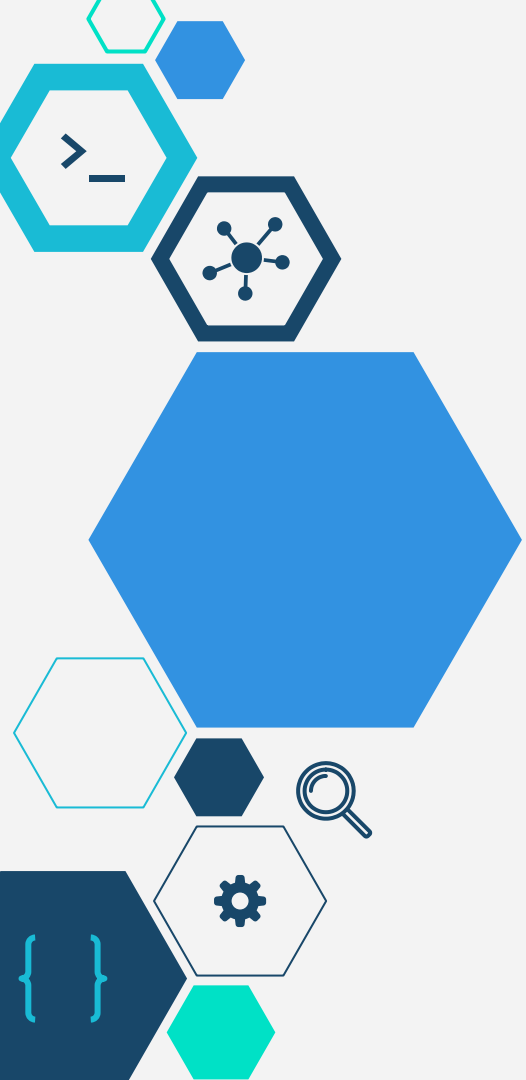
API Workshop

University of Virginia

2018-08-10

Valerie Addonizio
vaddoniz@jhu.edu

Mark Custer
mark.custer@yale.edu



Introduction

A spectrum of experience

Mark

- An **accidental Archivist**, by way of:
  - a playbill digitization project; and
  - an EAD conversion project.
- Thinks **XML** is still cool
- Has no idea how to use **MS Access**
- Enjoys **counting laps in a pool** (that's not a joke about programmatic loops)
- Would rank **List the Iconclast** in the top 5 best Simpsons episodes ever

Valerie

- Former **Processing Archivist**
- Never met a task she didn't want to solve with an **MS Access database**
- Helped migrate to ASpace and felt like **a fish climbing a tree**
- Enjoys **geocaching, woodworking,** and **planning ambitious camping trips**
- Knows **data modeling** and **systematic analysis** of the *really hard* problems

A broad audience

You can be here

Or here

Because we’re here for everyone

Aims

1.

- Practical, real-life application
- ...including the bad parts of real-life
- Resource-packed Git repository
- These voluminous (over 170!) slides

2.

Assurances:

- You are not alone
- I didn't become an archivist for this either (but some of you may have!)
- This can be difficult and frustrating
- You didn't miss a flight; the airport turned into a launch pad while you were in the parking lot

The 1-up learning experience

You can’t learn it all and we can’t teach it

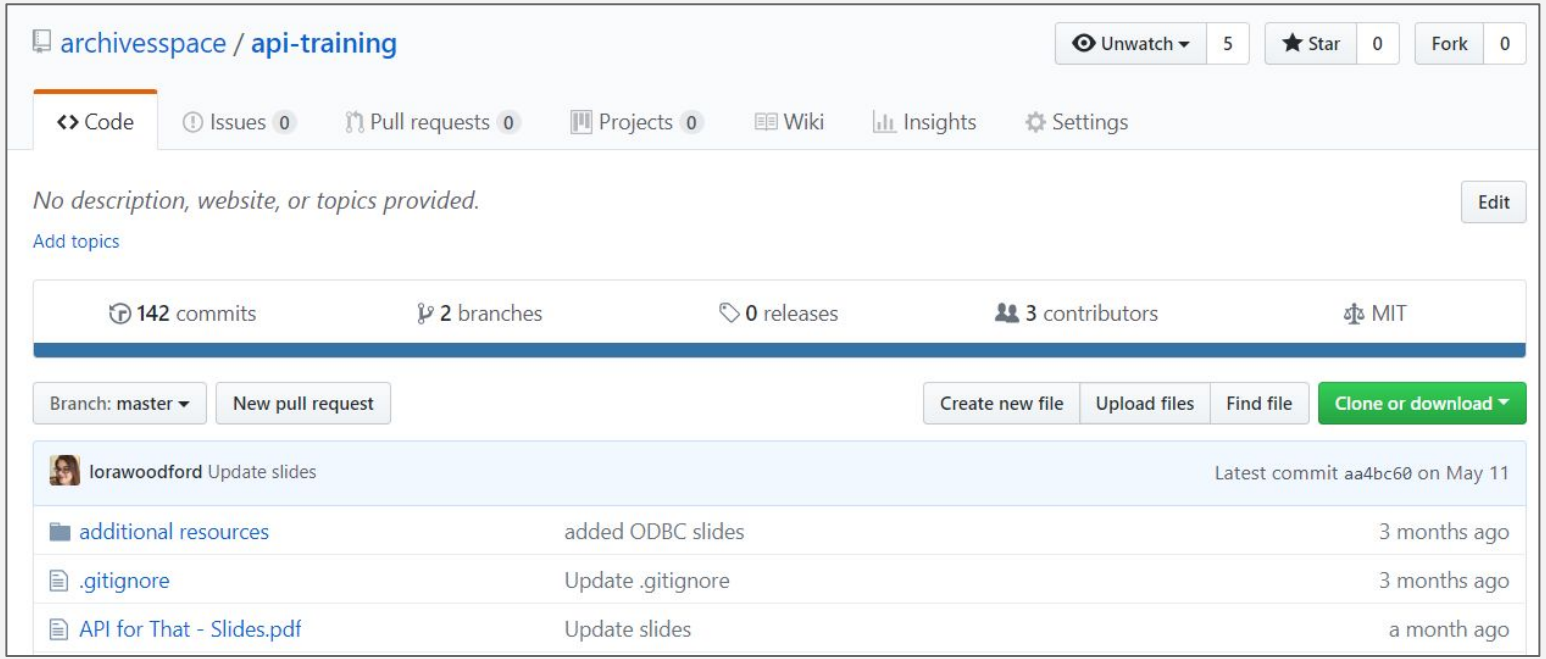
Are you a 1?	Are you a 2?	Are you a 3?
We hope you’ll leave a 2.	We hope you’ll leave a 3.	We hope to give you new ideas, scripts, and momentum.

Setting up your tech

We promise you're in the right workshop

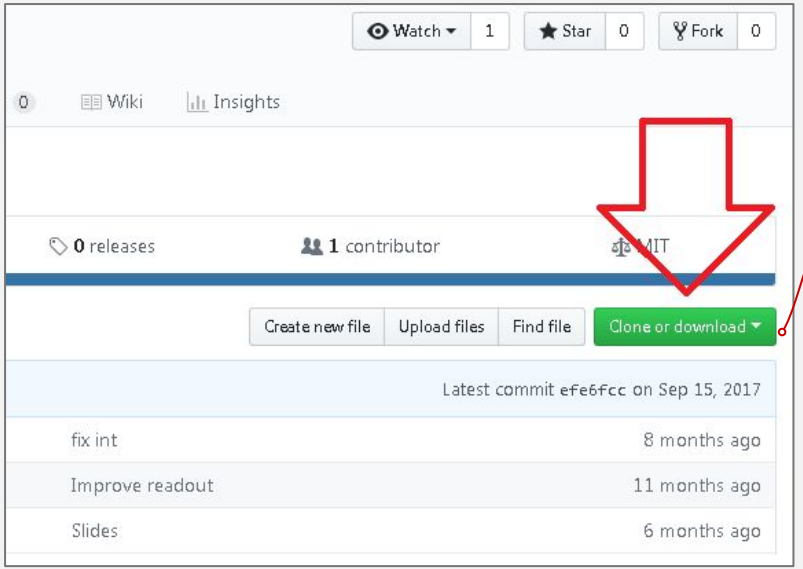
Pitstop: Clone the repo

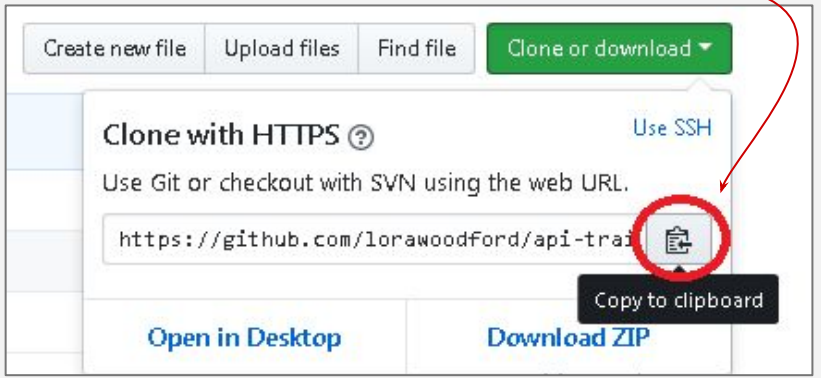
1. Open a browser and navigate to https://github.com/archivesspace/api-training
2. You should see this:



Pitstop: Clone the repo

1. Bookmark it!
2. Click the green button, click the little clipboard icon, and **Copy to clipboard**





Pitstop: Clone the repo

Mac

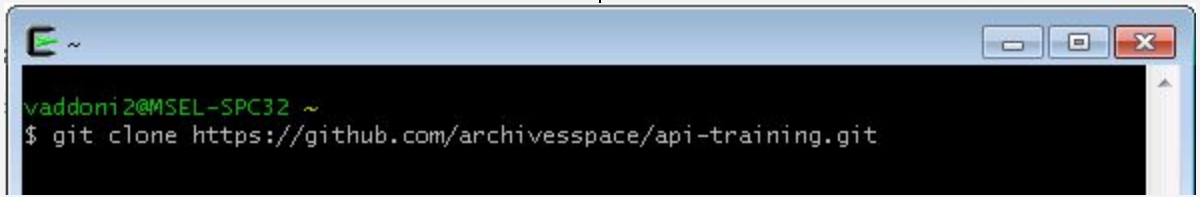
1. Open Terminal
2. Type `cd Desktop` and hit enter
3. Type `git clone [command+v then paste]`
4. Hit enter

(don't type this, we mean an action)

PC

1. Open Cygwin
2. Type `git clone [right-click then paste]`
3. Hit enter

(don't type this, we mean an action)




```
git clone https://github.com/archivesspace/api-training.git
```

Pitstop: Clone the repo

```
vaddoni2@MSEL-SPC32 ~
$ git clone https://github.com/archivesspace/api-training.git
Cloning into 'api-training'...
remote: Counting objects: 500, done.
remote: Compressing objects: 100% (190/190), done.
remote: Total 500 (delta 310), reused 493 (delta 303), pack-reused 0
Receiving objects: 100% (500/500), 25.58 MiB | 29.25 MiB/s, done.
Resolving deltas: 100% (310/310), done.
Checking connectivity... done.

vaddoni2@MSEL-SPC32 ~
$
```

Now you have a folder (either on your Mac's Desktop, or in C:/Cygwin/home/[username]) that contains all the **materials you'll need** for the rest of today's workshop.

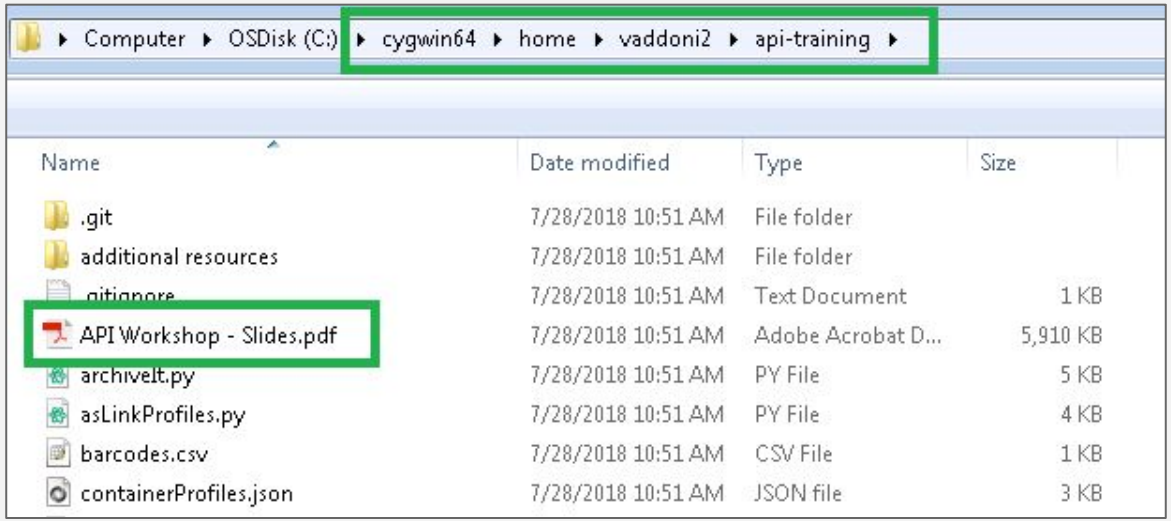
This folder, titled **"api-training,"** is a *direct clone* of what you see in your browser on github.com.

Pitstop: Clone the repo

Find the folder named “api-training” and open the PDF of these slides.

- Mac users: check your desktop
- Windows users: look in C:\Cygwin\home\[username]

Everyone should have the slides open by the end of this slide. Raise your hand if you can’t find it.



You can follow along starting from this slide

This terrible shade of yellow should be easy to find.

Reference slide numbers at key points

Technical pitstop:
The (free) applications

- **Atom**
  - A text editor that is handy for interacting with JSON, scripts, and all sorts of structured data
  - Can utilize additional packages to customize to your needs (e.g. a JSON “linter”)

- **Postman**
  - A GUI application for interacting with APIs

- **Cygwin** (Windows users)
  - A Unix emulator for Windows
  - Macs are already in a Unix environment (one reason for their preference among developers)

Pitstop: Pre-workshop steps

Pre-Workshop Instructions for Mac users

- Did you receive and complete the pre-workshop instructions?
- Did you install Atom?
- Postman?
- Did you check for gcc?
- Did you install Homebrew?

If you have not completed these steps, navigate to the api-training folder on your Desktop and open the Pre-Workshop instructions for Macs.

Pre-Workshop Instructions for Windows

- Did you receive and complete the pre-workshop instructions?
- Did you install Atom?
- Postman?
- Cygwin, with all the weird instructions?

If you have not completed these steps, navigate to C:/Cygwin/home/[username]/api-training and open the Pre-Workshop instructions for Windows.

Pitstop: Note this convention

If you see normal text, it's normal.

If you see `a different font highlighted in a different color`,
this indicates a command you need to type and execute.

Type or copy exactly what you see.

If you see weird stuff and lots of characters...
then type/copy weird stuff and lot of characters.

Pitstop: Installing packages

Mac users only:

1. Open Terminal
2. Type `brew update` and hit enter
3. Type `brew install pyenv` and hit enter
4. Copy and paste `echo -e 'if command -v pyenv 1>/dev/null 2>&1; then\n  eval "$(pyenv init -)"\nfi' >> ~/.bash_profile` and hit enter
5. Close and reopen terminal (required for your changes to take effect!)
6. Type (or copy/paste) `pyenv install 3.6.5` and hit enter
7. Type `pip3 install requests` and hit enter


Breathe

APIs Defined

Application Programming Interfaces

What does API stand for?

Application

As in a computer applications, like Word or Chrome or ArchivesSpace

Programming

As in computer “programming,” or taking steps to make a computer do something you want it to do

Interface

The place where two systems meet

What do APIs do?

As the prior slide suggests, APIs make it possible for **applications to interact (or interface) with one another**.

APIs are **not new**, and there are **many types** of APIs.

When you copy content from a Word document to your clipboard, then paste that content into an Outlook e-mail, it works because your computer operating system, which both your versions of Word and Outlook are programmed to run on, uses an API to **allow the interchange of information**.

APIs tell software developers the **rules of the road** that they must follow if they want their applications to play well with others.

For more: http://readwrite.com/2013/09/19/api-defined/

That’s not what I thought

Though anything that allows an interchange of information between two applications is *technically* a **form of an API**, what we typically mean today when we say “API” is a very specific thing:

That thing is a **web API**.

Ok, so what is a *web* API?

Complicated: A RESTful API is an **application program interface** (API) that uses HTTP requests to GET, PUT, POST and DELETE data.

Simple: You access it over the web, using URL-like directions, and are limited to 3-4 simple commands or activities.

For more: http://searchcloudstorage.techtarget.com/definition/RESTful-API

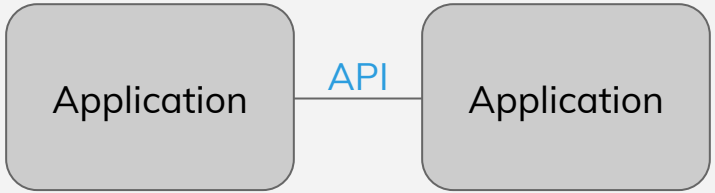
Extra nerdy sidebar:

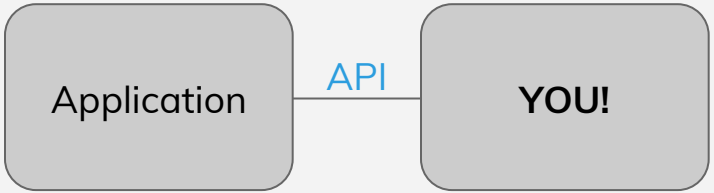
- Web APIs also come in several flavors, including **SOAP** and **REST**.
- We're going to be exclusively working with **RESTful APIs** today, as they're far more prevalent in archives/libraries technologies.
- REST stands for **"representational state transfer"** and was defined in 2000 in a doctoral dissertation by Roy Fielding.
- REST essentially dictates how an application should be able to **textually interact** with a web service.

But… I'm not an application

As librarians and archivists with collection descriptions and/or collections on the web, you probably **do** care about being able to **access** and **meaningfully manipulate** textual data on the web **at scale**.

In many of the exercises we will work through together today, **you** are, in fact, one of the "applications" interfacing with web-based data.





Questions

?

