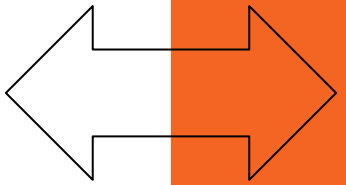
Mac users

Please sit together on this side of the room

PC users

Please sit together on this side of the room



Everyone

Please connect to the wifi

There's an API for That

Yale University
May 11, 2018

Lora Woodford
Valerie Addonizio

Introduction

The Odd Couple

We are stronger and smarter together

Lora

- Digital Archivist turned DevOps Specialist
- Never met a task she didn't want to solve with Python/Ruby
- Thinks hanging with the Bmore on Rails group sounds like a fun night out
- Helped migrate two institutions to ArchivesSpace and found it thrilling
- Enjoys craft beer, cross stitch, and the Pittsburgh Steelers
- Knows how to programmatically manipulate data

- CATS YAY!

Valerie

- Former Processing Archivist
- Never met a task she didn't want to solve with an MS Access database
- Knows rails are either "narrow gauge" or "standard" because ❤ steam trains
- Helped migrate to ASpace and felt like a fish climbing a tree
- Enjoys geocaching, blacksmithing, and planning ambitious camping trips
- Knows data modeling and systematic analysis of the *really hard* problems

- CATS YAY!

A Broad Audience

You can be here

Or here

Because we're here for everyone

Workshop Aims

1.

Practical, real-life application

...including the bad parts of real-life

Resource-packed GitHub

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!)

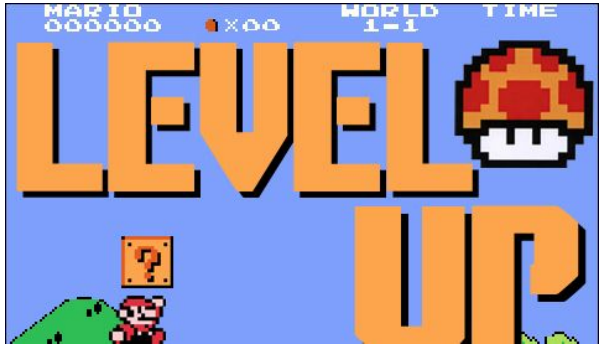
This is difficult and frustrating

You didn't miss a flight; the airport turned into a space launch 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.



Introduction to APIs

What does API stand for?

Application

As in a computer application, like Word or Chrome

Programming

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

Interface

As in the 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 at all what I thought an API was!

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.

Vocabulary pitstop: API Terms

- **GET**, **POST**, and **DELETE** are the 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

I’m not an application, I’m an archivist!

Why should I care?

As librarians and archivists with collection descriptions and/or collections themselves 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.

API possibilities

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

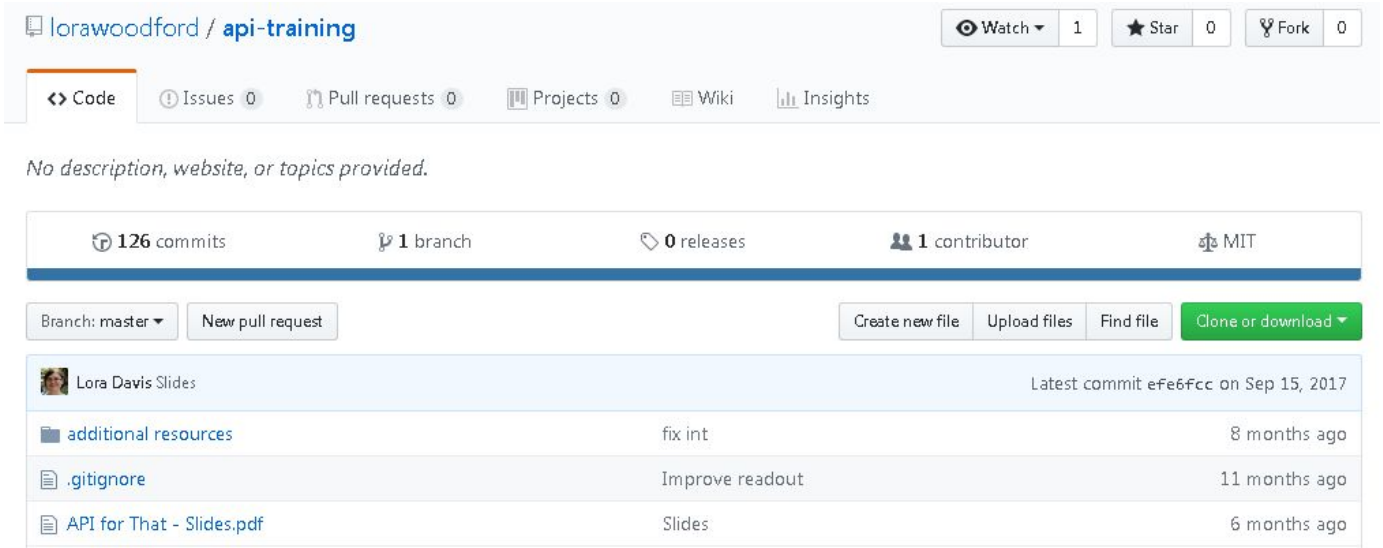
Questions?

Setting up your tech

(We promise you're in the right workshop)

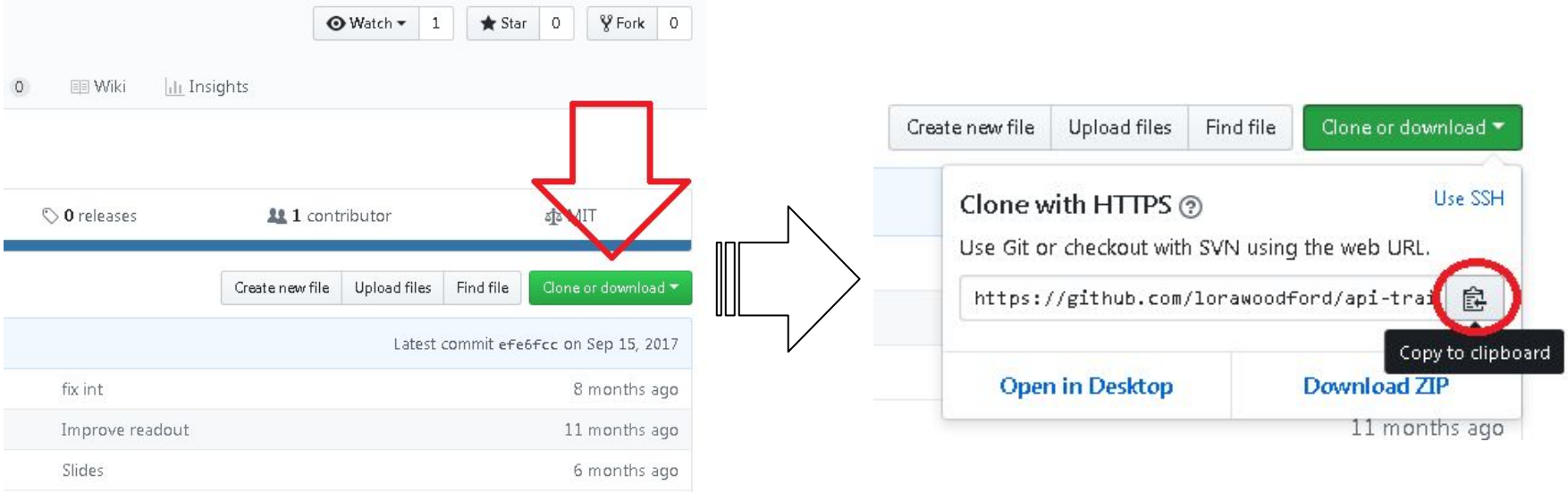
Resource Pitstop - Clone our repo

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



Resource Pitstop - Clone our repo

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



Resource Pitstop - Clone our repo

Mac

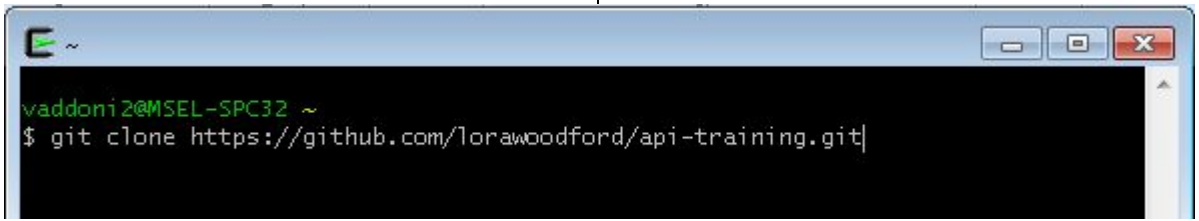
1. Open Terminal
2. Type `cd Desktop` and hit enter
3. Type `git clone [command+v then paste]`
4. Hit enter

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

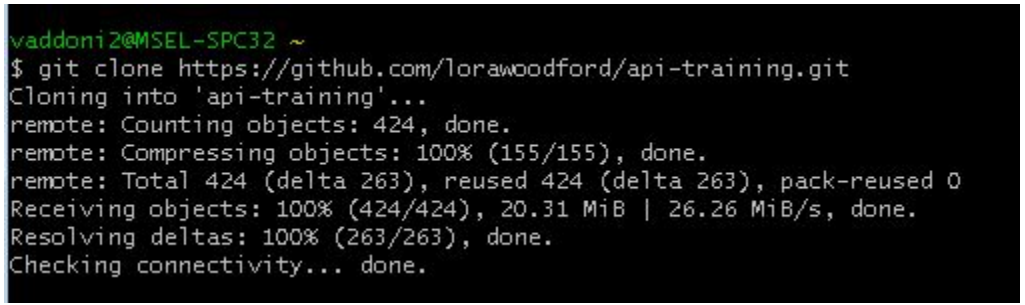
PC

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

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



Resource Pitstop - Clone our repo



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.

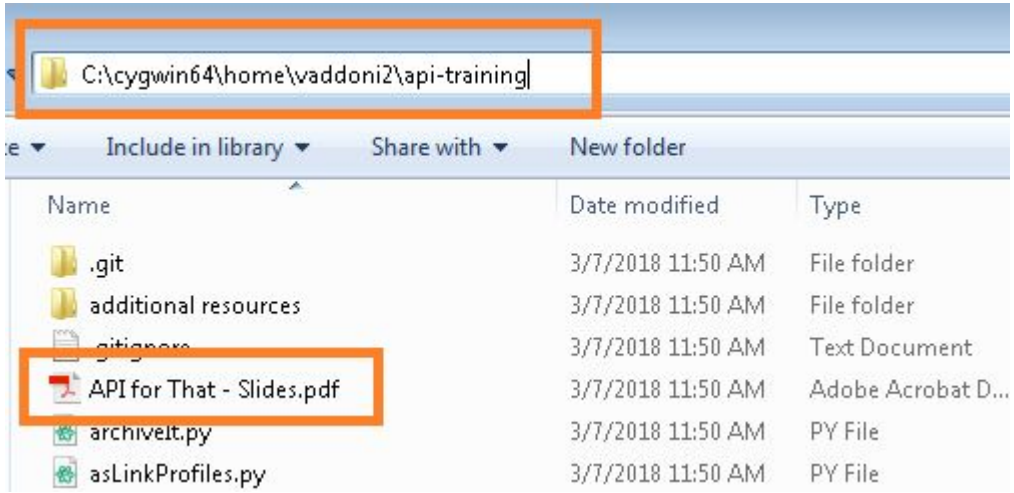
This folder, titled **"api-training,"** is a *direct clone* of what you see in your browser on github.com.

Resource Pitstop - Clone our 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.

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)

Technical Pitstop: Scripting set-up

Technical Pitstop - Scripting set-up

We are about to:

- Review the pre-workshop steps
- Get some important packages **installed** for our Mac users

Caveats:

- We **may lose some of you, both now and later**
- You *do* need to know this, but **we have other tofu to fry**
- Use these slides if you need to set up your own workstations **back at the office**

Housekeeping:

- We've got many different environments to troubleshoot and many different opportunities to fail - **please be patient**!
- If, at any point for the remainder of the workshop, you need **assistance**, just flag us down

Technical Pitstop - Pre-workshop instructions

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?
- Cgywin, 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.

Technical 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 2.7.14` and hit enter
7. Type `pyenv global 2.7.14` and hit enter
8. Type `pip install requests` and hit enter


Just
breathe

GET

API possibilities

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

GET with GUI - ProPublica

(and a bit about web searches versus APIs)

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

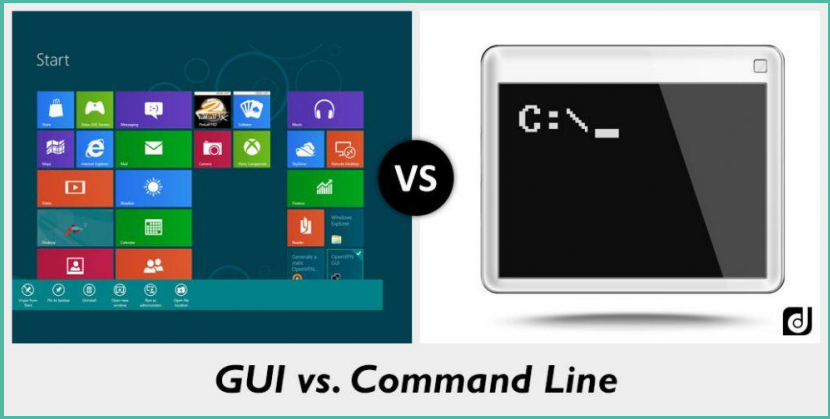


Image from http://www.differencebtw.com/difference-between-gui-and-command-line/

GET through a GUI

Scenario: A user wants access to data about every non-profit involving animal welfare in Nonprofit Explorer at ProPublica.

