Hey everyone!

We’re revising the slides

This version last updated Friday 5/12 at 12:30pm

You can use them now, but remember to check back, they’re going to improve.

And remember the GitHub!

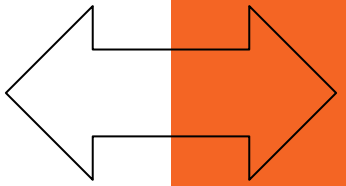
(Take home documentation *is coming*, we promise)

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
Create a new folder named “vagrant” on your desktop
Copy ***aspace.box*** off flash drive to this folder
(you only need to copy, do not open the file)

There's an API for That

Central New York Library Resources Council
May 25th, 2017

Lora J. Davis
Valerie Addonizio
Johns Hopkins University

Introduction

The Odd Couple

We are stronger and smarter together

Lora

- Digital Archivist
- 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

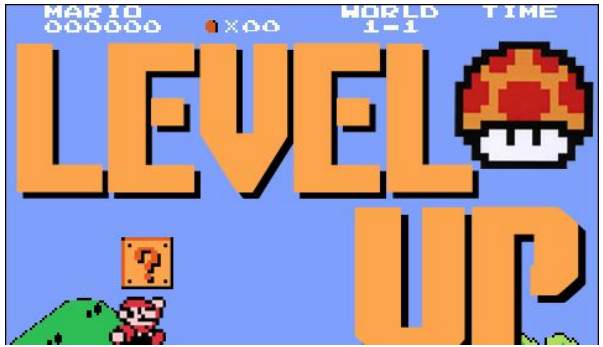
- (Former) Photo Archivist
- Never met a task she didn't want to solve with an MS Access database
- Knows rails are either "narrow gauge" or "standard" because ❤ steam trains
- Helped migrate to ASpace and felt like a fish climbing a tree
- Enjoys geocaching, hiking, and planning ambitious camping trips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

- CATS YAY!

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.



[insert corporate-looking clipart about learning and goal setting]

[this is definitely where that clipart should go]

[right here]

[inspiration, synergy, handshakes, the long road to success, get back on the horse]

Workshop aims

1.

Practical, real-life application

...including the bad parts of real-life

Resource-packed GitHub

These voluminous (over 160!) 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

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.

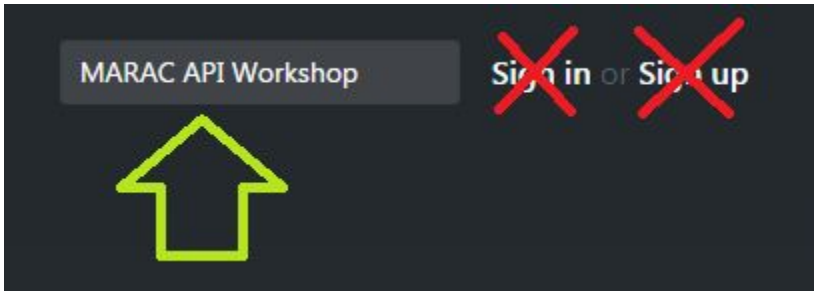
API possibilities

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?

Resource Pitstop - Get these slides

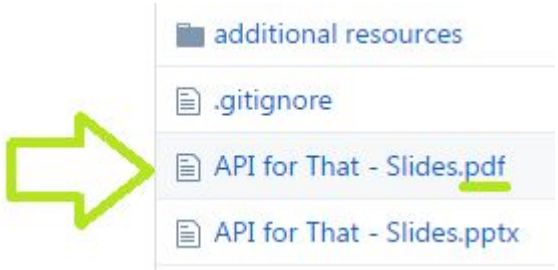
1. Navigate to GitHub.com
2. Search for “MARAC API Workshop” without quotes and hit enter (there is no search button)



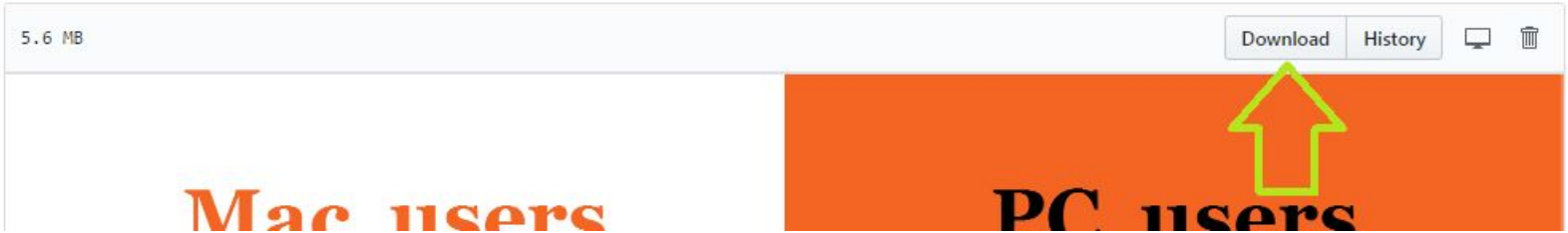
3. Select our “repo”

Resource Pitstop - Get these slides

1. Once in the repo, bookmark! You'll need this later.
2. Click "API for That - Slides.pdf"



2. Download and open. Leave these open; you will be using the slides on your own throughout the day.



Setting up your tech:

A long-ish pitstop

(We promise you're in the right workshop)

Please follow along starting from this slide

This terrible shade of yellow should be easy to find.

Everyone start with a Green post-it = good to go
Red/pink post-it = please assist!

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

VirtualBox
- A virtualization application that lets you run another machine (a “guest) within your existing computer (the “host”)
- We’re not using this in this workshop other than as a dependency for Vagrant, but you should check it out!

Vagrant
- A way to build and manage a virtual machine
- We’re using it to set you all up with a test instance of ArchivesSpace

Technical Pitstop: Scripting set-up

Technical Pitstop - Scripting set-up

We are about to:

- Show you a quick **shortcut** for opening the terminal/command prompt
- Get some important packages **installed** on your machines
- **Clone our GitHub** repository (download some scripts to your computer)

Caveats:

- This *is* scary, **we will lose some of you**
- 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:

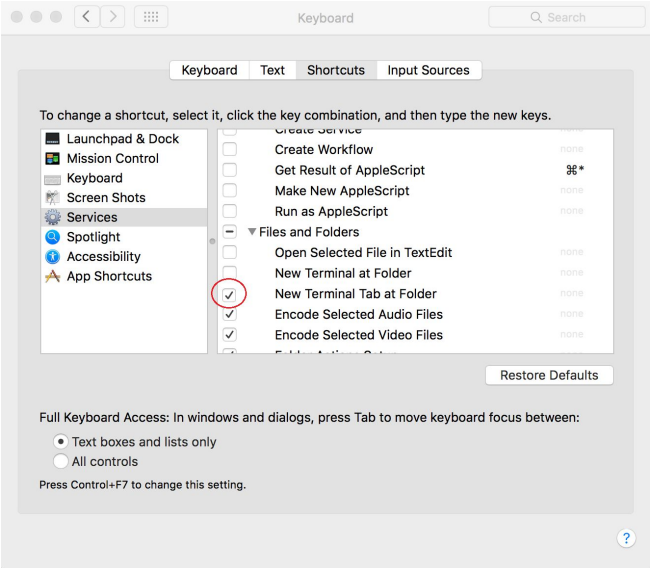
- With eight people in this class, we've got eight different environments to troubleshoot and eight different opportunities to fail - **please be patient**!
- If, at any point for the remainder of the workshop, you need **assistance**, please place a red/**hot pink post-it** on the back of your laptop screen

Technical Pitstop - Scripting set-up

This is a handy shortcut that you'll need later

Mac

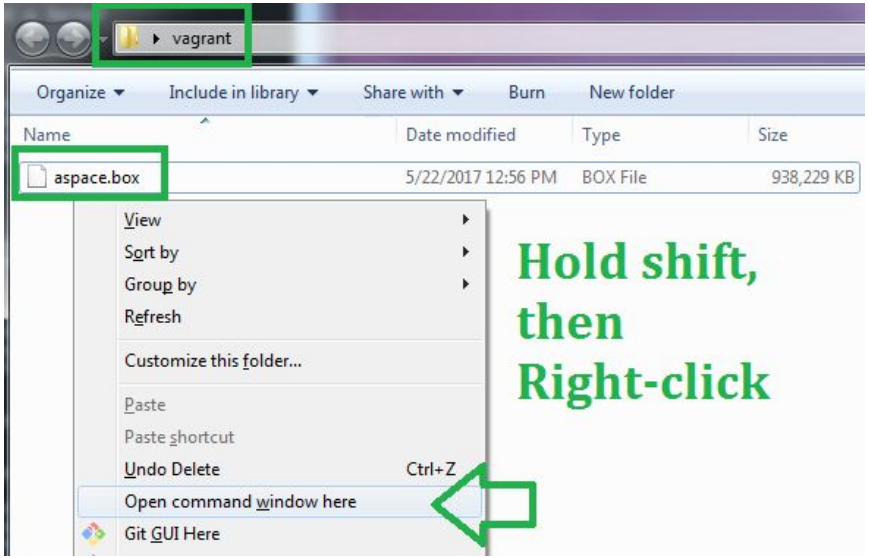
System Preferences > Keyboard > Shortcuts tab > Services



Note: Some (especially older) Mac OSes may not have this option! If so, no harm. Alert an instructor and we'll walk you through this manually.

PC

Practice: Open your new vagrant folder, then hold shift + right click anywhere in the window

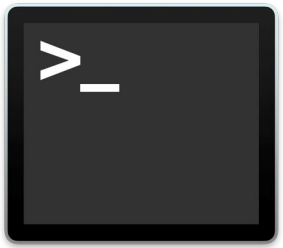


This is just practice, we don't actually need this window now.

Technical Pitstop - Scripting set-up

Mac

1. Please open the terminal:
   - Use spotlight search to search for “Terminal”
   - OR, open your Applications folder, then open the Utilities folder. Open the Terminal application.



PC

1. Please navigate to http://cygwin.com/install.html

   (that’s c-y-g-w-i-n)

2. Download and bring up the installer

Technical Pitstop - Scripting set-up

Mac

1. Type `gcc --version` and hit enter

If you are prompted to install, hit Install!
Once complete, type `gcc --version` again.

If you see the following, you're good to go.
Leave the terminal open.

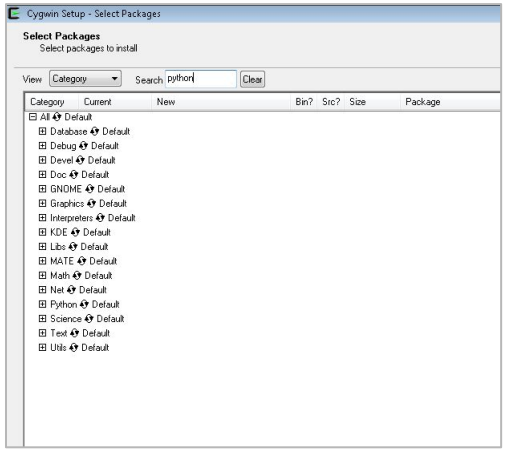
```
$ gcc --version

# After installing, check the gcc --version again.
# You should get the below message:

Configured with: --prefix=/Applications/Xcode.app/Contents/Developer/usr --with-gxx-include-dir=/usr/include/c++,
Apple LLVM version 8.0.0 (clang-800.0.38)
Target: x86_64-apple-darwin15.6.0
Thread model: posix
InstalledDir: /Applications/Xcode.app/Contents/Developer/Toolchains/XcodeDefault.xctoolchain/usr/bin
```

PC

1. Start downloading. Say yes to everything
2. Pick any site to install from and continue
3. Stop when you get to this screen:



Technical Pitstop - Scripting set-up

Mac

1. Open a browser and navigate to: **brew.sh**
   (yup that's a webpage)
2. Copy the long command and paste into terminal. Hit enter.

3. Enter your password if needed

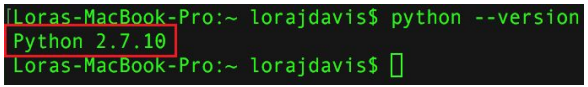
PC

1. Look for View: at the top of the screen and select Category
2. Search for "python2" (without quotes) but do not hit enter; the search happens as soon as you type

Technical Pitstop - Scripting set-up

Mac

1. Type `brew install python` and hit enter
2. Type `python --version` and hit enter to be sure Python appears:



3. Type `pip install requests`

(we will not remind you to hit enter after commands from now on)

PC

Go to next slide...

Windows users only: Install packages

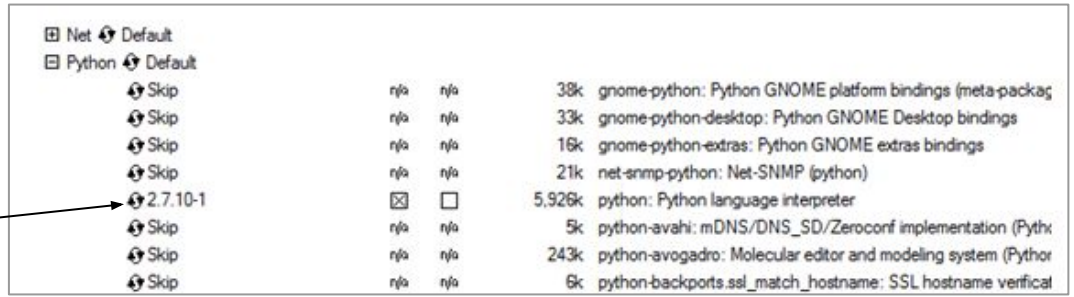
1. After you have searched for python2 locate and *unskip the following (the list is alphabetical and the kilobyte counts help)*:
   a. python2: Python 2 language interpreter **5,873k**
   b. python2-requests: Python HTTP/1.1 request module **84k**

2. Now search for git (but don't hit enter), expand the Devel heading, *unskip*:
   a. git: Distributed version control system **5,387k**

3. Finally, search for openssh (don't hit enter), expand the Net heading, *unskip*:
   a. openssh: The OpenSSH server and client programs **750k**

4. Once all four have been unskipped, proceed with install: Next > Next

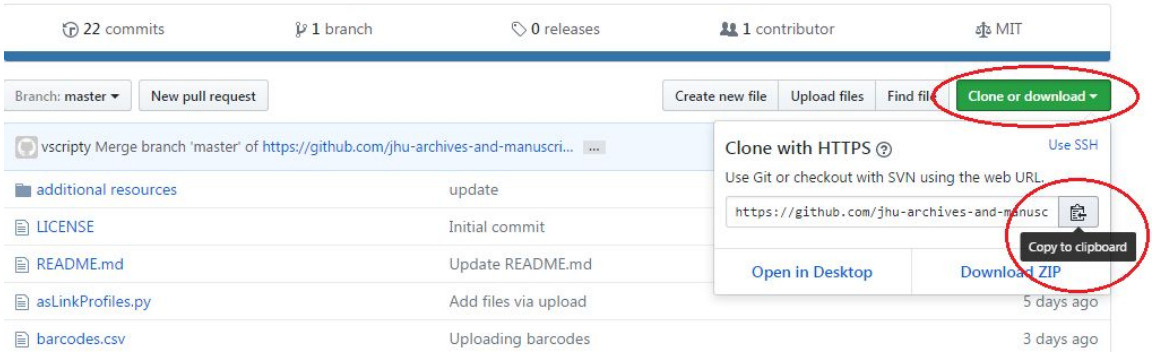
This screenshot shows what an "unskipped" line looks like



Technical Pitstop - Scripting set-up

Everyone

1. Go back to our GitHub repo, which you bookmarked earlier.
2. Then click the green button, and **Copy to clipboard**



Technical Pitstop - Scripting set-up

Mac

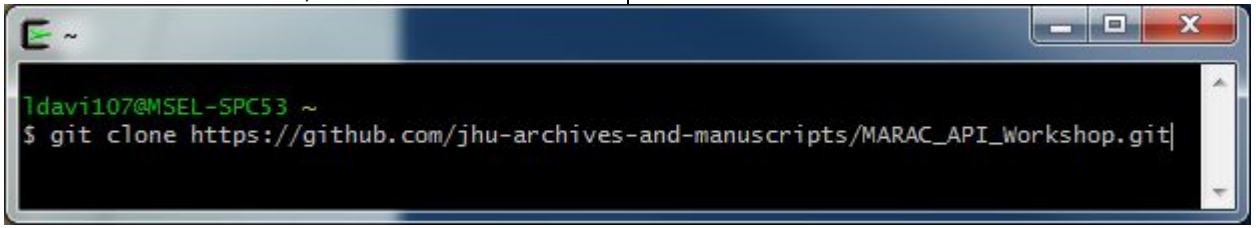
1. Open Terminal
2. Type `cd Desktop` and hit enter
3. Type `git clone [command+v then paste]`

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

PC

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

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



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 **"MARAC_API_Workshop,"** is a *direct clone* of what you see in your browser on github.com.


Just
breathe

15 minute break!