APIs in Action

Let’s start elsewhere, away from ArchivesSpace

APIs are pretty basic; it’s how you use them that gets complex

You have a few basic choices that you can combine:

COMMAND | Execution

Vocabulary pitstop:
API Commands

- **GET**, **POST**, and **DELETE** are three cornerstone commands for a RESTful API
- We will use these terms throughout
- Think of them as View, Save, and of course, Delete
- All APIs allow GETs, some let you POST, and few allow you to Delete
    - ASpace does all three, but allows you to tailor permissions for each

In our first exercise, we will be using

GET with a GUI

(COMMAND | Execution)

Vocabulary pitstop: GUI

- **GUI** (gooey) stands for Graphic User Interface: *every program* you use has a GUI
- But in the programming/scripting world, there is also the command line/powershell/terminal
- We will be using both: Postman is a GUI

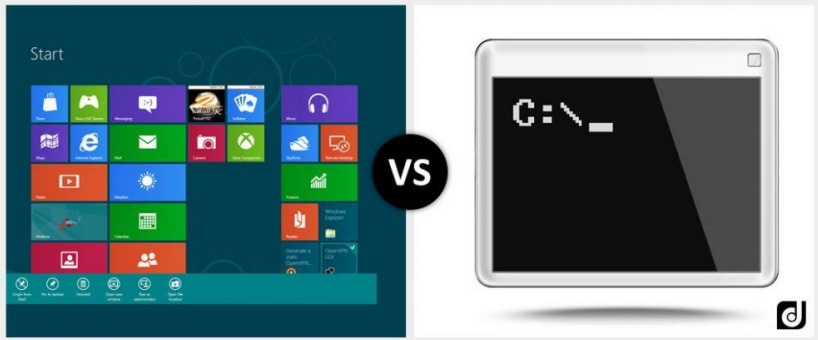


Image from http://www.differencebtw.com/difference-between-gui-and-command-line/

GET with GUI: ProPublica

Including:

- A lot of vocabulary
- Web searches versus APIs
- Repurposing data

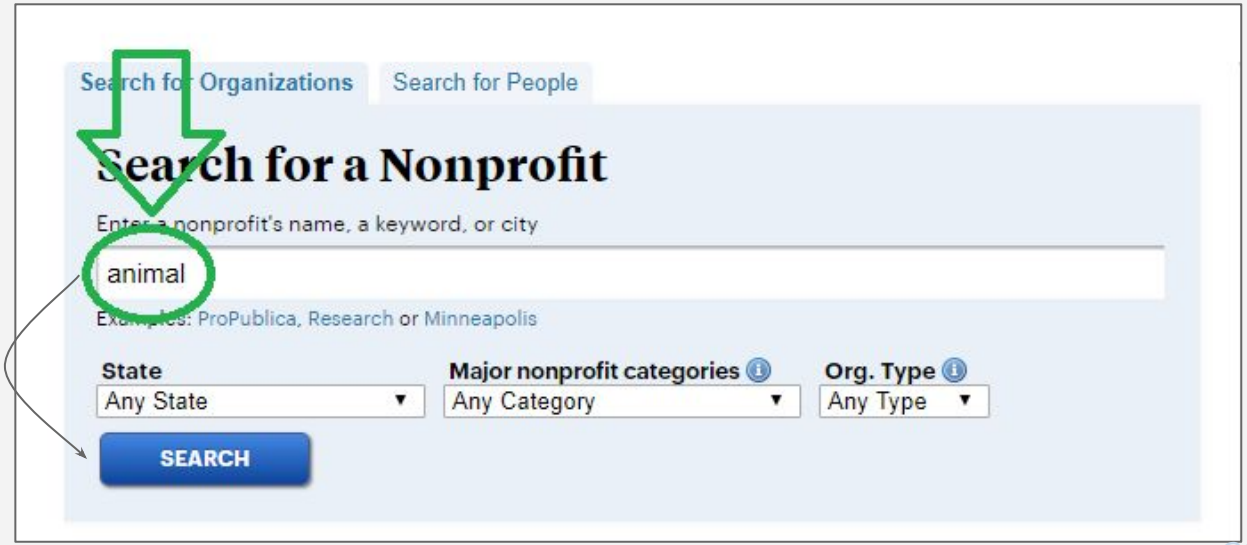
GET with GUI

ProPublica.org emphasizes research-based journalism and provides its data to other reporters and the public

1. Navigate to ProPublica.org
2. Click News Apps
3. Scroll to the Nonprofit Explorer
4. Next slide...

GET with GUI

1. Do a search for “animal” in the Nonprofit Explorer and hit Search. This is a list of animal-based charities
2. Look at the address bar…



Sidebar: Web URL vs API endpoint

Here's the address bar after that search. Note that your search term is in there:

https://projects.propublica.org/nonprofits/search?utf8=%E2%9C%93&q=animal&state%5Bid%5D=&ntee%5Bid%5D=&c_code%5Bid%5D=

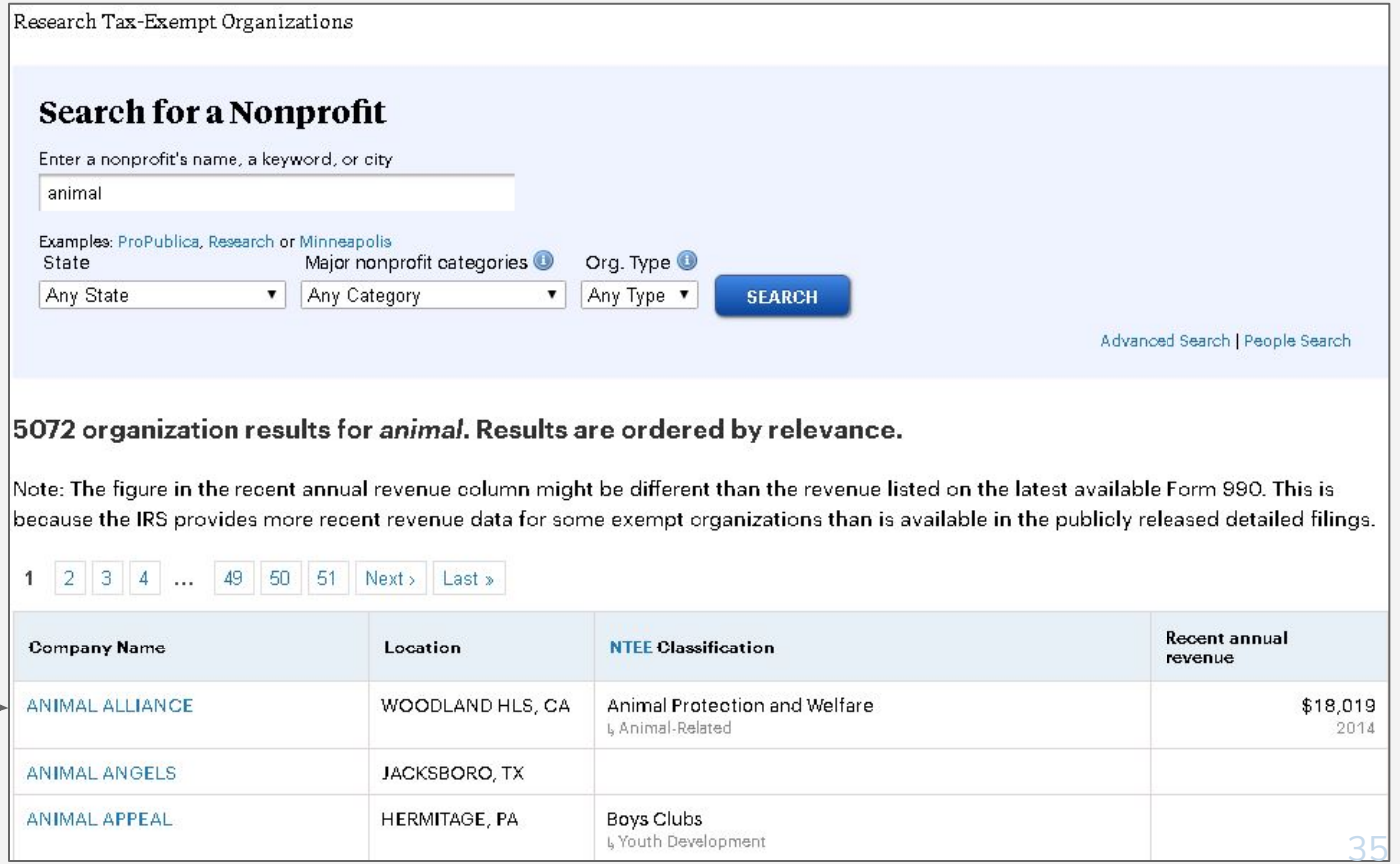
- We tend to think of URLs as *locations*...
- We call your attention to this as we discuss Endpoints.

GET with GUI

Back to the results.

Click on the first result,

Animal Alliance



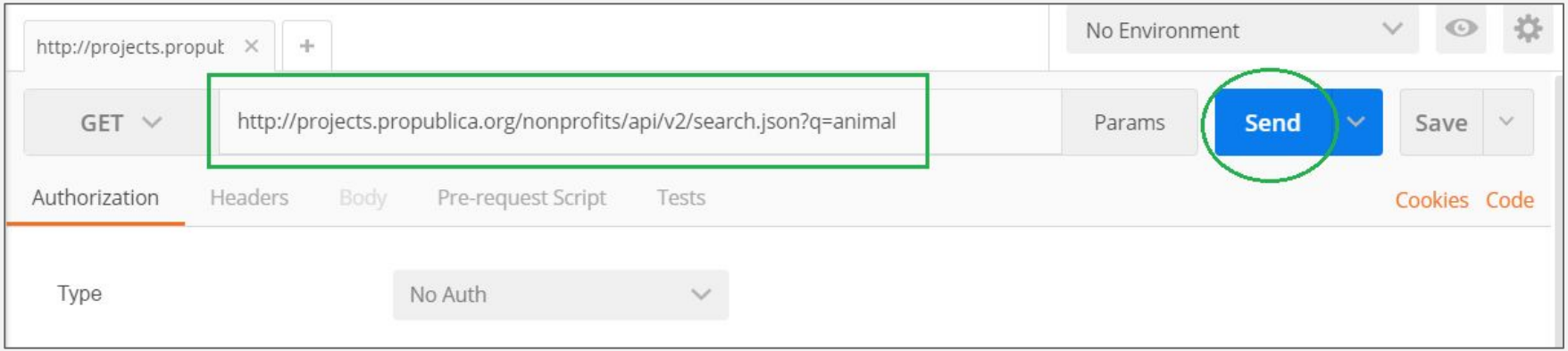
Sidebar: Web URL vs API endpoint

Note that the address for Animal Alliance has a unique identifier in it

https://projects.propublica.org/nonprofits/organizations/731663130

GET with GUI - ProPublica

1. Open Postman



2. Type this next to the GET command:

http://projects.propublica.org/nonprofits/api/v2/search.json?q=animal

GET with GUI - ProPublica

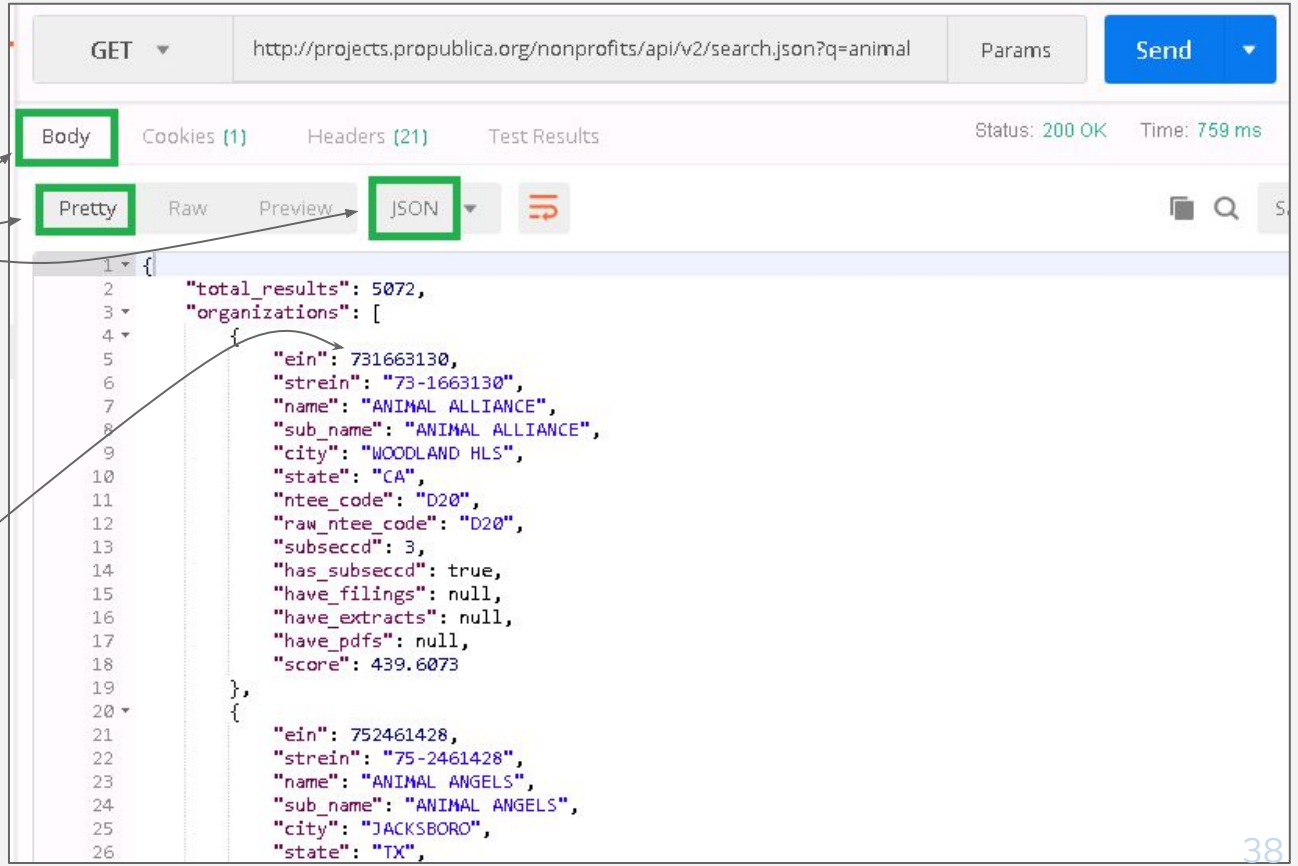
Do you see these results?

If not, check these settings.

Look at the first result.

There's that unique ID again.

It's an Employer Identification Number (EIN)



Sidebar:
Web search vs API

5072 organization results for *animal*. Results are ordered by relevance.

Note: The figure in the recent annual revenue column might be different than the revenue listed on the
because the IRS provides more recent revenue data for some exempt organizations than is available in t

1 2 3 4 ... 49 50 51 Next › Last »

Company Name	Location	NTEE Classification
ANIMAL ALLIANCE	WOODLAND HLS, CA	Animal Protection and Welfare ↳ Animal-Related
ANIMAL ANGELS	JACKSBORO, TX	
ANIMAL APPEAL	HERMITAGE, PA	Boys Clubs ↳ Youth Development

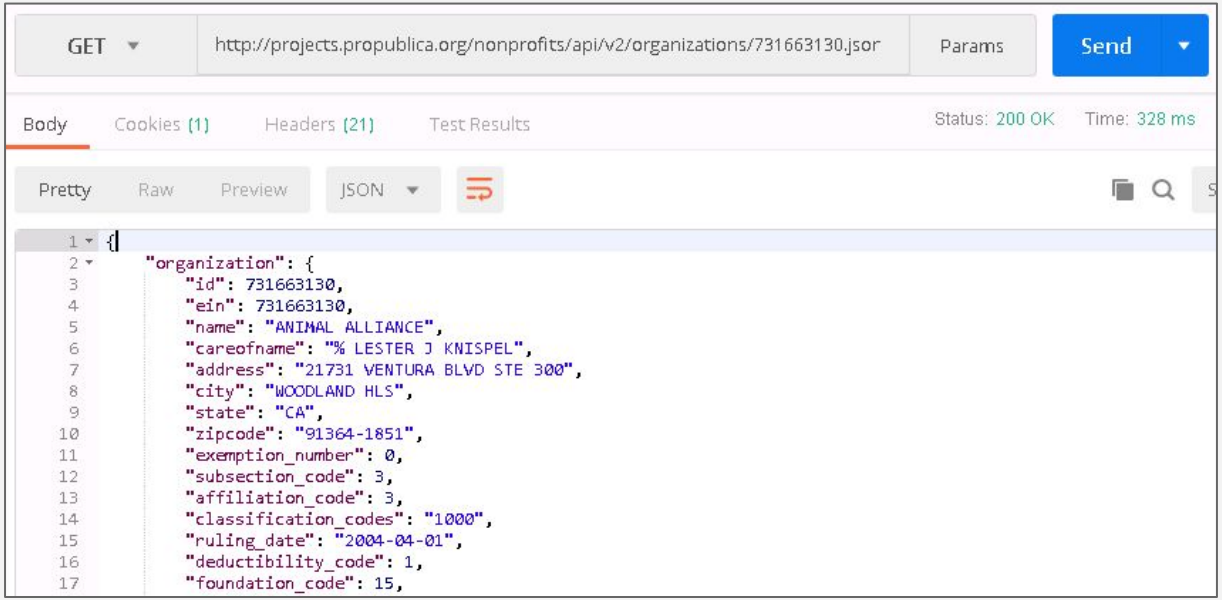

```
{
    "total_results": 5072,
    "organizations": [
        {
            "ein": 731663130,
            "name": "ANIMAL ALLIANCE",
            "city": "WOODLAND HLS",
            "state": "CA",
            "ntee_code": "D20",
            "raw_ntee_code": "D20",
            "subseccd": 3,
            "has_subseccd": true,
            "have_filings": null,
            "have_extracts": null,
            "have_pdfs": null,
            "score": 439.6073
        },
```


You don't get different *results*, you get the same results in a different format.

GET with GUI - ProPublica

That was fun. Do it again:

http://projects.propublica.org/nonprofits/api/v2/organizations/731663130.json



Vocabulary pitstop: **JSON**

- JSON (jason or jay-sohn) is the most typical data transmission standard in APIs
- It is lightweight and easy to read and NOT scary
- Consists of key-value pairs, “key”: “value”

```
EAD:
     <unittitle>Johns Hopkins University library
records</unittitle>

JSON:
     “Title”: “Johns Hopkins University library records”
```

Vocabulary pitstop: **Endpoint**

- An API Endpoint is simply a unique URL that
  - Calls a function, like a keyword search, or
  - Represents an object or a collection of objects
- In AS, an Endpoint can be a Resource record, an accession, a container, etc.
- All Endpoints are URLs, but not all URLs are Endpoints
- Constructing Endpoints is a fundamental requirement for using APIs
- You cannot paste what's in an address bar as an endpoint, but, addresses usually help you find your way

GET with GUI - ProPublica

Endpoints

http://projects.propublica.org/nonprofits/api/v2/search.json?q=animal

Asking the API to perform a keyword search

http://projects.propublica.org/nonprofits/api/v2/organizations/731663130.json

Asking the API to display a specific record with a unique identifier

GET with GUI - ProPublica

Endpoints

http://projects.propublica.org/nonprofits/api/v2/search.json?q=animal

Asking the API to perform a keyword search

http://projects.propublica.org/nonprofits/api/v2/organizations/731663130.json

Asking the API to display a specific record with a unique identifier

Hmmm...

http://aspace.library.jhu.edu/repositories/3/archival_objects/67503

Sidebar: Repurposing results

If you don’t want to work directly in JSON, that’s cool. Just convert it to what you’re comfortable with so that you can use the tools you know.

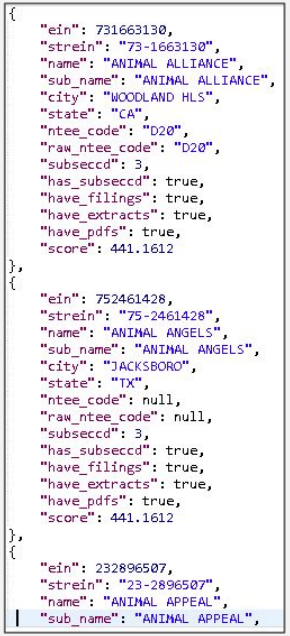
What’s more important is getting it back into JSON.

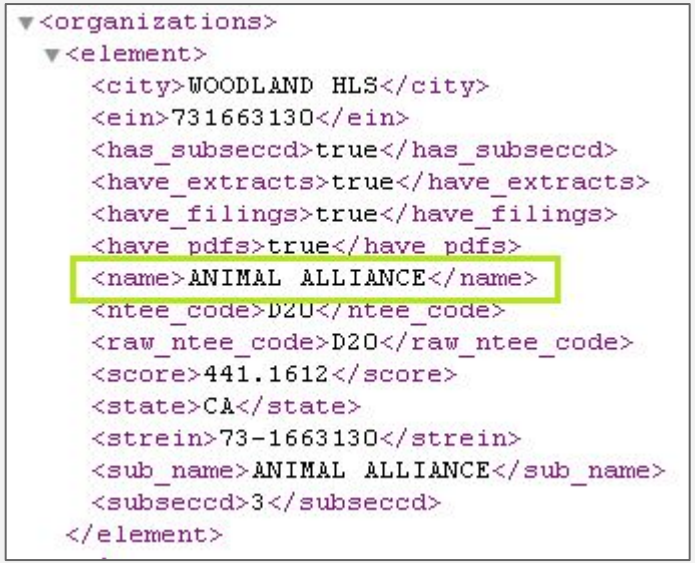
Sidebar: Repurposing results

Get data out	Do something to it	Put it back in
JSON MARC21 Any standardized data	OpenRefine MS Access XSLT Custom script (your choice) Find and Replace Hand encoding, copy and pasting, glue and popsicle sticks, *whatever it takes!*	This is the tough part.