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. Find the Nonprofit Explorer
4. Next slide...

Updated: Nonprofit Explorer

by **Mike Tigas**, **Sisi Wei**, and **Alec Glassford**,
Aug. 10, 2017, 11:37 a.m. EDT

We have added raw data from more than 1.9 million electronically filed Form 990 documents dating back to 2010.

GET through a GUI

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

Web search versus API

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=

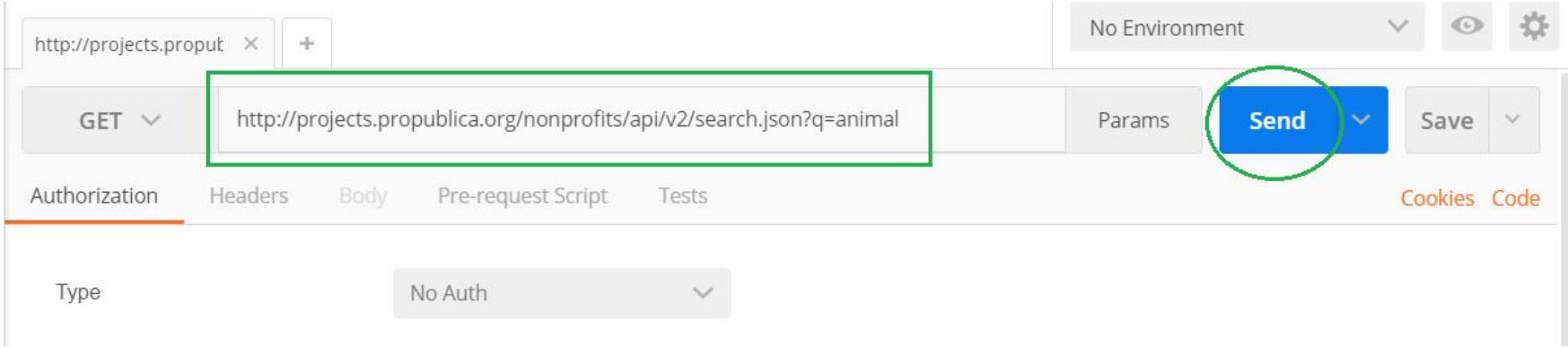
- We tend to think of URLs as *locations*…
- We call your attention to this as we discuss Endpoints.

Vocabulary pitstop: Endpoint

- An API Endpoint is simply a unique URL that 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
- As an introduction we will be comparing a search URL with an Endpoint, but just as an introduction
- Constructing Endpoints is a fundamental requirement for using APIs

GET through a GUI - ProPublica

1. Open Postman



2. Type this next to the GET command:

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

Web search versus 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 I
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.

Vocabulary pitstop: JSON

- JSON (jason) 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”

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

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

Re-purposing API data: JSON to 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)

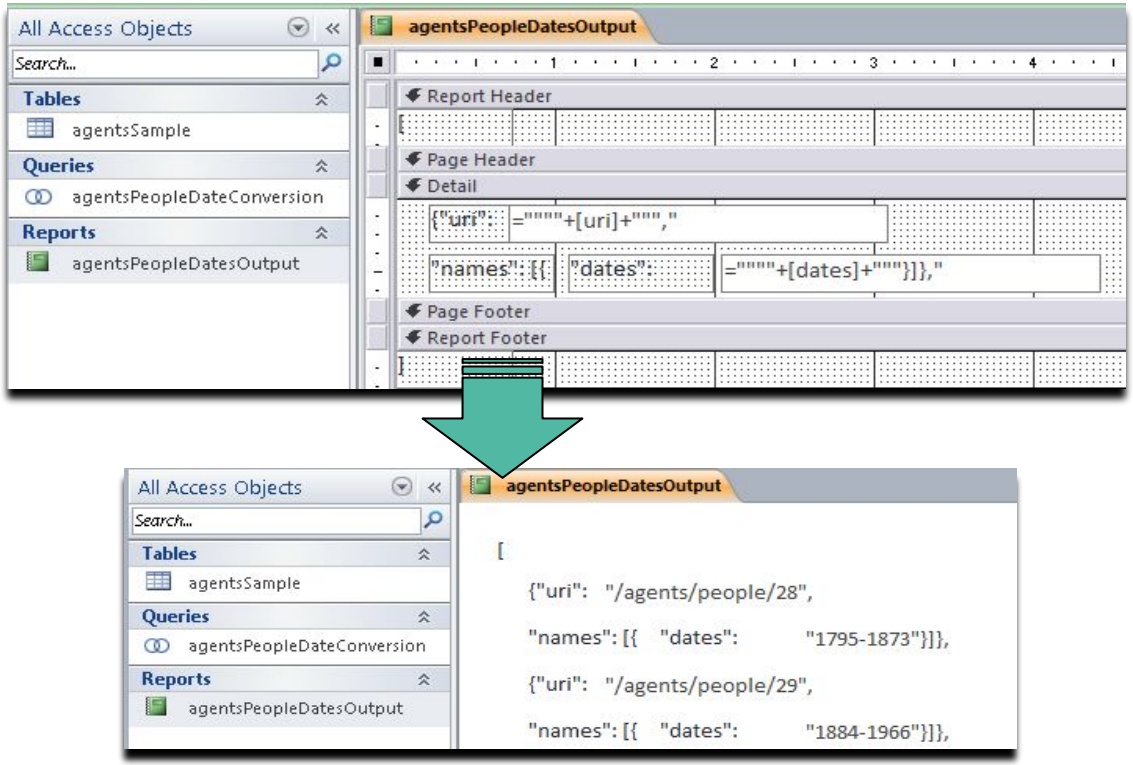
Re-purposing API data: JSON to CSV

```
{
    "ein": 731663130,
    "strein": "73-1663130",
    "name": "ANIMAL ALLIANCE",
    "sub_name": "ANIMAL ALLIANCE",
    "city": "WOODLAND HLS",
    "state": "CA",
    "ntee_code": "D20",
    "raw_ntee_code": "D20",
    "subseccd": 3,
    "has_subseccd": true,
    "have_filings": true,
    "have_extracts": true,
    "have_pdfs": true,
    "score": 441.1612
},
{
    "ein": 752461428,
    "strein": "75-2461428",
    "name": "ANIMAL ANGELS",
    "sub_name": "ANIMAL ANGELS",
    "city": "JACKSBORO",
    "state": "TX",
    "ntee_code": null,
    "raw_ntee_code": null,
    "subseccd": 3,
    "has_subseccd": true,
    "have_filings": true,
    "have_extracts": true,
    "have_pdfs": true,
    "score": 441.1612
},
{
    "ein": 232896507,
    "strein": "23-2896507",
    "name": "ANIMAL APPEAL",
    "sub_name": "ANIMAL APPEAL",
```

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

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

The reverse is also true: Access to JSON



This is a breakdown of the endpoint we just used:

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

Web search versus API

Address bar after search:

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

API endpoint:

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

Web search versus API

For those of you who weren't sure if you've ever used an API... you've been using them for years!

APIs, like wizards and geocaches, are all around you.

Yelp search? Check out the address bar.

Amazon search? Address bar.

Your bank account? Address bar!

Questions?

GET with Script-ProPublica

GET with a script - ProPublica

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.

15 minute 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

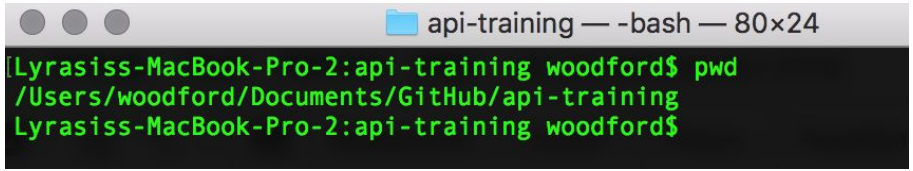
1. Open Cygwin 

Command line bootcamp: *Where are you?*

Everyone type `pwd` and then hit enter.

Mac

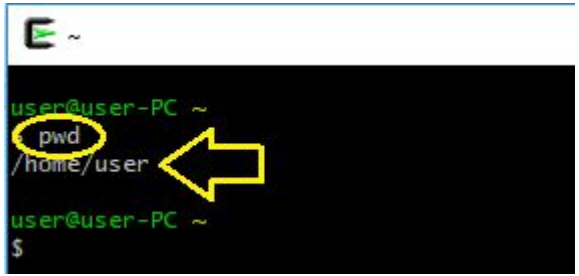
Mac users should see something like this:



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:

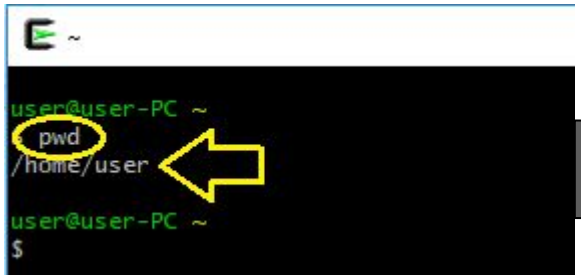


Command line bootcamp: *Where are you?*

PC

Windows users will ask: but **where** is that? This is non-intuitive, but *you're already in* C:\Cgywin 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`

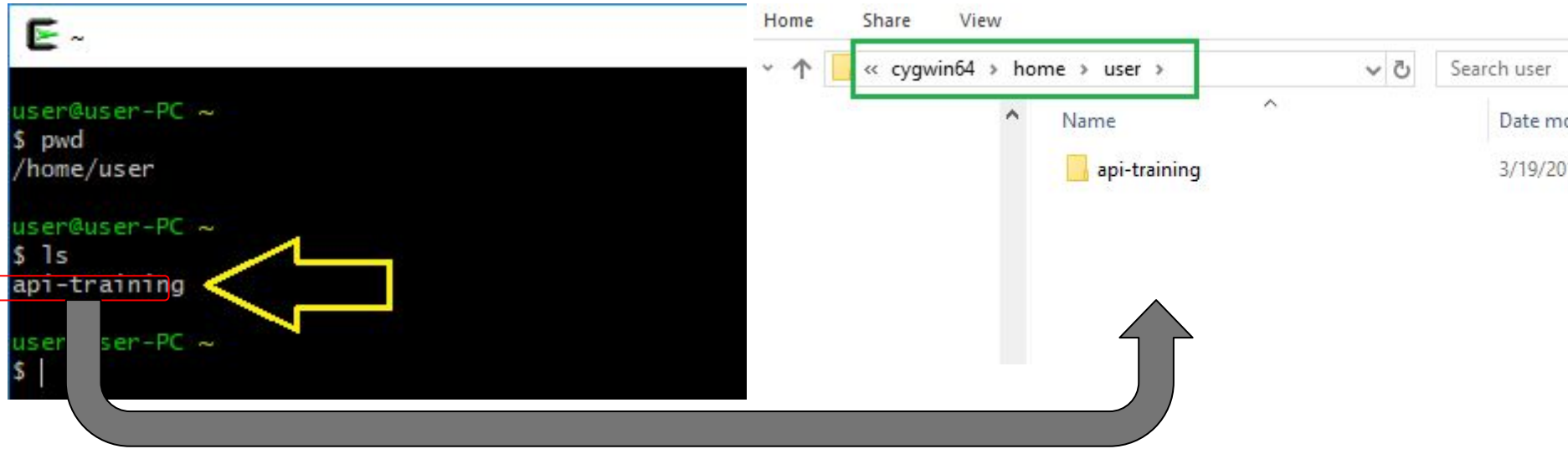
Command line bootcamp: *What is here?*

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 Valerie's PC, you won't have all these files):



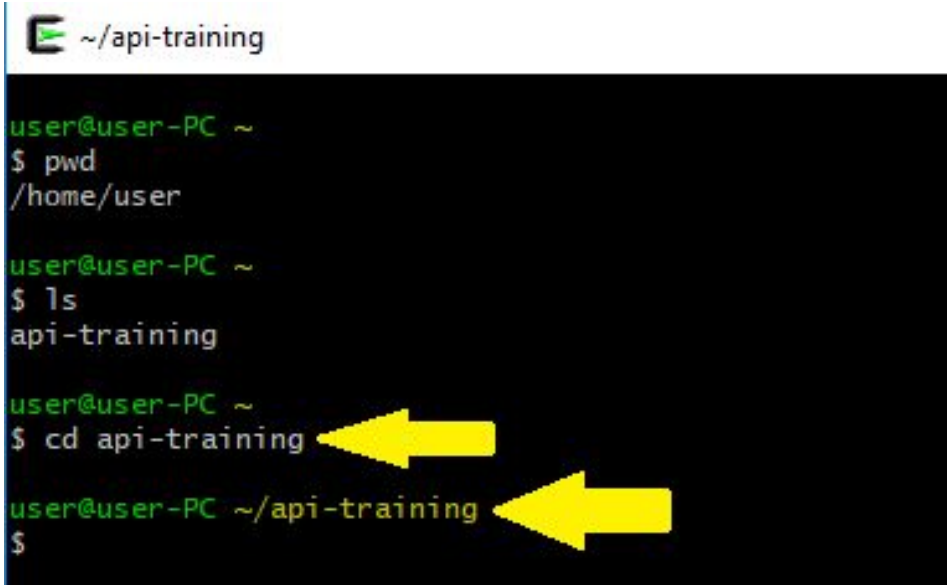
Unix commands for Mac and Cygwin

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

  (but this varies by user)
- And the *same path* in Cgywin looks like the new prompt, below:

Unix commands for Mac and Cygwin

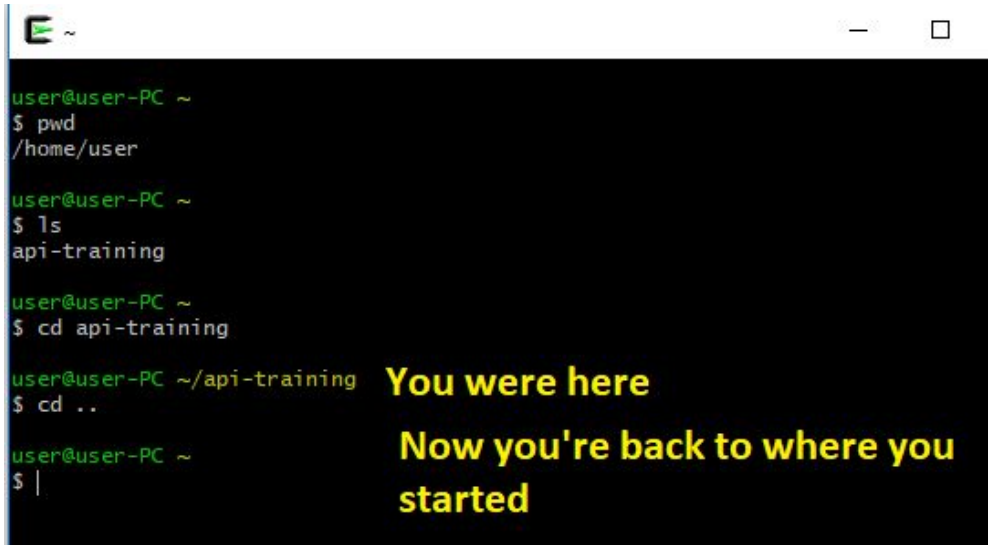
Where am I?	“print working directory”	`pwd`
What is here?	“list” (remember L as in List)	`ls`
How do I move from here to there?	“change directory”	`cd [type the name of the directory]` (don’t type this, we mean an action)

Command line bootcamp: *Move up*

Now you're going to move *from* the api-training clone folder back to the Cygwin home directory/Mac desktop:

Why? To demonstrate a simple command that means "go up one"

`cd ..` = "go up one"



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 command*

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

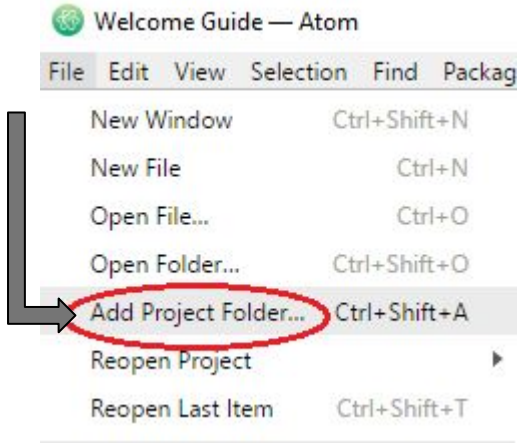
These are called Unix commands, so Google “unix commands” for other commands that will work on Macs and in Cygwin.

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.

Questions?

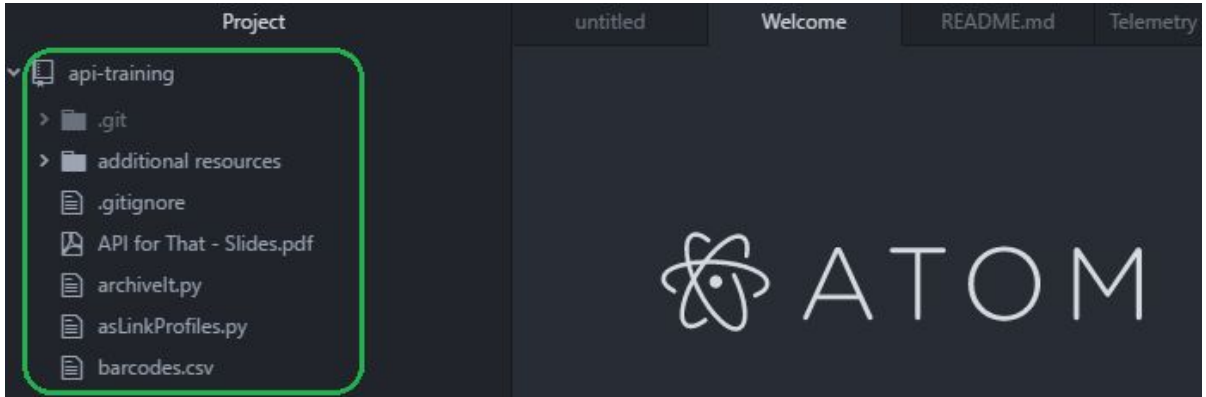
GET with a script - ProPublica

1. Open Atom (or your text editor of choice)
2. Add a project folder and select the api-training folder from earlier



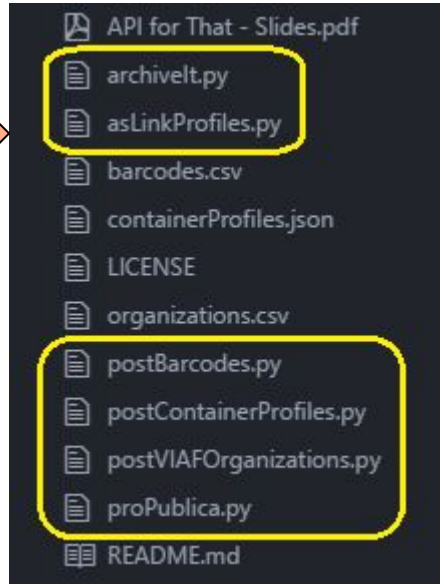
- Mac users: check your desktop
- Windows users: look in C:\Cygwin\home\[username]

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



GET with a script - ProPublica

Take note of everything ending in .py; these are Python scripts, and we will be using them throughout the workshop.



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

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

Locate the endpoint

```
7   raw_input('Press Enter t  continue...')
8
9   endpoint = 'http://projects.propublica.org/nonprofits/api/v2/search.json?q=animal'
10
11  output = requests.get(endpoint).json()
```

GET with a 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 `python 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 `python proPublica.py` and hit enter
5. Next slide...

GET with a 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:\cgywin\home\[username]\api-training

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

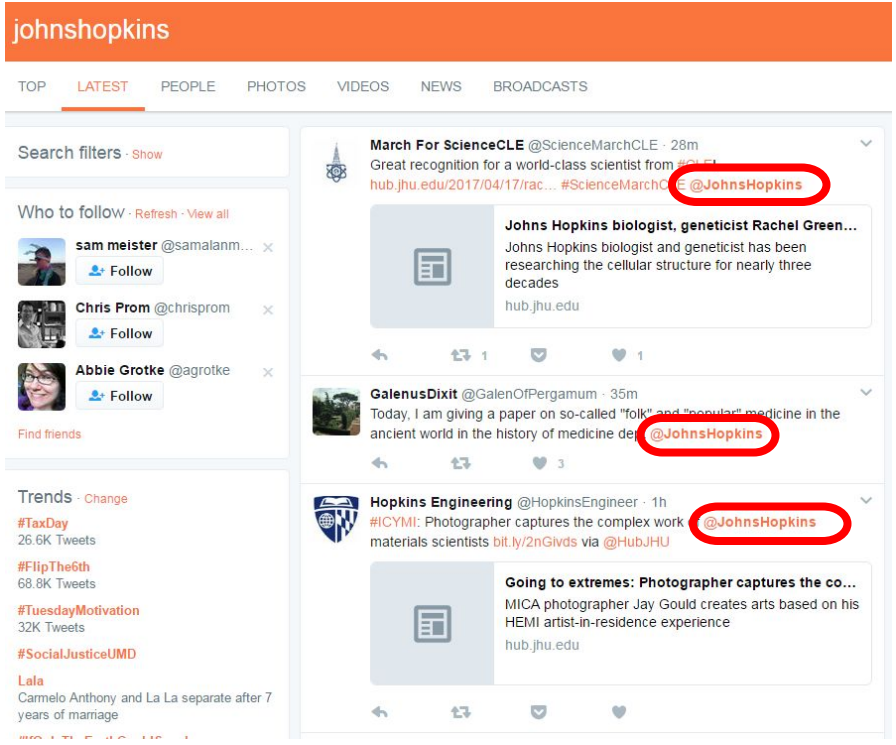
Questions?

GET with GUI - Twitter

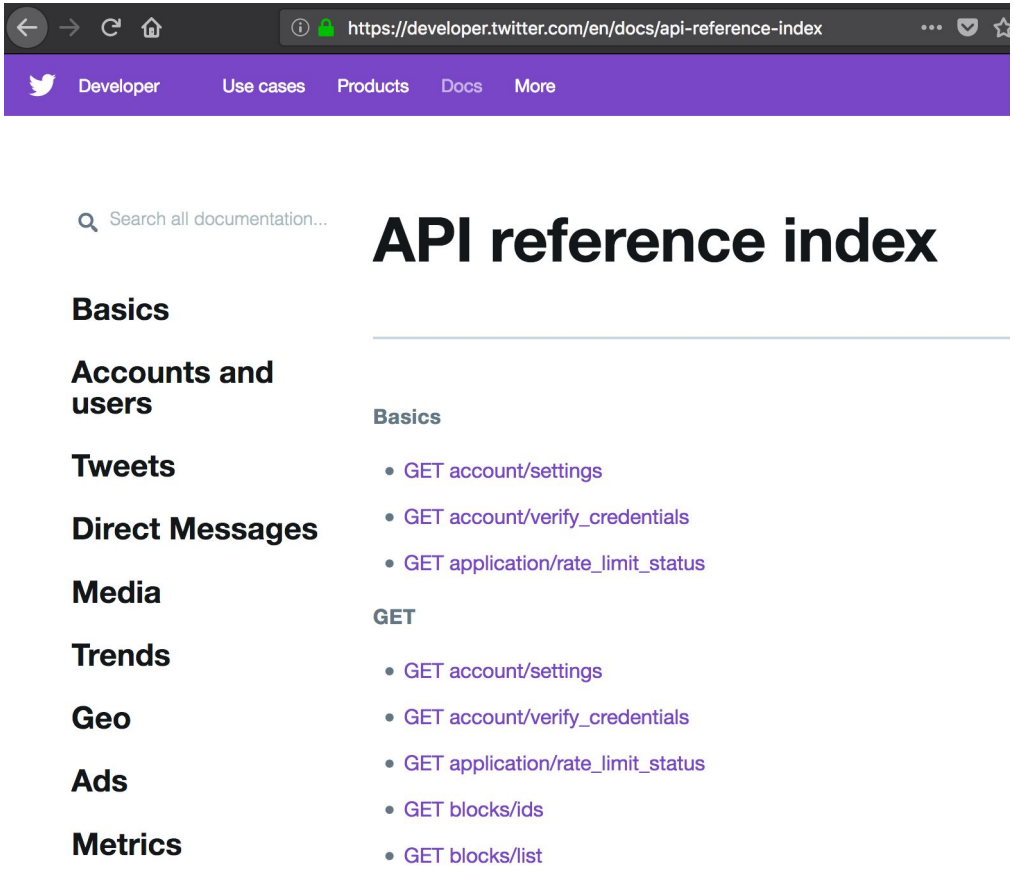
Using an API to collect records

@JohnsHopkins

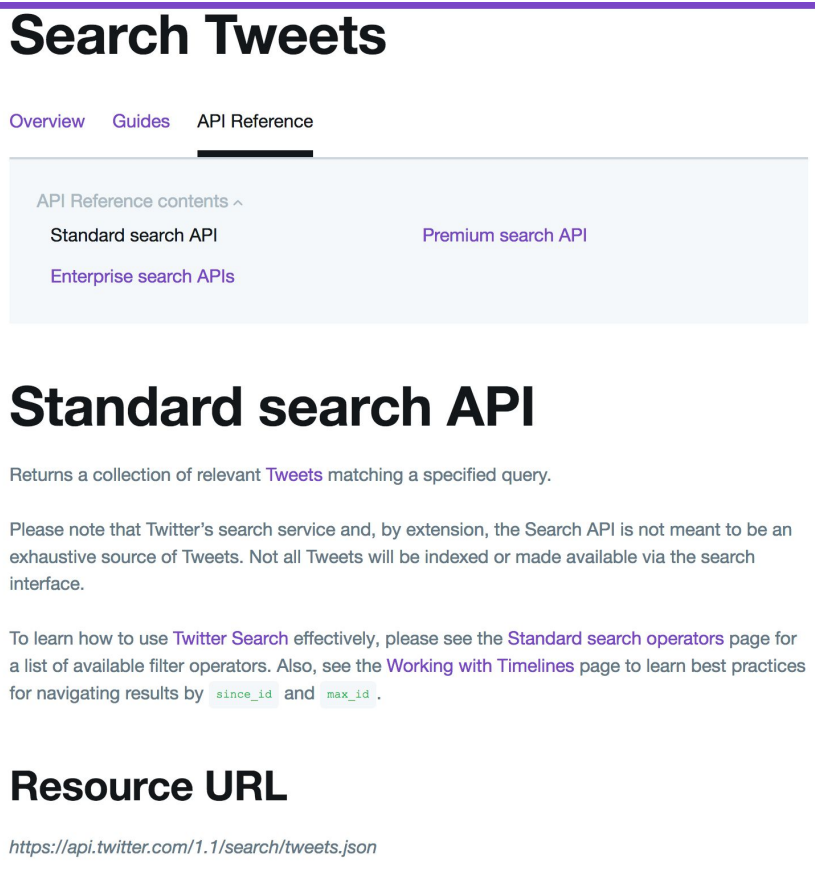
Scenario: As your university's records manager, you wish to regularly capture Tweets mentioning your university.



