
Begin your scripting journey

Not ready to engage your IT department or hosting provider yet?

2. Download and run AS locally on your machine

- I’m running AS locally on Windows this very moment
- You don’t need Linux, you don’t need a server
- It will be blank
- But you can test all you want
- The API URL will be http://localhost:8089
- Instructions in Playbook

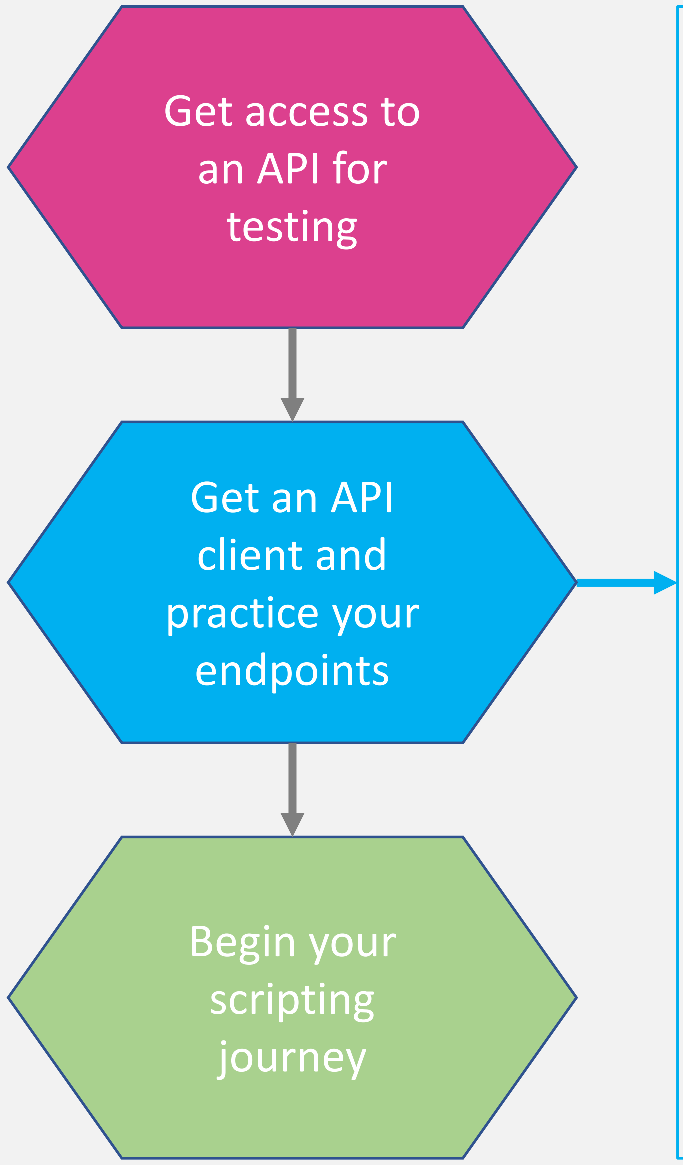
The API Playbook


Get access to an API for testing
Get an API client and practice your endpoints
Begin your scripting journey

Not ready to engage your IT department or hosting provider yet?

3. **Use the ASpace Sandbox API**
   - The API address is: http://sandbox.archivesspace.org/api/
   - It will get over-written
   - But you can test all you want
   - API URL in Playbook