Sidebar: Repurposing results

JSON

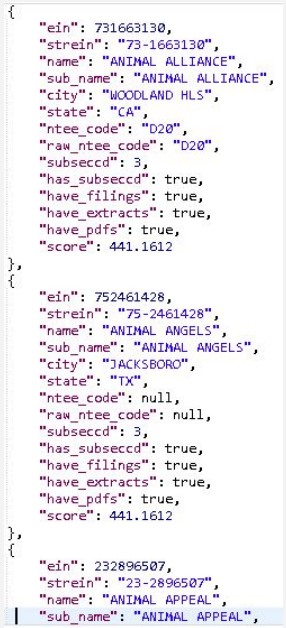




XML

Converting these JSON search results to a XML took less than 10 seconds using an online converter (we just googled “JSON to XML converter” and picked one)

Sidebar: Repurposing results



JSON

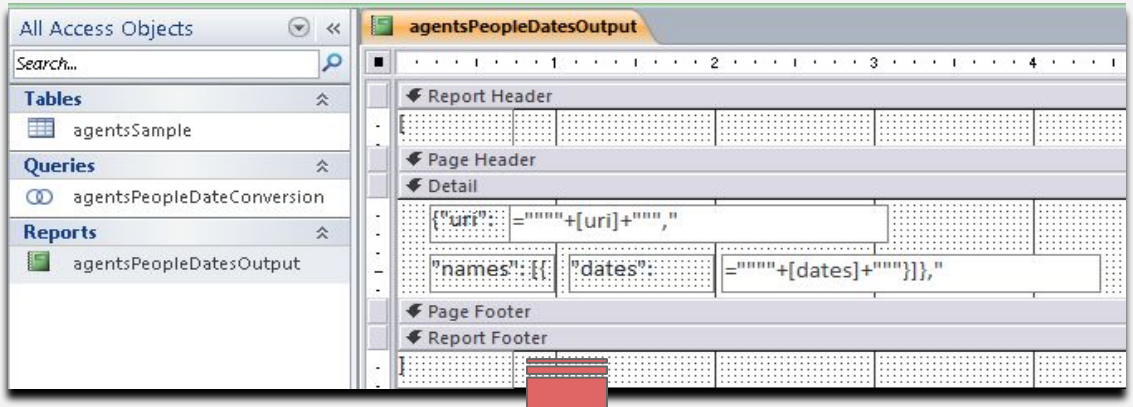
Spreadsheet

ein	strein	name	sub_name	city	state
731663130	73-1663130	ANIMAL ALLIANCE	ANIMAL ALLIANCE	WOODLAND HLS	CA
752461428	75-2461428	ANIMAL ANGELS	ANIMAL ANGELS	JACKSBORO	TX
232896507	23-2896507	ANIMAL APPEAL	ANIMAL APPEAL	HERMITAGE	PA
43654364	04-3654364	ANIMAL ASSISTANCE	ANIMAL ASSISTANCE	E BRUNSWICK	NJ
800099106	80-0099106	ANIMAL FAIR	ANIMAL FAIR	SEDALIA	MO

The above was again done with an online converter
(but remember that what you pass over the web is not secure).

Sidebar:
Repurposing results

MS Access report



"JSON"

This is a breakdown of the endpoint we just used:

http://projects.propublica.org/nonprofits/api/v2/search.json?q=animal

https://projects.propublica.org/nonprofits/api/v2	The address of the API. All requests must start with a similar string in order to point the request to where it needs to go. It essentially reads: go here.
/search.json	The search method set by the API. By appending this to the above URL, it essentially reads: go here, search, output that search in JSON.
?	Question marks denote that whatever follows is a parameter. Adding this to the above essentially reads: go here, and search, and use the following parameters. You can use more than one parameter.
q=	The parameter. The ProPublica API defines q as “A keyword search string. Searches using this parameter will search (in order) organization name, organization alternate name, city.”
animal	Any keyword supplied by the user. Go here, and use the keyword search parameter to search for *animal*, which will output as JSON.

ProPublica API documentation: https://projects.propublica.org/nonprofits/api

This is a breakdown of the other endpoint we just used:

http://projects.propublica.org/nonprofits/api/v2/organizations/731663130.json

https://projects.propublica.org/nonprofits/api/v2	The address of the API. All requests must start with a similar string in order to point the request to where it needs to go. It essentially reads: go here.
/organizations	By appending this to the above URL, it essentially reads: go here; once you get there, go to this specific place
/:ein	The colon means that your input is required.This essentially reads: put EIN number here. Together, it reads: go here; once you get there, go to this specific place; and once you're there, find this exact record.
.json	Output the record as JSON

ProPublica API documentation: https://projects.propublica.org/nonprofits/api

We just covered a whole lot.

- We did a search on ProPublica.org, and then we looked at a specific result of that search.
- Then we used the GET command in a GUI interface to do the same search through an API, using a search endpoint.
- We got those results in a data standard called JSON, and they were exactly the same as the ones we got on the web.
- We then used the GET command to get an individual record by using its unique endpoint.
- Once we got the results, we explored ways of manipulating them to suit our skills or needs.

Questions

?

GET with Script

In our second exercise, we will be using

GET with a Script

(COMMAND | Execution)

Scripts

Interacting with APIs through a GUI is a great way to “get it.”

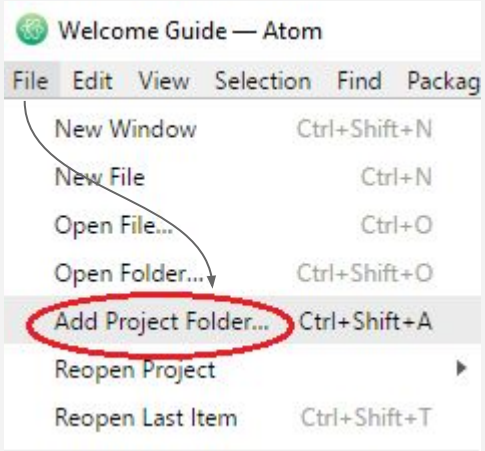
But it isn’t the only way.

We’re about to step into some unfamiliar territory for some.

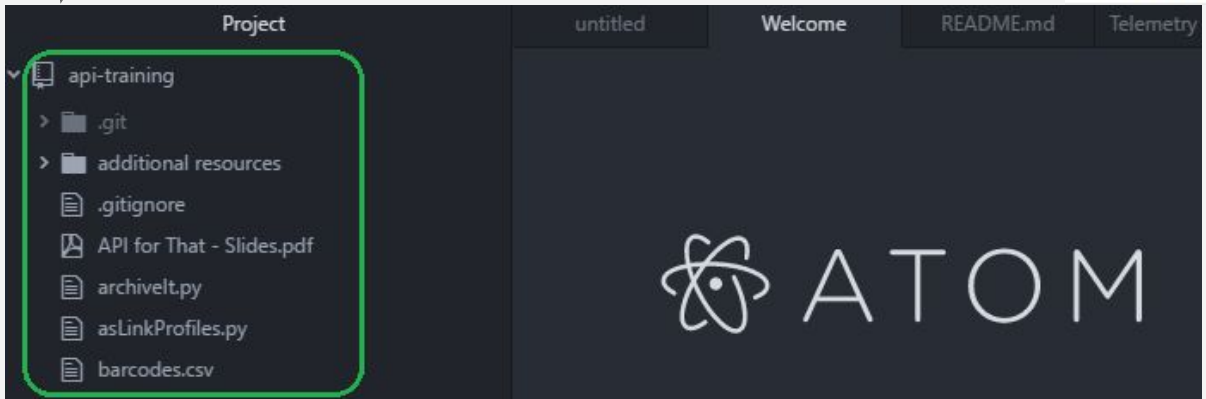
GET with Script- ProPublica

1. Open Atom
2. Add a project folder
3. Select the api-training folder

   - Mac users: check your desktop
   - Windows users: look in C:\Cygwin\home\[username]



3. You should now have a directory down the lefthand side, similar to this



GET with Script- ProPublica

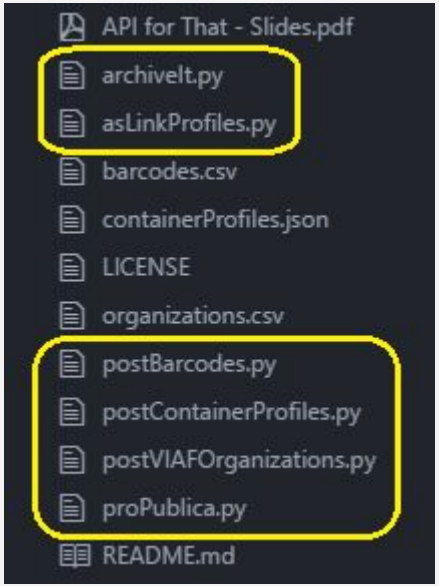
Take note of everything ending in .py

These are **python3** scripts, and we will be using them throughout the workshop.

We cannot teach you python3 today, but we can explore it together.

Open the first script of the day: **proPublica.py**

Locate the endpoint




```
7  raw_input('Press Enter to continue...')
8
9  endpoint = 'http://projects.propublica.org/nonprofits/api/v2/search.json?q=animal'
10
11 output = requests.get(endpoint).json()
```

Let’s take a break

Command line bootcamp

- Some very simple Unix commands are necessary in this workshop
- But more important is being able to use them effectively.
- Mac users, and PC users in Cygwin, will be using the same commands...
- ...but will be working in different directories.
- So navigating your own way is super important.

Command line bootcamp

Where are you and where do you want to go?

Mac

1. Open Terminal 

PC

1. Open Cygwin 

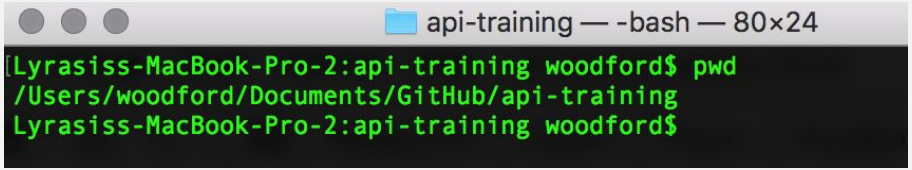
Command line bootcamp

Where are you?

Everyone type `pwd` and then hit enter.

Mac

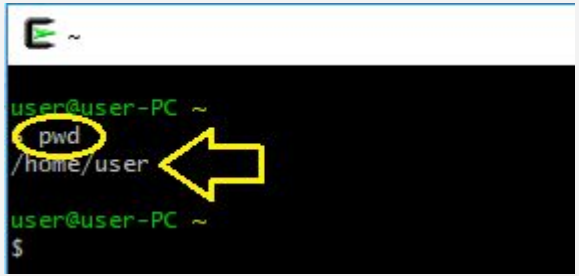
Mac users should see something like this



Note: There will be more screenshots for Windows users than Macs for the next few slides as we help PC users determine where they are. If your work computer is Windows, this will eventually matter to you.

PC

PC users should see something like this



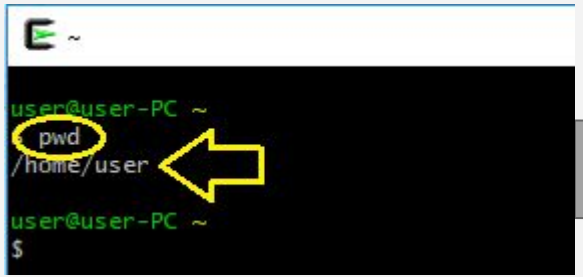
Command line bootcamp

Where are you?

PC

Windows users will ask: but **where** is that? This is non-intuitive, but *you're already in* C:\Cygwin because you're using the Cygwin window, so

This location:



Is this location:

Unix commands for Mac and Cygwin

Where am I?	“print working directory”	`pwd`

Command line bootcamp

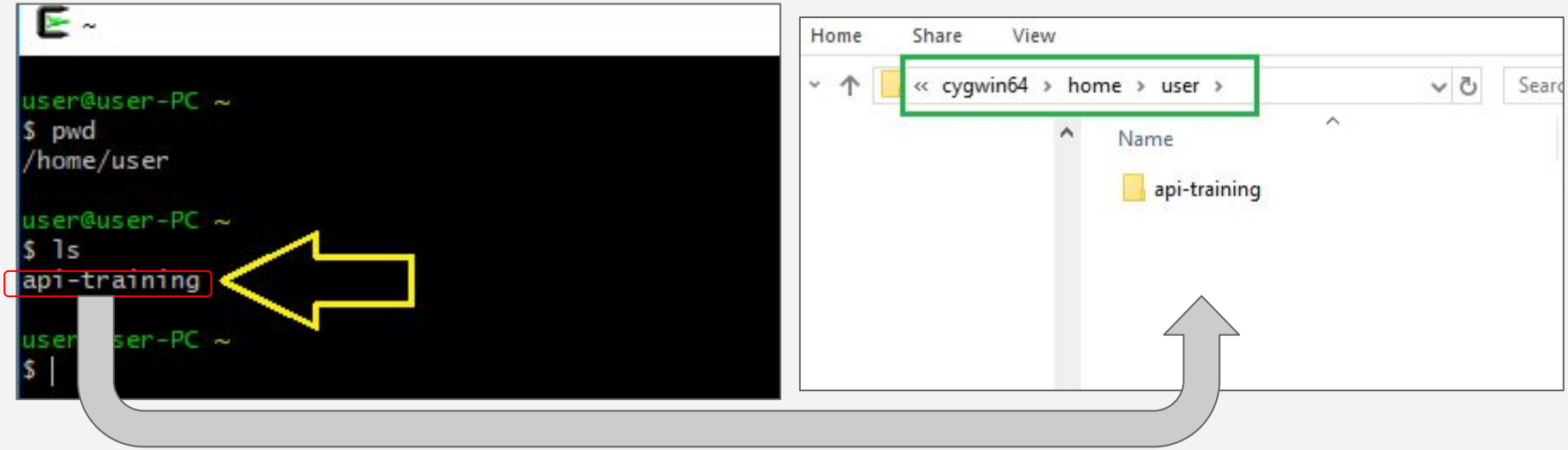
What is here?

Everyone type `ls` and then hit enter (that is **L** as in **L**ist)

Command line bootcamp

What is here?

PC



`ls` shows the same list of contents that I see if I navigate to C:\cygwin64\home\[user name] in Windows (this is a screenshot from Windows 7)

Unix commands for Mac and Cygwin

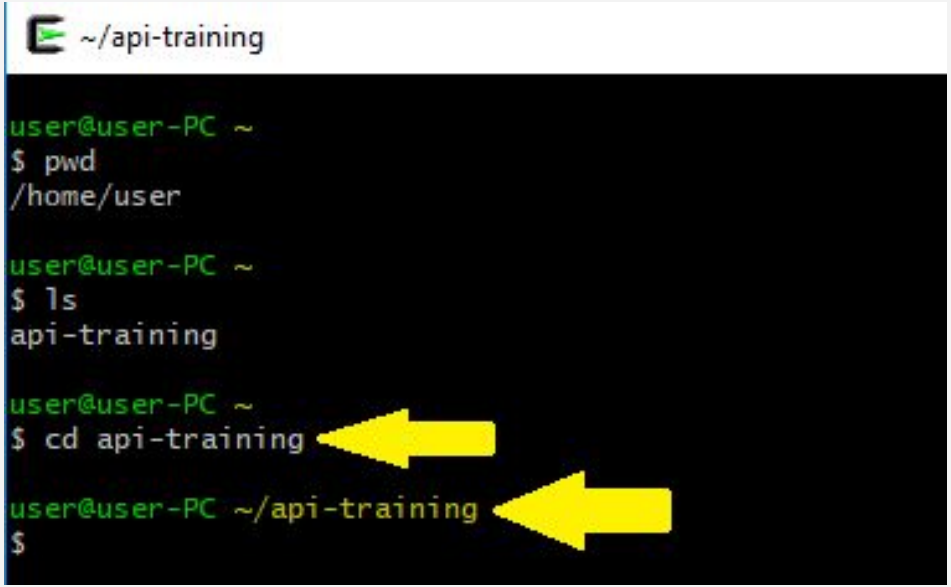
Where am I?	“print working directory”	`pwd`
What is here?	“list” (remember L as in List)	`ls`

Command line bootcamp

Move around

Now you're going to move *from* where you are *into* the api-training clone folder:

To move into that folder type `cd` (change directory), leave a space, and then type the name of the directory you want to go into: `cd api-training`



Command line bootcamp

Move around

PC

Happily, the directory you're in now is more obvious with that handy yellow text.

So remember:

- The path in Windows is: C:\cygwin64\home\[user name]\api-training
- And the *same path* in Cygwin looks like the new prompt, below:

Unix commands for Mac and Cygwin

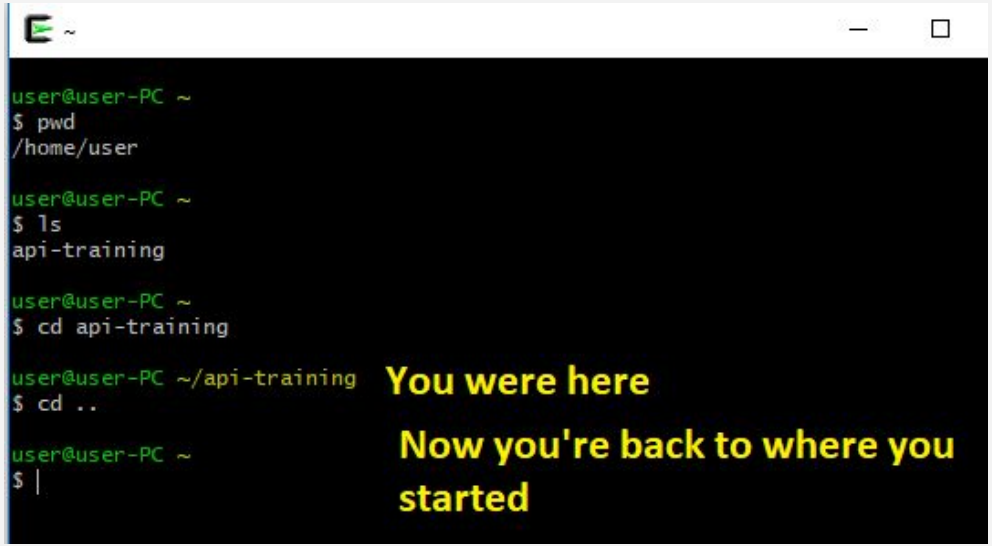
Where am I?	“print working directory”	`pwd`
What is here?	“list” (remember L as in List)	`ls`
How do I move from here to there?	“change directory”	`cd [type the name of the directory]` (don’t type this, we mean an action)

Command line bootcamp

Go up one level

`cd ..` = “go up one”

You’re going to move *from* the api-training clone folder back to the Cygwin home directory/Mac desktop



Unix commands for Mac and Cygwin

Where am I?	“print working directory”	`pwd`
What is here?	“list” (remember L as in List)	`ls`
How do I move from here to there?	“change directory”	`cd [type the name of the directory]` (don’t type this, we mean an action)
Move up one level		`cd ..`

Command line bootcamp

Repeat commands

Lastly, let's go back into the api-training directory, because that's where we need to be.

This is a good time to try the up-arrow on your keyboards to get back to a command you already issued:

- Try hitting the up-arrow a few times
- Pick the command that you need in order to get back into the api-training directory
- Use a command that will confirm where you are
- You may need to do this again, you have your handy cards to help you!