GET

API possibilities

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

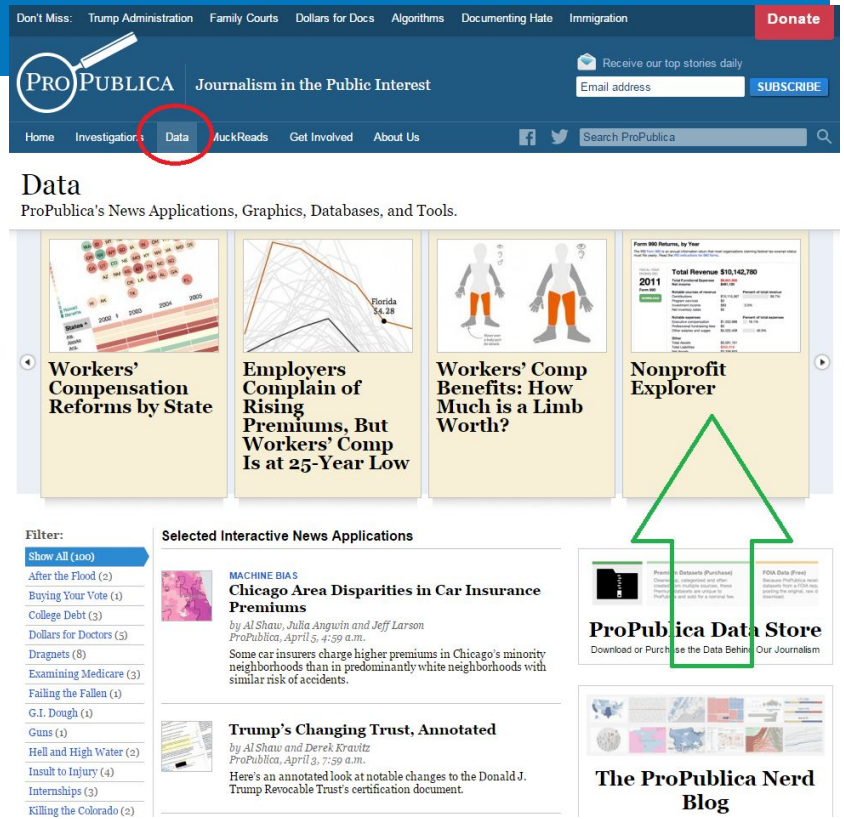
GET with GUI - ProPublica

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 is a research-based journalism site that provides its data to other reporters and the public

1. Navigate to ProPublica.org
2. Click Data
3. Click Nonprofit Explorer
4. Do a search for “animal” and hit enter.This is a list of animal-based charities
5. Look at the address bar



https://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

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/terminal
- We will be using both: Postman is a GUI

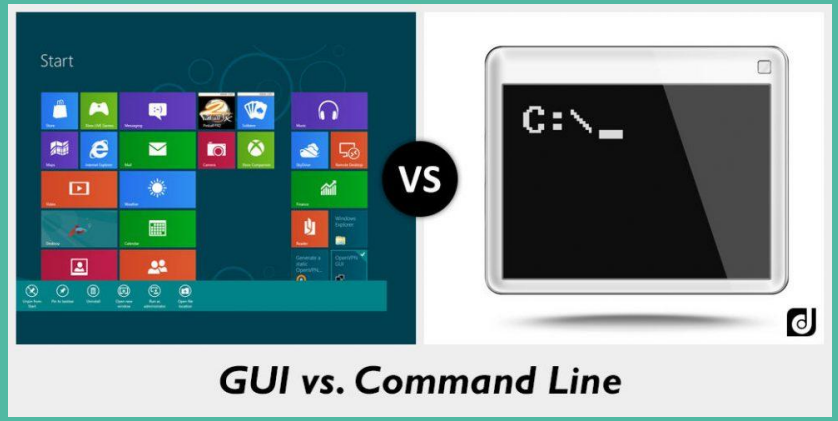
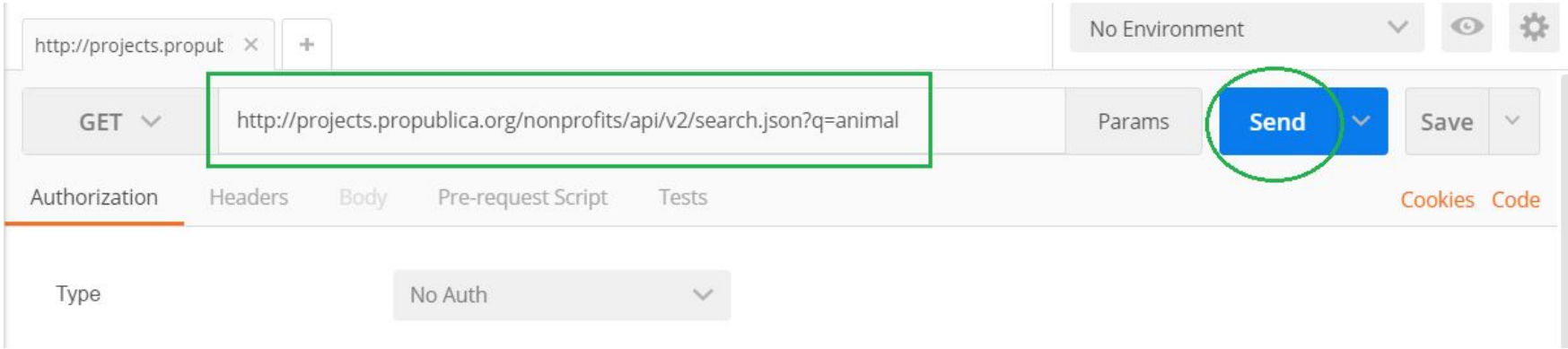


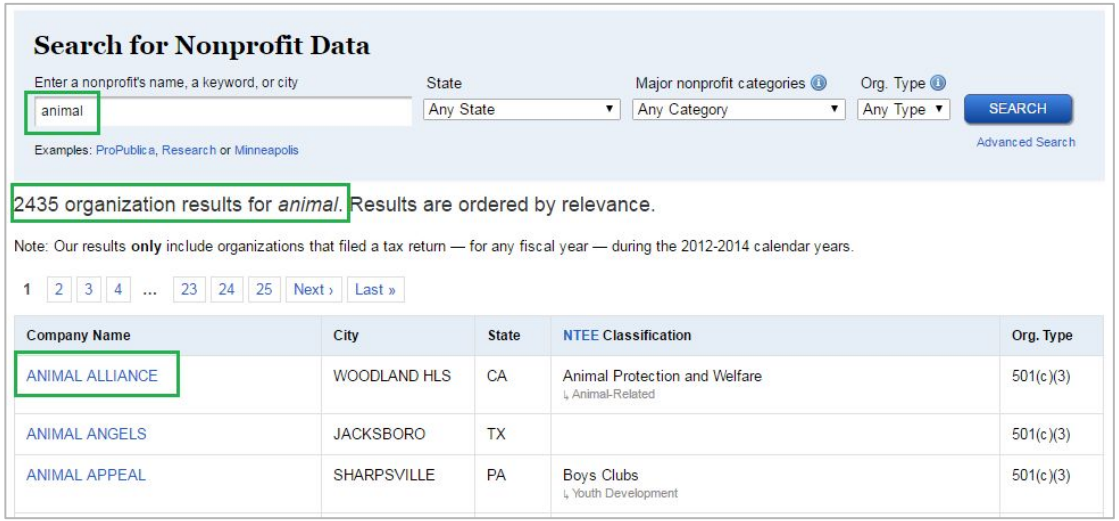
Image from http://www.differencebtw.com/difference-between-gui-and-command-line/

GET through a GUI - ProPublica



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

Web search versus API



Search for Nonprofit Data

Enter a nonprofit's name, a keyword, or city: animal

State: Any State | Major nonprofit categories: Any Category | Org. Type: Any Type | SEARCH

Examples: ProPublica, Research or Minneapolis

Advanced Search

2435 organization results for *animal*. Results are ordered by relevance.

Note: Our results **only** include organizations that filed a tax return — for any fiscal year — during the 2012-2014 calendar years.

1 2 3 4 ... 23 24 25 Next › Last »

Company Name	City	State	NTEE Classification	Org. Type
ANIMAL ALLIANCE	WOODLAND HLS	CA	Animal Protection and Welfare ↳ Animal-Related	501(c)(3)
ANIMAL ANGELS	JACKSBORO	TX		501(c)(3)
ANIMAL APPEAL	SHARPSVILLE	PA	Boys Clubs ↳ Youth Development	501(c)(3)


```
"total_results": 2435,
-"organizations": [Array[100]
  -0: {
      "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": 443.46564
  },
```