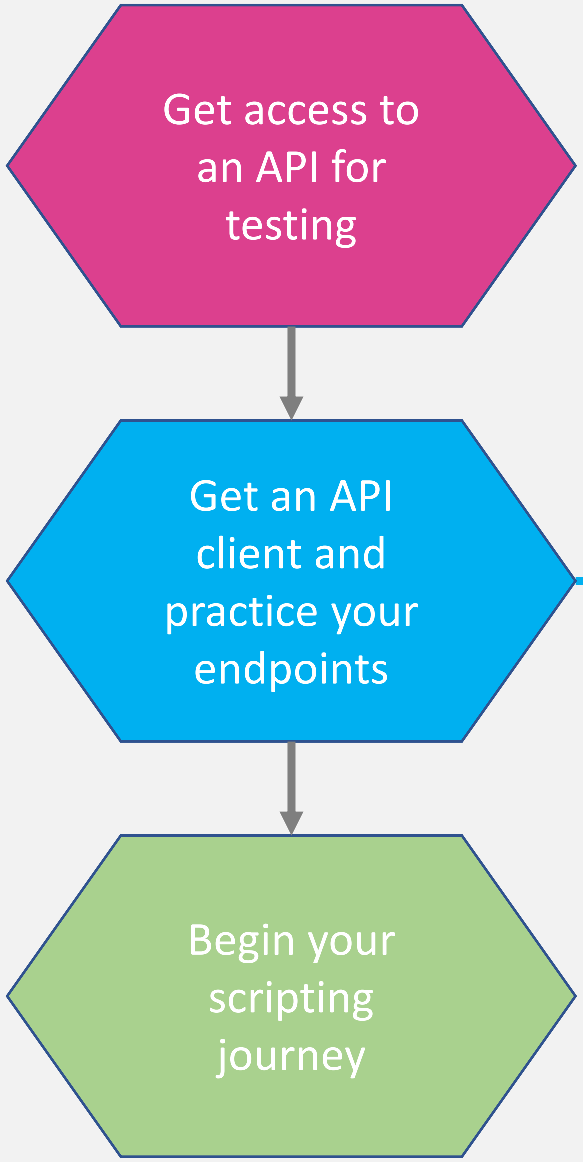
The API Playbook


Get access to an API for testing
Get an API client and practice your endpoints
Begin your scripting journey

Download Postman, a free API client

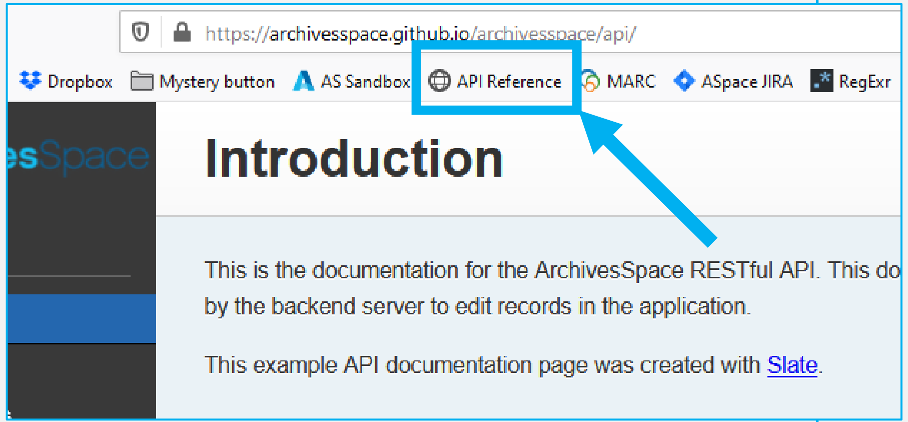
- I don't *think* you need admin privileges on your computer, but I'm not sure
- Use the Playbook to authenticate and GET your first record
- Experiment. A lot.
  - Get all the major record types
  - Read the JSON
  - Explore arrays
  - Compare what you see in the interface to what you see in the JSON
  - Show your colleagues what you're learning

The API Playbook


Get access to an API for testing
Get an API client and practice your endpoints
Begin your scripting journey

Practice using endpoints

- Start with the endpoints for major record types, the ones you use all the time
  - Accessions
  - Resources
  - Subjects and Agents
  - Top containers
  - Locations