Unix commands for Mac and Cygwin

Where am I?	“print working directory”	`pwd`
What is here?	“list” (remember L as in List)	`ls`
How do I move from here to there?	“change directory”	`cd [type the name of the directory]` (don’t type this, we mean an action)
Move up one level		`cd ..`
Repeat command		Up arrow on keyboard

These are called Unix commands, so Google “unix commands” for other commands that will work on Macs and in Cygwin.

To make your life harder, remember that these same commands do not work in the Windows command prompt/Powershell; those are MS-DOS commands. This is why Mac users are smug.


Breathe

And now back to...

GET with Script: ProPublica

GET with Script- ProPublica

Mac

1. In the Finder navigate to your api-training directory
2. Ctrl+click the api-training directory, and select “New Terminal at Folder”
3. Type `ls` and examine the contents of that folder
4. Type `python3 proPublica.py` and hit enter
5. Next slide...

PC

1. Open Cygwin
2. Type `cd api-training` to enter the api-training directory
3. Type `ls` and examine the contents of that folder
4. Type `python3 proPublica.py` and hit enter
5. Next slide...

GET with Script- ProPublica

Mac

6. A new file called “proPublicaRecord.json” should now be in your api-training directory
7. Go back to Atom and open “proPublicaRecord.json”

Directory reminder: Desktop/api-training

8. Click *Packages* > *Atom Beautify* > *Beautify*
9. Take a look!

PC

6. A new file called “proPublicaRecord.json” should now be in your api-training directory
7. Go back to Atom and open “proPublicaRecord.json”

Directory reminder:
C:\cygwin\home\[username]\api-training

8. Click *Packages* > *Atom Beautify* > *Beautify*
9. Take a look!

Questions

?

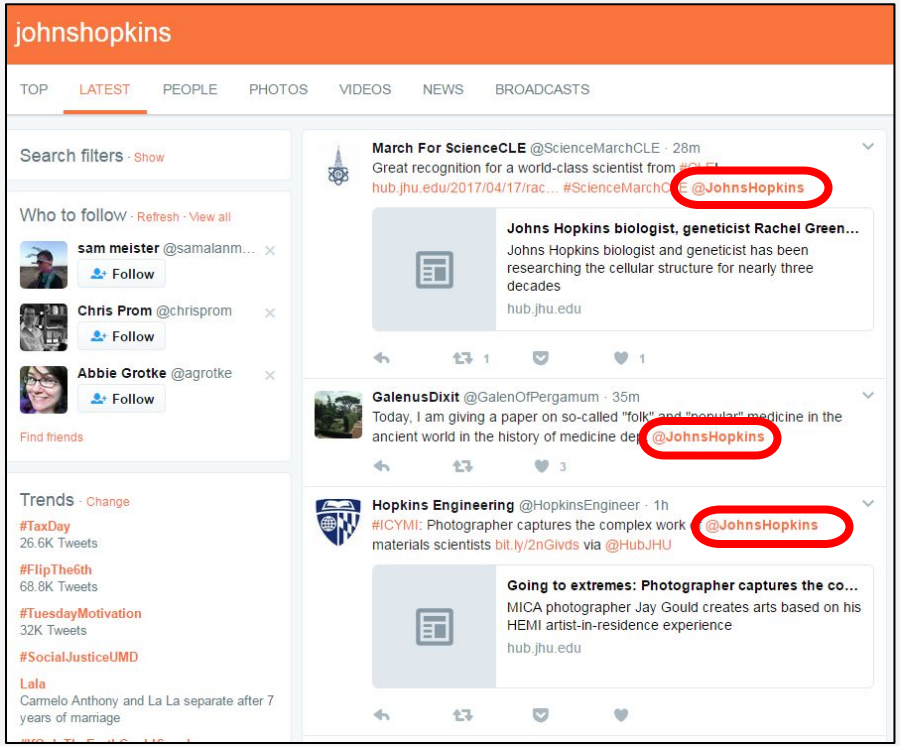
GET with GUI: Twitter

Using an API to collect records

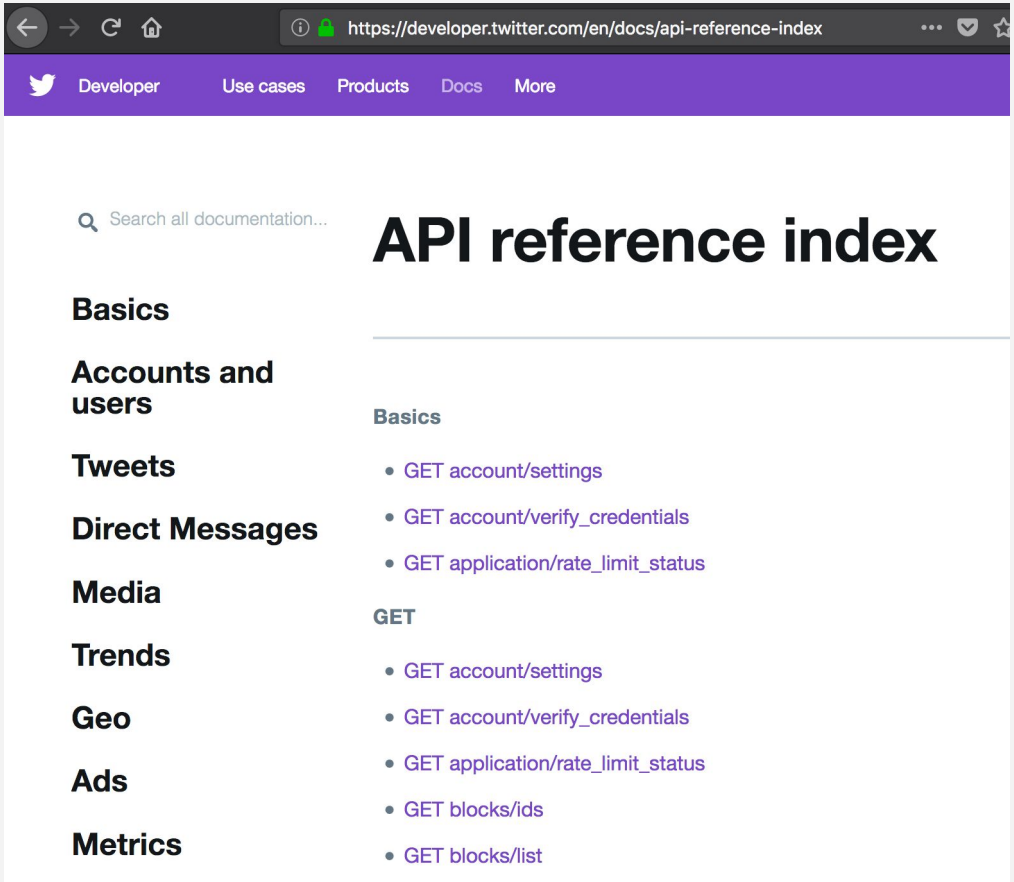
Using an API to collection records

@JohnsHopkins

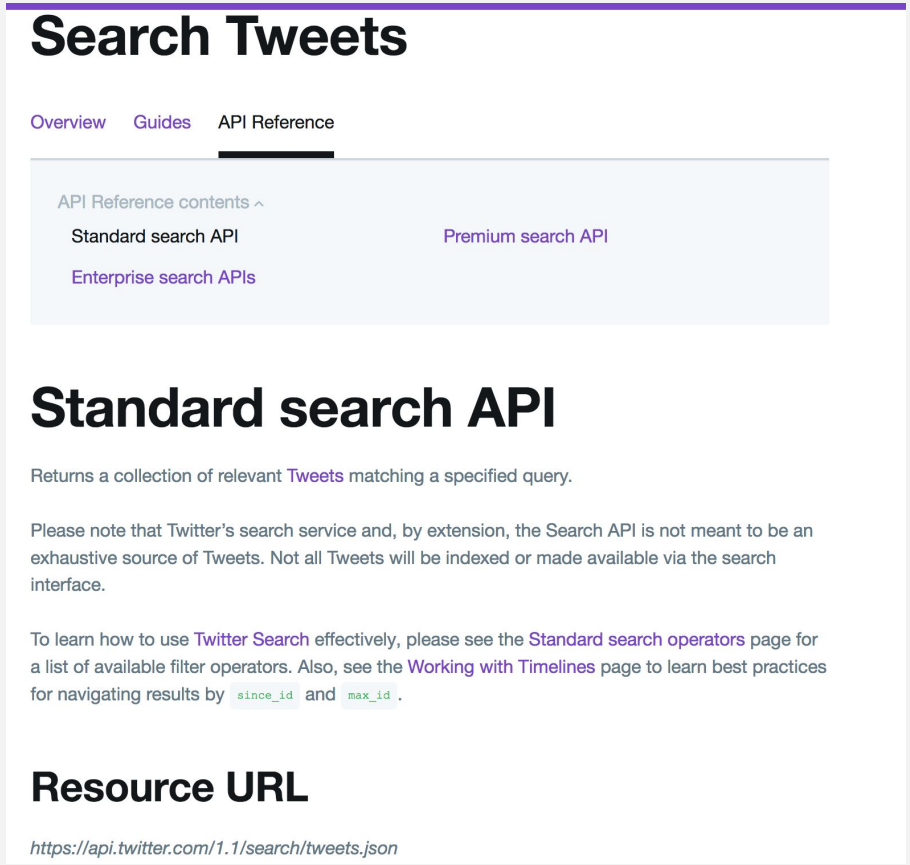
Scenario: As your university's records manager, you wish to regularly capture Tweets mentioning your university.



Using an API to collection records



Using an API to collection records



Search Tweets

Overview Guides API Reference

API Reference contents

Standard search API

Premium search API

Enterprise search APIs

Standard search API

Returns a collection of relevant Tweets matching a specified query.

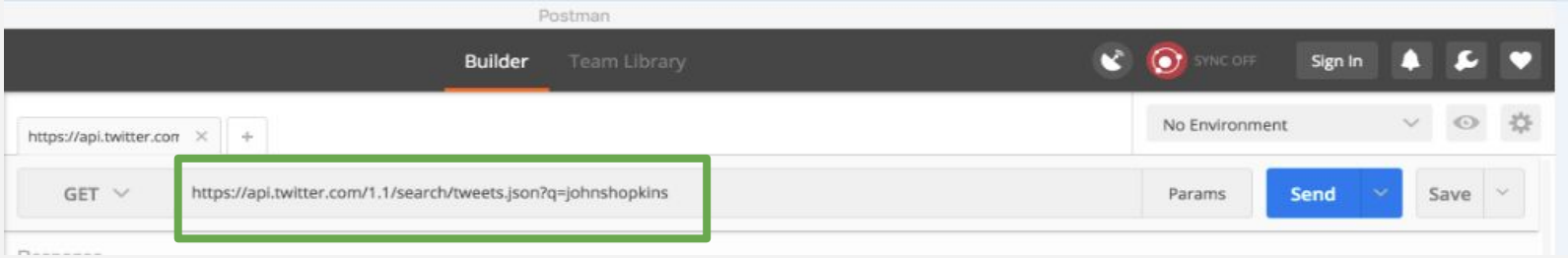
Please note that Twitter's search service and, by extension, the Search API is not meant to be an exhaustive source of Tweets. Not all Tweets will be indexed or made available via the search interface.

To learn how to use Twitter Search effectively, please see the Standard search operators page for a list of available filter operators. Also, see the Working with Timelines page to learn best practices for navigating results by `since_id` and `max_id`.

Resource URL

https://api.twitter.com/1.1/search/tweets.json

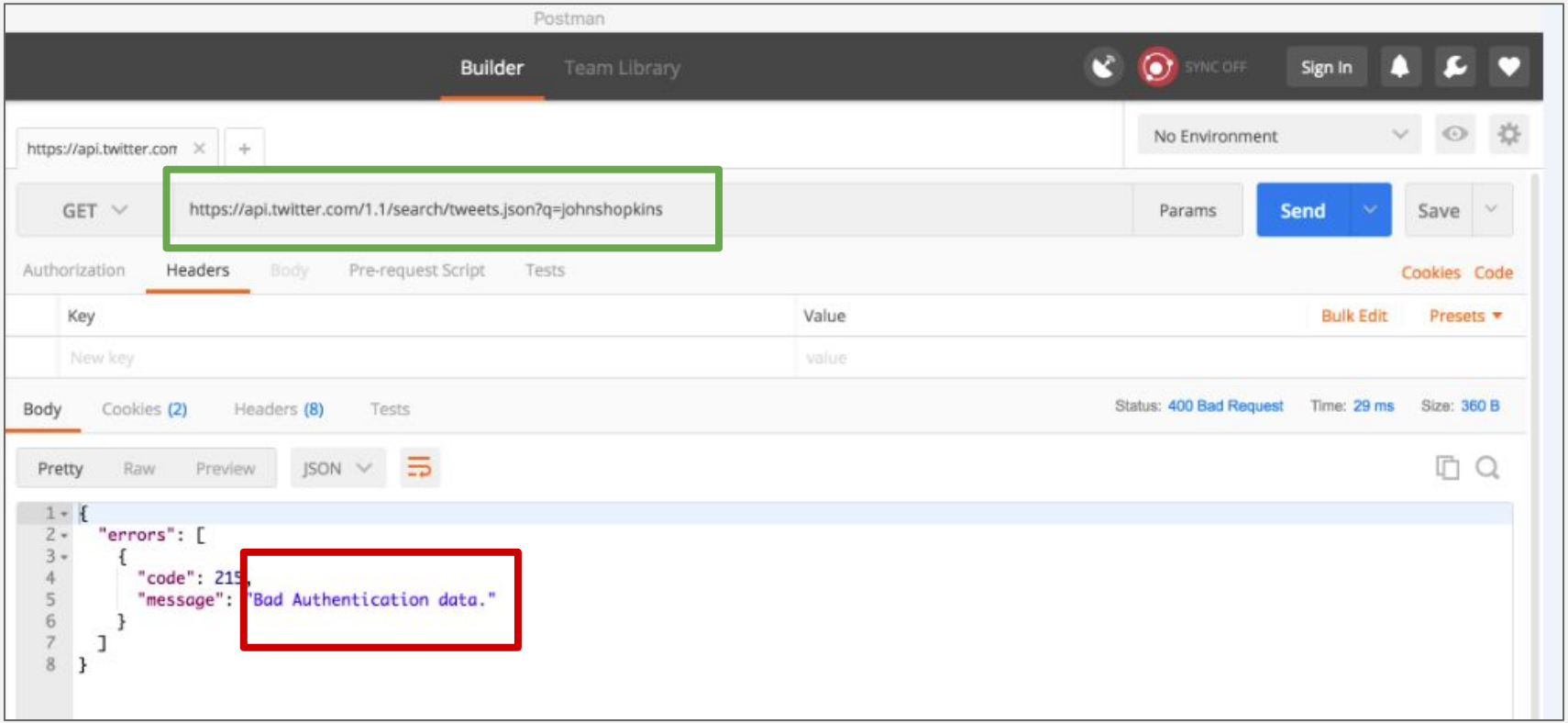
Using an API to collection records



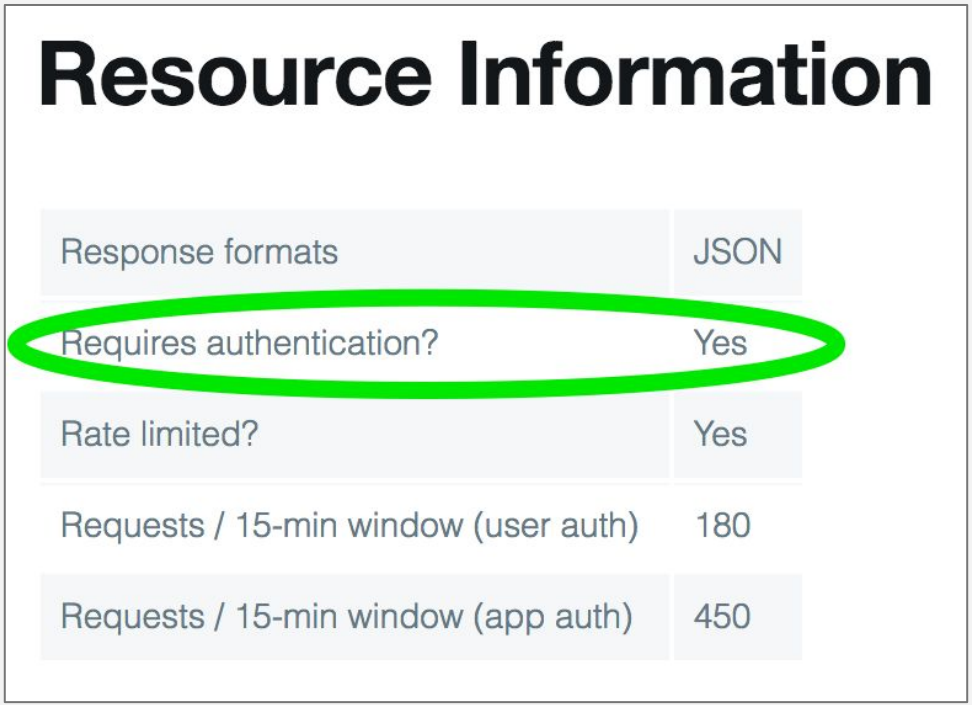
https://api.twitter.com/1.1/search/tweets.json?q=johnshopkins

Let's give it a try!

Using an API to collection records



Using an API to collection records



Resource Information

Response formats	JSON
Requires authentication?	Yes
Rate limited?	Yes
Requests / 15-min window (user auth)	180
Requests / 15-min window (app auth)	450

Source:
https://developer.twitter.com/en/docs/tweets/search/api-reference/get-search-tweets

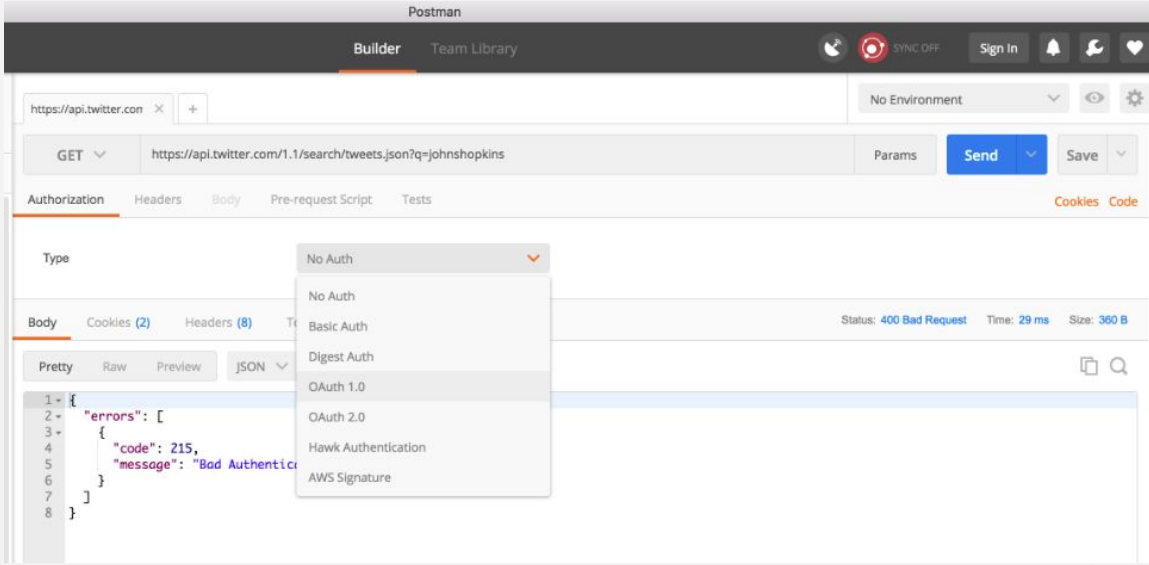
Using an API to collection records

So, we must **authenticate** in order to use Twitter's search API.