Using an API to collect records



Using an API to collect 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.

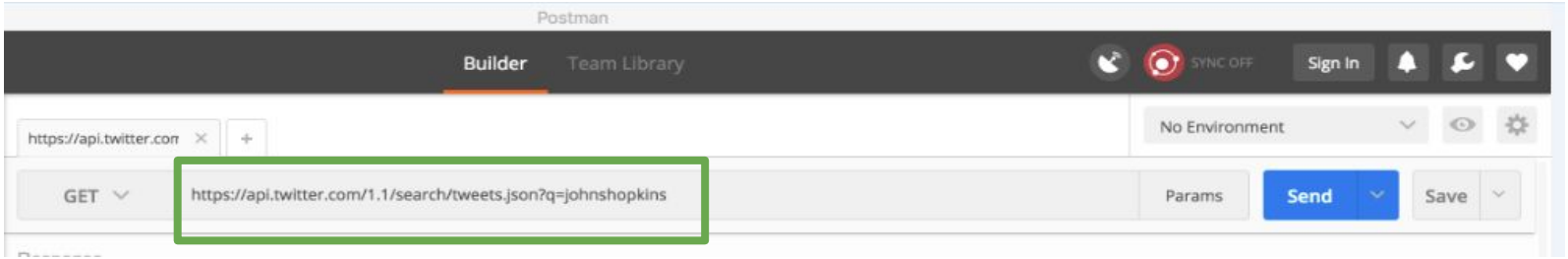
Please note that Twitter's search service and, by extension, the Search API is not meant to be an exhaustive source of Tweets. Not all Tweets will be indexed or made available via the search interface.

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

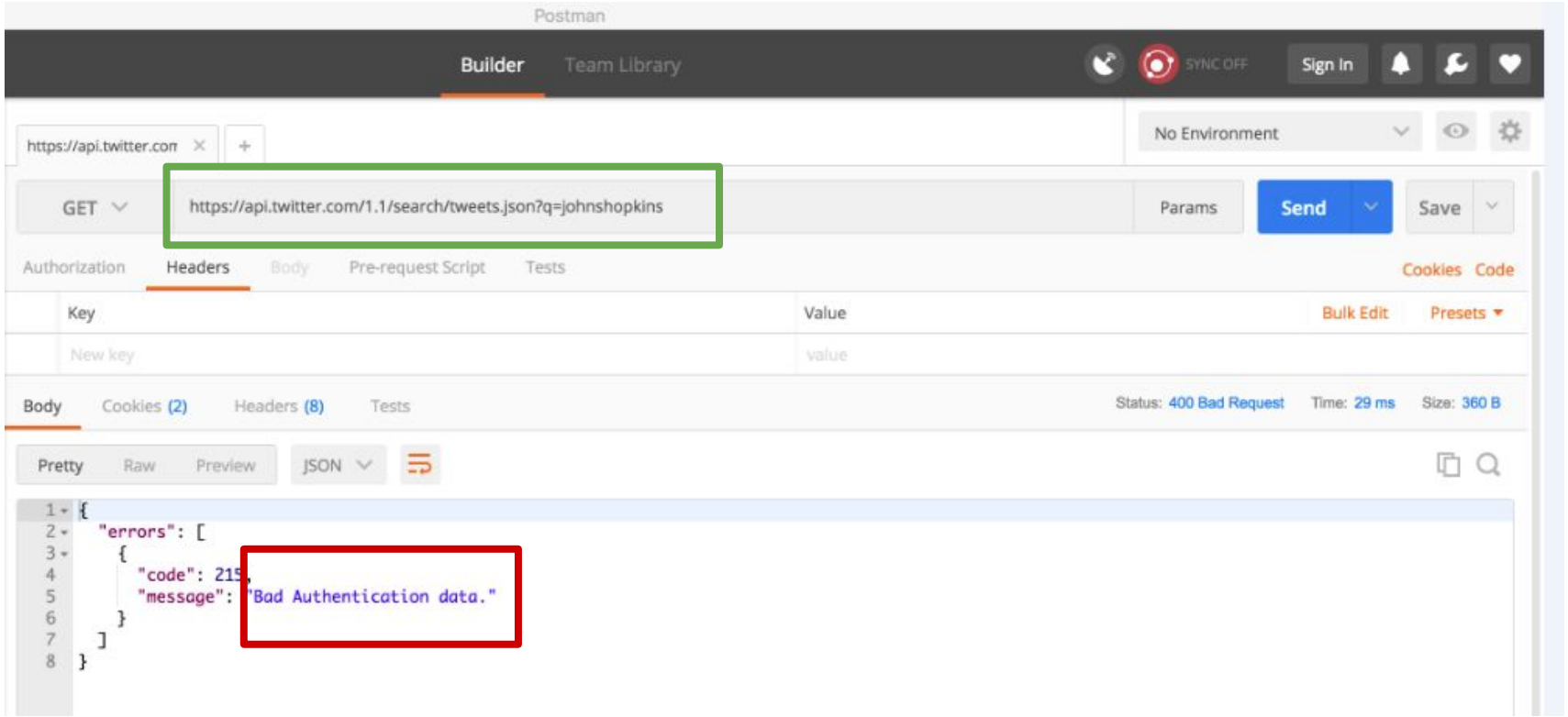
Using an API to collect records



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

Let’s give it a try!