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”
```

GET through a GUI - ProPublica

So taken to extremes, I wonder what would happen if you copied the ProPublica search address into Postman and hit Send….

What about a Google search? Look at the address bar.

Amazon search? Address bar.

Your bank account?? Andress bar!

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

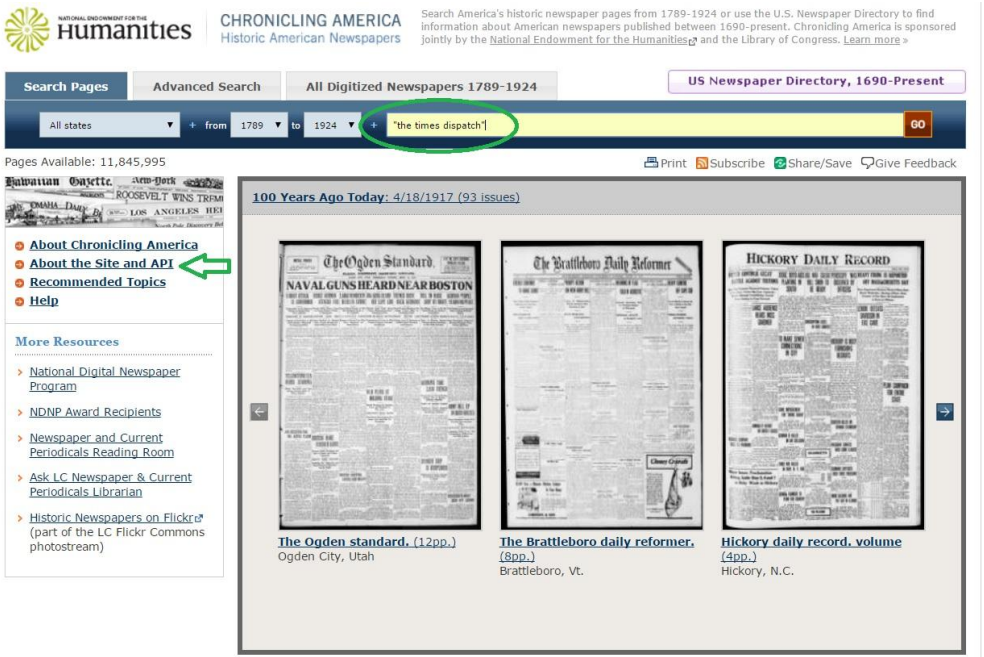
Questions???

GET with GUI - Chronicling America

Web search versus API

Scenario: You wish to link to every digitized edition of a certain newspaper hosted in Chronicling America.

1. Navigate to ChroniclingAmerica.loc.gov
2. Search for “the times dispatch” in quotes



Web search versus API

Scenario: You wish to link to every digitized edition of a certain newspaper hosted in Chronicling America.

1. Click any record
2. Click All Issues

Web search versus API

This lists every issue, but not that helpful. I wonder if there's another way.

Browse Issues: The times dispatch.
Richmond, Va. (1903-1914)

Browse Issues | About | Libraries that Have It | MARC Record

Issues for: 1903 ▾

Show all front pages

Single edition: dates in **bold**.

Multiple editions: dates in ***bold italics***.

January, 1903

S	M	T	W	T	F	S
				1	2	3
4	5	6	7	8	9	10
11	12	13	14	15	16	17
18	19	20	21	22	23	24
25	26	**27**	**28**	**29**	**30**	**31**

February, 1903

S	M	T	W	T	F	S
1	2	**3**	**4**	**5**	**6**	**7**
8	9	**10**	**11**	**12**	**13**	**14**
15	16	**17**	**18**	**19**	**20**	**21**
22	23	**24**	**25**	**26**	**27**	**28**

March, 1903

S	M	T	W	T	F	S
1	2	**3**	**4**	**5**	**6**	**7**
8	9	**10**	**11**	**12**	**13**	**14**
15	16	**17**	**18**	**19**	**20**	**21**
22	23	**24**	**25**	**26**	**27**	**28**
29	30	**31**				

April, 1903

S	M	T	W	T	F	S
			1	**2**	**3**	**4**
5	6	**7**	**8**	**9**	**10**	**11**
12	13	**14**	**15**	**16**	**17**	**18**
19	20	**21**	**22**	**23**	**24**	**25**
26	27	**28**	**29**	**30**		

May, 1903

S	M	T	W	T	F	S
					1	**2**
3	4	**5**	**6**	**7**	**8**	**9**
10	11	**12**	**13**	**14**	**15**	**16**
17	18	**19**	**20**	**21**	**22**	**23**
24	25	**26**	**27**	**28**	**29**	**30**
31						

June, 1903

S	M	T	W	T	F	S
	1	**2**	**3**	**4**	**5**	**6**
7	8	**9**	**10**	**11**	**12**	**13**
14	15	**16**	**17**	**18**	**19**	**20**
21	22	**23**	**24**	**25**	**26**	**27**
28	29	**30**				

July, 1903

S	M	T	W	T	F	S
			1	**2**	**3**	**4**
5	6	**7**	**8**	**9**	**10**	**11**
12	13	**14**	**15**	**16**	**17**	**18**
19	20	**21**	**22**	**23**	**24**	**25**
26	27	**28**	**29**	**30**	**31**	

August, 1903

S	M	T	W	T	F	S
						1
2	3	**4**	**5**	**6**	**7**	**8**
9	10	**11**	**12**	**13**	**14**	**15**
16	17	**18**	**19**	**20**	**21**	**22**
23	24	**25**	**26**	**27**	**28**	**29**
30	31					

September, 1903

S	M	T	W	T	F	S
		1	**2**	**3**	**4**	**5**
6	7	**8**	**9**	**10**	**11**	**12**
13	14	**15**	**16**	**17**	**18**	**19**
20	21	**22**	**23**	**24**	**25**	**26**
27	28	**29**	**30**			

October, 1903

S	M	T	W	T	F	S
				1	**2**	**3**
4	5	**6**	**7**	**8**	**9**	**10**
11	12	**13**	**14**	**15**	**16**	**17**
18	19	**20**	**21**	**22**	**23**	**24**
25	26	**27**	**28**	**29**	**30**	**31**

November, 1903

S	M	T	W	T	F	S
1	2	**3**	**4**	**5**	**6**	**7**
8	9	**10**	**11**	**12**	**13**	**14**
15	16	**17**	**18**	**19**	**20**	**21**
22	23	**24**	**25**	**26**	**27**	**28**
29	30					

December, 1903

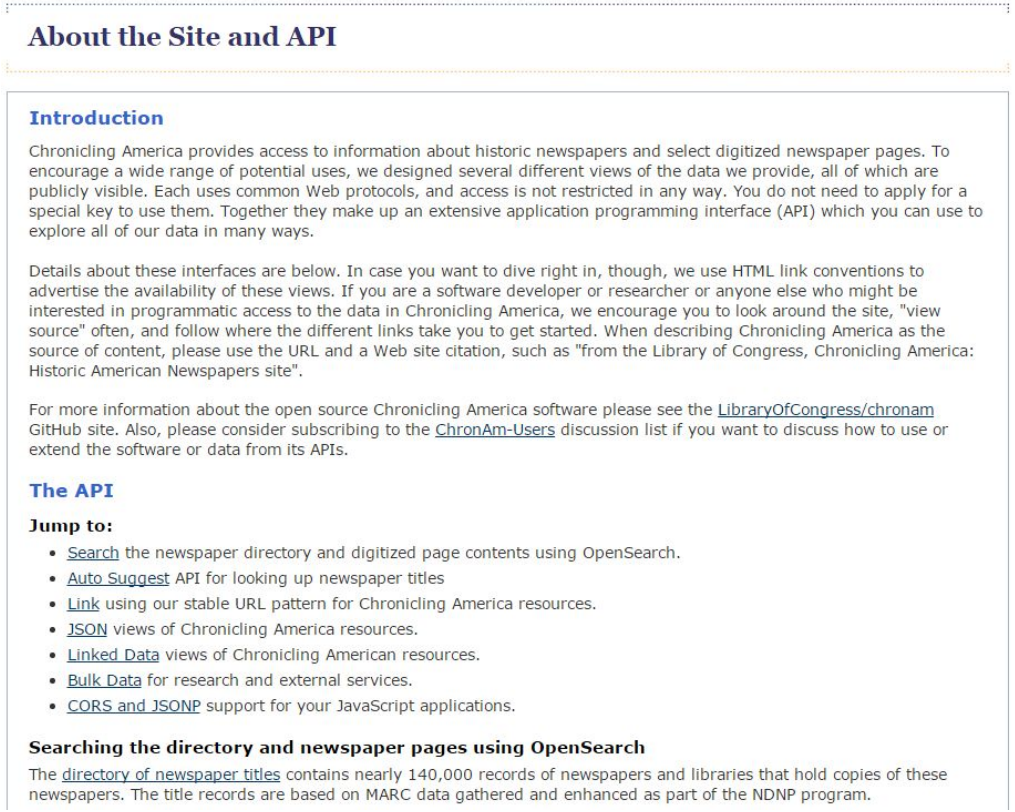
S	M	T	W	T	F	S
		1	**2**	**3**	**4**	**5**
6	7	**8**	**9**	**10**	**11**	**12**
13	14	**15**	**16**	**17**	**18**	**19**
20	21	**22**	**23**	**24**	**25**	26
27	28	**29**	**30**	**31**		

GET with GUI - Chronicling America

Does **Chronicling America** have an **API** we can use to access this information we're seeing in our browsers?

YES!

Back to Postman!



About the Site and API

Introduction

Chronicling America provides access to information about historic newspapers and select digitized newspaper pages. To encourage a wide range of potential uses, we designed several different views of the data we provide, all of which are publicly visible. Each uses common Web protocols, and access is not restricted in any way. You do not need to apply for a special key to use them. Together they make up an extensive application programming interface (API) which you can use to explore all of our data in many ways.

Details about these interfaces are below. In case you want to dive right in, though, we use HTML link conventions to advertise the availability of these views. If you are a software developer or researcher or anyone else who might be interested in programmatic access to the data in Chronicling America, we encourage you to look around the site, "view source" often, and follow where the different links take you to get started. When describing Chronicling America as the source of content, please use the URL and a Web site citation, such as "from the Library of Congress, Chronicling America: Historic American Newspapers site".

For more information about the open source Chronicling America software please see the LibraryOfCongress/chronam GitHub site. Also, please consider subscribing to the ChronAm-Users discussion list if you want to discuss how to use or extend the software or data from its APIs.

The API

Jump to:

- Search the newspaper directory and digitized page contents using OpenSearch.
- Auto Suggest API for looking up newspaper titles
- Link using our stable URL pattern for Chronicling America resources.
- JSON views of Chronicling America resources.
- Linked Data views of Chronicling American resources.
- Bulk Data for research and external services.
- CORS and JSONP support for your JavaScript applications.

Searching the directory and newspaper pages using OpenSearch

The directory of newspaper titles contains nearly 140,000 records of newspapers and libraries that hold copies of these newspapers. The title records are based on MARC data gathered and enhanced as part of the NDNP program.

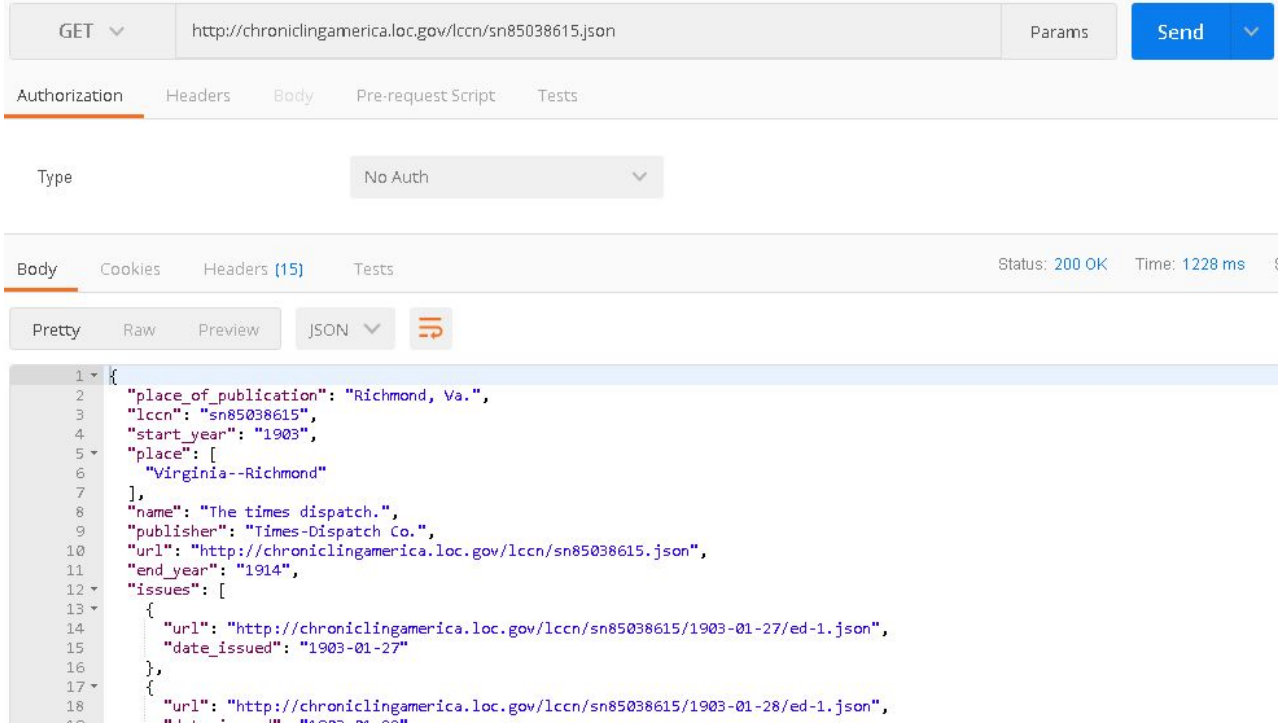
GET with GUI - Chronicling America

Scenario: You wish to link to every digitized edition of a certain newspaper in Chronicling America.

http://chroniclingamerica.loc.gov/lccn/sn85038615.json

GET with GUI - Chronicling America

Scenario: You wish to link to every digitized edition of a certain newspaper in Chronicling America.



Re-purposing API data

	A	B
1	**url**	**date_issued**
2	http://chroniclingamerica.loc.gov/lccn/sn85038615/1903-01-27/ed-1.json	1903-01-27
3	http://chroniclingamerica.loc.gov/lccn/sn85038615/1903-01-28/ed-1.json	1903-01-28
4	http://chroniclingamerica.loc.gov/lccn/sn85038615/1903-01-29/ed-1.json	1903-01-29
5	http://chroniclingamerica.loc.gov/lccn/sn85038615/1903-01-30/ed-1.json	1903-01-30
6	http://chroniclingamerica.loc.gov/lccn/sn85038615/1903-01-31/ed-1.json	1903-01-31
7	http://chroniclingamerica.loc.gov/lccn/sn85038615/1903-02-01/ed-1.json	1903-02-01
8	http://chroniclingamerica.loc.gov/lccn/sn85038615/1903-02-03/ed-1.json	1903-02-03
9	http://chroniclingamerica.loc.gov/lccn/sn85038615/1903-02-04/ed-1.json	1903-02-04
10	http://chroniclingamerica.loc.gov/lccn/sn85038615/1903-02-05/ed-1.json	1903-02-05
11	http://chroniclingamerica.loc.gov/lccn/sn85038615/1903-02-06/ed-1.json	1903-02-06
12	http://chroniclingamerica.loc.gov/lccn/sn85038615/1903-02-07/ed-1.json	1903-02-07
13	http://chroniclingamerica.loc.gov/lccn/sn85038615/1903-02-08/ed-1.json	1903-02-08
14	http://chroniclingamerica.loc.gov/lccn/sn85038615/1903-02-10/ed-1.json	1903-02-10
15	http://chroniclingamerica.loc.gov/lccn/sn85038615/1903-02-11/ed-1.json	1903-02-11
16	http://chroniclingamerica.loc.gov/lccn/sn85038615/1903-02-12/ed-1.json	1903-02-12
17	http://chroniclingamerica.loc.gov/lccn/sn85038615/1903-02-13/ed-1.json	1903-02-13
18	http://chroniclingamerica.loc.gov/lccn/sn85038615/1903-02-14/ed-1.json	1903-02-14
19	http://chroniclingamerica.loc.gov/lccn/sn85038615/1903-02-15/ed-1.json	1903-02-15
20	http://chroniclingamerica.loc.gov/lccn/sn85038615/1903-02-17/ed-1.json	1903-02-17
21	http://chroniclingamerica.loc.gov/lccn/sn85038615/1903-02-18/ed-1.json	1903-02-18
22	http://chroniclingamerica.loc.gov/lccn/sn85038615/1903-02-19/ed-1.json	1903-02-19
23	http://chroniclingamerica.loc.gov/lccn/sn85038615/1903-02-20/ed-1.json	1903-02-20
24	http://chroniclingamerica.loc.gov/lccn/sn85038615/1903-02-21/ed-1.json	1903-02-21
25	http://chroniclingamerica.loc.gov/lccn/sn85038615/1903-02-22/ed-1.json	1903-02-22
26	http://chroniclingamerica.loc.gov/lccn/sn85038615/1903-02-24/ed-1.json	1903-02-24
27	http://chroniclingamerica.loc.gov/lccn/sn85038615/1903-02-25/ed-1.json	1903-02-25
28	http://chroniclingamerica.loc.gov/lccn/sn85038615/1903-02-26/ed-1.json	1903-02-26
29	http://chroniclingamerica.loc.gov/lccn/sn85038615/1903-02-27/ed-1.json	1903-02-27
30	http://chroniclingamerica.loc.gov/lccn/sn85038615/1903-02-28/ed-1.json	1903-02-28

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

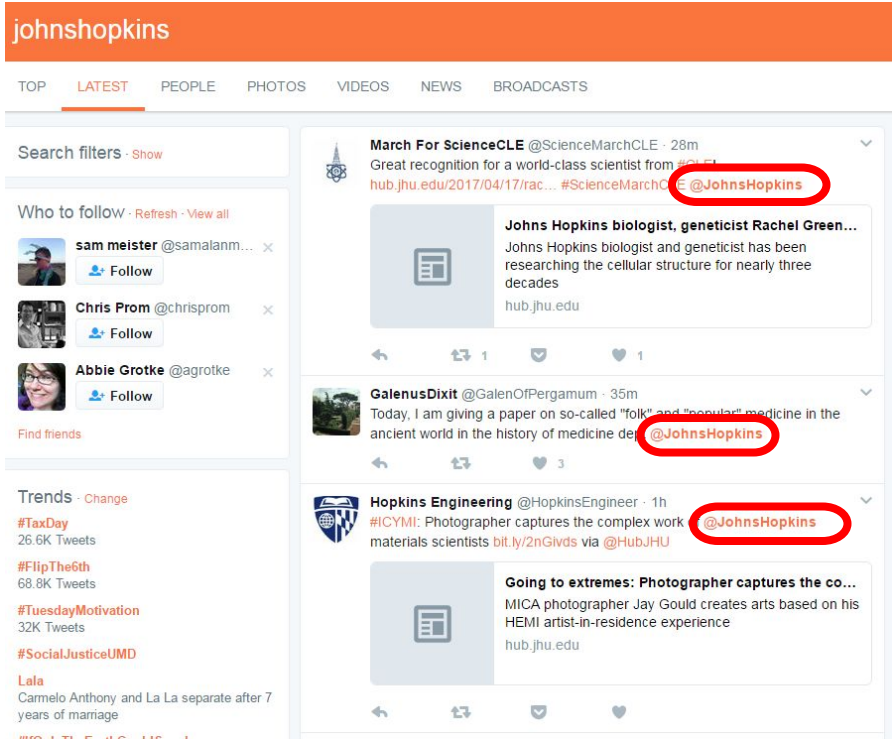
Questions???

GET with GUI - Twitter

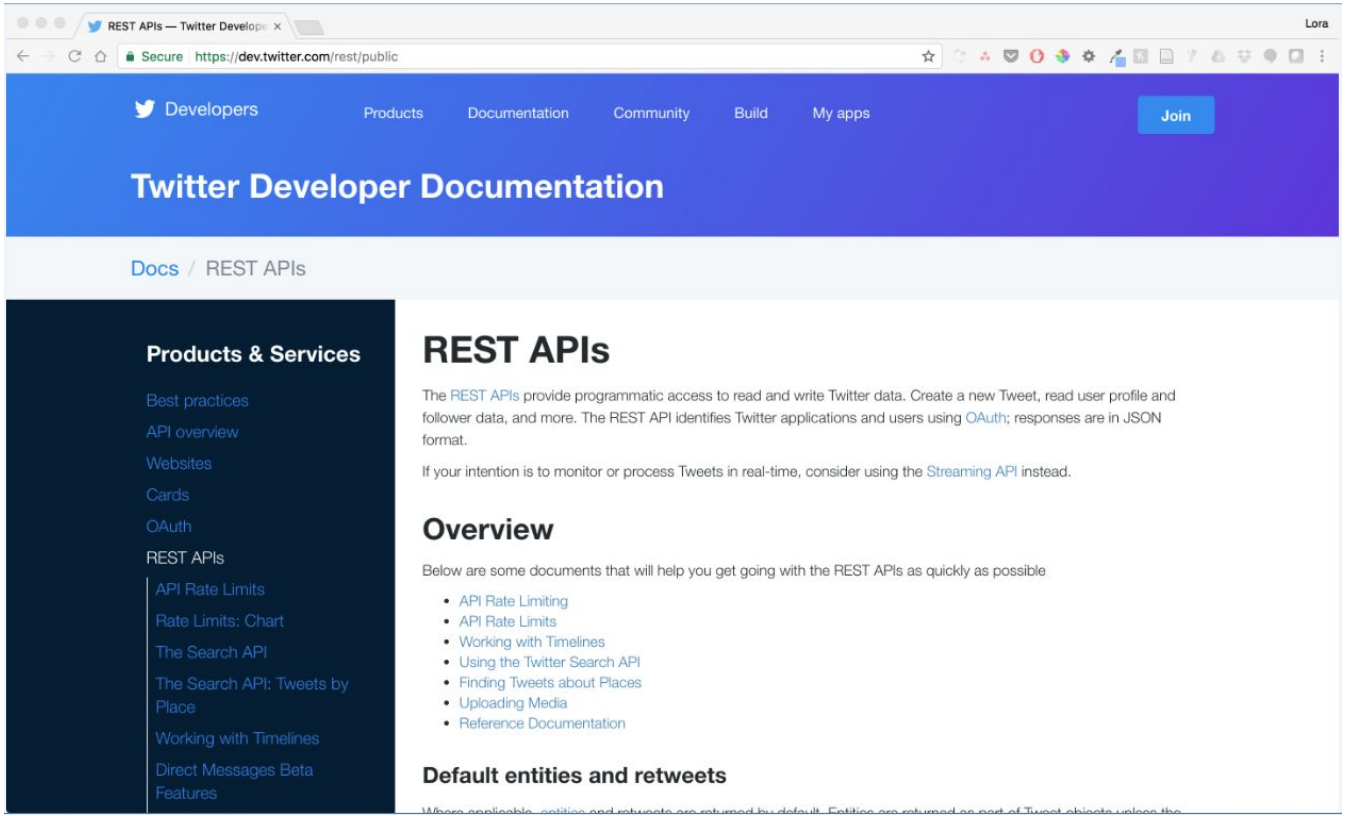
Using an API to collect records

@JohnsHopkins

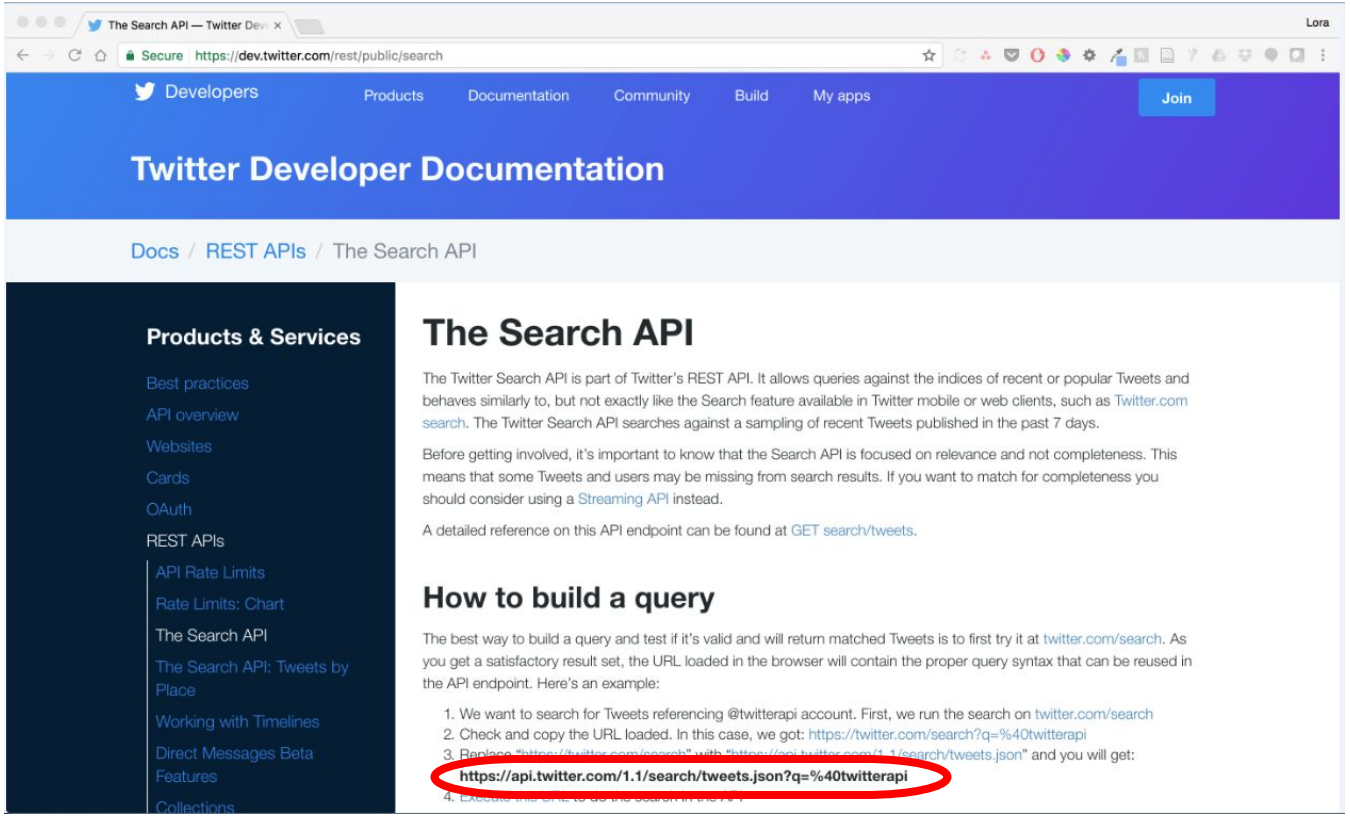
Scenario: As your university's records manager, you wish to regularly capture Tweets mentioning your university.



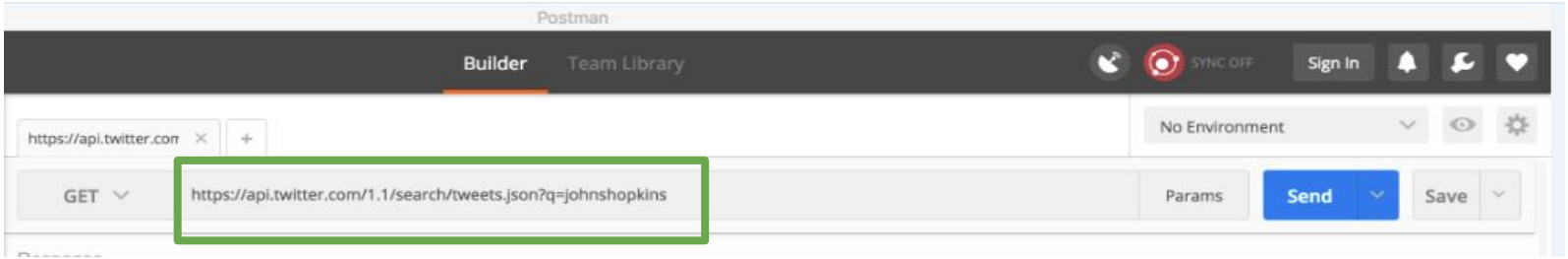
Using an API to collect records



Using an API to collect records



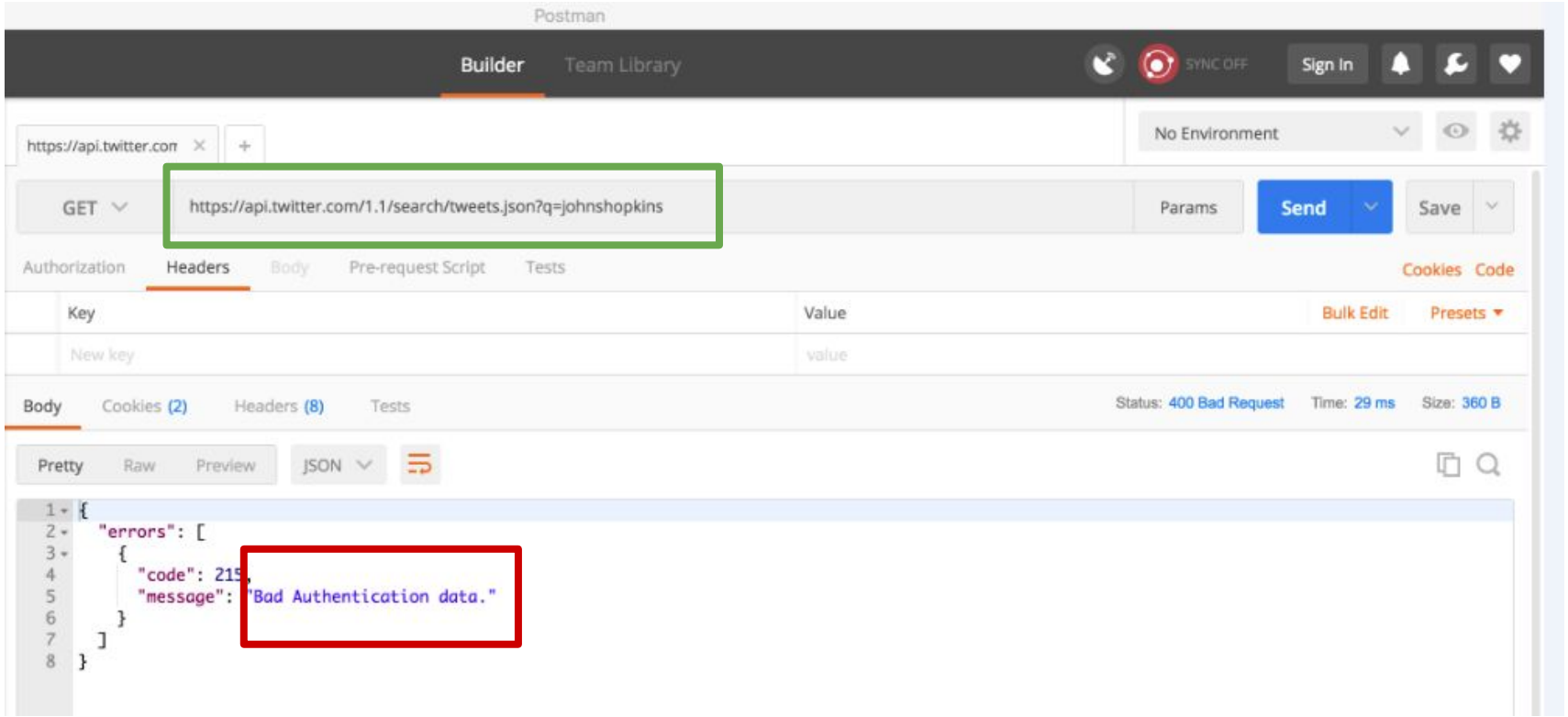
Using an API to collect records



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

Authentication on all endpoints

Applications must authenticate all requests with OAuth 1.0a or Application-only authentication. This allows us to prevent abusive behavior, and it also helps us to further understand how categories of applications are using the API. We apply this understanding to better meet the needs of developers as we continue to evolve the platform.

Source: https://dev.twitter.com/rest/public

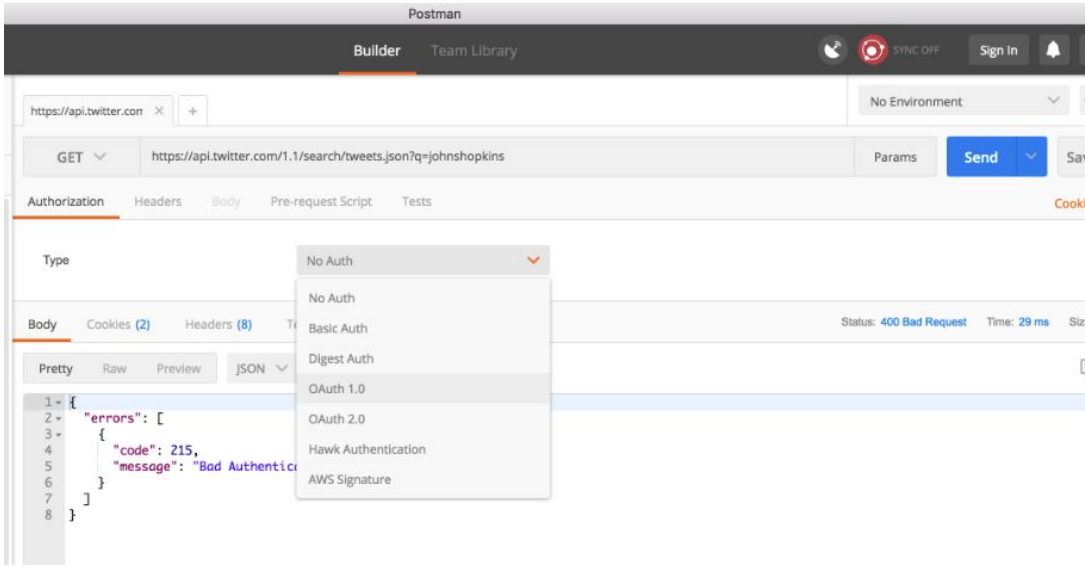
Using an API to collect records

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": "ScienceMarchCLE",
            "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
            ]
```