The type of authentication Twitter requires is **OAuth**, an open protocol for authentication (see: https://oauth.net)

Postman will help walk us through OAuth1.0 authentication, but you must have a Twitter developer account in order to do so!

Show and tell!



Using an API to collection records

In the end, we've got JSON. Though, be wary of the Twitter **developer license** you signed!


```
{
  "statuses": [
    {
      "created_at": "Tue Apr 18 14:16:03 +0000 2017",
      "id": 854337735129092096,
      "id_str": "854337735129092096",
      "text": "RT @ScienceMarchCLE: Great recognition for a world-class scientist from #CLE! https://t.co/i2GtX37EMr #ScienceMarchCLE @JohnsHopkins",
      "truncated": false,
      "entities": {
        "hashtags": [
          {
            "text": "CLE",
            "indices": [
              72,
              76
            ]
          },
          {
            "text": "ScienceMarchCLE",
            "indices": [
              102,
              118
            ]
          }
        ],
        "symbols": [],
        "user_mentions": [
          {
            "screen_name": "ScienceMarchC
            "name": "March For ScienceCL
            "id": 824422865206472706,
            "id_str": "824422865206472706
            "indices": [
              3,
              19
            ]
          },
          {
            "screen_name": "JohnsHopkins"
            "name": "Johns Hopkins U.",
            "id": 14441010,
            "id_str": "14441010",
            "indices": [
              119,
              132
            ]
          }
        ],
```

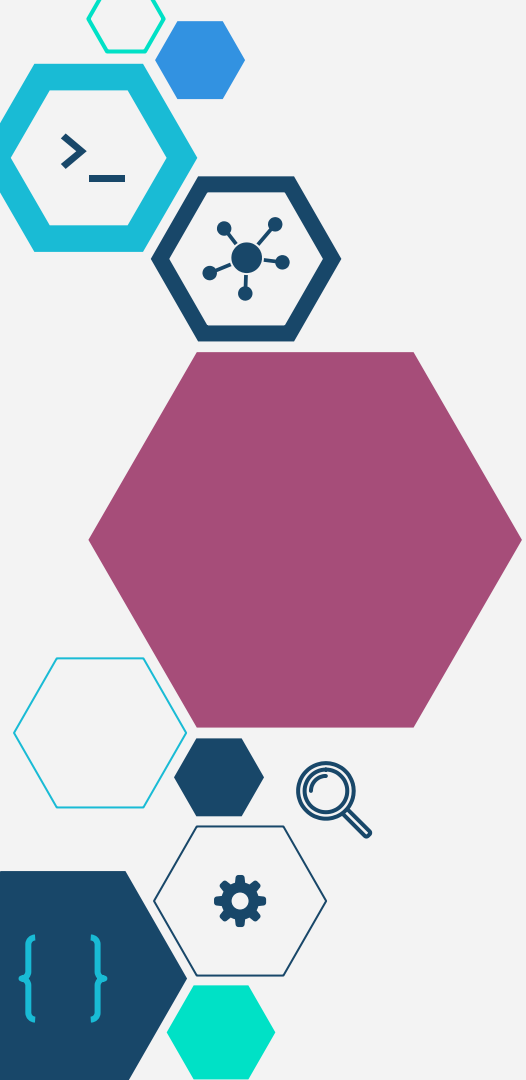

Tweet compliance

Overview Guides API Reference

One of Twitter's core values is to defend and respect the user's voice. This includes respecting their expectations and intent when they delete or modify the content they choose to share on Twitter. We believe that this is critically important to the long term health of one of the largest public, real-time information platforms in the world. Twitter puts controls in the hands of its users, giving individuals the ability to control their own Twitter experience. We believe that business consumers that receive Twitter data have a responsibility to honor the expectations and intent of end users.

Questions

?

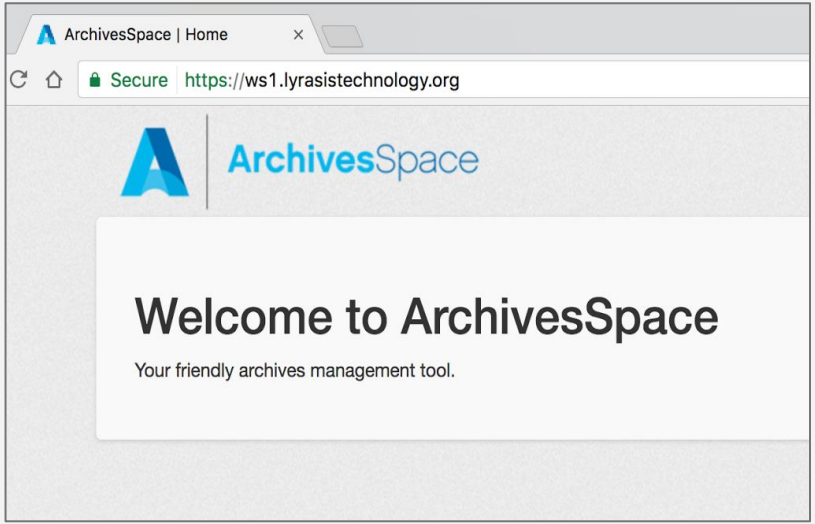


The ASpace API

Finally!

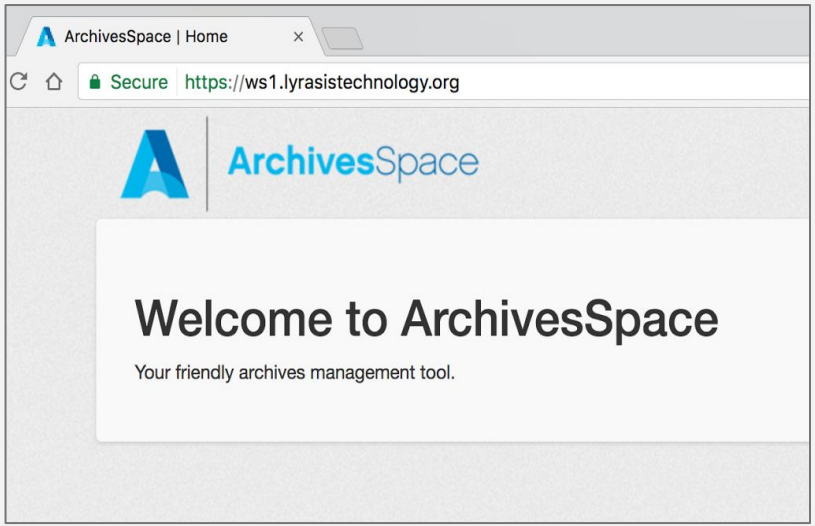
ArchivesSpace!

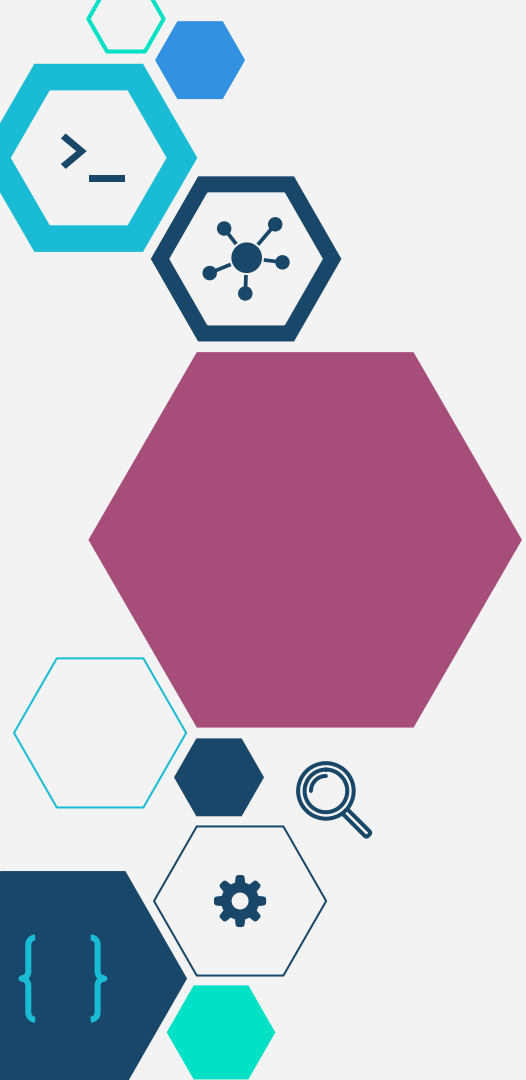
- You've each been provided a test instance of ArchivesSpace by our gracious hosts at Lyrasis
- The address of the **staff interface** of your instance is: https://ws[your#].lyrtech.org/staff/ (ex. https://ws0.lyrtech.org/staff/)
- The address of the **API** of your instance is: https://ws[your#].lyrtech.org/staff/api
- Go check out the **staff interface** now!
  username: admin
  password: admin
- You already know your number!



ArchivesSpace!

- We will be in ASpace for the rest of the day
- We will be starting with three simple actions, then stepping up in complexity.
- These scenarios are specific and may not help you; think about your own work as a series of steps, not circumstances
- Ask us questions!



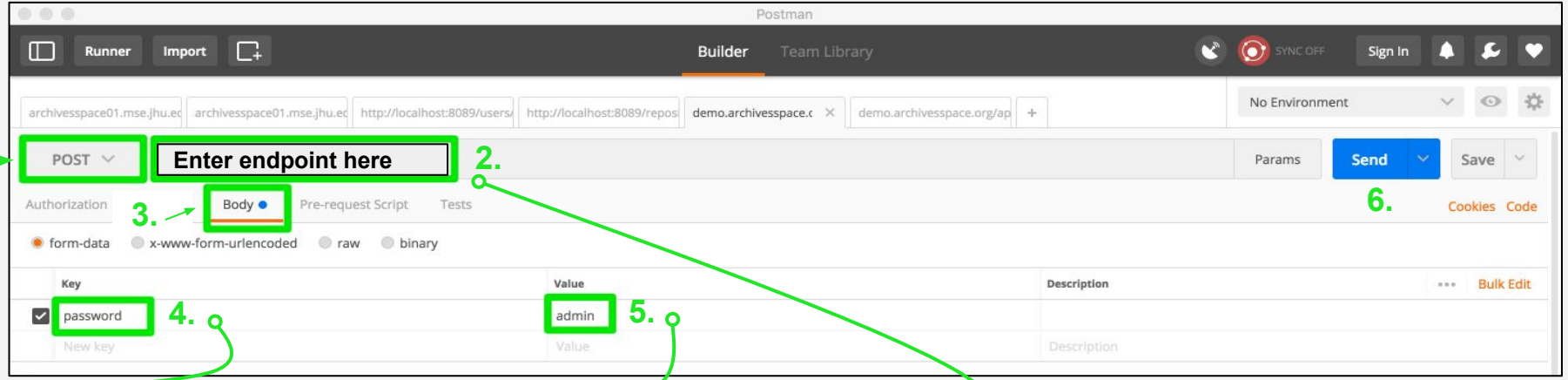


Authenticate and GET

ASpace with GUI

Authenticate to AS with GUI

Before we do anything, we need to authenticate (just like you do when logging in) so let's do that in 6 steps:



Type **password** under Key ; type **admin** under Value

Endpoint: https://ws[your#].lyrtech.org/staff/api/users/admin/login

Authenticate to AS with GUI

We just POSTed our username and password. In return, AS acknowledges that we have permission to proceed and provides a “session key.” We need to copy that.

Authenticate to AS with GUI

Now we're going to take our authentication info and put it where it should go, in 3 steps:

Type **X-ArchivesSpace-Session** under Key ; paste from the clipboard under Value

GET from AS with GUI

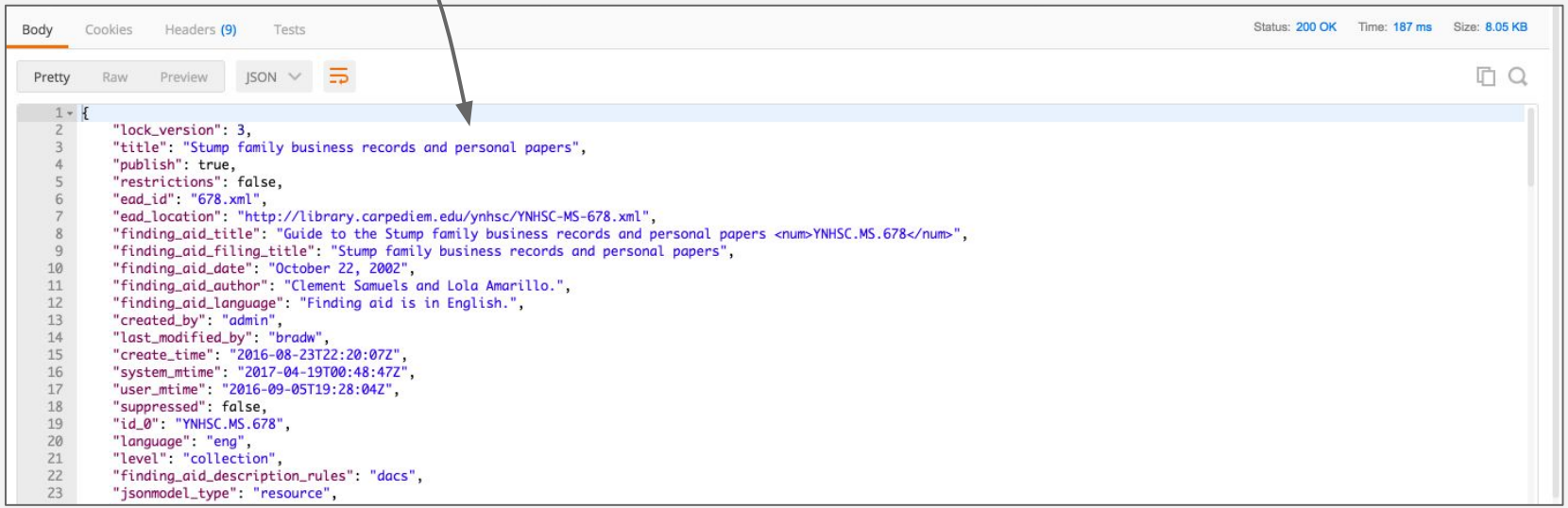
With our authentication info in place, let's GET a resource record in 3 steps.

Endpoint: https://ws[your#].lyrtech.org/staff/api/repositories/2/resources/2

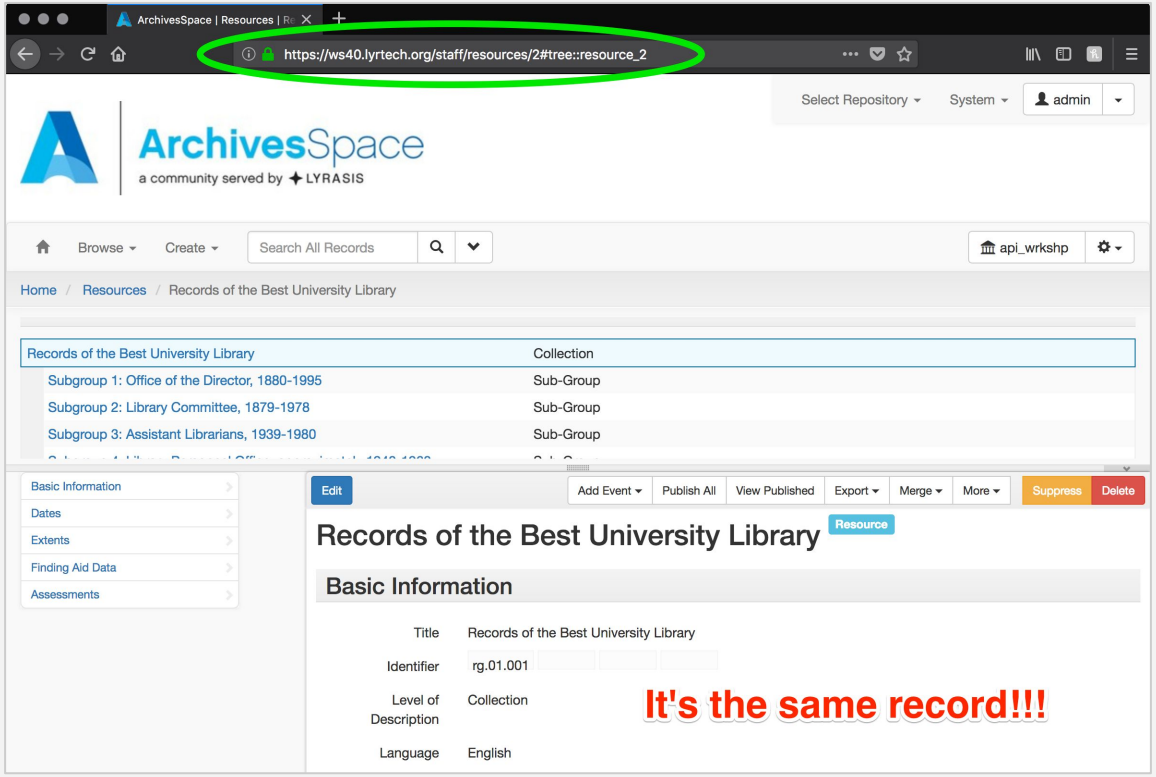
GET

GET from AS with GUI

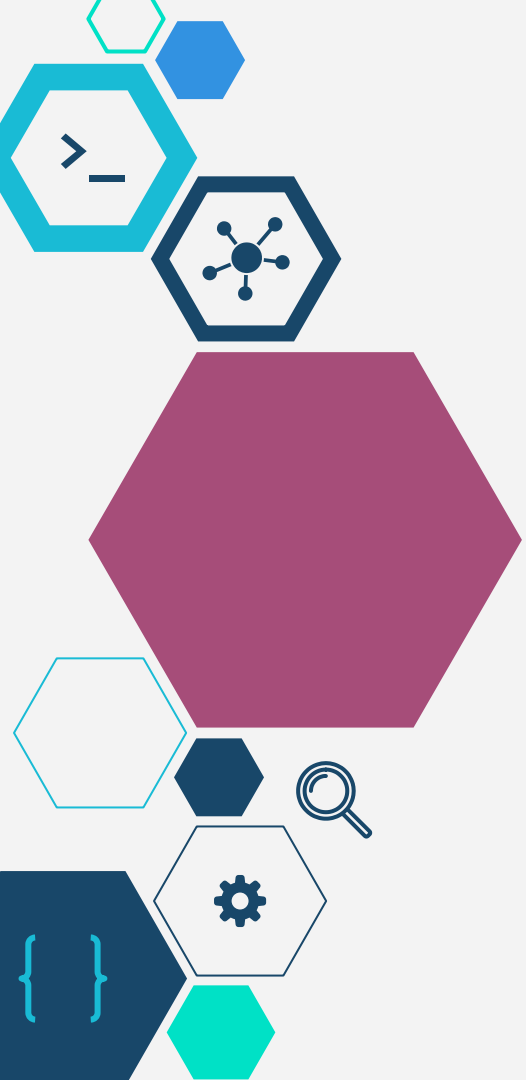
Did you get a result?



GET from AS with GUI



Instructor cue: Auth and GET if you haven't already



POST to ASpace

POST to AS

Scenario: Back when JHU migrated to 1.5.4, we found ourselves with the following problems. They are simple but good introductions to project work with the API.