Using an API to collect records



Using an API to collect 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

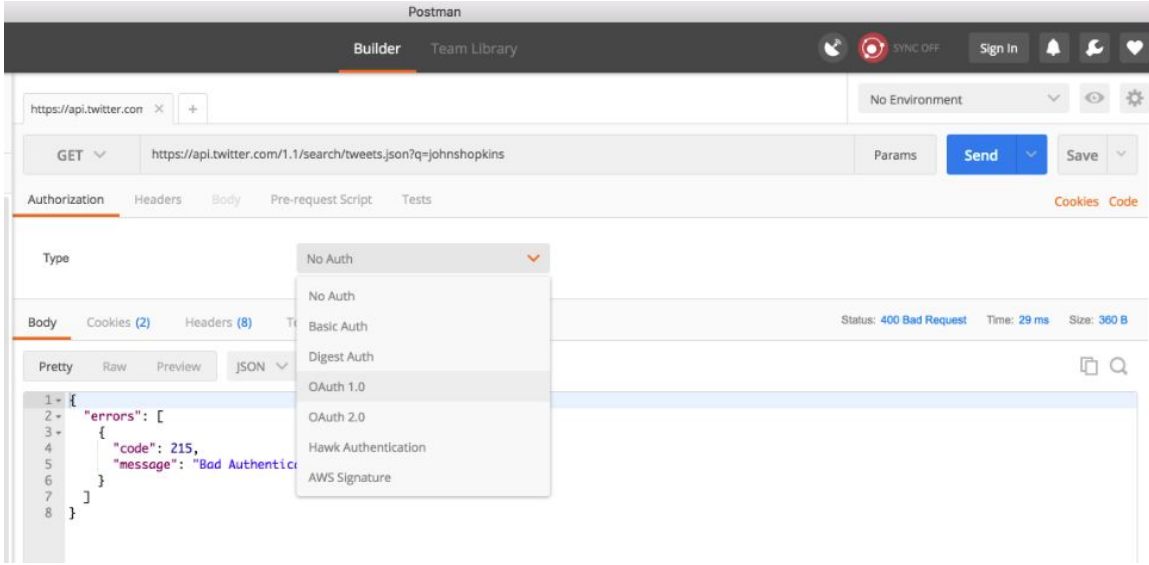
Using an API to collect records

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 collect 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 ScienceCLE
            "id": 824422865206472706,
            "id_str": "824422865206472706
            "indices": [
              3,
              19
            ]
          },
          {
            "screen_name": "JohnsHopkins"
            "name": "Johns Hopkins U.",
            "id": 14441010,
            "id_str": "14441010",
            "indices": [
              119,
              132
            ]
          }
        ],
        "urls": [
```


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.

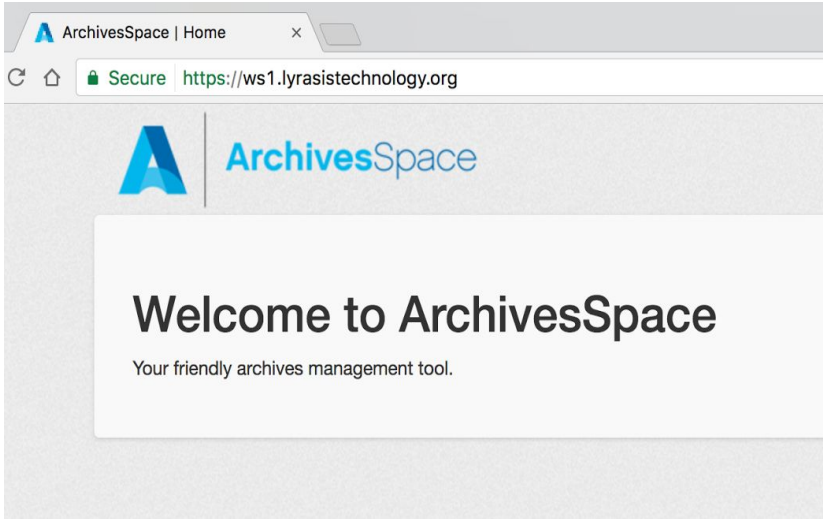
Questions?

ArchivesSpace

(I know, finally right?)

ArchivesSpace!

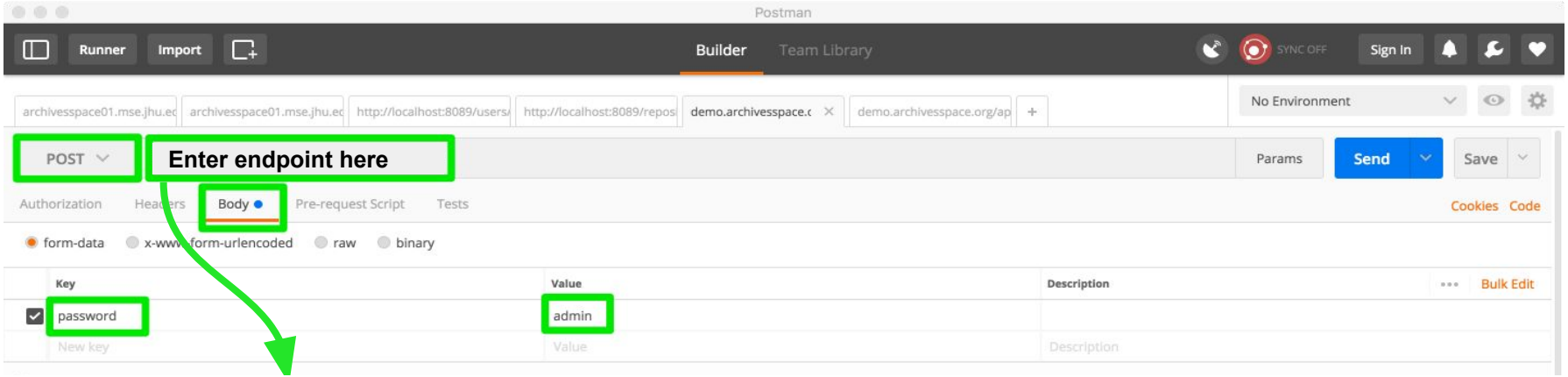
- You've each been provided a test instance of ArchivesSpace by our gracious hosts at Lyrasis
- The address of the **staff interface** of your instance is: https://ws[your#].lyrtech.org/staff/ (https://ws0.lyrtech.org/staff/)
- The address of the **API** of your instance is: https://ws[your#].lyrtech.org/staff/api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



Authenticate and GET - AS with GUI

Authenticate to AS with GUI

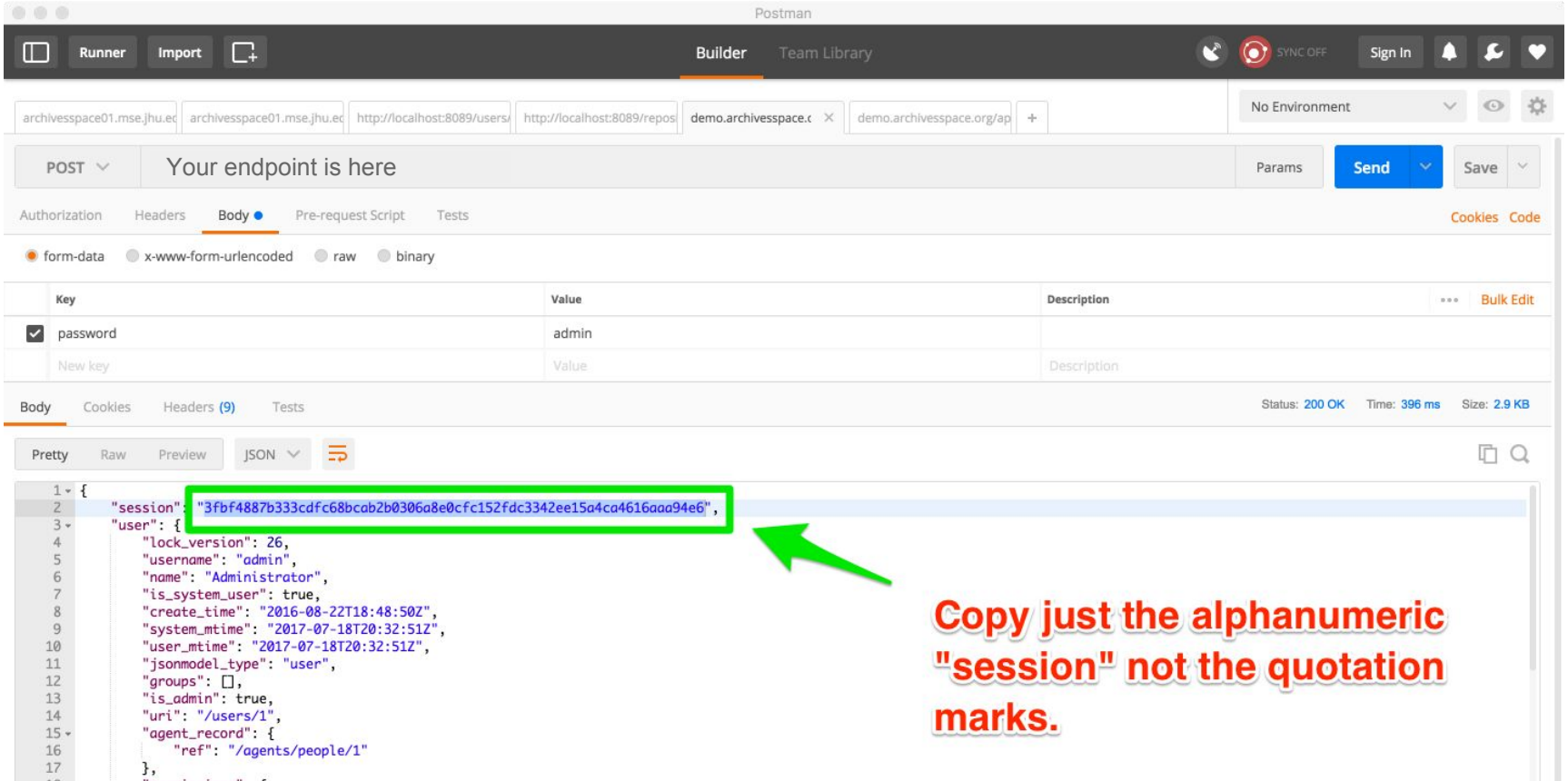
Before we start posting to AS, we need to authenticate (just like you do when logging in) so let's do that:



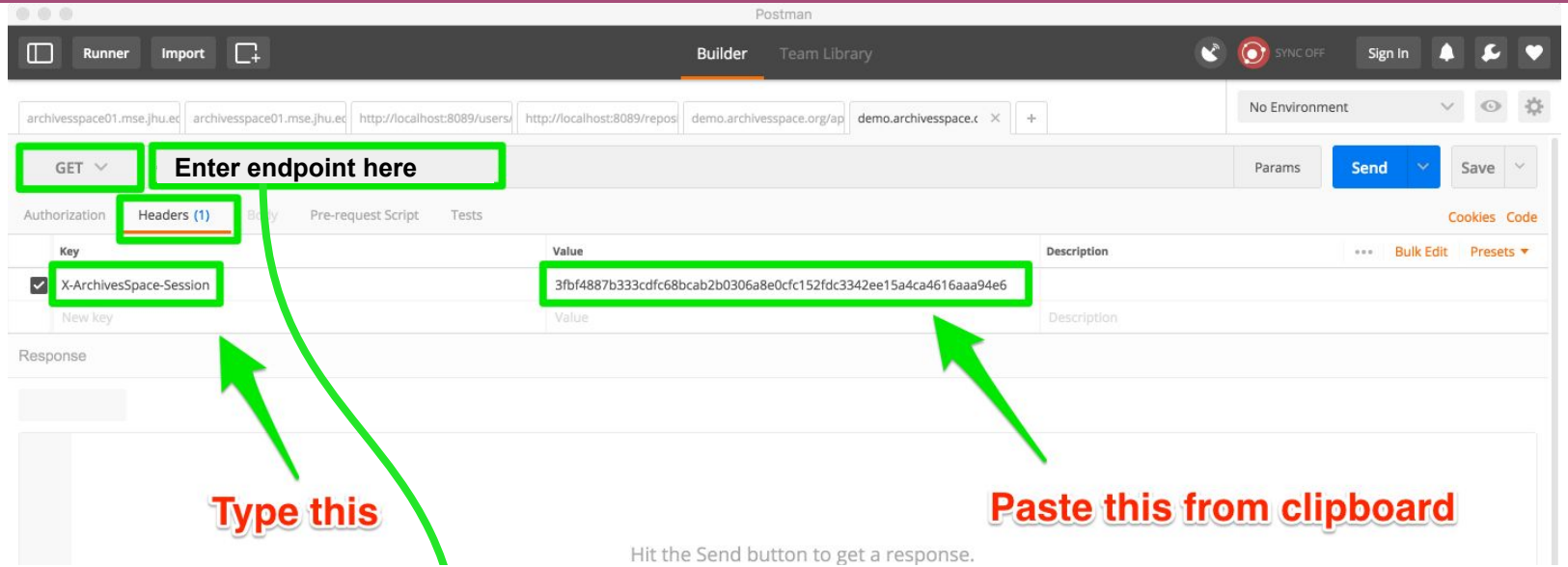
Endpoint: https://ws[your#].lyrtech.org/staff/api/users/admin/login

password: admin

Authenticate to AS with GUI



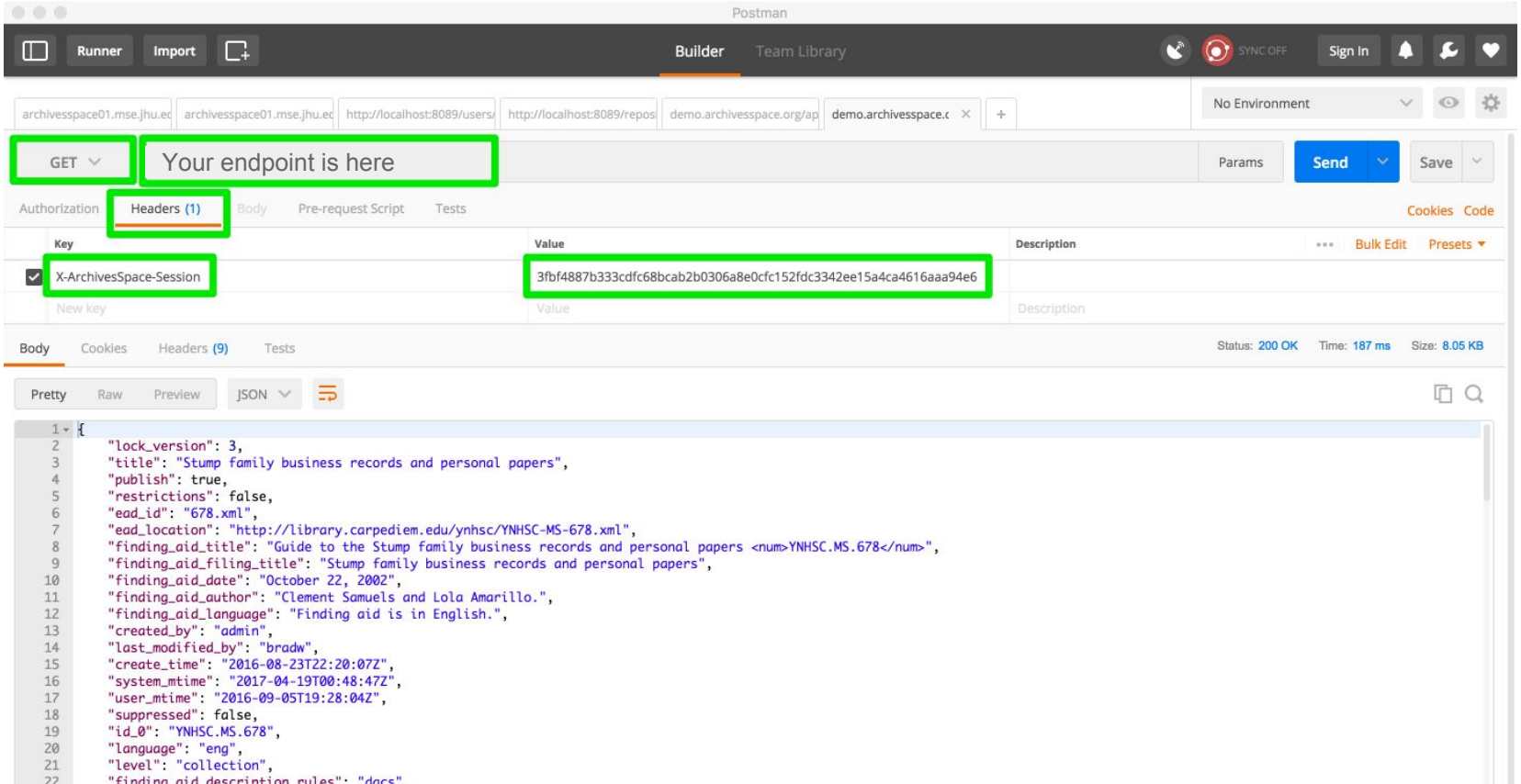
GET from AS with GUI



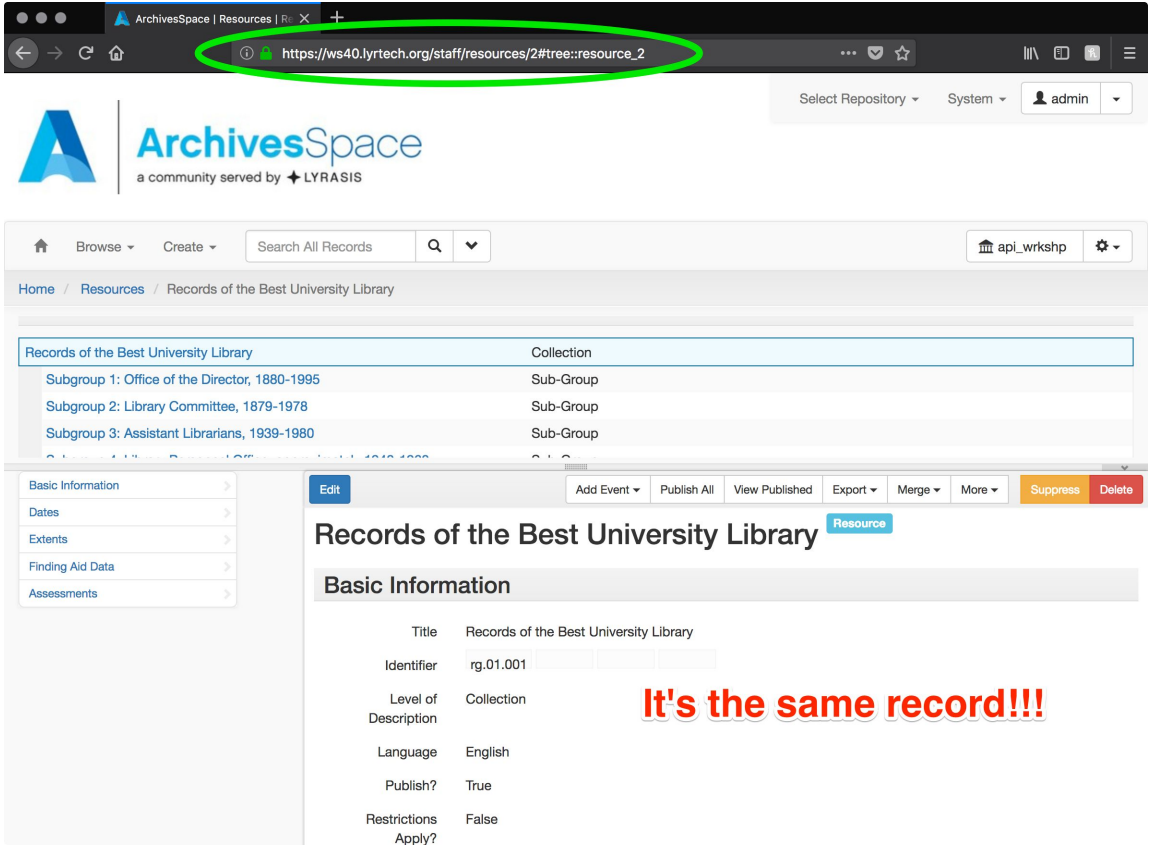
Key: X-ArchivesSpace-Session

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

GET from AS with GUI - You did it!



GET from AS with GUI



POST: ArchivesSpace

ArchivesSpace!

Scenario: You have successfully migrated into -- or have begun to use -- an instance of ASpace at your institution, but…

- There's all this new functionality, what do I do with it?
  - I don't have barcodes for my containers, or I have faux codes
  - I do have barcodes, but they're not in ArchivesSpace. How do I get them in without ruining a student worker's semester?
- There are new fields where there were no fields before
  - I'd love to use URIs for Agents, but that's a lot of work
  - BARCODES, again with the barcodes
- We didn't use Archivists' Toolkit for accessions, how do I get them in now?
- Suppressing and unsuppressing, publishing and unpublishing, and how do I publish everything but not *those* things?

As some of you know, it's a huge undertaking and you might have dozens/hundreds/thousands of old and new problems.

ArchivesSpace!

Scenario: You have successfully migrated into -- or have begun to use -- an instance of ASpace at your institution, but this is a short workshop, so here are our problems for today:

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

Extra archivisty sidebar

- These are, in fact, all problems we've addressed (or are addressing) at Johns Hopkins. And this is exactly how we did (or will be) solving these issues.
- 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

API possibilities

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

POST to AS with GUI-Container profiles

Container profiles

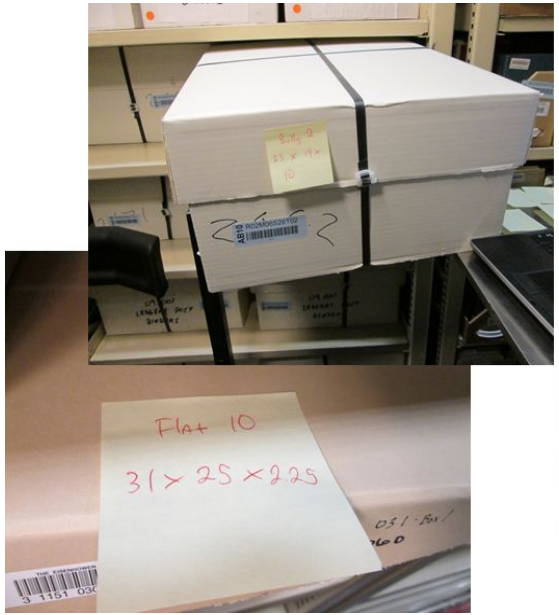
What's a container profile?

ASpace offers robust "container modeling" for the first time in the archives world.

Every type of box (ex. record carton) in your library gets its own record (a profile), which records its height, width, and depth. This helps calculate space on a huge scale, and is a game-changer for some repositories.

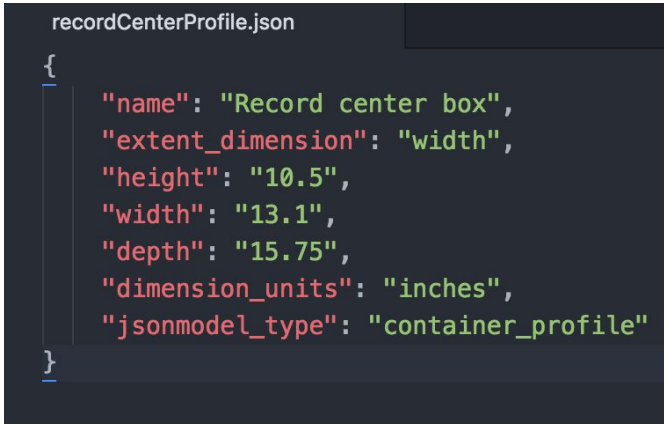
So, we have boxes o'plenty ---------->

But to use this feature, we need to get their profiles into AS.



POST to AS with GUI - Container profiles

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, and go back to Postman

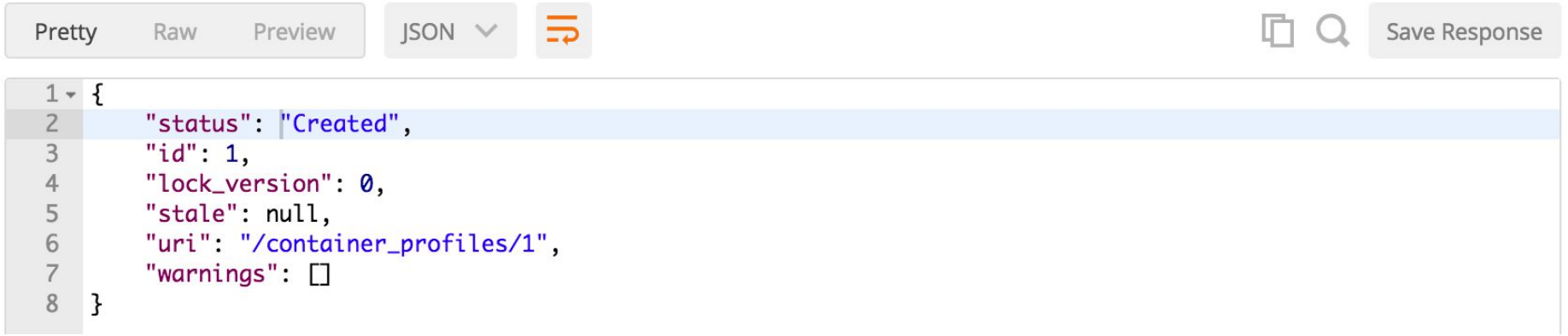


POST to AS with GUI - Container profiles

Endpoint:

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

POST to AS with GUI - Container profiles



Success!!!

POST to AS with GUI - Container profiles

1. Go back to Atom and open “containerProfiles.json” from the directory with our cloned GitHub repo

   Mac users: Desktop

   Windows users: C:\cygwin64\home\[username]\api-training
2. Packages > Atom Beautify > Beautify
3. Here are *all* the profiles, ready to go
4. Copy everything, and go back to Postman

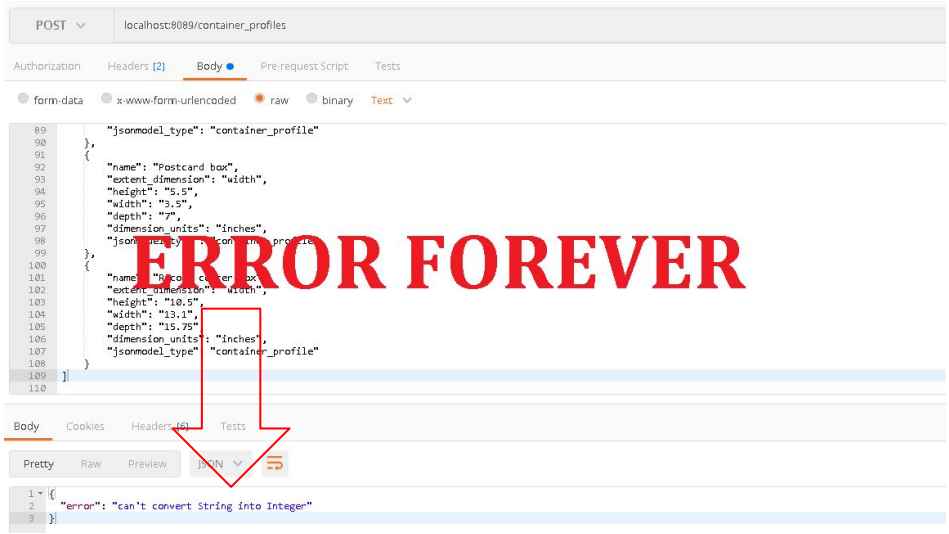