F. Be a Good Partner to Twitter

1. Follow the guidelines for using Tweets in broadcast if you display Tweets offline.
2. If you provide Content to third parties, including downloadable datasets of Content or an API that returns Content, you will only distribute or allow download of Tweet IDs and/or User IDs.
    a. You may, however, provide export via non-automated means (e.g., download of spreadsheets or PDF files, or use of a "save as" button) of up to 50,000 public Tweets and/or User Objects per user of your Service, per day.
    b. Any Content provided to third parties via non-automated file download remains subject to this Policy.

Questions???

Scripting

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, because:

- 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)

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: Navigation

Where are you, and where do you want to go?

Mac

1. In the Finder navigate to your MARAC_API_Workshop directory
2. Ctrl+click the MARAC_API_Workshop directory, and select “New Terminal at Folder”

PC

1. Open Cygwin

Command line bootcamp: Navigation

Where are you?

Everyone type `pwd` and then hit enter.

Mac

Mac users should see something like this:

```
Loras-MacBook-Pro:MARAC_API_Workshop lorajdavis$ pwd
/Users/lorajdavis/Desktop/MARAC_API_Workshop
Loras-MacBook-Pro:MARAC_API_Workshop lorajdavis$
```

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:

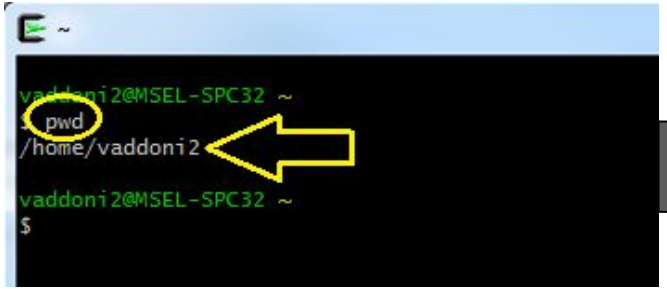
Command line bootcamp: Navigation

Where are you, and where do you want to go?

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:

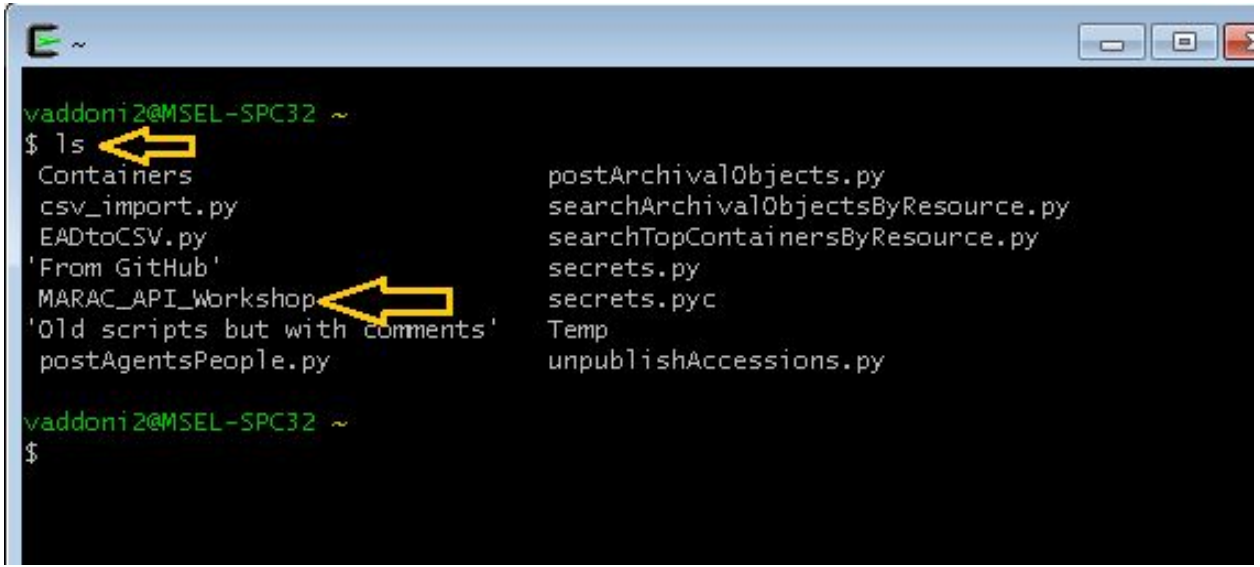
Unix commands for Mac and Cygwin

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

Command line bootcamp: ls

What is here?

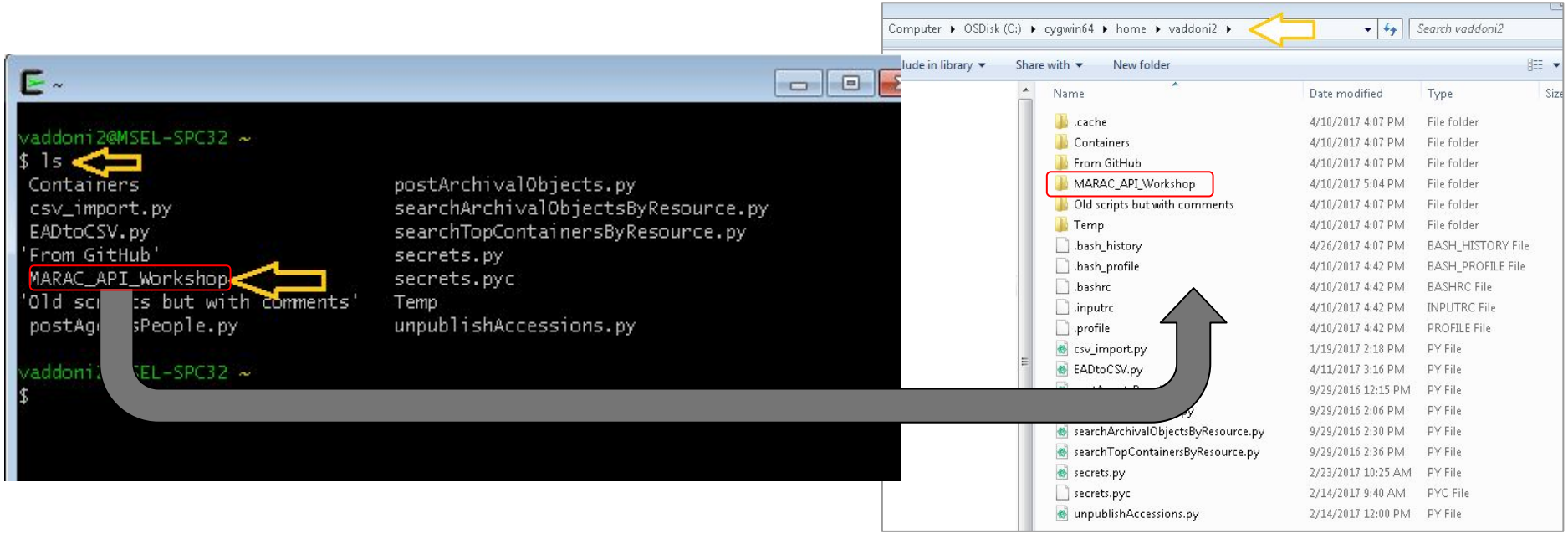
Everyone type `ls` and then hit enter (that is **L** as in **L**ist)



Command line bootcamp: ls

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

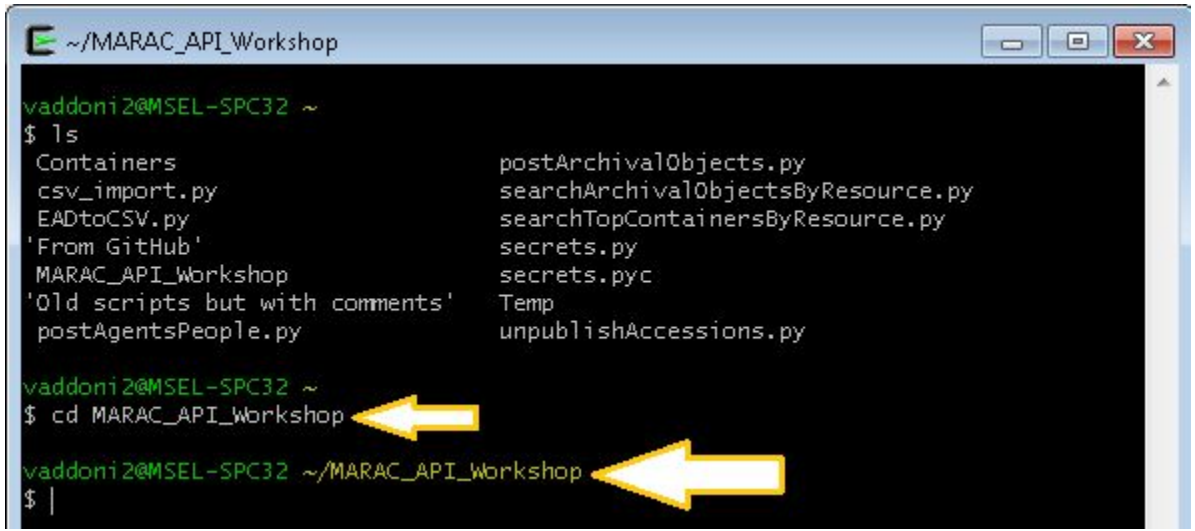
Where am I?	“print working directory”	`pwd`
What is here?	“list” (remember L as in List)	`ls`

Command line bootcamp: Navigation

Where are you, and where do you want to go?

Now you're going to move *from* where you are *into* the MARAC API Workshop clone folder:

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



Command line bootcamp: Navigation

Where are you, and where do you want to go?

PC

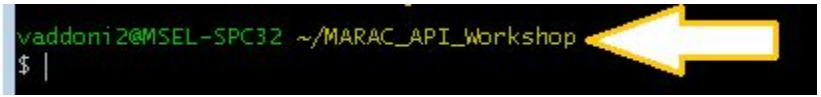
Happily, the directory you're in now is more obvious with that handy yellow text.