1. We didn't have **container profiles** in ArchivesSpace and wanted to, so we needed to create some
2. In following the migration instructions for 1.5, we had to add **faux codes** to our containers; we'd like to use our *actual* barcodes
3. Once we created container profiles, we needed to link them to *actual* **top containers**

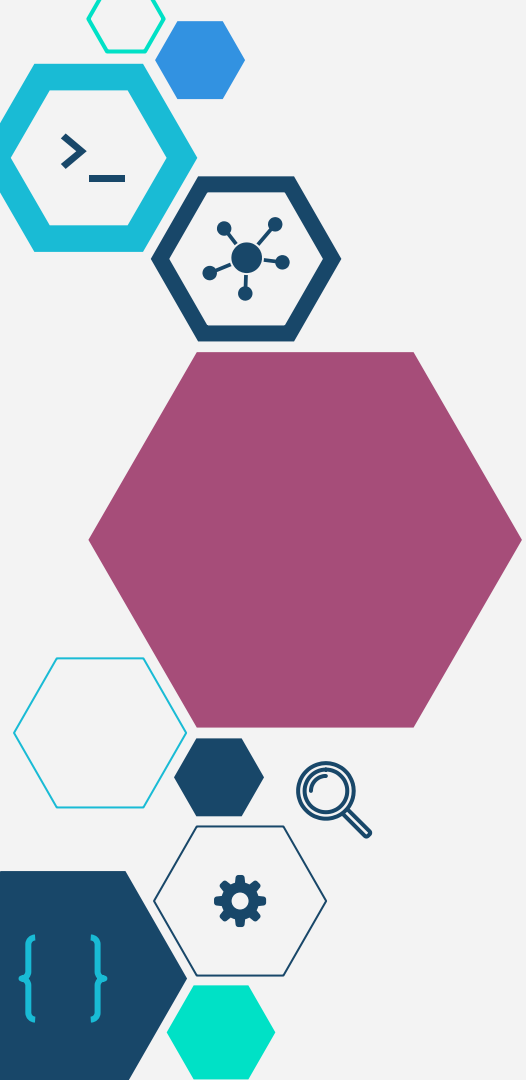
If you switch out **"container profiles"** for **"agent records"** or **"subject headings"** or **"digital objects,"** the steps are similar and will likely transfer. Namely:

- Create new records
- Modify existing records
- Link records

Sidebar: Repurposing results

Get data out → Do something to it → Put it back in

Lunch!



POST to ASpace with GUI

Container profiles

Creating new records

Container profiles

What's a container profile?

A simple record consisting of dimensions. Height x Width x Depth.

They are an optional feature of the container management module.

They are very SIMPLE, which is why we're using them.

POST to AS with GUI

Container profiles ⟶ Creating new records

The JSON for a profile is simple and easy to read:

1. Open Atom
2. Open “recordCenterProfile.json” from the directory with our cloned GitHub repo
3. Packages > Atom Beautify > Beautify
4. Here is the container profile for a record center carton in JSON, ready to go
5. Copy the whole thing (including brackets), and go back to Postman

recordCenterProfile.json


```
{
    "name": "Record center box",
    "extent_dimension": "width",
    "height": "10.5",
    "width": "13.1",
    "depth": "15.75",
    "dimension_units": "inches",
    "jsonmodel_type": "container_profile"
}
```

POST to AS with GUI

Container profiles ⟶ Creating new records

Endpoint:

https://ws[your#].lyrtech.org/staff/api/container_profiles

POST to AS with GUI

Container profiles ⟶ Creating new records

Pretty Raw Preview JSON Save Response


```
{
    "status": "Created",
    "id": 1,
    "lock_version": 0,
    "stale": null,
    "uri": "/container_profiles/1",
    "warnings": []
}
```


Success!!

POST to AS with GUI

Container profiles ⟶ Creating new records

Let's do more!

1. Go back to Atom and open "containerProfiles.json"
2. Packages > Atom Beautify > Beautify
3. Here are *all* the profiles, ready to go
4. Copy everything, and go back to Postman


```
[{
        "name": "Flat box01",
        "extent_dimension": "width",
        "height": "3",
        "width": "12",
        "depth": "16",
        "dimension_units": "inches",
        "jsonmodel_type": "container_profile"
    },
    {
        "name": "Flat box02",
        "extent_dimension": "width",
        "height": "3",
        "width": "21",
        "depth": "25",
        "dimension_units": "inches",
        "jsonmodel_type": "container_profile"
    },
    {
        "name": "Flat box03",
        "extent_dimension": "width",
        "height": "3",
        "width": "9",
        "depth": "11",
        "dimension_units": "inches",
        "jsonmodel_type": "container_profile"
    },
```

POST to AS with GUI

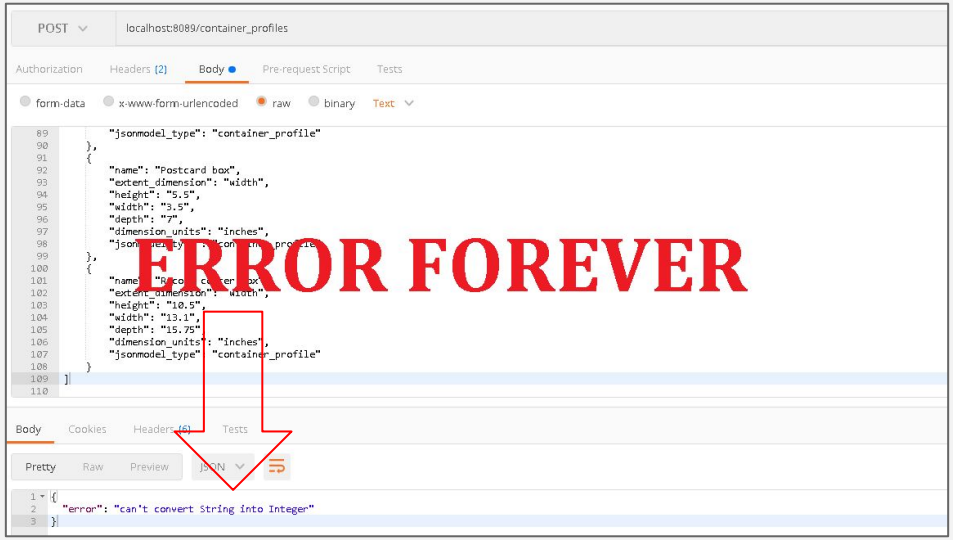
Container profiles ⟶ Creating new records

Endpoint:

https://ws[your#].lyrtech.org/staff/api/container_profiles

POST to AS with GUI

Container profiles ⟶ Creating new records



Don't hate us: you <u>cannot</u> post multiple records through the GUI. This frustrating exercise might save you a month.

(use your month wisely: take a vacation from computers)


Breathe
(hydrate)

Scripting

>_

Scripting


MARION ELECTRICAL INSTRUMENT CO. U.S.A.
5
10
0
15
YOUR ENTHUSIASM
DIRECT CURRENT
MR36W015DCMA
HM3

Scripting - Why?

Using a GUI application like **Postman** to interact with APIs can be a great way to *learn*, *explore*, and *troubleshoot*, but ultimately you'll hit a brick wall:

- It takes an **awful lot of clicks** to get out a small amount of data (relatively speaking)
- If you want to get multiple full records OUT you've got to run a GET as **many times** as there are records you want to retrieve
- While you can POST many one-off changes using a GUI like Postman, you can rarely get a GUI to make **intelligent, iterative POSTs at scale**
- **Manually authenticating** is a pain
- Though we told you that you will be sometimes playing the role of "application" in this API world, you don't *always* want to **be the application**!

Scripting - How?

Yes, this is a huge barrier to entry for most users, but it can be mitigated:

- We (defined here as both **archivists** and **developers**) are a **community** that likes sharing!
- There is no **“one right language”** to make this work
  - If you have *any* prior knowledge of a particular scripting language, **start there**
  - All the scripts you will use in this workshop are **python** because: 1) python was Lora’s preferred hammer, and 2) unscientifically speaking, it seems that python is the preferred language of archivists (which means there’s more to ~~steal~~ borrow)
  - But, if you want, you can use a **Ruby** or **Perl** or **PHP** or **JavaScript** shaped hammer!
- The Internet is full of **helpful advice**!
  - Just don’t feed the trolls
  - And StackOverflow is an amazing resource, filled with answers that may or may not work out for you.

Scripting- No really, how?

Remember all the legwork you did both at home and during the early part of this workshop? You've:

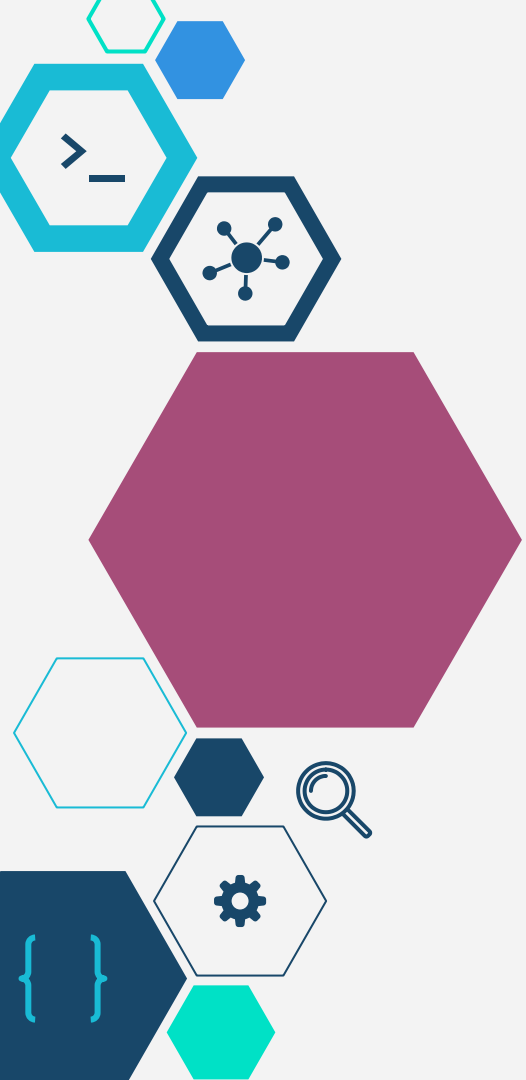
- Installed applications, including the **text editor** *Atom*
- Installed (or located) a **shell**, namely *Terminal* (Mac) or *Cygwin* (Windows)
- Installed (or confirmed installation of) **python3**

Guess what? You've set up a **python3 development environment** already! Good work!

With that work complete, for the remainder of this workshop you should only need to type `python3 [name of script here].py` into Terminal/Cygwin, and you'll be **executing python3 scripts!** Just remember:

- You should be located in the same directory as the script (and any files it is reliant on) before you type your command (you can always `ls` to confirm the script is there!)

For more, see: http://www.shubhro.com/2014/05/29/development-environment/ and/or http://python3-guide-pt-br.readthedocs.io/en/latest/starting/install/osx/ (Mac specific)



POST to ASpace with Script

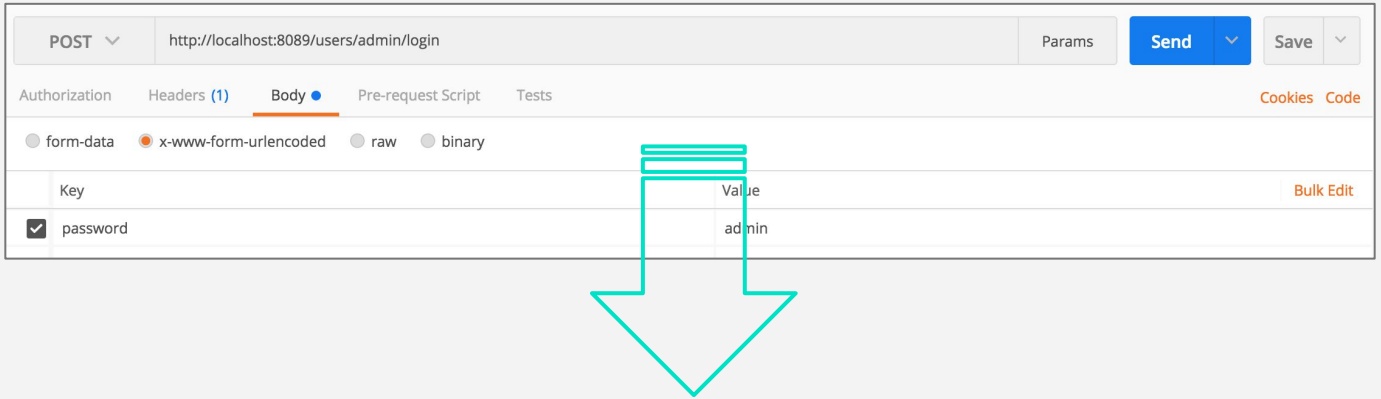
Container profiles

Creating new records

POST to AS with Script

Container profiles ⟶ Creating new records

Remember that before we start posting to AS, we need to authenticate, so how do we do that with scripts?




```
secrets.py
1  baseURL='http://localhost:8089'
2  user='admin'
3  password='admin'
```

POST to AS with Script

Container profiles ⟶ Creating new records

"Keep it secret, keep it safe." - Gandalf the Grey

secrets.py

```
1  baseURL='http://localhost:8089'
2  user='admin'
3  password='admin'
```

This means no manual authenticating! Learn to script just for that and call it a win!

```
auth = requests.post(baseURL + '/users/'+user+'/login?password='+password).json()
session = auth["session"]
headers = {'X-ArchivesSpace-Session':session, 'Content_Type':'application/json'}
```

POST to AS with Script

Container profiles ⟶ Creating new records

- We're all connecting to different instances of ArchivesSpace (so we don't clash with each others' work!), so we need to tell **secrets.py** where each of our individual instances live.
- Open **secrets.py** in Atom
- Change the line:
  baseURL='http://localhost:8089'
  to
  baseURL='https://ws[your#].lyrtech.org/staff/api'

- Then save.

POST to AS with Script

Container profiles ⟶ Creating new records

Mac

1. In the Finder navigate to your api-training directory
2. Ctrl+click the api-training directory, and select “New Terminal at Folder”
3. Type `ls` and examine the contents of that folder
4. Type `python3 postContainerProfiles.py` (case sensitive!) and hit enter
5. Navigate back to AS in your browser (https://ws[your#].lyrtech.org/staff/container_profiles)

PC

1. Open Cygwin
2. Type `cd api-training` to enter the api-training directory
3. Type `ls` and examine the contents of that folder
4. Type `python3 postContainerProfiles.py` (case sensitive!) and hit enter
5. Navigate back to AS in your browser (https://ws[your#].lyrtech.org/staff/container_profiles)

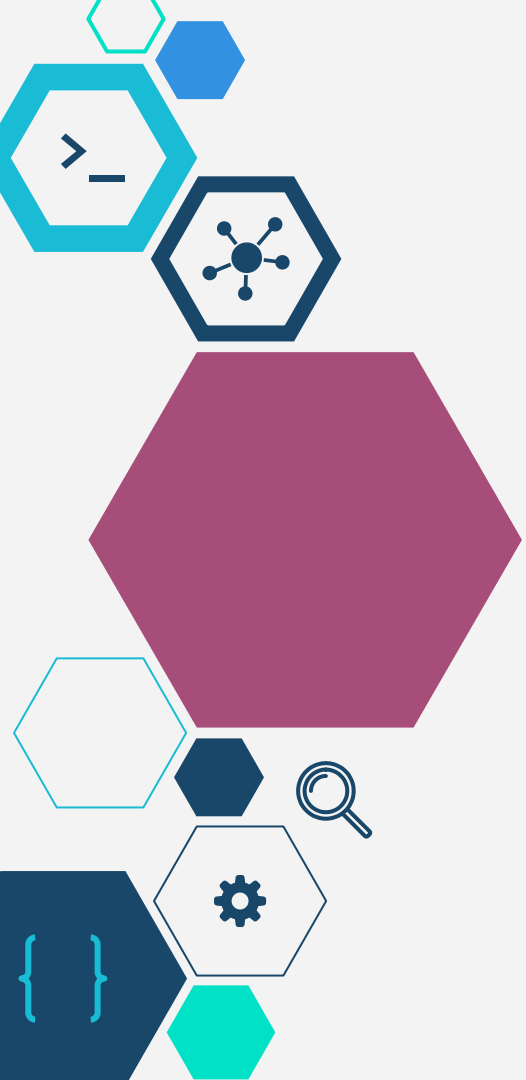
POST to AS with Script

Container profiles ⟶ Creating new records

Scenario: Back when JHU migrated to 1.5.4, we found ourselves with the following problems. They are simple but good introductions to project work with the API.

1. ~~We didn't have **container profiles** in ArchivesSpace and wanted to, so we needed to create some~~
2. In following the migration instructions for 1.5, we had to add **faux codes** to our containers; we'd like to use our *actual* barcodes
3. Once we created container profiles, we needed to link them to *actual* **top containers**

Pro-tip: Remember that this container profile scenario represented "create" in AS; if you have records you need to create, this is how to start thinking about them.



POST to ASpace with Script

Edit barcodes

Modify existing records

Barcodes

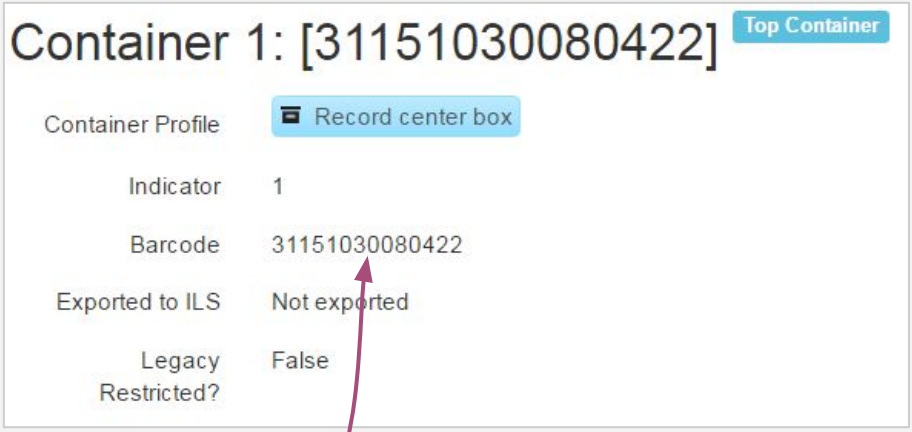
Every container in your collections gets a record, and this is called a top container.

Simply put, this is the thing you put a number on: Box 1.

So, your archives might have hundreds or thousands of boxes called “Box 1”

AS assigns its own unique number to every top_container, but you might also want to use YOUR unique number.

At JHU, we wanted to use our actual barcodes.



POST to AS with Script

Edit barcodes ⟶ Modify existing records

1. Navigate to the Gérard Defaux papers (https://ws[your#].lyrtech.org/staff/resources/1)
2. Expand Research Materials > click on any file > scroll down to Instances > see fake barcodes

These are the barcodes generated by the barcoder plugin. Hopkins has thousands of them.

3. Navigate to our GitHub and look at *barcodes.csv*

POST to AS with Script

Edit barcodes ⟶ Modify existing records

Mac

1. Type `ls` and examine the contents of that folder
2. Type `python3 postBarcodes.py` (case sensitive!) and hit enter
3. Navigate back to AS in your browser (https://ws[your#].lyrtech.org/staff/resources/1)

PC

1. Type `ls` and examine the contents of that folder
2. Type `python3 postBarcodes.py` (case sensitive!) and hit enter
3. Navigate back to AS in your browser (https://ws[your#].lyrtech.org/staff/resources/1)

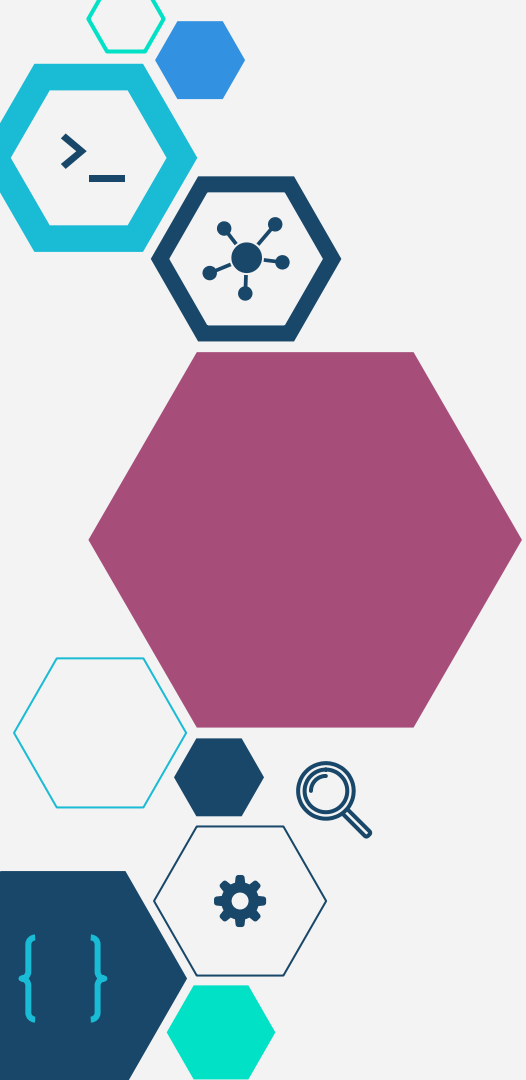
POST to AS with Script

Edit barcodes ⟶ Modify existing records

Scenario: Back when JHU migrated to 1.5.4, we found ourselves with the following problems. They are simple but good introductions to project work with the API.