```
[{
        "name": "Flat box01",
        "extent_dimension": "width",
        "height": "3",
        "width": "12",
        "depth": "16",
        "dimension_units": "inches",
        "jsonmodel_type": "container_profile"
    },
    {
        "name": "Flat box02",
        "extent_dimension": "width",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

Endpoint:

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

POST to AS with GUI - Container profiles



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)


Just
breathe

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!
  - Frankly, if you're sitting down to write scripts from scratch, you're **doing it wrong**
- 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 (and, to a lesser degree, Ruby) is 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

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) **python**

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

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

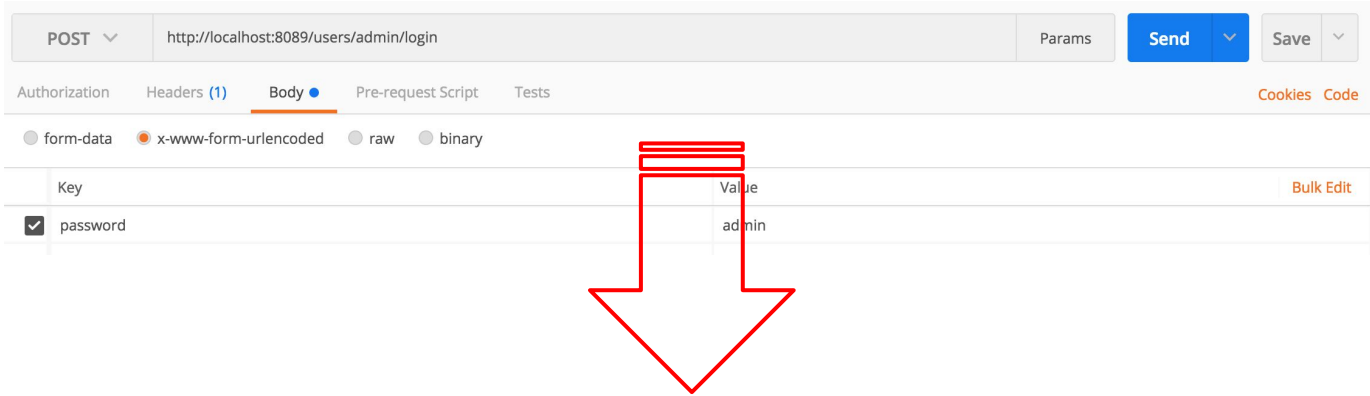
- You should be located in the same directory as the script (and any files it is reliant on) before you type your command (you can always `ls` to confirm the script is there!)

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

POST to AS with script-Container profiles

POST to AS with script

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



secrets.py

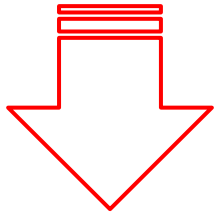
```
1  baseURL='http://localhost:8089'
2  user='admin'
3  password='admin'
```