So remember:

- The path in Windows is: C:\cygwin64\home\[user name]\MARAC_API_Workshop

  (but this varies by user)

- And the *same path* in Cgywin looks like the new prompt, below:



Unix commands for Mac and Cygwin

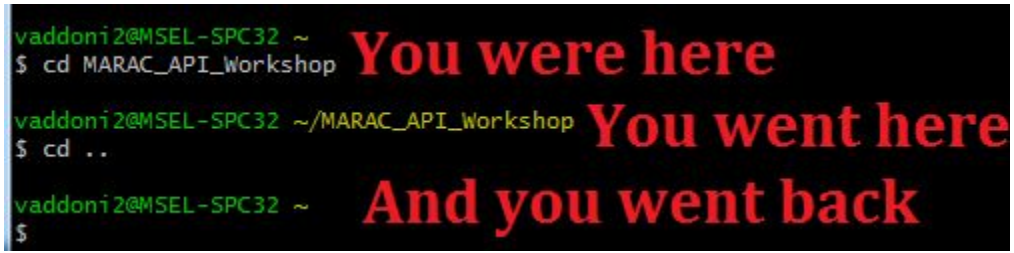
Where am I?	“print working directory”	`pwd`
What is here?	“list” (remember L as in List)	`ls`
How do I move from here to there?	“change directory”	`cd [type the name of the directory]` (don’t type this, we mean an action)

Command line bootcamp: Navigation

Now you're going to move *from* the MARAC API Workshop clone folder back to the Cgywin 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: Navigation

Lastly, let's go back into the MARAC_API_Workshop 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 MARAC_API_Workshop 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; those are MS-DOS commands.

GET with Script-ProPublica

GET with a script - ProPublica

Let's return to our ProPublica example from earlier.

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

But, this time, let's assume that this user isn't content just copying/pasting pages of data out of **Postman**. This researcher has **big plans** for this dataset, and wants a **standalone JSON file** instead!

> Scenario, v. 2: A user wants **a JSON file** containing ProPublica Nonprofit Explorer data about every non-profit involving animal welfare.

GET with a script - ProPublica

Mac

1. In the Finder navigate to your MARAC_API_Workshop directory
2. Ctrl+click the MARAC_API_Workshop 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 MARAC_API_Workshop` to enter the MARAC_API_Workshop 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 MARAC_API_Workshop directory
7. Launch Atom and open "proPublicaRecord.json"

Directory reminder: Desktop/MARAC_API_Workshop

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

PC

6. A new file called "proPublicaRecord.json" should now be in your MARAC_API_Workshop directory
7. Launch Atom and open "proPublicaRecord.json"

Directory reminder:
C:\cgywin\home\[username]\MARAC_API_Directory

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

Questions???

Lunch!

Load, GET, and compare - VIAF

API possibilities

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

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!

Load, GET, and compare - VIAF

1. Navigate to your MARAC_API_Workshop folder (Mac: Desktop; Windows: Cygwin folder)
2. Open “organizations.csv” in Excel/Numbers (or your preferred spreadsheet program)
3. Add a few additional corporate names of your choosing (universities, businesses, etc.) to column A
4. Save the file as a .csv
5. From Terminal/Cygwin type

`python viafReconciliationCorporate.py` and hit Enter

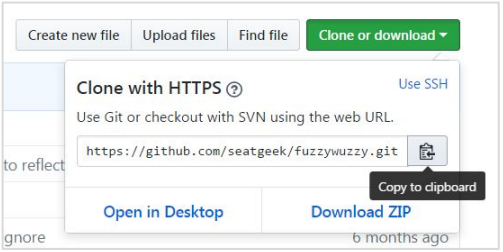
Load, GET, and compare - VIAF

Uh oh…

Technical Pitstop - Installing extra packages

Everyone

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, and copy the path to the clipboard
3. Confirm that terminal/cygwin is still in the MARAC_API_Workshop folder
4. Type `git clone {paste path}`, which should look like

   ```
   git clone
   https://github.com/seatgeek/fuzzywuzzy.git
   ```

Technical Pitstop - Installing extra packages

Everyone

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"
3. To execute that setup script, type `python setup.py install`
4. The package is **installed**!
5. Type `cd ..` to return you to the MARAC_API_Workshop directory

Load, GET, and compare - VIAF

Let's try this again!

1. Type `python viafReconciliationCorporate.py`
2. Success!
3. There is a newly created "viafCorporateResults.csv" file in your MARAC_API_Workshop directory

   Mac users: Desktop

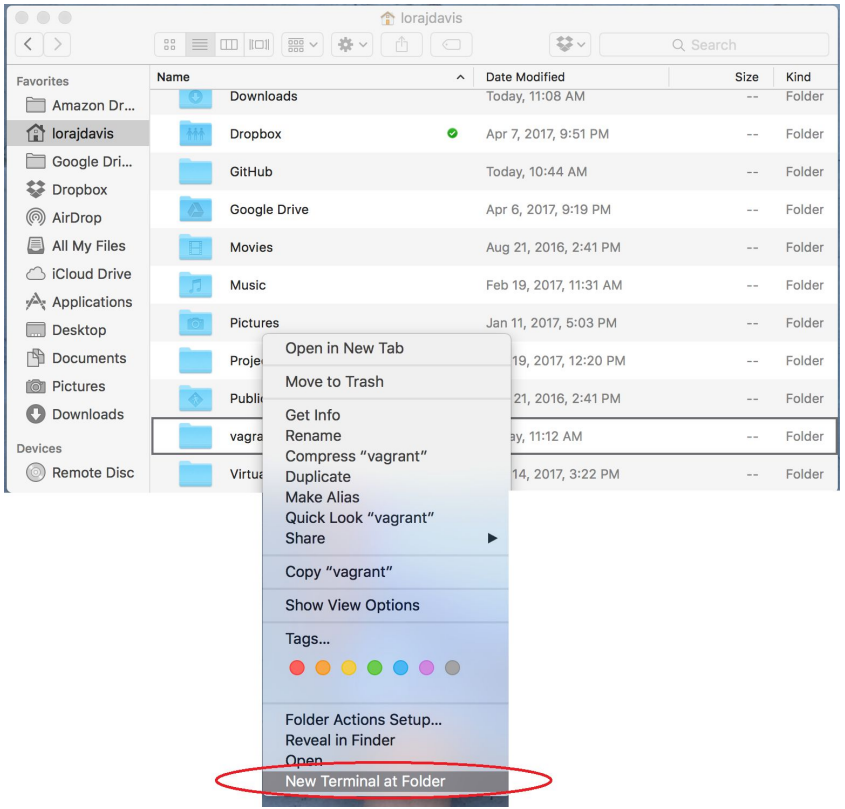
   Windows users: C:\cygwin64\home\[username]\MARAC_API_Workshop
4. Let's open and inspect it!

Questions???

Technical Pitstop: vagrant install and `vagrant up`

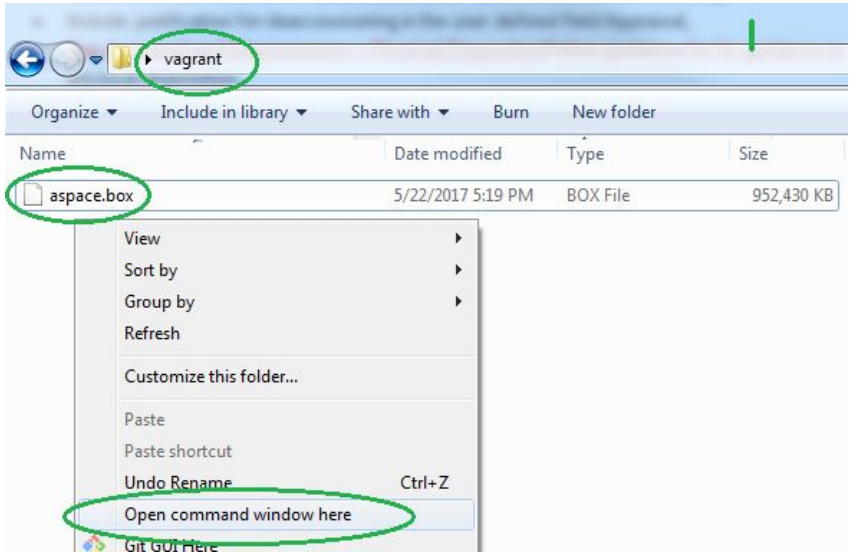
(this is super awesome)

Mac

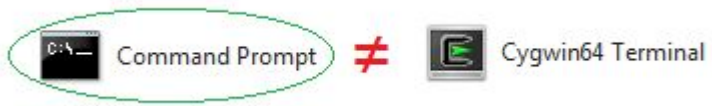


PC

Navigate to your new vagrant folder on the desktop. Open the command prompt in that folder with **shift+Right-click**:



Note: The command prompt is not Cgywin, we specifically mean the command prompt on this slide:

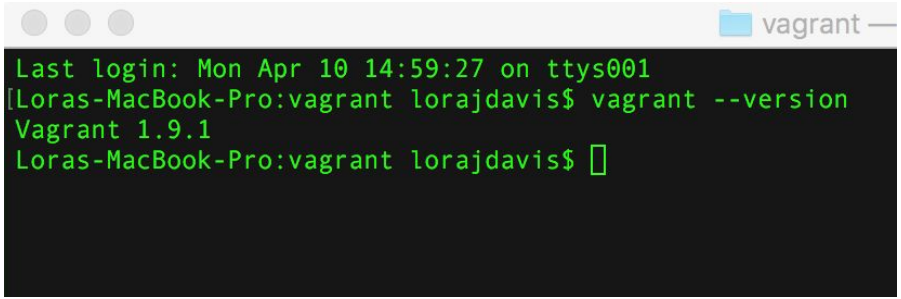


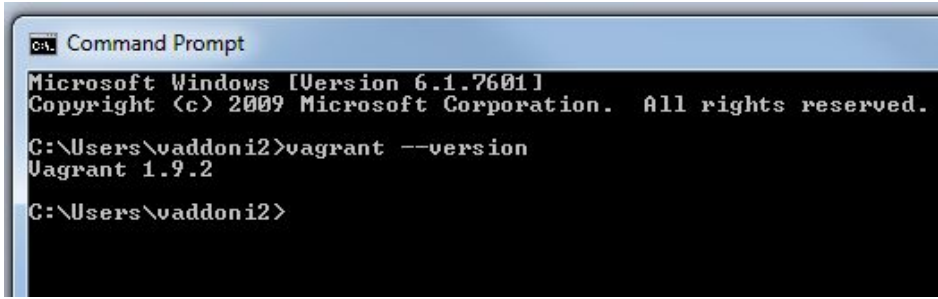
Technical Pitstop - Vagrant

Mac

PC

`vagrant --version`

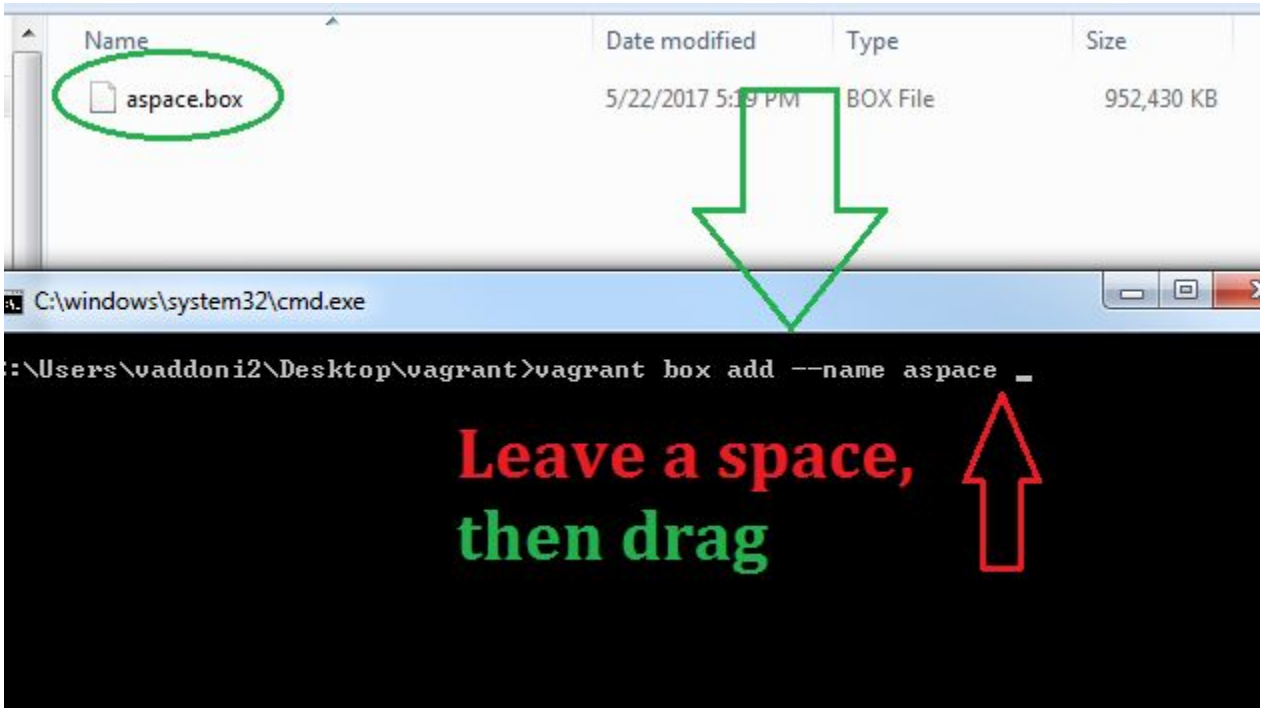




If you see any number that doesn't start with "1.9" or if you get an error altogether please use your post-its, or shout out!

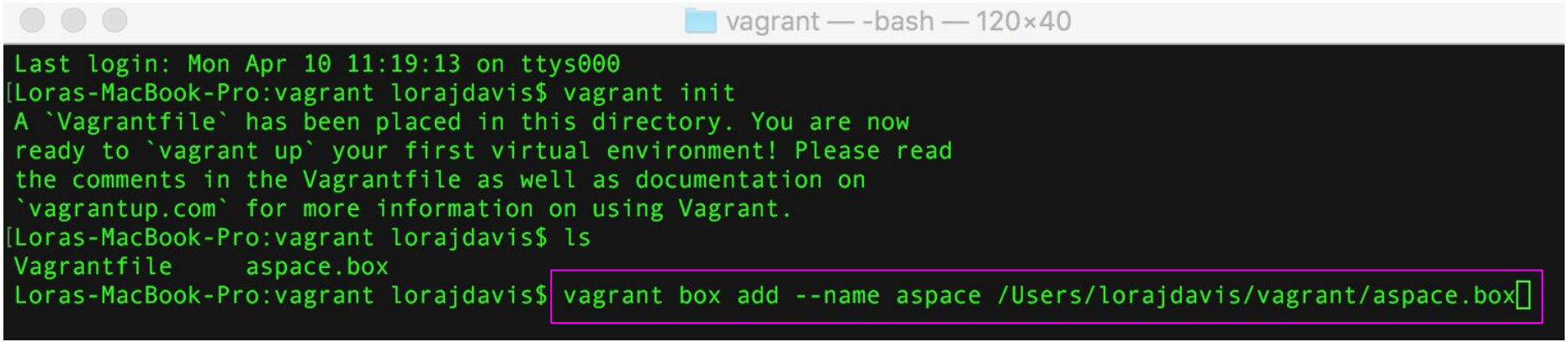
Everyone

Type `vagrant box add --name aspace` and leave a space, then drag:
(see next slide to confirm before hitting enter)



Everyone

It should look like this now. If it does, hit Enter:

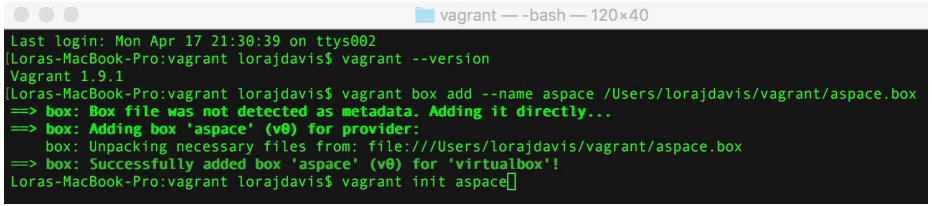


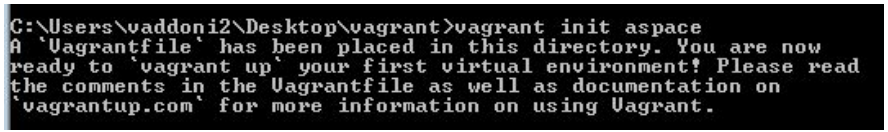
Technical Pitstop - Initialize Vagrant

Mac

PC

```
vagrant init aspace
```



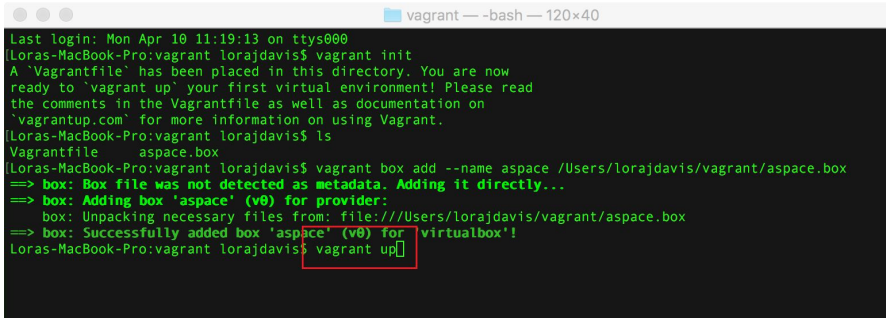


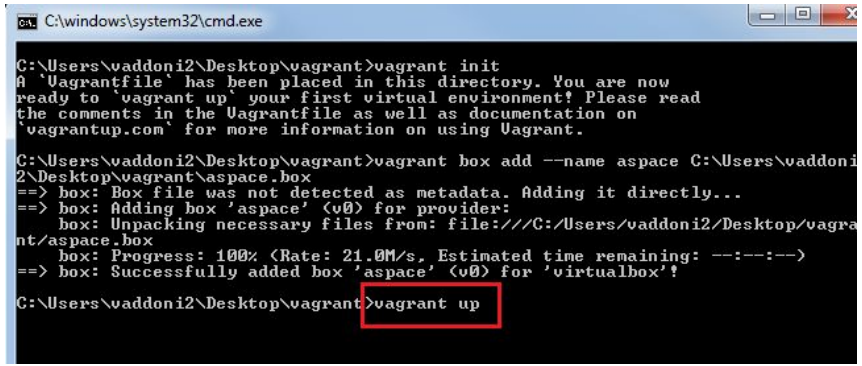
Technical Pitstop - Vagrant up

Mac

PC

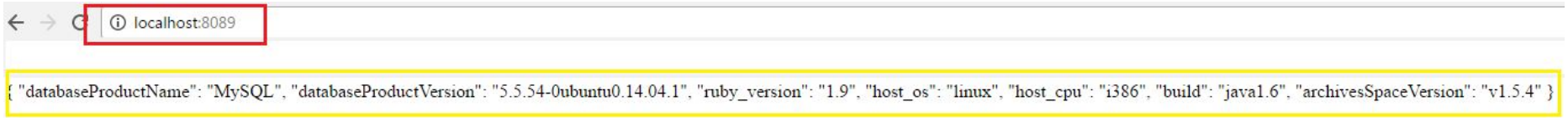
`vagrant up`





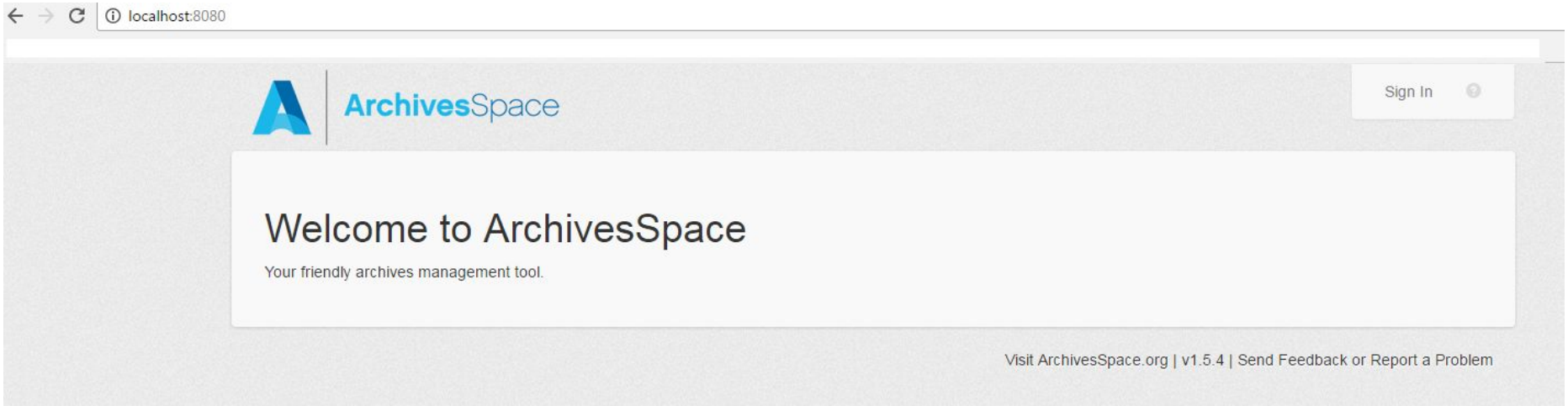
Technical Pitstop - Check Vagrant backend

Open browser, navigate to:
http://localhost:**8089**



Technical Pitstop - Check Vagrant staff interface

Try: http://localhost:**8080**



Now you have a personal ArchivesSpace! It's yours and it's re-useable! Look around.

Username: admin
Password: admin

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 "container modeling" for the first time in the archives world.

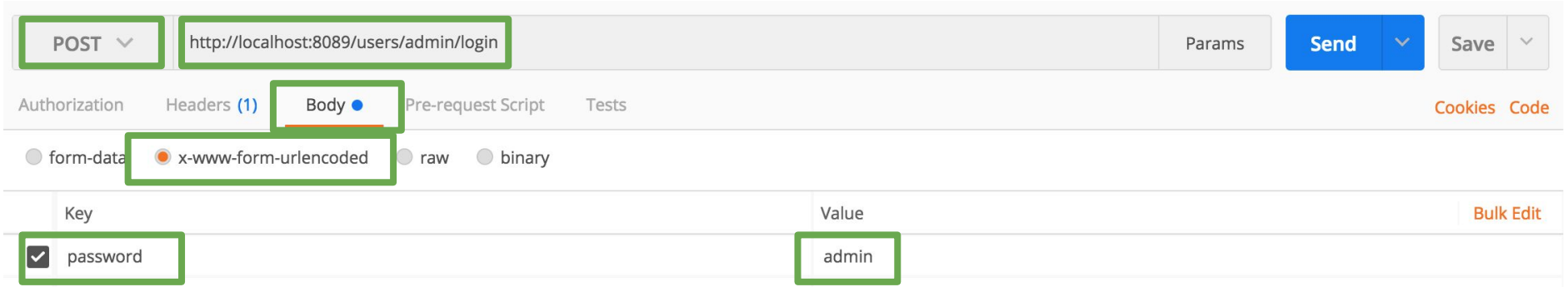
Every type of box (ex. record carton) in your library gets its own record (a profile), which records its height, width, and depth. This helps calculate space on a huge scale, and is a game-changer for some repositories.

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

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

POST to AS with GUI

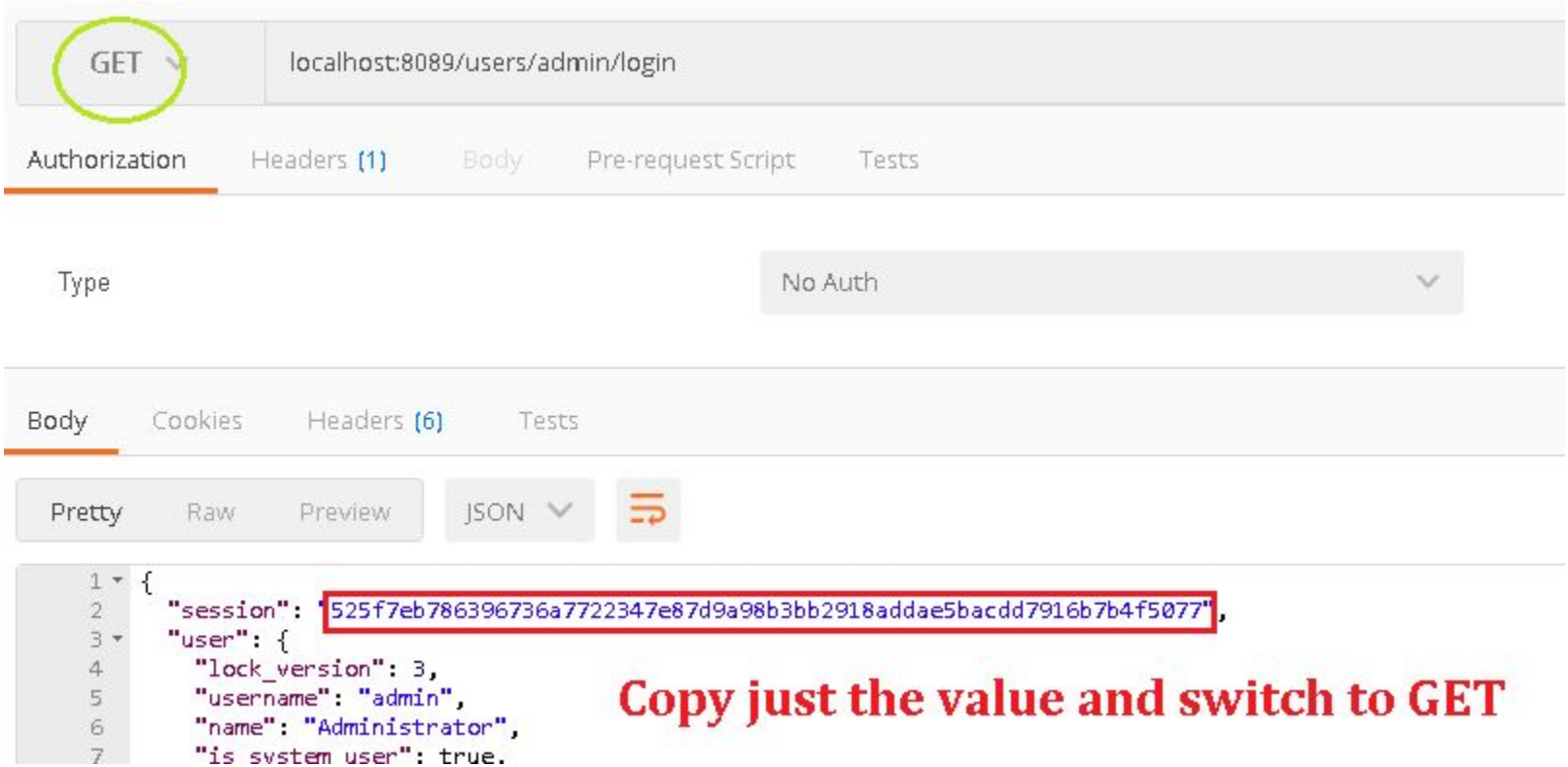
Before we start posting to AS, we need to authenticate, so let's do that and try a GET first:



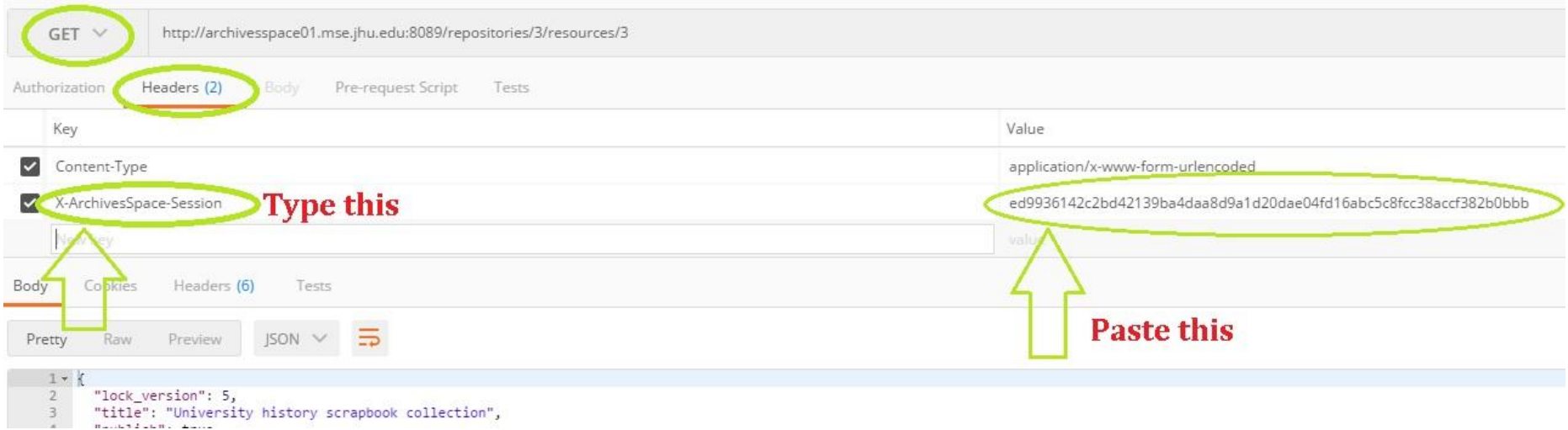
Endpoint: http://localhost:8089/users/admin/login

password: admin

POST to AS with GUI



POST to AS with GUI



Key: X-ArchivesSpace-Session

Endpoint: http://localhost:8089/repositories/2/resources/1

POST to AS with GUI - Container profiles

1. Navigate to the directory with our cloned GitHub repo

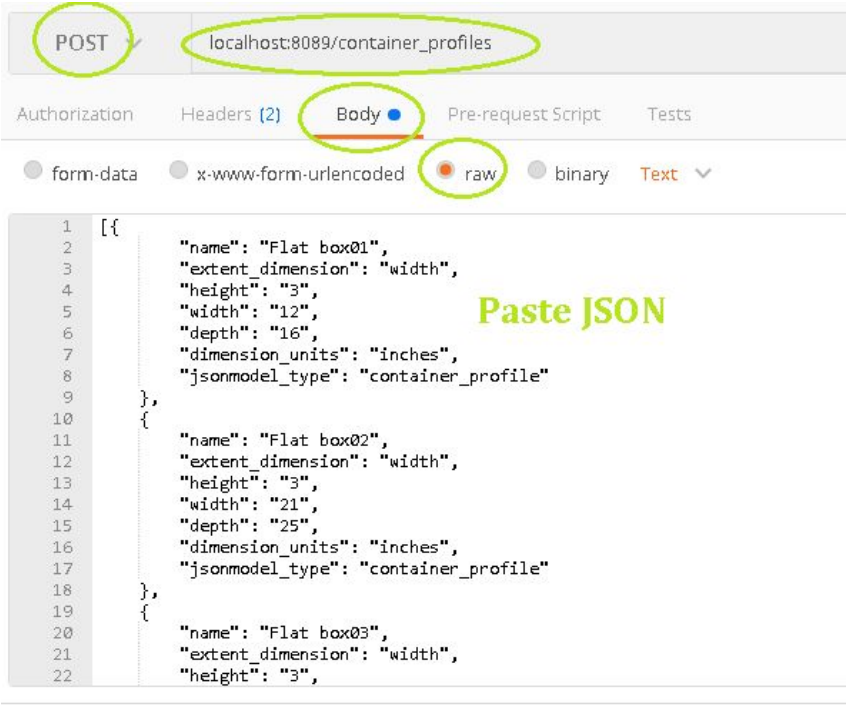
   Mac users: Desktop

   Windows users:

   C:\cygwin64\home\[username]\MARAC_API_Workshop
2. Open “containerProfiles.json” with Atom
3. Here are the profiles, ready to go
4. Copy, and go back to Postman