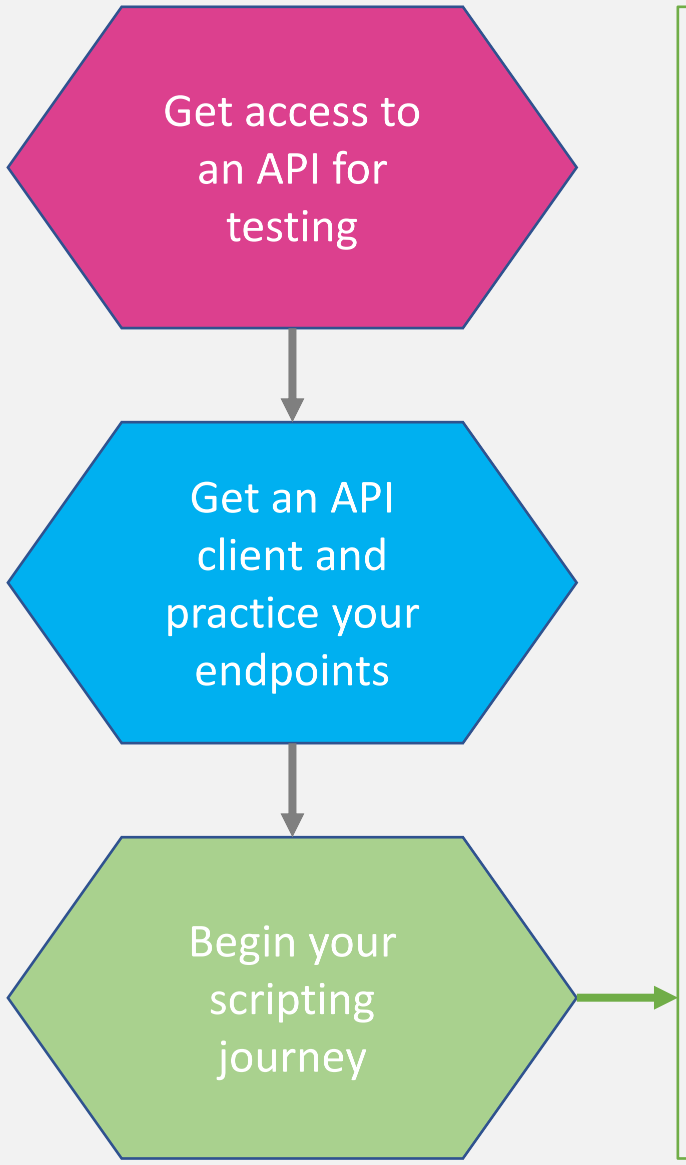
Introduction

This is the documentation for the ArchivesSpace RESTful API. This do
by the backend server to edit records in the application.

This example API documentation page was created with Slate.

- Then move on to endpoints you don't recognize
  - It's okay if you can't figure some out
  - Even people who use the API have no idea sometimes
- Remember that the API documentation reflects the most recent version of Aspace, which might not be the version you're testing with