POST to AS with script

“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}
```

POST to AS with script

- We're all connecting to different instances of ArchivesSpace (so we don't overwrite and/or clash with each others' work!), so we need to tell `secrets.py` where each of our individual instances live.
- Navigate to the ASpace_API_Workshop directory we cloned from GitHub earlier:
  - Mac users: This should be your Desktop
  - Windows users: This should be "C:\cygwin64\home\[username]\ASpace_API_Workshop"
- Open `secrets.py` in Atom
- Change the line:
  baseURL='http://localhost:8089'
  to
  baseURL='https://ws[your#].lyrtech.org/staff/api'

POST to AS with script- Container Profiles

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 `python 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 `python postContainerProfiles.py` (case sensitive!) and hit enter
5. Navigate back to AS in your browser (https://ws[your#].lyrtech.org/staff/container_profiles)

ArchivesSpace!

Scenario: You have successfully migrated into -- or have begun to use -- an instance of ASpace at your institution, but this is a short workshop, so here are our problems for today:

1. ~~We don't have **container profiles** in ArchivesSpace and would like to, so we need to create some~~
2. In following the migration instructions for 1.5, we had to add **faux codes**; we'd like to use our *actual* barcodes
3. Now that we have container profiles, we need 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 AS with script- Edit barcodes

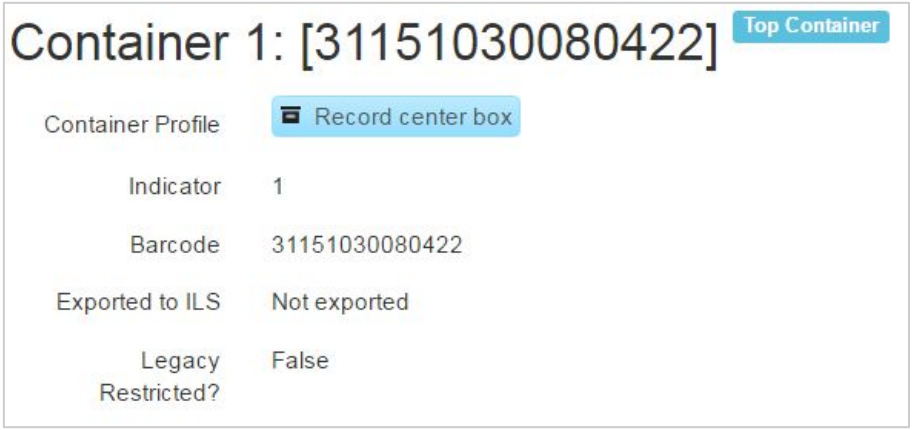
Barcodes/top_containers

Répète: ASpace offers “container modeling” for the first time in the archives world.

In addition to profiles, every actual box in your collections *also* 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”

Barcodes make that sane for AS. Hence, AS 1.5 and beyond requires some sort of unique code in every top container record.



Barcodes/top_containers

Every marathon runner and every top_container must have a unique ID to participate.

Barcodes/top_containers

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

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

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

Barcodes/top_containers

If you're in ASpace, you will have some version of this problem, which is why we're featuring it.

Your top containers might:

- Have barcodes already! Well, this is still a lesson in editing records
- Have "faux codes," like the ones in the AS vagrant
- Have nothing, and you have no idea where to start. We'll have to refer you to the AS 1.5 instructions and this plugin by Chris Fitzpatrick

So let's fix our problem and imagine it working at scale.

POST to AS with script- Edit barcodes

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 `python postBarcodes.py` (case sensitive!) and hit enter
5. Navigate back to AS in your browser (https://ws[your#].lyrtech.org/staff/resources/1)

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 `python postBarcodes.py` (case sensitive!) and hit enter
5. Navigate back to AS in your browser (https://ws[your#].lyrtech.org/staff/resources/1)

ArchivesSpace!

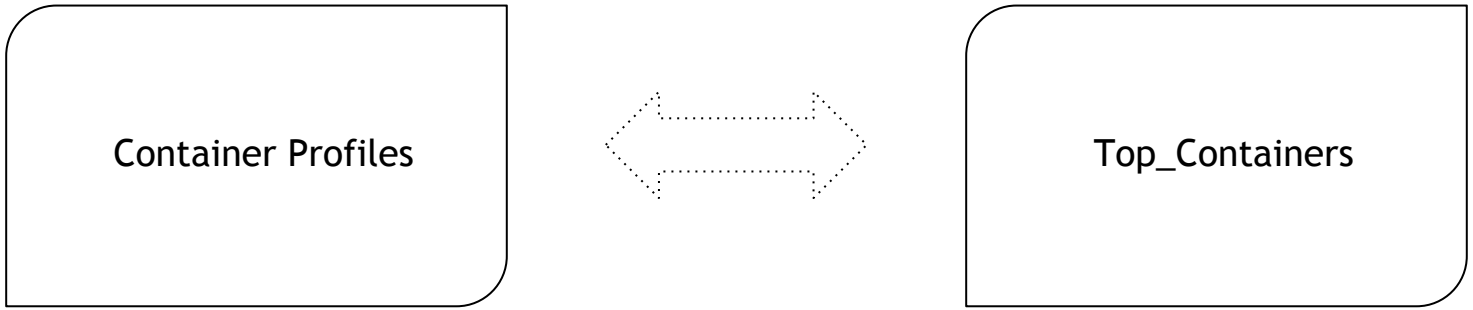
Scenario: You have successfully migrated into -- or have begun to use -- an instance of ASpace at your institution, but this is a short workshop, so here are our problems for today:

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

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

POST to AS with script-Link profiles

Linking profiles to containers

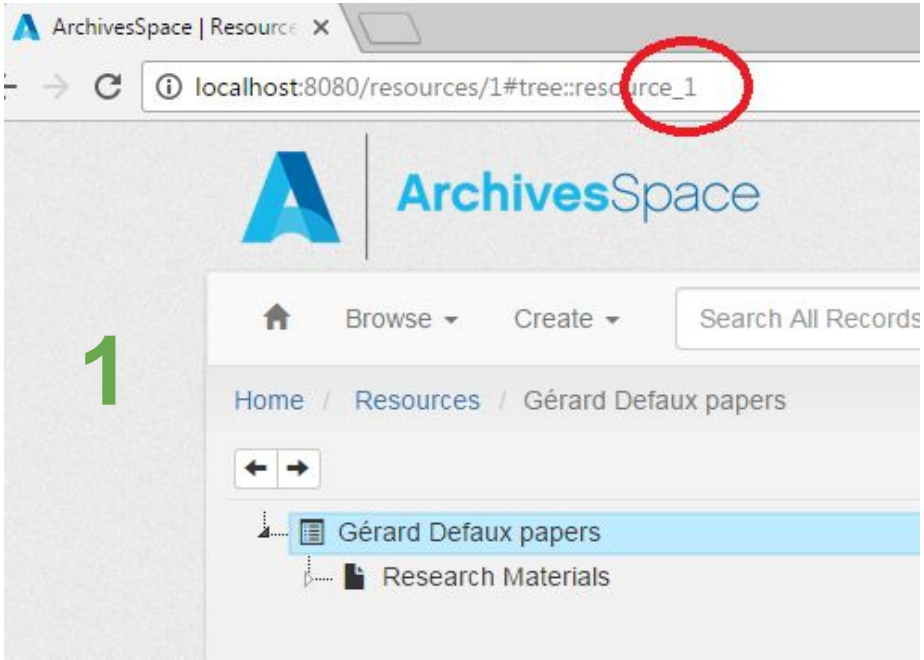


1. Type `ls`
2. Type `python asLinkProfiles.py` (case sensitive!)
3. The script should prompt you for something...

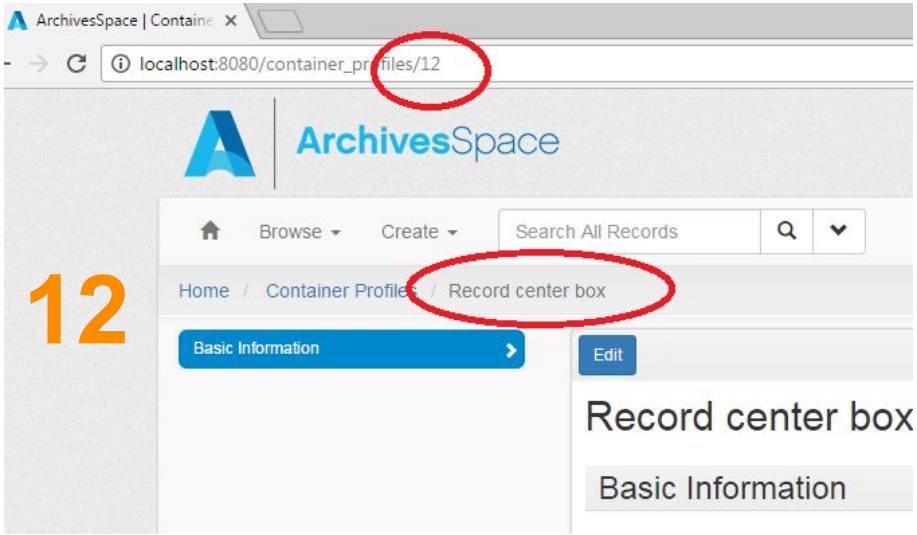
POST to AS with script - Linking profiles

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:



~~ArchivesSpace! You did it!~~

1. ~~We don't have **container profiles** in ArchivesSpace and would like to, so we need to create some~~
2. ~~In following the migration instructions for 1.5, we had to add **faux codes**; we'd like to use our *actual* barcodes~~
3. ~~Now that we have container profiles, we need 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.

Lunch!

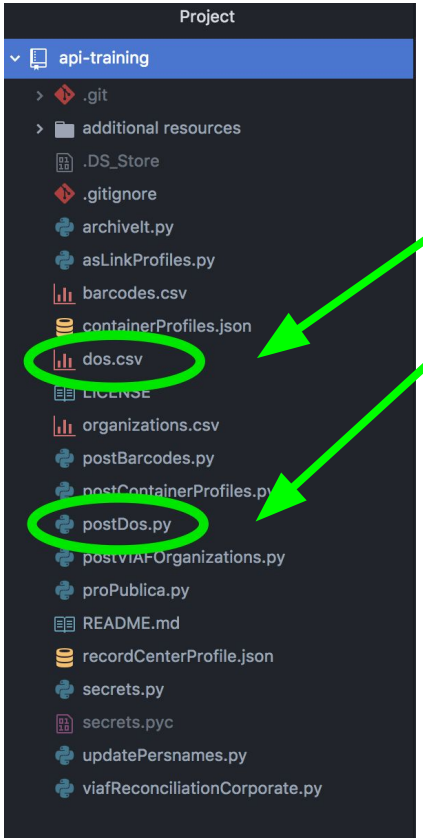
Creating Digital Objects in Bulk

Creating Digital Objects in Bulk

Scenario:

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

Creating Digital Objects in Bulk



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!

Creating Digital Objects in Bulk

Mac

1. Type `python 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 `python 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?

Load, GET, and compare with VIAF

Load, GET, and compare - VIAF

Scenario: You have an existing spreadsheet containing a number of organizational names that are either subjects or creators of some of your collections. Now, you want to take this manually-made spreadsheet and actually do some authority control work!

Pro-tip: We know your Agents are probably not in a CSV, but, if you like this script, you can GET all your agents as JSON, convert to CSV, and proceed.

Load, GET, and compare - VIAF

1. Let's take a look at "organizations.csv" online at:
   https://github.com/lorawoodford/api-training/blob/master/organizations.csv
2. From Terminal/Cygwin type `python viafReconciliationCorporate.py` and hit Enter

Load, GET, and compare - VIAF

Uh oh…

In the wilds of the internet you will encounter scripts that you want to use. Those scripts may call commands from libraries/packages (in other languages, gems, etc.) that you don't have installed. This error message and its solution is an example of real-life application.

We already successfully ran this:

proPublica.py

```
1  import json, requests, time
2
```

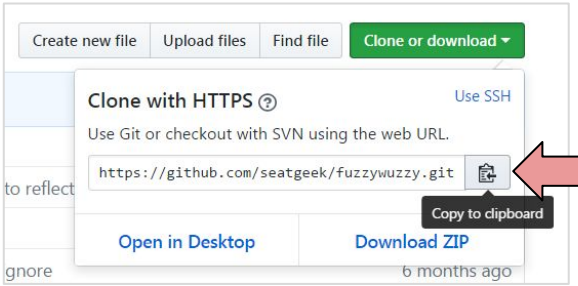
Compare that script to what we're running now:

viafReconciliationCorporate.py