1. ~~We didn't have **container profiles** in ArchivesSpace and wanted to, so we needed to create some~~
2. ~~In following the migration instructions for 1.5, we had to add **faux codes** to our containers; we'd like to use our *actual* barcodes~~
3. Once we created container profiles, we needed to link them to *actual* **top containers**

Pro-tip: Remember that this scenario represented "edit record" in AS; if you have records you need to edit, this is how to start thinking about it.



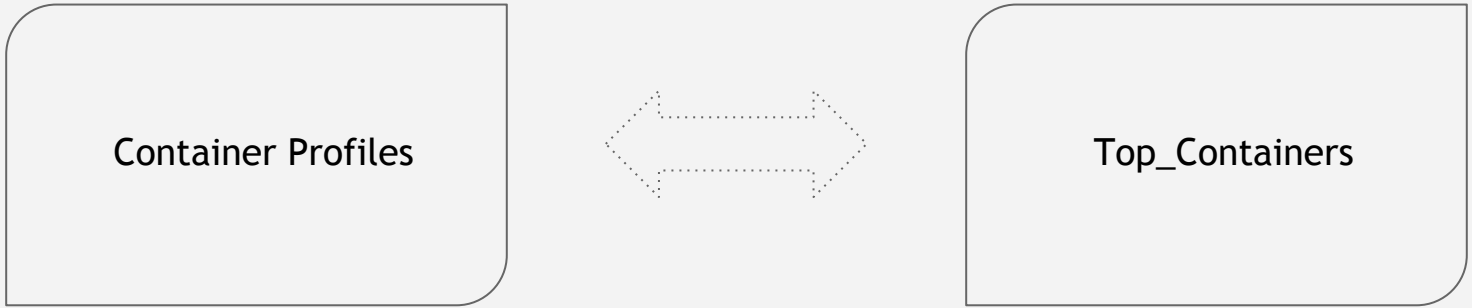
POST to ASpace with Script

Link profiles

Link existing records

Linking profiles to containers

Link profiles ⟶ Link existing records



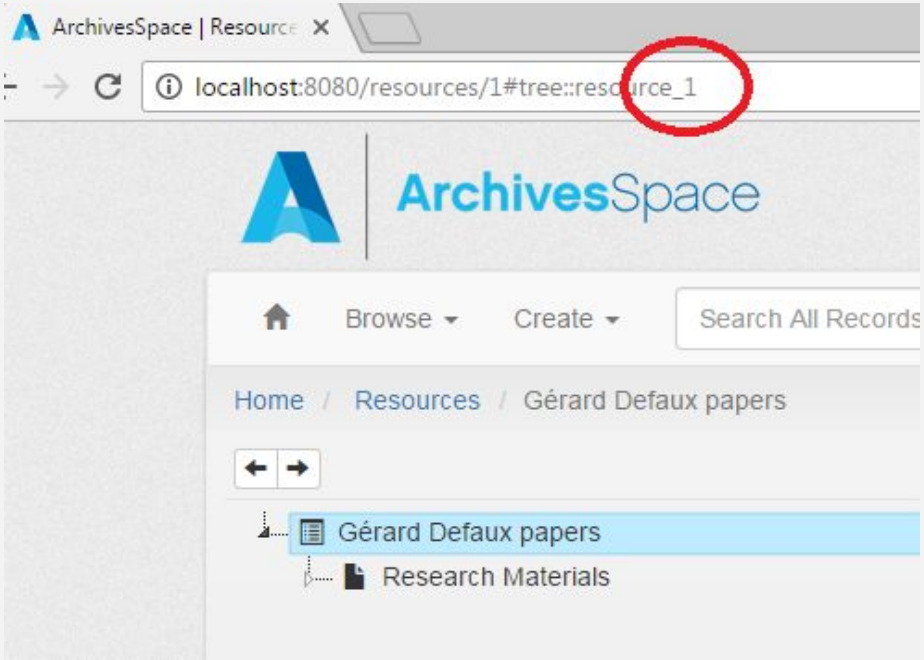
1. Type `ls`
2. Type `python3 asLinkProfiles.py` (case sensitive!)
3. The script should prompt you for a few things...

POST to AS with Script

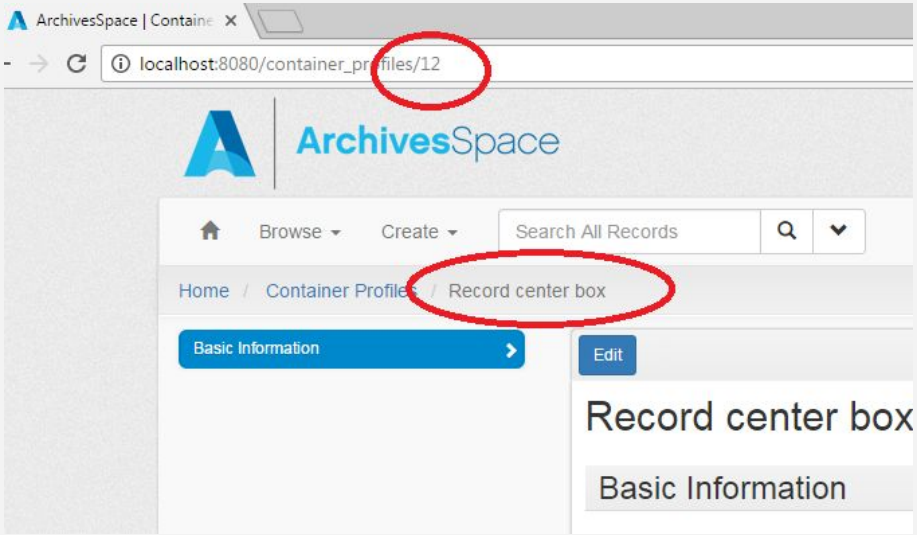
Link profiles ⟶ Link existing records

The interface — just another lens on the same data — is helpful for constructing API requests.

View a resource record for its resource number:



View a container profile for its profile number:



POST to AS with Script

Link profiles ⟶ Link existing records

Scenario: Back when JHU migrated to 1.5.4, we found ourselves with the following problems. They are simple but good introductions to project work with the API.

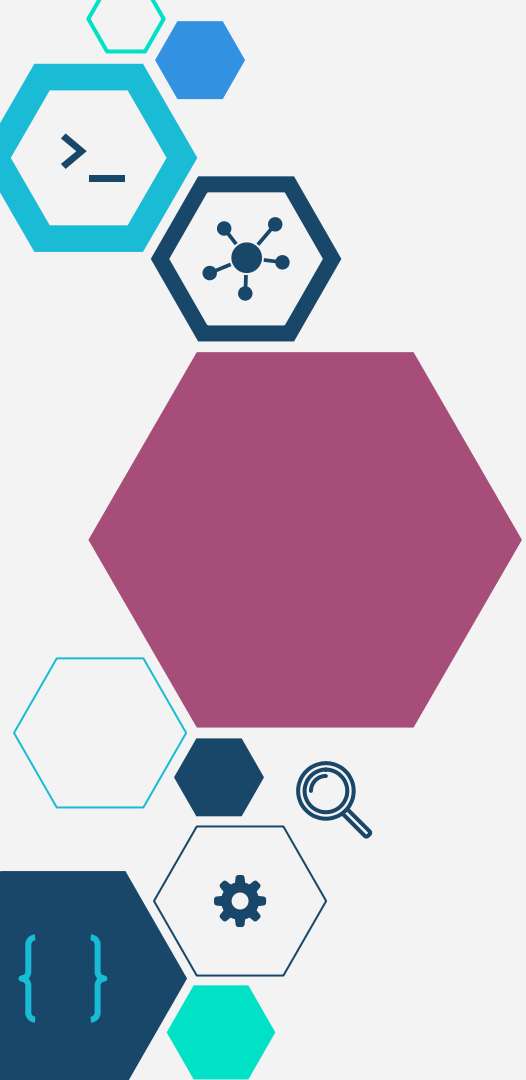
1. ~~We didn't have **container profiles** in ArchivesSpace and wanted to, so we needed to create some~~
2. ~~In following the migration instructions for 1.5, we had to add **faux codes** to our containers; we'd like to use our *actual* barcodes~~
3. ~~Once we created container profiles, we needed to link them to *actual* **top containers***~~

Pro-tip: Remember that this link scenario represented any "link" in AS; if you have records you need to link, this is how to start thinking about it.

Remember this

- You did it!
- Yes, we wrote the scripts.
- But now they’re yours.

- If you wish to move forward, remember these. Examine them, and those we provide at the end of the workshop.



Bonus:
Create Digital Objects in Bulk

One more simple create script

Create DOs in Bulk

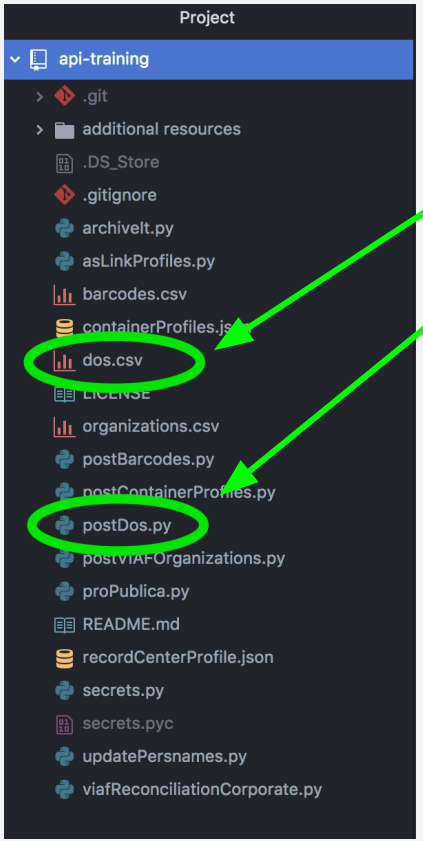
Workshop feedback works!

Scenario:

We have a spreadsheet of bare minimal digital object description that we just want to get into ArchivesSpace with little effort.

Create DOs in Bulk

Workshop feedback works!



Spreadsheet of digital objects (dos.csv)

Script to post new digital objects (postDos.py)

```
39  # Construct JSON to post from csv
40  doList = []
41  for row in csv_dict:
42      file_uri = row['fileuri']
43      title = row['title']
44      digital_object_id = row['objectid']
45      doRecord = {'title': title, 'digital_object_id': digital_object_id,
46      doRecord['file_versions'] = [{'file_uri': file_uri, 'publish': Fals
47      doRecord = json.dumps(doRecord)
48      post = requests.post(baseURL + '/repositories/2/digital_objects', h
49      print post
50      # Save uri to new csv file
51      uri = post['uri']
52      f.writerow([title]+[digital_object_id]+[uri])
53
54  # Feedback to user
55  print 'New .csv saved to working directory.  Go have a look!'
56
```

Let's take a look at this script!

Create DOs in Bulk

Workshop feedback works!

Mac

1. Type `python3 postDos.py` and hit enter
2. Navigate back to AS in your browser (https://ws[your#].lyrtech.org/staff/resources/1)
3. Browse > Digital Objects

Explore results in ASpace...

4. Switch to Atom and explore the postDos.py script
5. Can you tell where the feedback/results from running the script might have been saved from reviewing the script?

PC

1. Type `python3 postDos.py` and hit enter
2. Navigate back to AS in your browser (https://ws[your#].lyrtech.org/staff/resources/1)

Explore results in ASpace...

3. Switch to Atom and explore the postDos.py script
4. Can you tell where the feedback/results from running the script might have been saved from reviewing the script?


Breathe

Until now, we've been following the basic construct we introduced earlier

COMMAND | Execution

Now we're moving into complex scenarios where multiple commands get executed at once.

API to API:
GET MARC records

GET MARC records

Demo time!

GET MARC records

Mac

1. Type `ls` and examine the contents of the folder
2. Type `python3 postMARCtoArchivesSpace.py`

PC

1. Type `ls` and examine the contents of the folder
2. Type `python3 postMARCtoArchivesSpace.py`

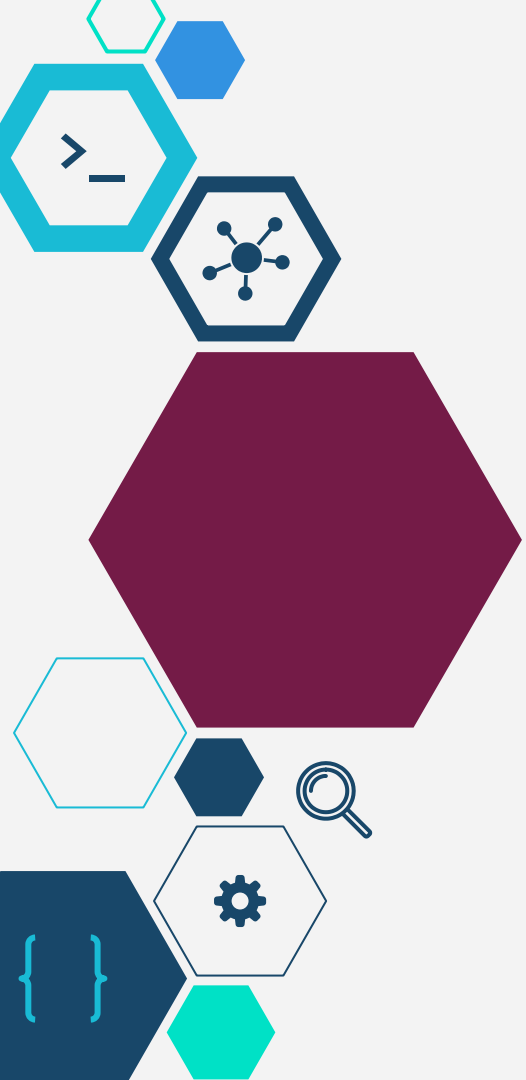
Questions

?

Leveraging linked data - SNAC

Leveraging linked data

Let’s talk about SNAC!

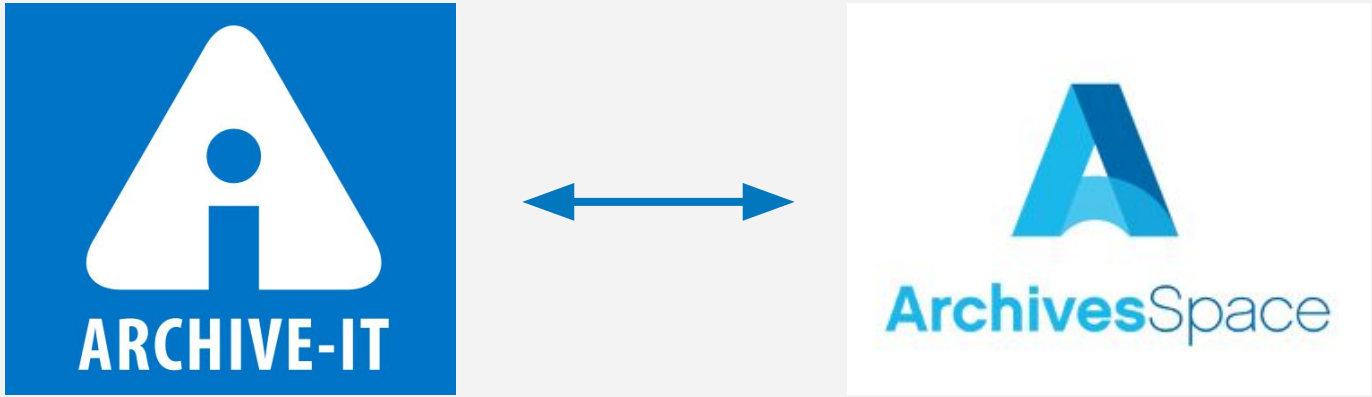


App-to-app communications

Automating (some) archival description

App-to-app communication

Scenario: As your university's web archivist, you wish to make your Archive-It web crawls accessible to users who access your collections via ArchivesSpace without having to individually create digital objects every time you run a new Archive-It crawl.



App-to-app communication

1. In ArchivesSpace, navigate to the “Records of the Best University Library” resource

2. Expand Subgroup 7: Library Website

3. Click on library.jhu.edu

4. Note that archival object’s level

App-to-app communication

We now know that we can access ArchivesSpace's archival object records via the **ArchivesSpace API**, right?

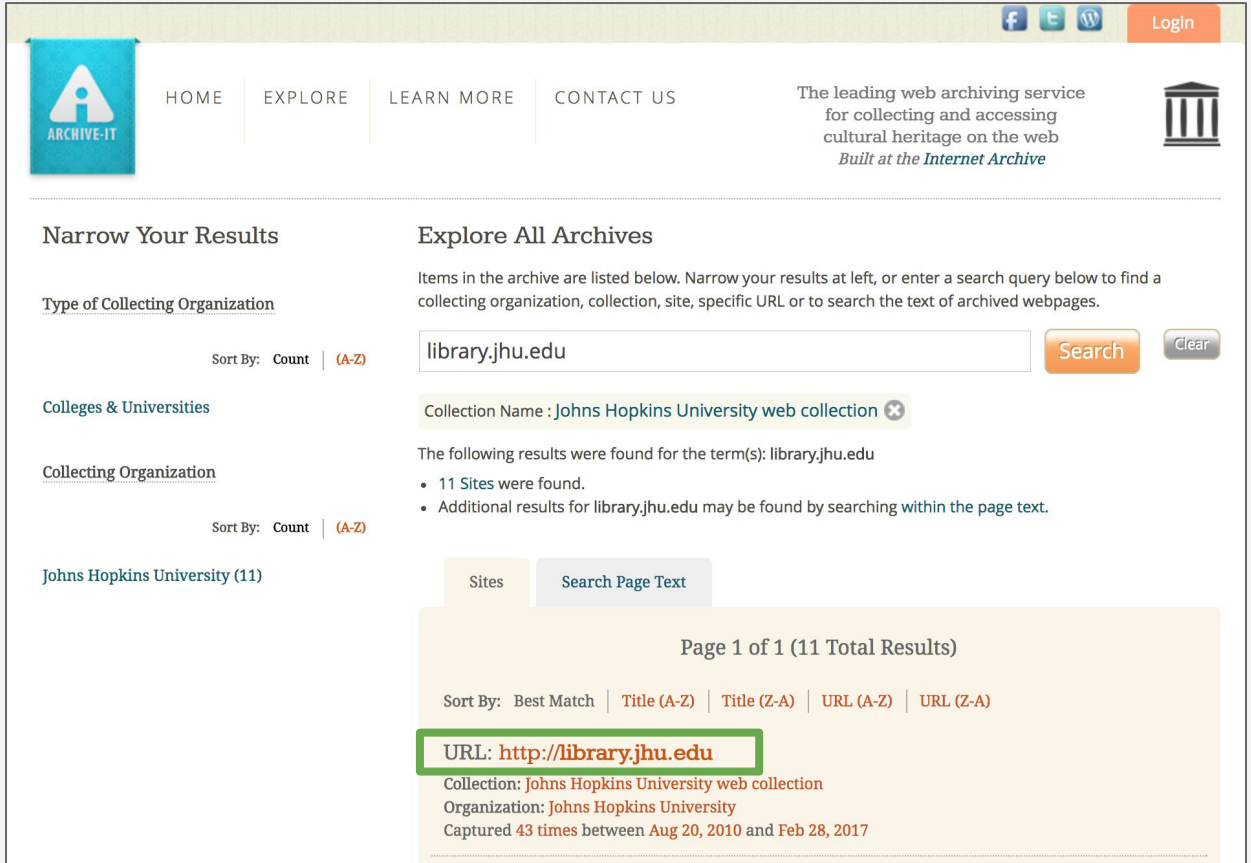
In fact, with a decent enough search we could probably even have a script return JUST those archival objects with the level **"Web archive."**

Since we're going to want to keep programmatically working with/altering this data after we find it, we'll use a **python3 script**, instead of Postman to run this search.

```
39    # search AS for archival_object's with level "Web archive"
40    query =
      '/search?page=1&filter_term[]={"primary_type":"archival_object"}&filter_term[]={"level":"Web
      archive"}'
41    ASoutput = requests.get(baseURL + query, headers=headers).json()
42    print 'Found ' + str(len(ASoutput['results'])) + ' archival objects with the instance type "Web
      archive."'
```

Code snippet from archiveIt.py

App-to-app communication



App-to-app communication

Does **Archive-It** have an **API** we can use to access this information we're seeing in our browsers?