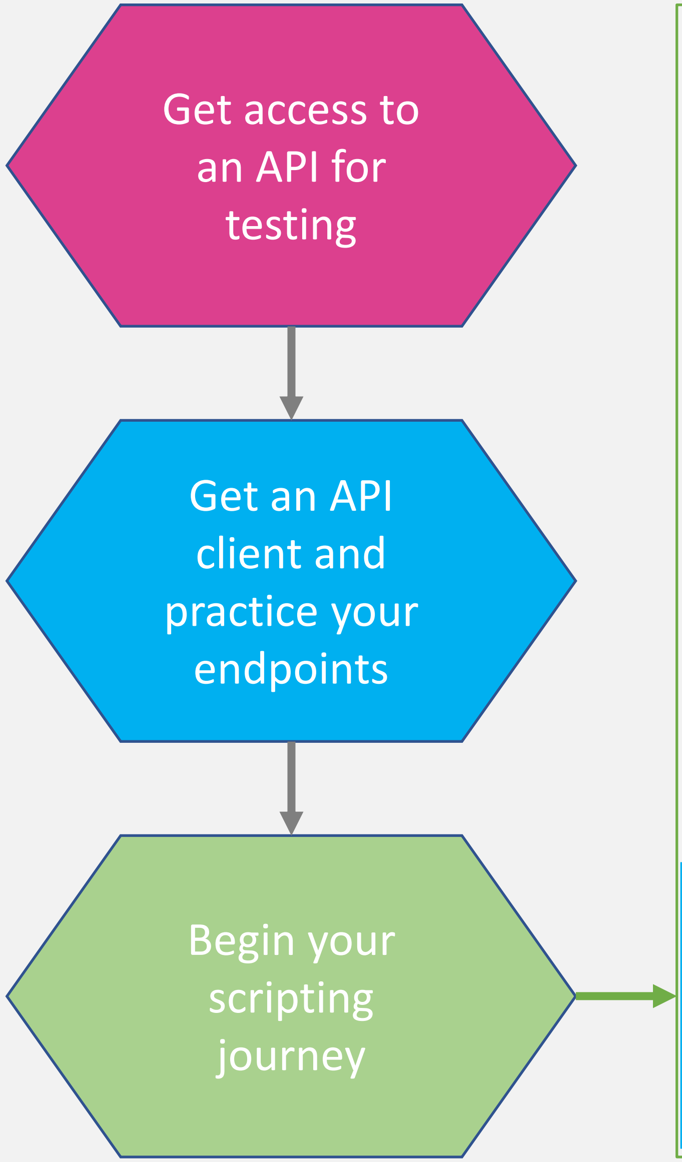
The API Playbook


Get access to an API for testing
Get an API client and practice your endpoints
Begin your scripting journey

Acknowledge that this is a journey.

- You might not want to take on an entirely new career goal
- Or maybe you do
- Whichever it is, it will take time
- You will get frustrated and hit dead ends
- You might want structure where there is none
  - i.e. there isn't an 8-week "Python for Archivists using the Aspace API" course, you will have to cobble that together for yourself

The API Playbook


Get access to an API for testing
Get an API client and practice your endpoints
Begin your scripting journey

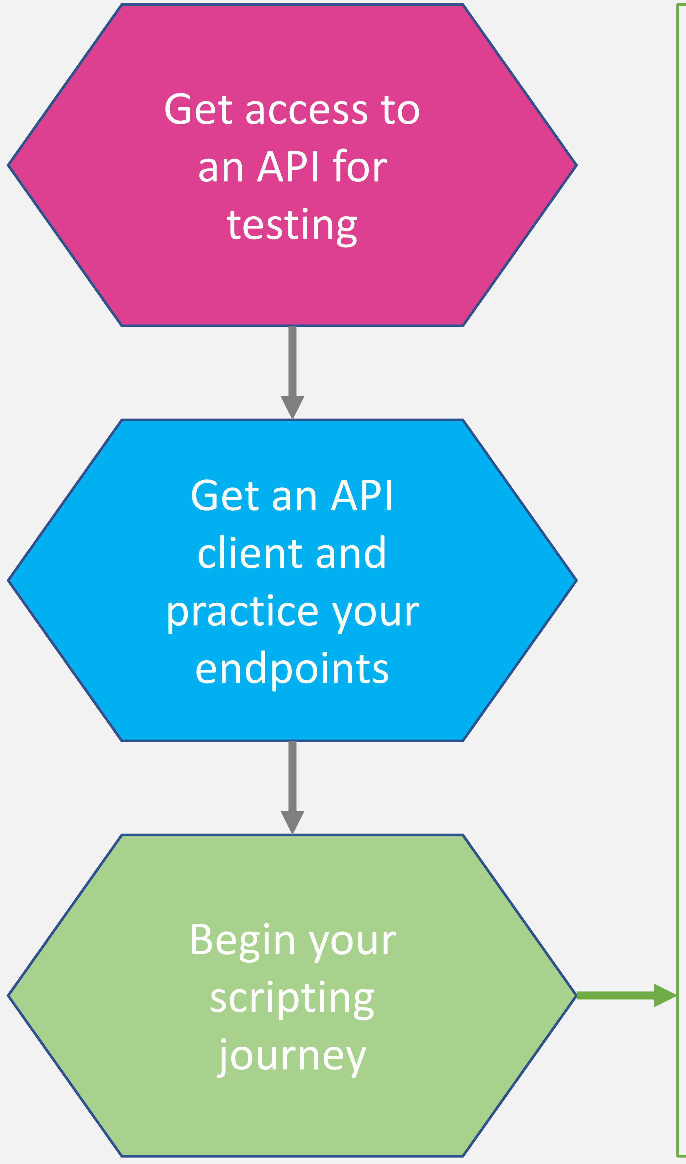
Share it. Cultivate buy-in.