```
import requests, csv, json, urllib, time
from fuzzywuzzy import fuzz
```

Technical Pitstop - Installing missing packages

Downloading the fuzzy wuzzy python package:



1. Google “fuzzy wuzzy github” and it should be the first result
2. Click the green “Clone or Download” button, click the little clipboard icon, and copy the path to the clipboard
3. Confirm that terminal/cygwin is still in the api-training folder
4. Type `git clone ` then paste the path [command+v or right-click +paste], so that it looks like the following:

   ```
   git clone https://github.com/seatgeek/fuzzywuzzy.git
   ```
5. Hit enter

Technical Pitstop - Installing missing packages

Installing the fuzzy wuzzy python package:

1. Type `cd fuzzywuzzy` to enter the newly created fuzzywuzzy directory
2. Type `ls` to see what's in the directory and note the script "setup.py" (if you don't see it, you're not there)
3. To execute that setup script, type `python setup.py install`
4. The package is **installed**!
5. Type `cd ..` to return you to the api-training directory

Load, GET, and compare - VIAF

Let's try this again!

1. Type or up-arrow `python viafReconciliationCorporate.py`
2. Success!
3. The script created this new file for you. Let's inspect it!
4. Open the newly created "viafCorporateResults.csv" file from the api-training directory

    Mac users: Desktop/api-training

    Windows users: C:\cygwin64\home\[username]\api-training

Post from a CSV - VIAF

With the VIAF **name source added** to ArchivesSpace, next:

1. Confirm you still have a file called “viafCorporateResults.csv” in your api-training directory
2. From within the api-training directory in Terminal/Cygwin execute

   ```
   python postVIAFOrganizations.py
   ```
3. Go back to the ArchivesSpace staff interface at: https://ws[your#].lyrtech.org/staff
4. In the top left click “Browse > Agents”
5. Voila!

Questions?

GET and POST across two applications with Python

Leveraging Linked Data

Leveraging Linked Data

Scenario:

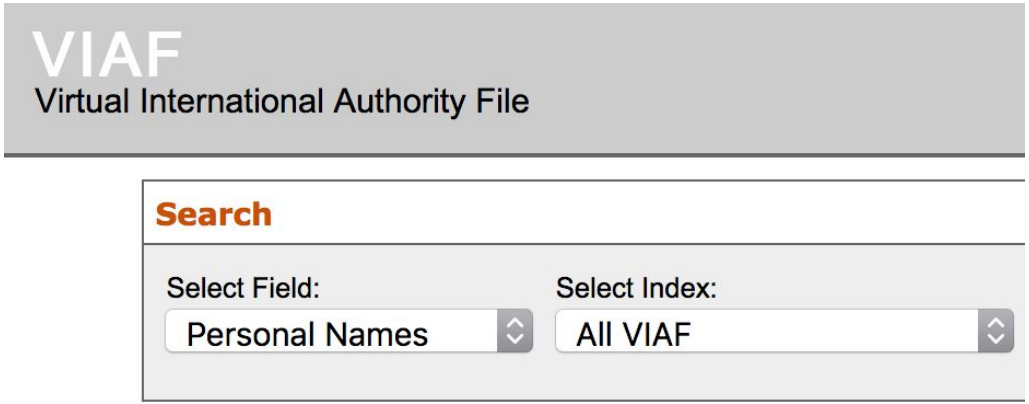
We have a number of agents (people, specifically) in ArchivesSpace that were imported from legacy data. None have VIAF identifiers (URIs), we're pretty certain some of the name forms are wrong/non-authorized, and some are outdated (sadly, people die and death dates happen!). How can we automatically update these agents in ArchivesSpace leveraging the VIAF API?

Leveraging Linked Data

1. Explore the existing person agents by navigating to Agents and filtering for Person in your instance of ArchivesSpace
2. Some of these don't quite look right, huh?

Let's fix it!

3. Execute `python updatePersnames.py`
4. Let's go back and review those agents by navigating to Agents and filtering for Person in ArchivesSpace



Pro-tip: We've made this a "best case scenario" for the purposes of this workshop, but you'd obviously want to **think carefully** about the ramifications of this script before running in production!

Leveraging Linked Data

If you paid attention to the prompt/feedback you received at the start of the script you might have noticed it said:

> *This script queries existing person agent records in ArchivesSpace with the source of "viaf" and updates them with the proper/updated name form from VIAF (if one exists) and appends the VIAF URI to the existing records.* ***Please note: This is a PROOF OF CONCEPT script, and should not be used in production settings without thinking this through!***

But why? Things I'd do differently:

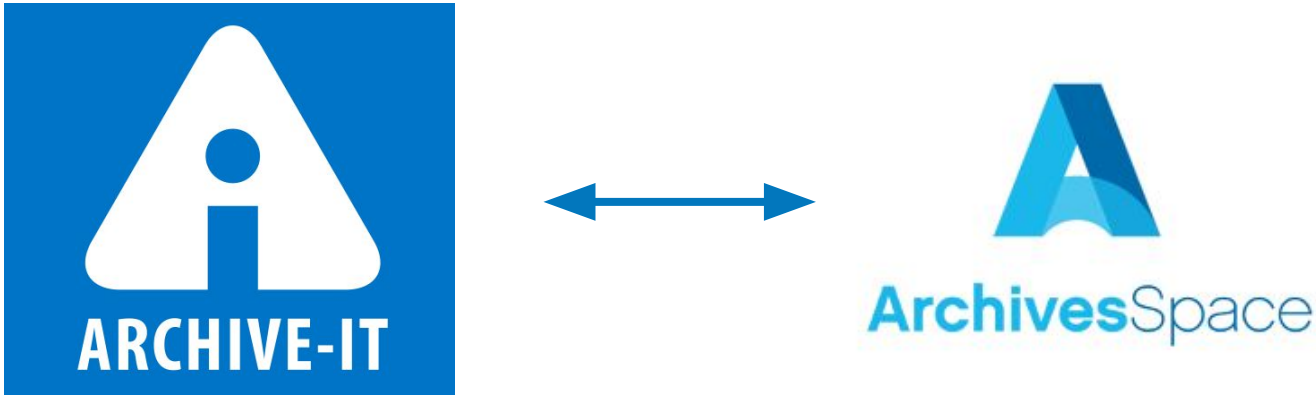
- Better logging
- Don't overwrite existing name form, but append another and ensure it is marked as authorized and display name (e.g. keep legacy data)
- If I had total control, I'd try and have those VIAF ids in there already so that we're not putting so much faith in keyword searches of VIAF

15 minute break!

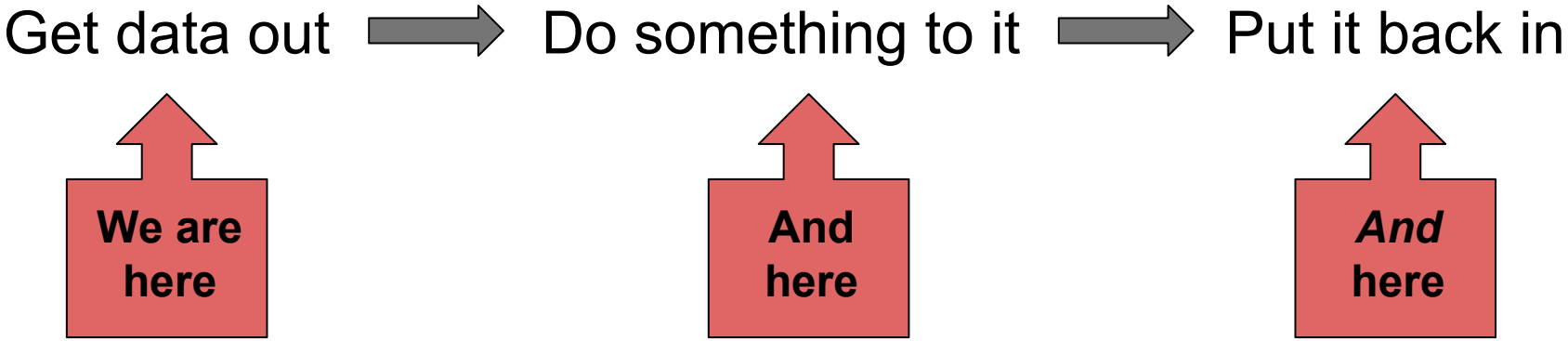
Automating (some) Archival Description

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.



API possibilities



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?

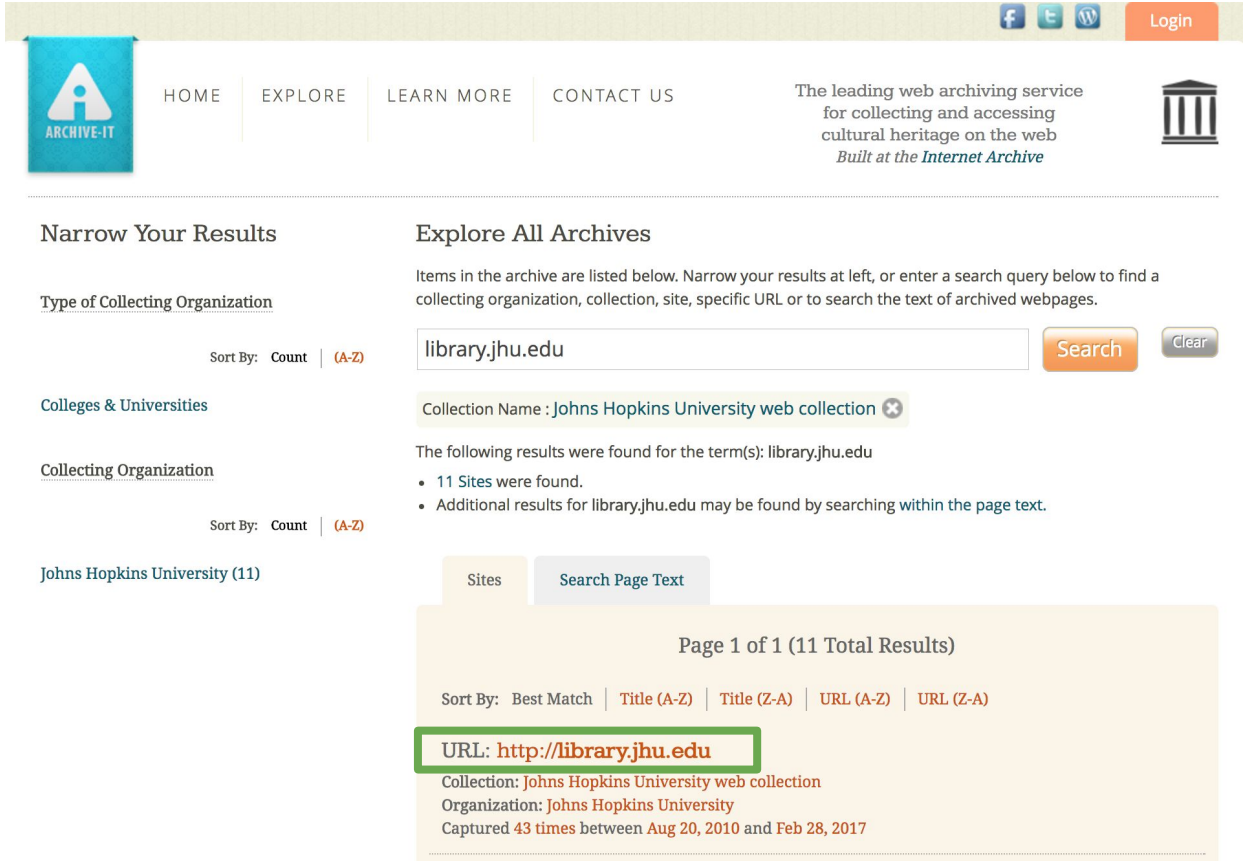
In fact, with a decent enough search we could probably even have a script return JUST those archival objects with the level **"Web archive."**

Since we're going to want to keep programmatically working with/altering this data after we find it, we'll use a **Python 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!

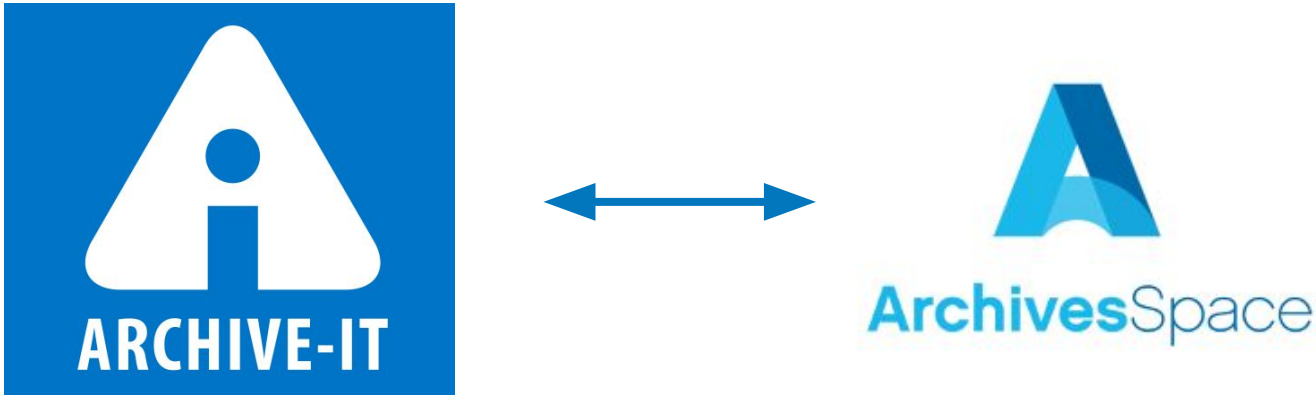
```
67        for crawl in crawlList:
68            doid = 'https://wayback.archive-it.org' + '/' + archiveit_coll + '/' +
          •     crawl['timestamp'] + '/' + crawl['original']
69            query = '/search?page=1&filter_term[]={"primary_type":"digital_object"}&q=' + doid
70            existingdoID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

Code snippet from archiveIt.py

App-to-app Communication

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

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



User stories

Reporting

- The API is not a good solution for improving AS reporting
- But this is:

https://github.com/lorawoodford/api-training/blob/master/additional%20resources/ArchivesSpace_Phila2016_Celia-ODBC.pdf

Thanks to Celia Caust-Ellenbogen, credit to Nancy Enneking, and Laurel McPhee.

Can the API bulk import data?

- Yup! The simpler the data, the better. You'll need to
  - Map what you have to extant fields
  - Get it into JSON however you can (Access, scripts)
  - 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

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

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

Re-purpose powerful scripts

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

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

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

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

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

What if you want to see all the AOs associated with a single collections?	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

Scripts you don’t have to write

And for those looking to write 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.

Icing and Advice

Icing: Interpreting (ASpace) API endpoints

Official: http://archivesspace.github.io/archivesspace/api

GET /repositories/:repo_id/resources/:id

Description

Get a Resource

Parameters

Integer id – The ID of the record

Integer repo_id – The Repository ID – The Repository must exist

[String] resolve – A list of references to resolve and embed in the response

Returns

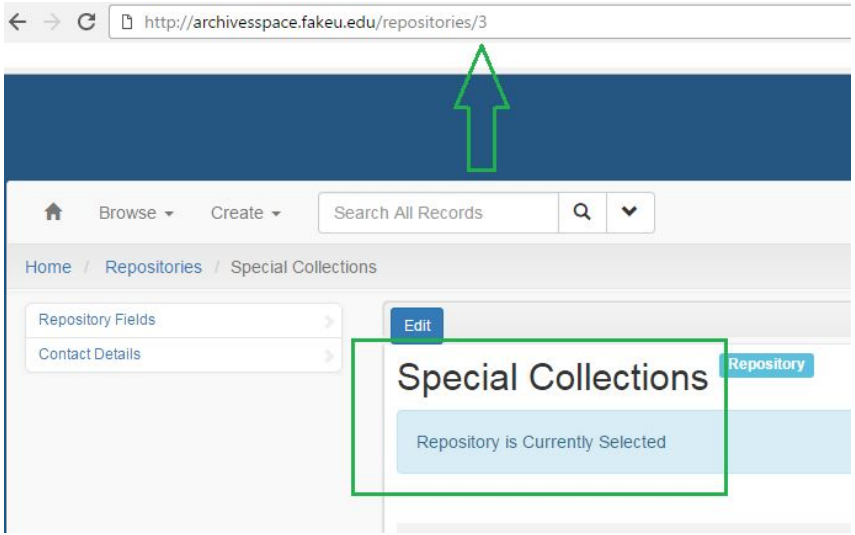
200 – (:resource)

```
curl -H "X-ArchivesSpace-Session: $SESSION" "http://localhost:8089/repositories/:repo_id/resources/1"
```

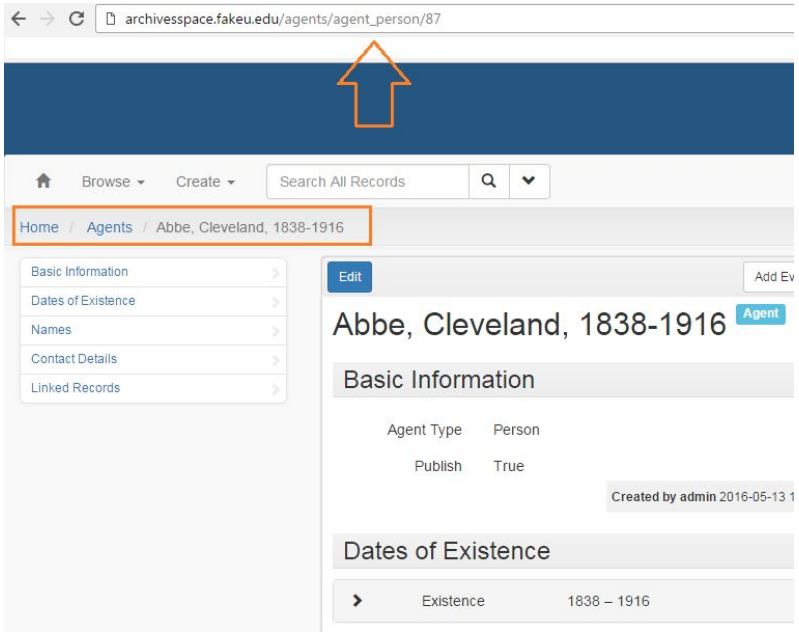
Icing: Interpreting (ASpace) API endpoints

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

Determining the repository number:



Determining an agent number:



Sample endpoint from documentation: https://ws[your#].lyrtech.org/staff/api/repositories/:repo_id/resources/1

Example "fake" endpoint that mimics real life: http://archivesspace.fakeu.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.fakeu.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.fakeu.edu:8089/repositories/3
/resources	Other examples are /accessions or /top_containers. EX. http://archivesspace.fakeu.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.fakeu.edu:8089/repositories/3/resources/1

Icing: There will ALWAYS be “gotchas”

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.

Icing: A frequent ASpace “gotcha”

it·er·a·tion
ˌidəˈrāSH(ə)n/
noun

1. the repetition of a process or utterance.
    - repetition of a mathematical or computational procedure applied to the result of a previous application, typically as a means of obtaining successively closer approximations to the solution of a problem.

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.

Icing: A frequent ASpace frustration

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
- Ask your ASpace tech person to up the session time 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/

Icing: What IS GitHub anyway?

The least and most helpful thing you'll hear is, "It's in our GitHub!"

There are 10 million GitHub intro videos on YouTube

Even casual users will benefit from using other people's scripts (that's how devs work!)

https://github.com/lorawoodford/api-training

Thanks!

Please fill out the post-workshop survey!

Join us at api-alums.slack.com

Lora Woodford | lora.woodford@lyrasis.org
Valerie Addonizio | vaddoniz@jhu.edu