YES!

INTERNET ARCHIVE
WayBackMachine

Wayback Machine APIs

The Internet Archive Wayback Machine supports a number of different APIs to make it easier for developers to retrieve information about Wayback capture data.

The following is a listing of currently supported APIs. This page is subject to change frequently, please check back for the latest info.

Updated on September, 24, 2013

Wayback Availability JSON API

This simple API for Wayback is a test to see if a given url is archived and currenlty accessible in the Wayback Machine. This API is useful for providing a 404 or other error handler which checks Wayback to see if it has an archived copy ready to display. The API can be used as follows:

`http://archive.org/wayback/available?url=example.com`
which might return:

```
{
    "archived_snapshots": {
        "closest": {
            "available": true,
            "url": "http://web.archive.org/web/20130919044612/http://example.com/",
            "timestamp": "20130919044612",
            "status": "200"
        }
    }
}
```

App-to-app communication

With the right amount of trial and error, we can also get information about our Archive-It holdings out of the Archive-It API with a **python script** as well!

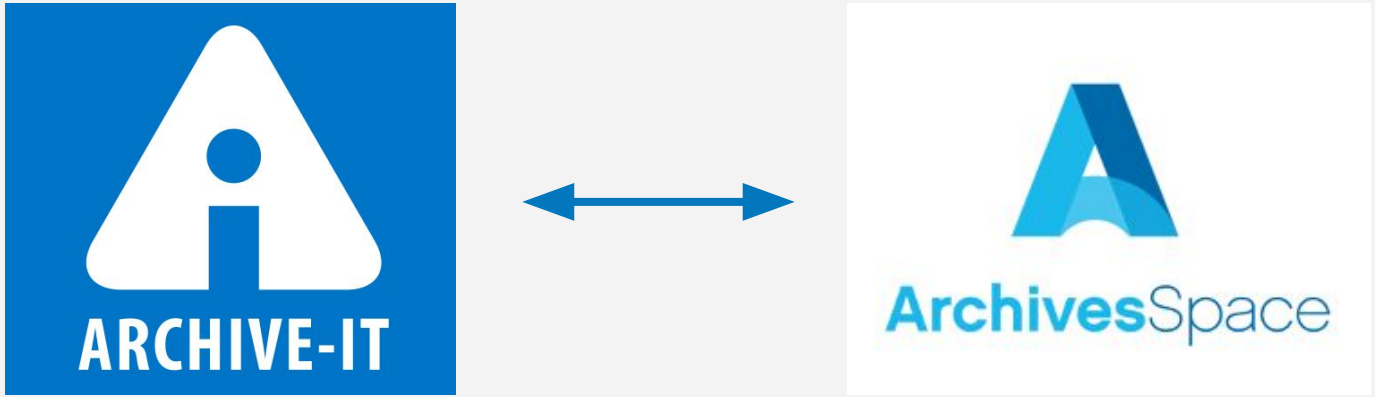
```
67        for crawl in crawlList:
68            doid = 'https://wayback.archive-it.org' + '/' + archiveit_coll + '/' +
              crawl['timestamp'] + '/' + crawl['original']
69            query = '/search?page=1&filter_term[]={"primary_type":"digital_object"}&q=' + doid
70            existingdoID = requests.get(baseURL + query, headers=headers).json()
```

Code snippet from archiveIt.py

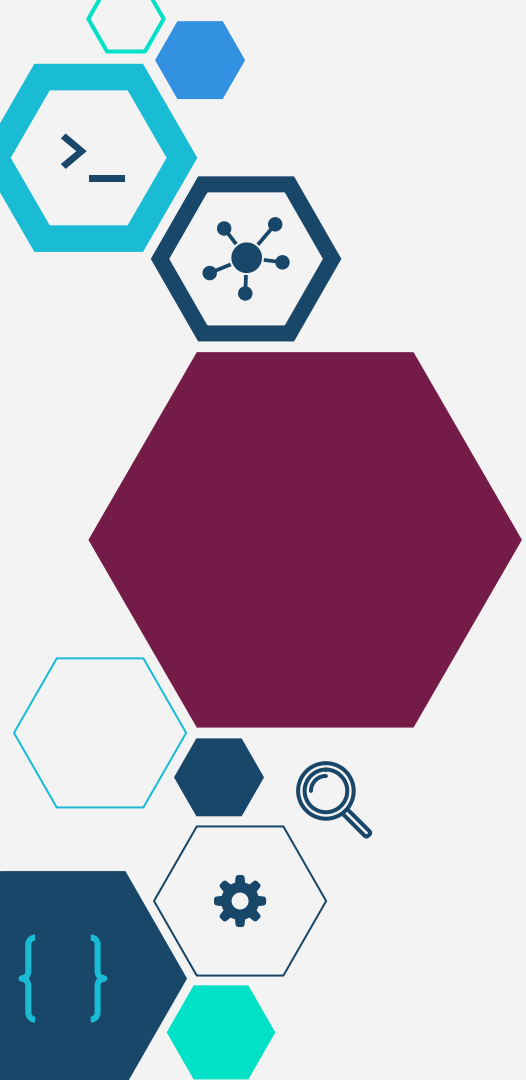
App-to-app communication

1. In Terminal/Cygwin run `python3 archiveIt.py` and let's see what happens!

2. Go check out that "Records of the Best University Library" resource record once again.



Questions ?



Anatomy of a script

Let’s explore a script together

Anatomy of a script

We’ve spent a lot of time running scripts, but not a lot of time talking about them.

Let’s take a break

User stories

Ask questions!

Re-purpose powerful scripts

But… YOU wrote all these scripts. What am **I** going to do?

We're about to show you some powerful scripts that are universal enough to aid in a lot of your real life scenarios.

Unfortunately, they were written in Python 2, and your environment is now setup for Python 3.

Since these aren't ours, we can't promise to re-write them, but e-mail Valerie at vaddoniz@jhu.edu or join our Slack channel and we'll get you set up to use them.

(You really can e-mail Valerie at any time with any question, she really digs helping people)

So right now, bring to mind a scenario. See if any of these sound useful:

Re-purpose powerful scripts

Want to GET all resources on one screen so that you can manipulate something globally?	getResources.py	This GET scripts retrieves all of the resources from a particular repository into a JSON file which is specified in variable 'f' on line 16. This GET script can be adapted to other record types by editing the 'endpoint' variable on line 13 (e.g. 'repositories/[repo ID]/accessions' or 'agents/corporate_entities').
And then post them all back?	postOverwrite.py	This POST script will overwrite existing ArchivesSpace records based the 'uri' and can be used with any ArchivesSpace record type (e.g. resource, accession, subject, agent_people, agent_corporate_entity, archival_object, etc.). It requires a properly formatted JSON file (specified where [JSON File] appears in the 'records' variable on line 13) for the particular ArchivesSpace record type you are trying to post.

Eric Hanson's GitHub: https://github.com/ehanson8/archivesspace-api

Pro-tip: These are basic scripts. You can modify the endpoints of these scripts to GET and then POST all of any record type: accessions, AOs, containers, agents, DOs, etc.

Re-purpose powerful scripts

Do you have legacy data outside AS? If you can format it to JSON, this script will POST new records.	postNew.py	This POST script will post new records to a generic API endpoint based the record type, 'agents/people' in this example. This script can be modified to accommodate other data types (e.g. 'repositories/[repo ID]/resources' or 'agents/corporate_entities'). It requires a properly formatted JSON file (specified where [JSON File] appears in the 'records' variable on line 13) for the particular ArchivesSpace record type you are trying to post.
You can post from a CSV, too. This script posts containers from a CSV and can be modified.	postContainersFromCSV.py	This script works to create instances (consisting of top_containers) from a separate CSV file. The CSV file should have two columns, indicator and barcode. The directory where this file is stored must match the directory in the filePath variable. The script will prompt you first for the exact name of the CSV file, and then for the exact resource or accession to attach the containers to.

Eric Hanson's GitHub: https://github.com/ehanson8/archivesspace-api

Re-purpose powerful scripts

What if you want to see all the AOs associated with a single collection?	getArchivalObjectsByResource.py	A GET script to extract all of the archival objects associated with a particular resource. Upon running the script, you will be prompted enter the resource ID (just the number, not the full URI).
Or unpublish all AOs associated with a single collection?	unpublishArchivalObjectsByResource.py	This script un-publishes all archival objects associated with the specified resource. Upon running the script, you will be prompted enter the resource ID (just the number, not the full URI).

Eric Hanson's GitHub: https://github.com/ehanson8/archivesspace-api

Re-purpose powerful scripts

Links to some Duke scripts (Python, XSLT, SQL) for working with the ArchivesSpace API or backend database: https://github.com/duke-libraries/archivesspace-duke-scripts

- You can find many more examples of these by searching for “ArchivesSpace” on github: https://github.com/search?q=archivesspace+scripts

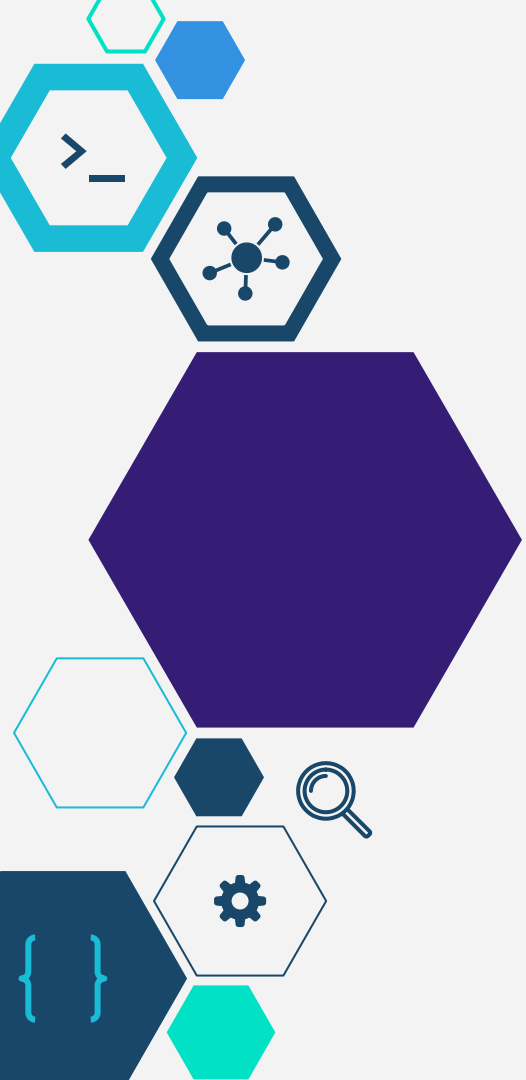
App-to-app communication

A few of you are looking into connecting two or more applications.

1. Determine if the other apps have APIs, and then find their documentation. Do you need to authenticate? Start with ProPublica's documentation, which is very user-friendly
2. Postman is useful for these initial testing phases; try authenticating, try a few GETs, build your confidence with the other API
3. Then it's time to pick your hammer; this workshop was done with Python3, but if you have other programming experience, start there

Can the API import in bulk?

- Yup! The simpler the data, the easier to learn. You'll need to
  - Map what you have to extant fields
  - Get it into JSON however you can (don't overthink this part, really)
  - Iterate the posting to ASpace
- A good starting point: pick some legacy data, make a record in ASpace with it, then GET that record out using the GUI, and map the fields. Then, POST one record in through the GUI to see if it works.
- For the rest, try modifying Eric Hanson's **postNew.py**, located here: https://github.com/ehanson8/archivesspace-api



Advice

Reference these slides to help you get started

I don’t have access/how do I experiment?

Every instance of AS has an API, but access to it can get political. After all, you CAN do real damage.

If your API is locked down, ask IT if they can deploy you a Development instance that you can run locally. You could test scripts with your own data, and then provide the scripts to IT. At that point, they may reconsider...

Configuring the API

Session time and page limits

- Ever been timed out of your bank account? Frustrating but vital
- The amount of time you have after authenticating is called “session time”
- ASpace default is very short
- The page limit is very short, too
- Ask your ASpace tech person to up the session time and page limit in the AS config
- But remember that the session time will apply to your regular login

```
##########################################################################
## This file shows the ArchivesSpace configuration options that are available,
## and the default value for each.
##
## Note that there is no need to uncomment these unless you plan to change the
## value from its default.
##########################################################################

##
## This section contains the most commonly changed ArchivesSpace settings
##

# Sessions marked as expirable will timeout after this number of seconds of inactivity
AppConfig[:session_expire_after_seconds] = 3600

# Sessions marked as non-expirable will eventually expire too, but after a longer period.
AppConfig[:session_nonexpirable_force_expire_after_seconds] = 604800

AppConfig[:default_page_size] = 10
AppConfig[:max_page_size] = 250
```

See: http://archivesspace.github.io/archivesspace/user/configuring-archivesspace/

Endpoints for your ASpace

- If this was your first rodeo, try experimenting with ProPublica's documentation
- Then move on to ASpace's documentation
- Then do a close reading of the next slide, and use these slides to help you authenticate to Postman, which is handy for experimentation

Sample endpoint from documentation: http://localhost:8089/repositories/:repo_id/resources/1

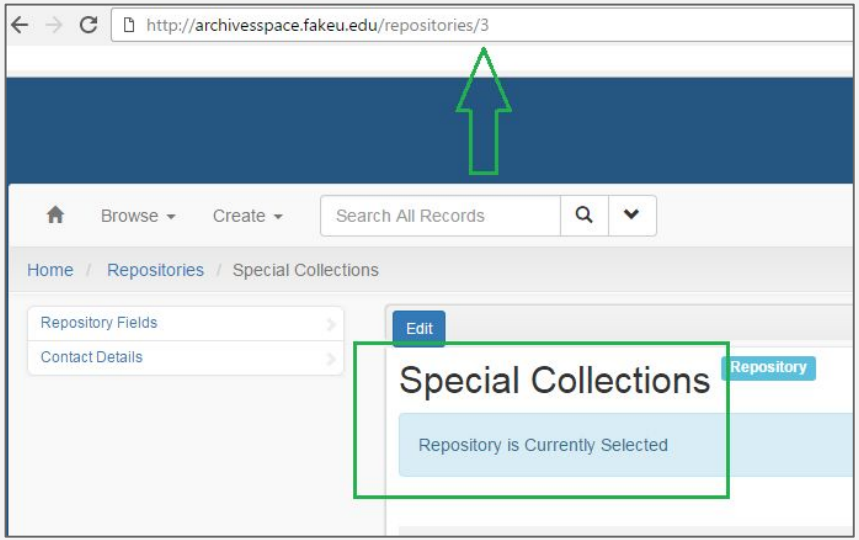
Example "fake" endpoint that mimics real life: http://archivesspace.fake.edu:8089/repositories/3/resources/1

http://localhost:8089	The address of your instance of ASpace. You will ONLY replace "local host," the colon and port number remain. EX. http://archivesspace.fakeu.edu:8089
/repositories	The presence of "repositories" here means that this endpoint is repository-specific. Some non-repo specific requests in AS are for Agents and Access Points, which span all of AS. EX. http://archivesspace.fake.edu:8089/agents
/:repo_id	The presence of this colon means this value will be unique to your institution. How can you determine the repository number? You can use the repo endpoint, or, from within AS navigate Systems > Manage Repositories > select repository > and look at the address bar. EX. http://archivesspace.fake.edu:8089/repositories/3
/resources	Other examples are /accessions or /top_containers. EX. http://archivesspace.fake.edu:8089/repositories/3/accessions
/1	The first resource. How can you determine resource numbers? Navigate to the resource in the interface and its number will be in the address bar. EX. http://archivesspace.fake.edu:8089/repositories/3/resources/1

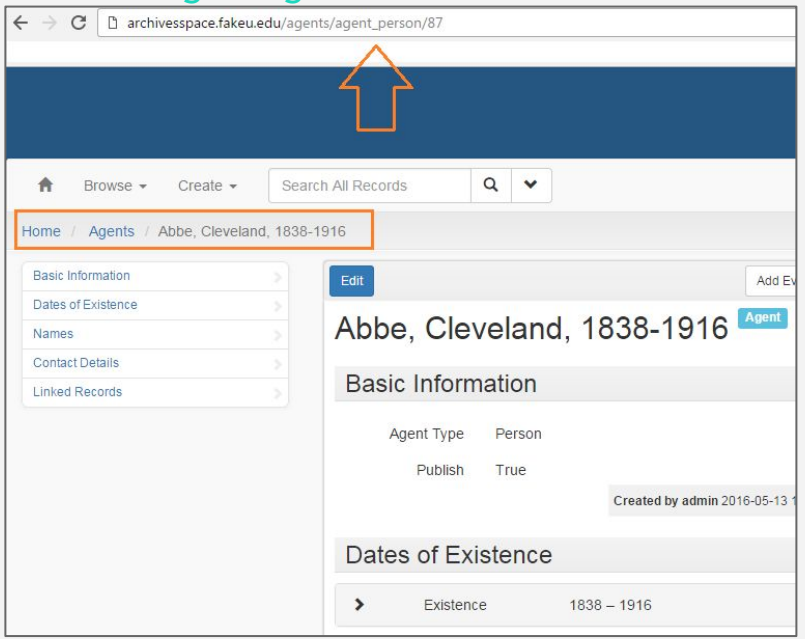
Endpoints for your ASpace

The interface — just another lens on the same data — is helpful for constructing API requests.

Determining the repository number:



Determining an agent number:



Gotcha!

We purposely made you “fail” a few times today. Get used to it!

- You WILL not succeed on the first try.
- You WILL hit unanticipated snafus, oftentimes due to data models and/or poorly written documentation (aka, due to no fault of your own!).
- You WILL be fitter, happier, and more productive if you start building a community now and asking questions.

Gotcha!

Remember that iterating over AS data changes the lock version. This will trip you up when you least expect it.

Lock version - a value that incrementally increases every time an AS record is altered. In practice, this means work cannot and should not continue on the data in question, i.e. your team **has** to stop work while you make changes.

For the 3’s in the room

For those already writing their own scripts, check out ArchivesSnake if you haven’t already:

https://github.com/archivesspace-labs/ArchivesSnake

Purpose

As institutions have adopted ArchivesSpace, a variety of practitioners and institutions have written scripts making use of the backend API to accomplish various bulk tasks not supported (yet) by the interface. ArchivesSnake is intended to be a comprehensive client library, to reduce duplication of effort and simplify scripting ArchivesSpace.

Thanks!

Remember to take the post-workshop survey
Join us at api-alums.slack.com

Credits

This workshop was written and developed by Valerie Addonizio and Lora Woodford
Original Python2 scripts by Lora Woodford and Eric Hanson
Python3 scripts adapted by Mark Custer
Slide template adapted from SlidesCarnival