- This is real advice, buy-in is important
- Are there others in your organization that you can learn with?
    - Take classes together
    - Have meetups
    - Inevitably you will teach each other
- Even if you're solo, share it
    - Demonstrate using the API to your manager
    - Present about it at your next staff meeting

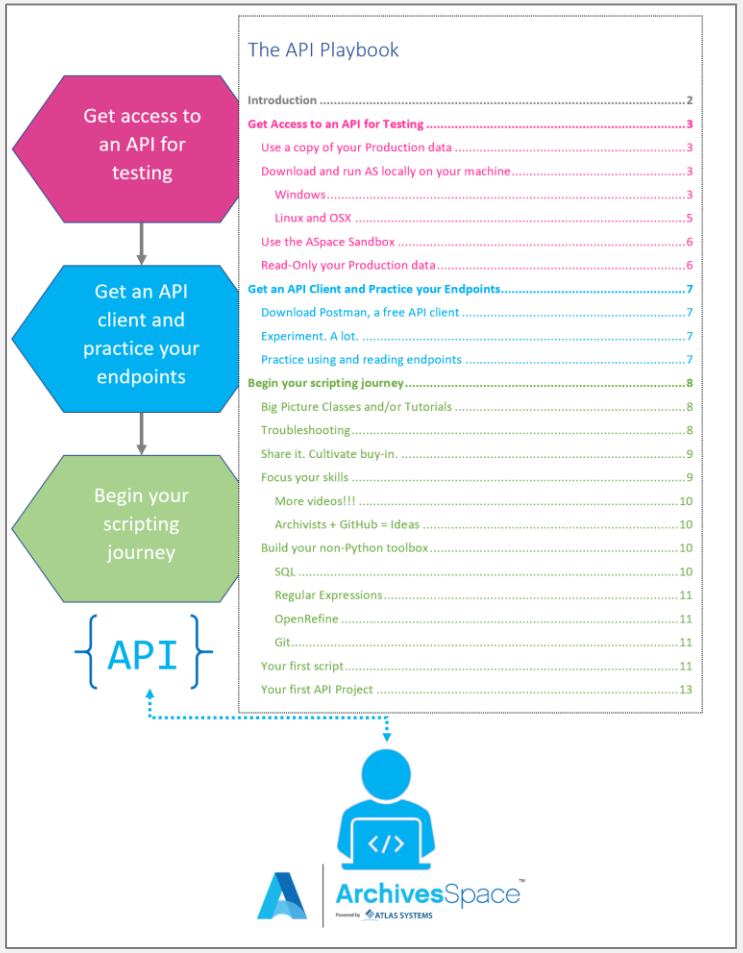
Managers:

Consider funding and time for Python classes for staff, and know that this is going to take awhile
Or, advocate for new relationships inside your organization
i.e. having an IT/archivists working group where there was none before, advocating for staff to have admin privileges on their machines, cultivating trust for archivists with growing tech skills

The API Playbook


Get access to an API for testing
Get an API client and practice your endpoints
Begin your scripting journey

- **Big Picture Classes and/or Tutorials**
- **Troubleshooting**
- **Share it. Cultivate buy-in.**
- **Focus your skills**
  - More videos!!!
  - Archivists + GitHub = Ideas
- **Build your non-Python toolbox**
  - SQL
  - Regular Expressions
  - OpenRefine
  - Git
- ***Your first script***
- ***Your first API Project***


Get access to an API for testing
Get an API client and practice your endpoints
Begin your scripting journey
{API}
ArchivesSpace™
Powered by ATLAS SYSTEMS

The API Playbook

End of Prepared Content!

Questions?

Improvisation?

Thanks to
Lora Woodford and Eric Hanson! #triad
Dumbledore's Army for support and camaraderie | Mark Cyzyk and Alicia Detelich for SQL revelations
Austin Schaffer, Dustin Stokes, and all my supportive colleagues at Atlas Systems