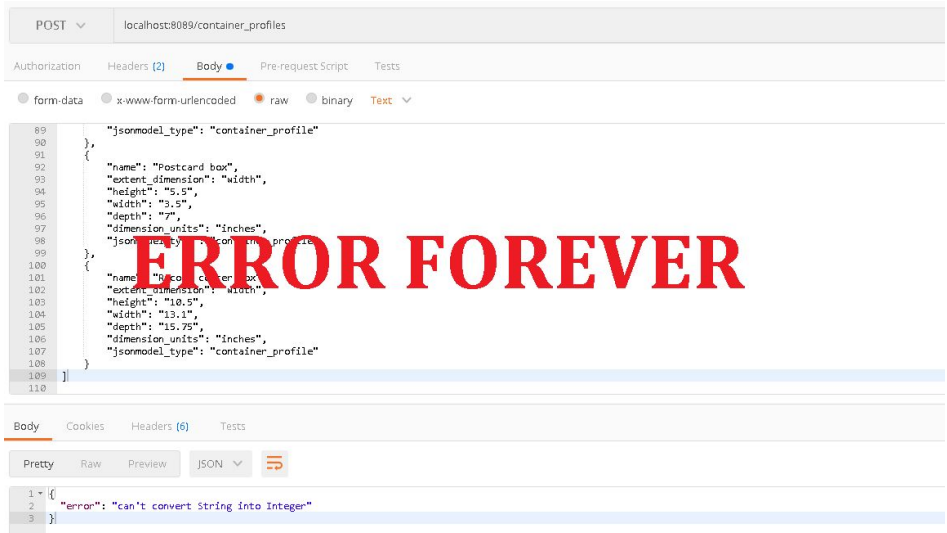

```
[{
        "name": "Flat box01",
        "extent_dimension": "width",
        "height": "3",
        "width": "12",
        "depth": "16",
        "dimension_units": "inches",
        "jsonmodel_type": "container_profile"
    },
    {
        "name": "Flat box02",
        "extent_dimension": "width",
        "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: http://localhost:8089/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 will save you a month

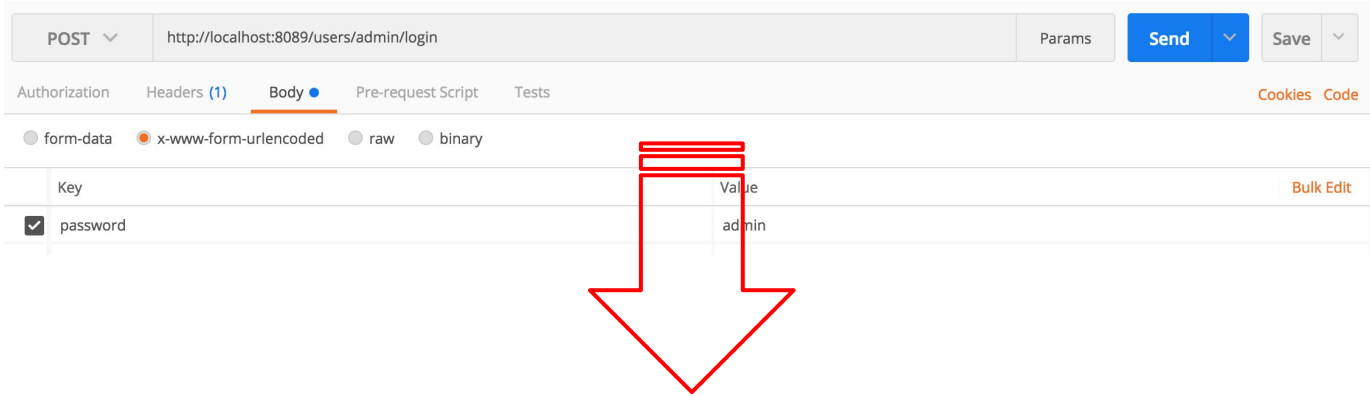
(use your month wisely - take a vacation from ArchivesSpace)


Just
breathe

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

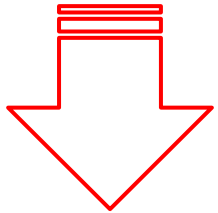
```
1  baseURL='http://localhost:8089'
2  user='admin'
3  password='admin'
```

POST to AS with script

“Keep it secret, keep it safe.” - Gandalf

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- Container Profiles

Mac

1. In the Finder navigate to your MARAC_API_Workshop directory
2. Ctrl+click the MARAC_API_Workshop directory, and select “New Terminal at Folder”
3. Type `ls` and examine the contents of that folder
4. Type `python postContainerProfiles.py` (case sensitive!)
5. Navigate back to AS in your browser (http://localhost:8080)

PC

1. Open Cygwin
2. Type `cd MARAC_API_Workshop` to enter the MARAC_API_Workshop directory
3. Type `ls` and examine the contents of that folder
4. Type `python postContainerProfiles.py` (case sensitive!)
5. Navigate back to AS in your browser (http://localhost:8080)

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**

POST to AS with script- Edit barcodes

Barcodes/top_containers

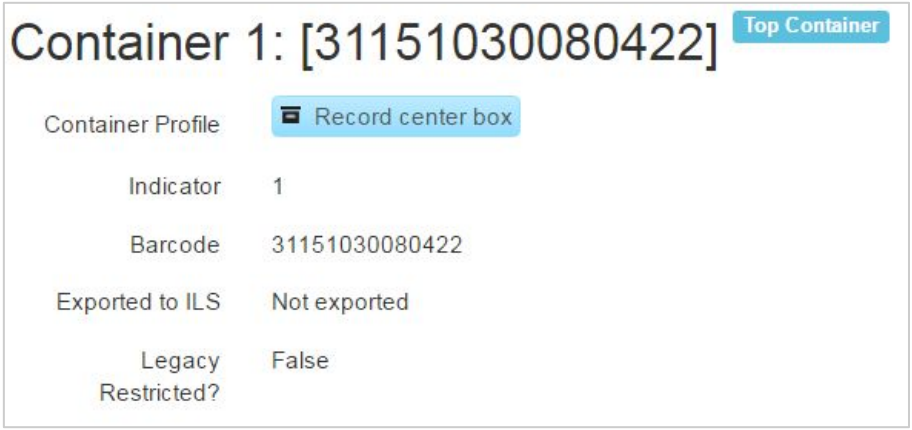
Répète: ASpace offers “container modeling” for the first time in the archives world.

Every type of box (ex. record carton) in your library gets its own record (a profile), which records its height, width, and depth.

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 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 back to ASpace in your browser (http://localhost:8080)
2. Browse > Resources > Gérard Defaux papers > View > expand Research Materials > click on any file > scroll down to Instances > see fake barcode
3. These are the barcodes generated by the barcoder plugin. Hopkins has thousands of them.
4. 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 MARAC_API_Workshop directory
2. Ctrl+click the MARAC_API_Workshop directory, and select “New Terminal at Folder”
3. Type `ls` and examine the contents of that folder
4. Type `python postBarcodes.py` (case sensitive!)
5. Navigate back to AS in your browser (http://localhost:8080)

PC

1. Open Cygwin
2. Type `cd MARAC_API_Workshop` to enter the MARAC_API_Workshop directory
3. Type `ls` and examine the contents of that folder
4. Type `python postBarcodes.py` (case sensitive!)
5. Navigate back to AS in your browser (http://localhost:8080)

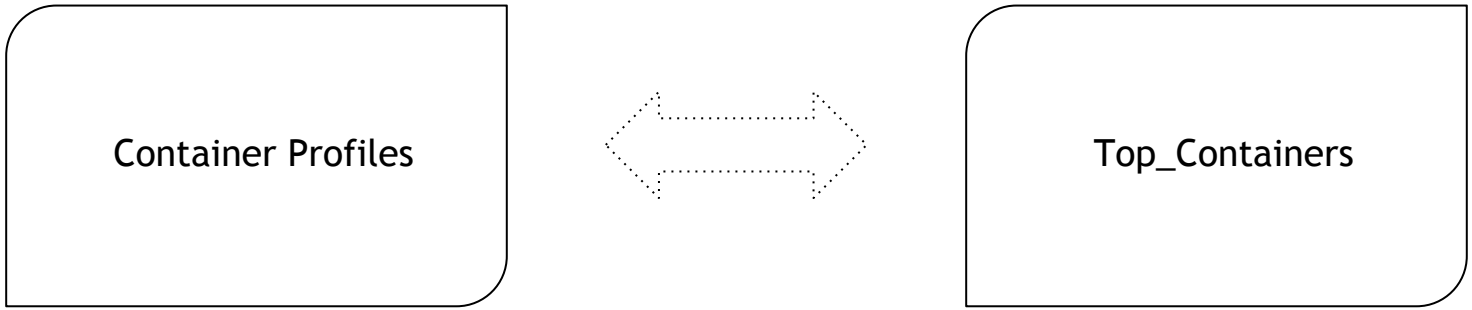
ArchivesSpace!

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**

POST to AS with script-Link profiles

Linking profiles to containers



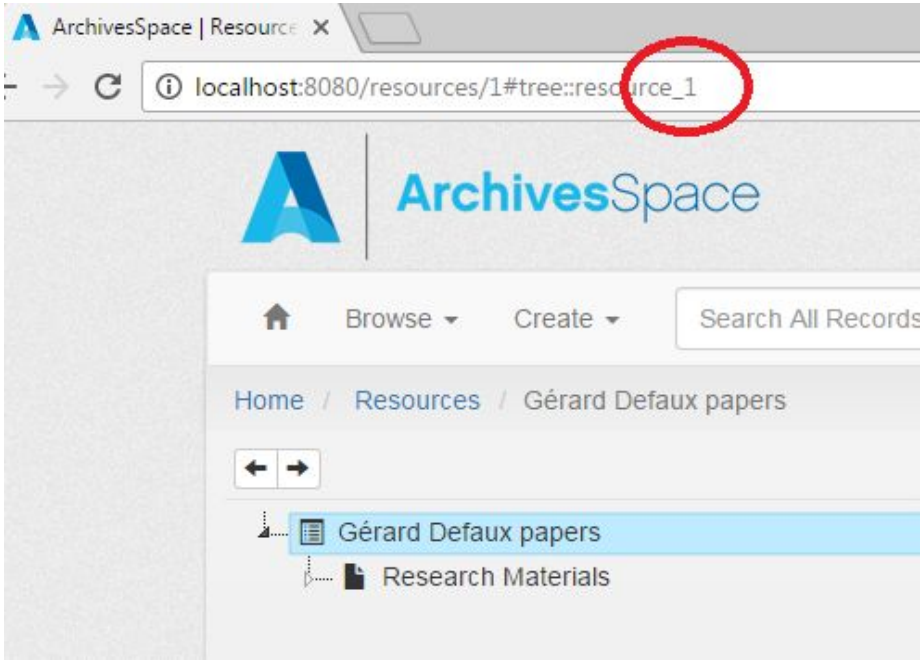
POST to AS with script - Linking profiles

1. Type `ls` and examine the contents of that folder
2. Type `python asLinkProfiles.py` (case sensitive!)
3. You will be prompted for a resource id and a container id… how do you determine what you need to know?
4. Let’s return to ASpace real quick

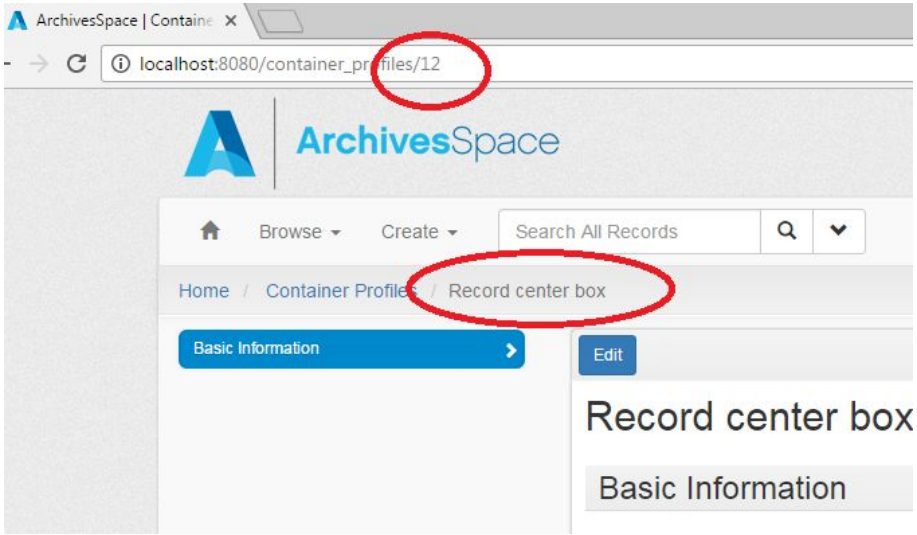
POST to AS with script - Linking profiles

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:



POST to AS with script - Linking profiles

1. Type `ls` and examine the contents of that folder
2. Type `python asLinkProfiles.py` (case sensitive!)
3. You will be prompted for a resource id and a container id… how do you determine what you need to know?
4. Let’s return to ASpace real quick

~~ArchivesSpace! You did it!~~

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:

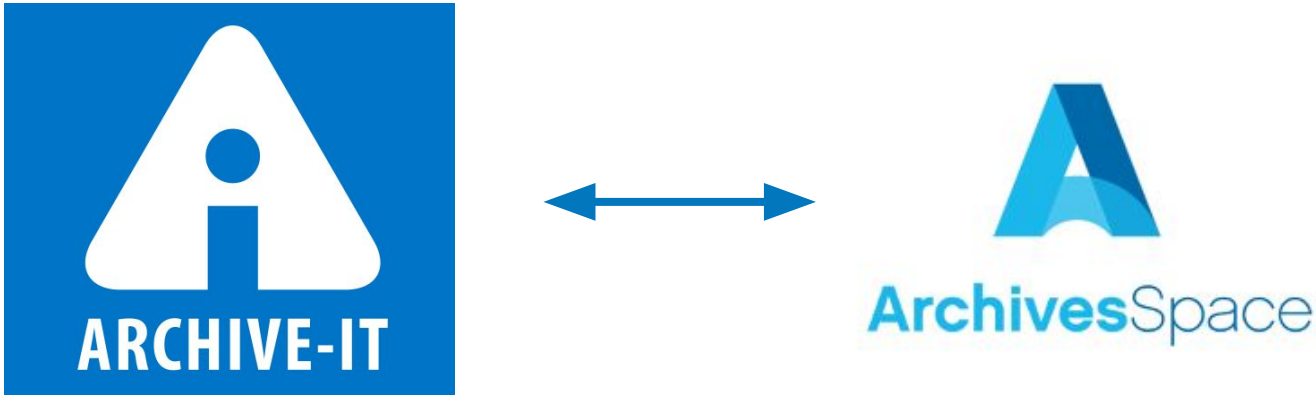
1. ~~We don't have **container profiles** in ArchivesSpace and would like to, so we need to create some~~
2. ~~In following the migration instructions for 1.5, we had to add **faux codes**; we'd like to use our *actual* barcodes~~
3. ~~Now that we have container profiles, we need to link them to *actual*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

15 minute break!

GET and POST across two applications with Python

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 Johns Hopkins University Library” resource

2. Expand Subgroup 12: 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?

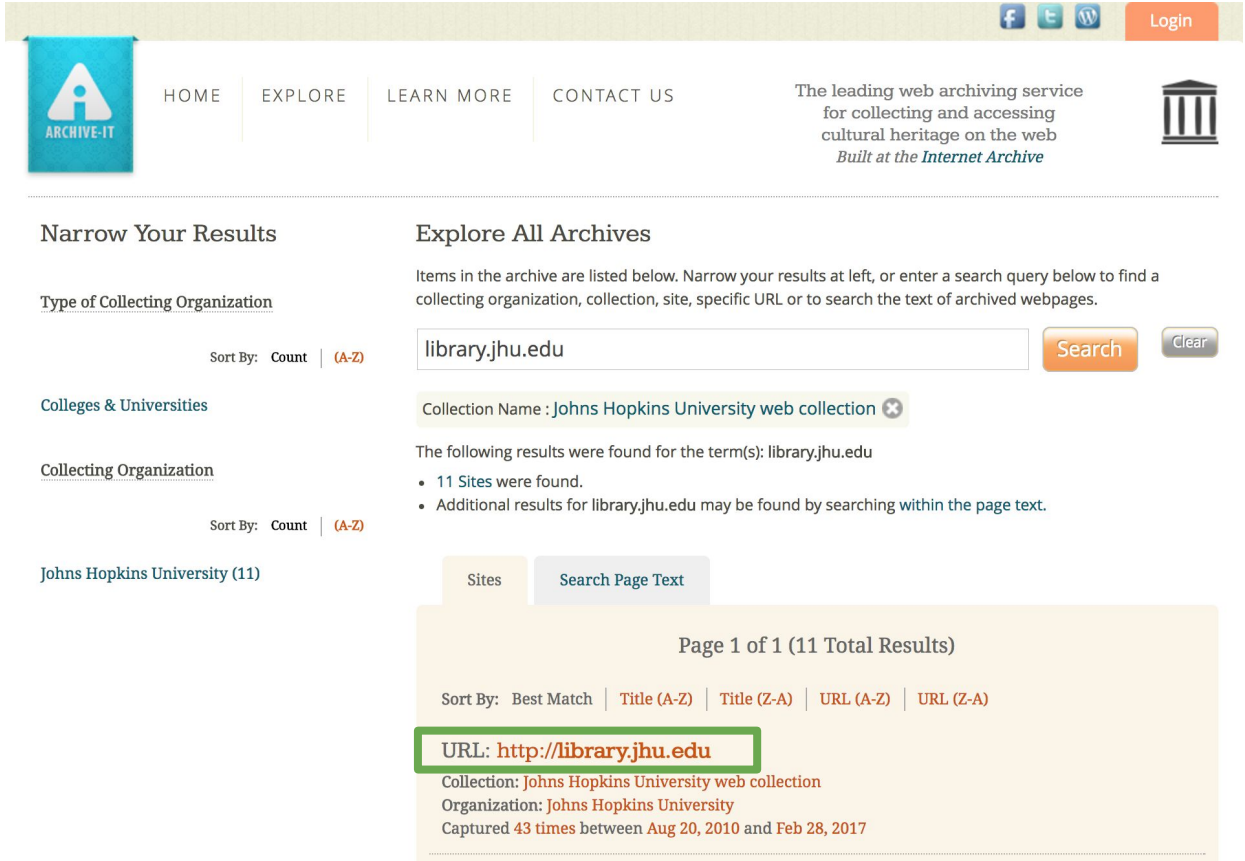
In fact, with a decent enough search we could probably even have a script return JUST those archival objects with the level **"Web archive."**

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!

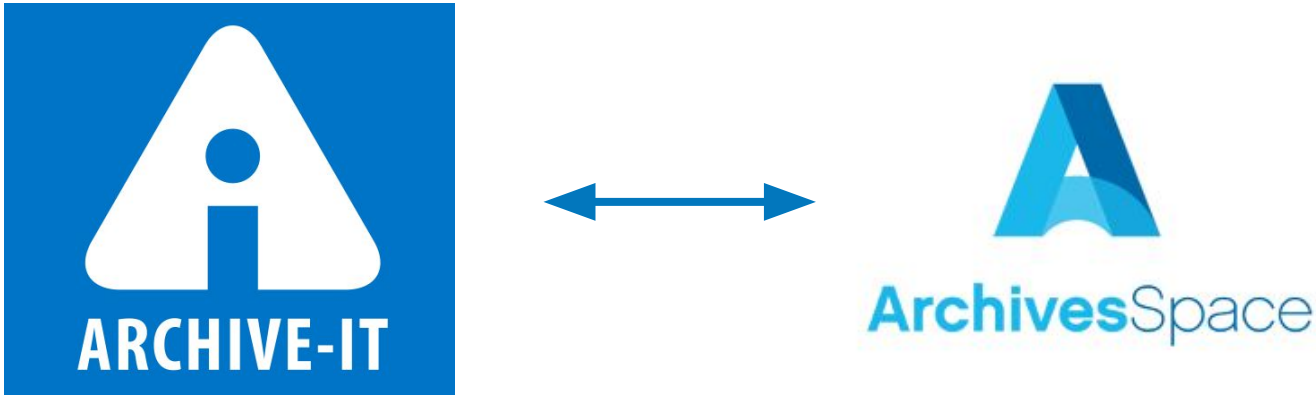
```
67        for crawl in crawlList:
68            doid = 'https://wayback.archive-it.org' + '/' + archiveit_coll + '/' +
          •     crawl['timestamp'] + '/' + crawl['original']
69            query = '/search?page=1&filter_term[]={"primary_type":"digital_object"}&q=' + doid
70            existingdoID = 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 Johns Hopkins University Library" resource record once again.



User stories

Can the API create reports?

- Yes… but an API is the wrong tool for reporting
- Like using a jack-hammer indoors: yes it will work, but it will be more effort with the wrong tool
- For AS users in particular:
  - Wait for reporting to improve, because it will
  - But in the meantime (or for more customized reporting) we suggest connecting AS to MS Access. If you don't know Access, it's easier to learn that than learning to script just for reports.
    - You can find explanatory slides in the workshop GitHub > additional resources
    - Credit to Nancy Enneking, Head of Institutional Records at the Getty and Celia Caust-Ellenbogen, Friends Historical Library of Swarthmore College for the method

Can APIs change the staff/user interface?

- No: the API is only a way of manipulating data
- Look at the API endpoints and see if one relates to the change you want to make
- We did write something that changed enumerations

Can APIs improve my agent records?

- Yes!
- Hopkins made changes to **almost all** of our Agent and Subject records through the API
- We added VIAF **ids** to Agents and converted LCSH to FAST
- You already have improved Corporate Names from earlier
- Let’s put those to work!

POST with script - VIAF

Post from a CSV - VIAF

	A	B
1	**url**	**date_issued**
2	http://chroniclingamerica.loc.gov/lccn/sn85038615/1903-01-27/ed-1.json	1903-01-27
3	http://chroniclingamerica.loc.gov/lccn/sn85038615/1903-01-28/ed-1.json	1903-01-28
4	http://chroniclingamerica.loc.gov/lccn/sn85038615/1903-01-29/ed-1.json	1903-01-29
5	http://chroniclingamerica.loc.gov/lccn/sn85038615/1903-01-30/ed-1.json	1903-01-30
6	http://chroniclingamerica.loc.gov/lccn/sn85038615/1903-01-31/ed-1.json	1903-01-31
7	http://chroniclingamerica.loc.gov/lccn/sn85038615/1903-02-01/ed-1.json	1903-02-01
8	http://chroniclingamerica.loc.gov/lccn/sn85038615/1903-02-03/ed-1.json	1903-02-03
9	http://chroniclingamerica.loc.gov/lccn/sn85038615/1903-02-04/ed-1.json	1903-02-04
10	http://chroniclingamerica.loc.gov/lccn/sn85038615/1903-02-05/ed-1.json	1903-02-05
11	http://chroniclingamerica.loc.gov/lccn/sn85038615/1903-02-06/ed-1.json	1903-02-06
12	http://chroniclingamerica.loc.gov/lccn/sn85038615/1903-02-07/ed-1.json	1903-02-07
13	http://chroniclingamerica.loc.gov/lccn/sn85038615/1903-02-08/ed-1.json	1903-02-08
14	http://chroniclingamerica.loc.gov/lccn/sn85038615/1903-02-10/ed-1.json	1903-02-10
15	http://chroniclingamerica.loc.gov/lccn/sn85038615/1903-02-11/ed-1.json	1903-02-11
16	http://chroniclingamerica.loc.gov/lccn/sn85038615/1903-02-12/ed-1.json	1903-02-12
17	http://chroniclingamerica.loc.gov/lccn/sn85038615/1903-02-13/ed-1.json	1903-02-13
18	http://chroniclingamerica.loc.gov/lccn/sn85038615/1903-02-14/ed-1.json	1903-02-14
19	http://chroniclingamerica.loc.gov/lccn/sn85038615/1903-02-15/ed-1.json	1903-02-15
20	http://chroniclingamerica.loc.gov/lccn/sn85038615/1903-02-17/ed-1.json	1903-02-17
21	http://chroniclingamerica.loc.gov/lccn/sn85038615/1903-02-18/ed-1.json	1903-02-18
22	http://chroniclingamerica.loc.gov/lccn/sn85038615/1903-02-19/ed-1.json	1903-02-19
23	http://chroniclingamerica.loc.gov/lccn/sn85038615/1903-02-20/ed-1.json	1903-02-20
24	http://chroniclingamerica.loc.gov/lccn/sn85038615/1903-02-21/ed-1.json	1903-02-21
25	http://chroniclingamerica.loc.gov/lccn/sn85038615/1903-02-22/ed-1.json	1903-02-22
26	http://chroniclingamerica.loc.gov/lccn/sn85038615/1903-02-24/ed-1.json	1903-02-24
27	http://chroniclingamerica.loc.gov/lccn/sn85038615/1903-02-25/ed-1.json	1903-02-25
28	http://chroniclingamerica.loc.gov/lccn/sn85038615/1903-02-26/ed-1.json	1903-02-26
29	http://chroniclingamerica.loc.gov/lccn/sn85038615/1903-02-27/ed-1.json	1903-02-27
30	http://chroniclingamerica.loc.gov/lccn/sn85038615/1903-02-28/ed-1.json	1903-02-28

Way back during the Chronicling America exercise, we showed you how to convert JSON to CSV using a free online converter...

Post from a CSV - VIAF

result.csv

	total_results	organizations__ein	organizations__strein	organizations__name	organizations__sub_name	organizations__city	organizations__state	organizations__ntee_code	organizations__raw_ntee_code	organizations_
001	2435	731663130	73-1663130	ANIMAL ALLIANCE	ANIMAL ALLIANCE	WOODLAND HLS	CA	D20	D20	3
002		752461428	75-2461428	ANIMAL ANGELS	ANIMAL ANGELS	JACKSBORO	TX	null	null	3
003		232896507	23-2896507	ANIMAL APPEAL	ANIMAL APPEAL	SHARPSVILLE	PA	O21	O21	3
004		43654364	04-3654364	ANIMAL ASSISTANCE	ANIMAL ASSISTANCE	E BRUNSWICK	NJ	D20	D20	3
005		800099106	80-0099106	ANIMAL FAIR	ANIMAL FAIR	SEDALIA	MO	D20	D20	3
006		752391827	75-2391827	ANIMAL HAVEN	ANIMAL HAVEN	DALLAS	TX	null	null	3
007		953793138	95-3793138	ANIMAL HELPINE	ANIMAL HELPINE	MORONGO VLY	CA	D200	D200	3
008		570921521	57-0921521	ANIMAL MISSION	ANIMAL MISSION	COLUMBIA	SC	D200	D200	3
009		462455876	46-2455876	ANIMAL NETWORK	ANIMAL NETWORK	LAS VEGAS	NV	D20	D20	3
010		10462882	01-0462882	ANIMAL ORPHANAGE	ANIMAL ORPHANAGE	ORONO	ME	D200	D200	3
011		454902841	45-4902841	ANIMAL PAD	ANIMAL PAD	SAN DIEGO	CA	D20	D20	3
012		810679304	81-0679304	ANIMAL ADVOCATES	ANIMAL ADVOCATES	BARNWELL	SC	D11	D11	3
013		20469900	02-0469900	ANIMAL ALLIES	ANIMAL ALLIES	MANCHESTER	NH	D20	D20	3
014		680630714	68-0630714	ANIMAL BALANCE	ANIMAL BALANCE	EL CERRITO	CA	Z99	Z99	3
015		953967883	95-3967883	ANIMAL COMPANIONS	ANIMAL COMPANIONS	PIPE CREEK	TX	null	null	3
016		237237862	23-7237862	ANIMAL FUND	ANIMAL FUND	GREENBRAE	CA	D013	D013	3
017		204444813	20-4444813	ANIMAL LIFELINE	ANIMAL LIFELINE	WARRINGTON	PA	D20	D20	3
018		680200668	68-0200668	ANIMAL PLACE	ANIMAL PLACE	GRASS VALLEY	CA	D200	D200	3
019		800610723	80-0610723	ANIMAL UNION	ANIMAL UNION	DALY CITY	CA	P80	P80	3
020		260796104	26-0796104	ANIMAL ADVOCATES	ANIMAL ADVOCATES	WESTCLIFFE	CO	D20	D20	3
021		223451569	22-3451569	ANIMAL ALLIES	ANIMAL ALLIES	EWING	NJ	D200	D200	3
022		571098821	57-1098821	ANIMAL ALLIES	ANIMAL ALLIES	SPARTANBURG	SC	D200	D200	3
023		208502915	20-8502915	ANIMAL ANGELS	ANIMAL ANGELS	MEDIAPOLIS	IA	D20	D20	3
024		262933425	26-2933425	ANIMAL DIPLOMACY	ANIMAL DIPLOMACY	CAMBRIDGE	MA	D20	D20	3
025		10714781	01-0714781	ANIMAL TRACKS	ANIMAL TRACKS	ACTON	CA	D34	D34	3
026		454027119	45-4027119	ANIMAL AID	ANIMAL AID	ATHENS	TN	D40	D40	3
027		272034873	27-2034873	ANIMAL WATCH	ANIMAL WATCH	BOULDER	CO	D30	D30	3
028		481287089	48-1287089	ANIMAL ADVOCATES	ANIMAL ADVOCATES	LOS ANGELES	CA	D20	D20	3
029		800530102	80-0530102	ANIMAL FOLKS	ANIMAL FOLKS	SAINT PAUL	MN	D20	D20	3
030		371768388	37-1768388	ANIMAL FOUNDATION	ANIMAL FOUNDATION	HERMANN	MO	P29	P29	3
031		203596003	20-3596003	ANIMAL MAGIC	ANIMAL MAGIC	BELLEVILLE	MI	D20	D20	3
032		953171867	95-3171867	ANIMAL SAMARITANS	ANIMAL SAMARITANS	THOUSAND PLMS	CA	D200	D200	3


```
- "organizations": [Array[100]
  - 0: {
      "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": 443.46564
    },
  - 1: {
      "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": 443.46564
    },
  - 2: {
      "ein": 232896507,
      "strein": "23-2896507",
      "name": "ANIMAL APPEAL",
      "sub_name": "ANIMAL APPEAL",
      "city": "SHARPSVILLE",
      "state": "PA",
      "ntee_code": "O21",
      "raw_ntee_code": "O21",
      "subseccd": 3,
      "has_subseccd": true,
      "have_filings": true,
      "have_extracts": true,
      "have_pdfs": true,
      "score": 443.46564
    },
```


...well, we can also convert a csv (like the file that resulted from running `python viafReconciliationCorporate.py`) to JSON!
And ArchivesSpace likes eating JSON!

Post from a CSV - VIAF

Before posting these VIAF corporate entities into ArchivesSpace, we must first add VIAF as a valid **“name source”** in ArchivesSpace

1. Go to the ArchivesSpace staff interface at: http://localhost:8080
2. In the top right select “System > Manage Controlled Value Lists”
3. In the drop-down you’re provided, select “Name Source (name_source)”
4. In the middle right click “Create Value”
5. Name this value “viaf”
    a. Note: The punctuation here is important since it must match what is in our Python script. All lowercase, no spaces.
6. Click “Create Value”

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 MARAC_API_Workshop directory
2. From within the MARAC_API_Workshop directory in Terminal/Cygwin execute `python postVIAFOrganizations.py`
3. Go back to the ArchivesSpace staff interface at: http://localhost:8080
4. In the top left click “Browse > Agents”
5. Voila!

Icing and Advice

Icing: Interpreting (ASpace) API endpoints

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

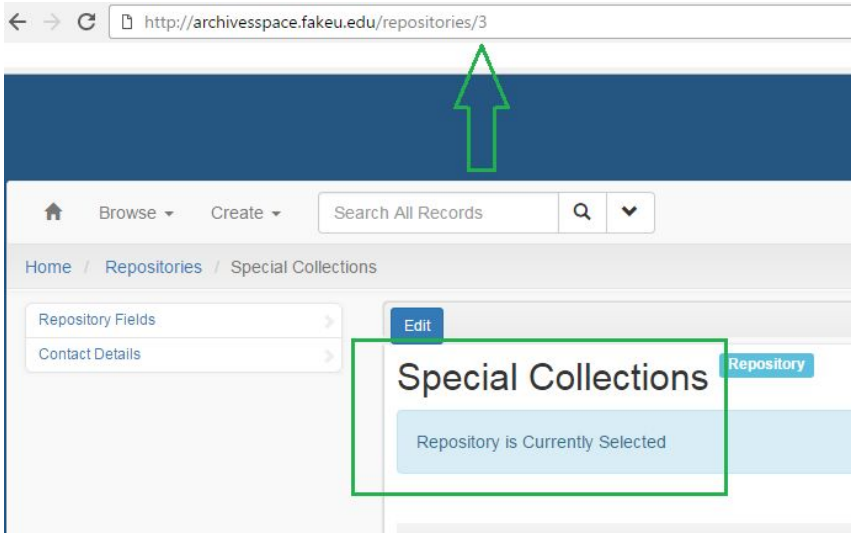
200 – (:resource)

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

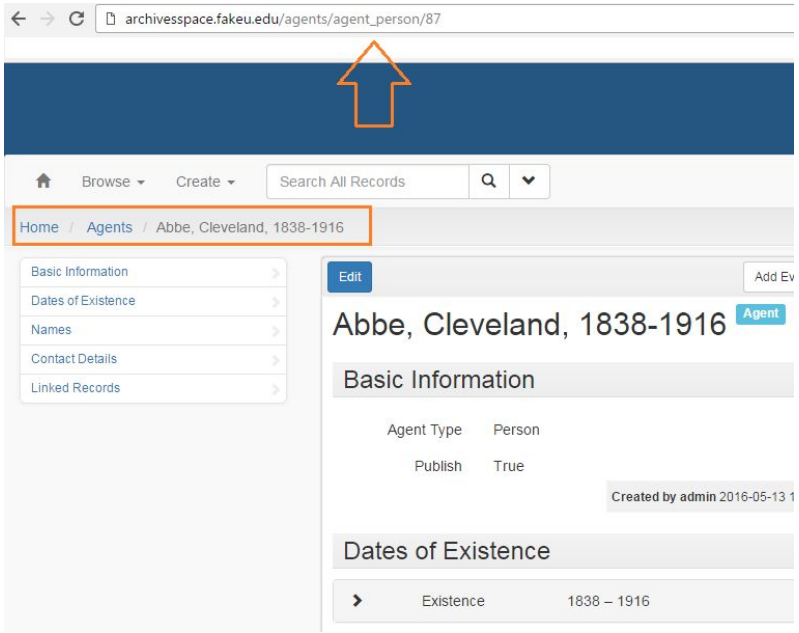
Icing: Interpreting (ASpace) API endpoints

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: http://localhost:8089/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: What IS GitHub anyway?

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

If you're serious about learning to script, you should watch the 10 million GitHub intro videos on YouTube

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

Let's go look at our repo together, we made it for **you**!

https://github.com/jhu-archives-and-manuscripts/MARAC_API_Workshop

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

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; we’ve provided instructions in the take home document (which really will happen, we promise)

Icing: What about XYZ application?

In your pre-workshop surveys, many of you (understandably) noted applications we didn't get a chance to cover today, notably:

- **ILSes** like Voyager;
- **Digital exhibition** applications like Omeka;
- **Digital collections/repository** applications like ContentDM or Islandora; and,
- **Cloud-based sharing applications** like Dropbox/Drive/Box.

While we didn't work through exercises with these applications today, hopefully you now know the **steps to take** to do future API work of your own, namely:

- **Research the API**, including authentication requirements and endpoint documentation;
- Play around in **Postman** (or another GUI API application);
- Determine whether your desired tasks can be accomplished through the **GUI**, or if you need a **scripting language**;
    - If the latter, determine if **someone else has already tackled** your task (for example, on GitHub)
- **Iteratively test** (in a *non-Production environment*!)
- **Profit!**

Wrap up!

Shutting down your Vagrant box - super important!

Do the following in Terminal/the command prompt:

1. Type `vagrant destroy` and confirm the command by typing `y` when prompted
   a. "Destroying" is equivalent to hitting "shut down" on your computer; the computer is still sitting on your desk with your work saved, but it is powered off.
2. Type `vagrant box remove aspace` to remove the box completely
   a. If "destroying" a box is like shutting down, "removing" a box is like taking a time machine back to the day you first cut the shrink wrap on your brand new computer. All work you have done in the virtual environment will be lost, but, once the box is re-added and upped again, you will be back to the exact version we shared with you here.

Thanks!

The Vagrant developed for this workshop was adapted by Lora Davis from **Dallas Pillen**'s *Archivagrant*. See his blog post for more information: http://archival-integration.blogspot.com/2016/01/archivesspace-vagrant-archivagrant.html

Additional Vagrant advice was offered by **Francis Kayiwa**.

The inspiration (if not the flawed code) for the ArchiveIt-ArchivesSpace exercise comes from the fantastic work of **Greg Wiedeman**. Check it out on GitHub: https://github.com/UAlbanyArchives/ASpace_WebArchives

Thanks to **Eric Hanson**, Metadata Librarian at JHU, for his troubleshooting and all-around good spirits (YAY ERIC!)

Several of these concepts and/or exercises were tested on JHU Sheridan Libraries' staff, including NDSR Resident **Elizabeth England**.

Anyone who has ever shared help/advice/support on blogs/listservs/bar stools who are too numerous to fully name (but we'll try in person if you ask us to!).