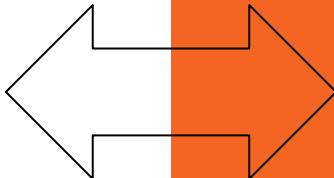


Mac users

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

Center for Jewish History
September 15, 2017

Lora Woodford
Valerie Addonizio

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?

We hope you'll leave a
2.

Are you a 2?

We hope you'll leave a
3.

Are you a 3?

We hope to give you
new ideas, scripts, and
momentum.



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](#)

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 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.

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!*

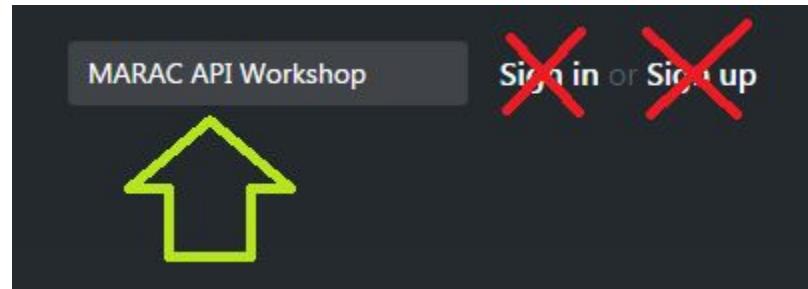
*needle coming off
the record*

This is the tough
part.

Questions?

Resource Pitstop - Get these slides

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

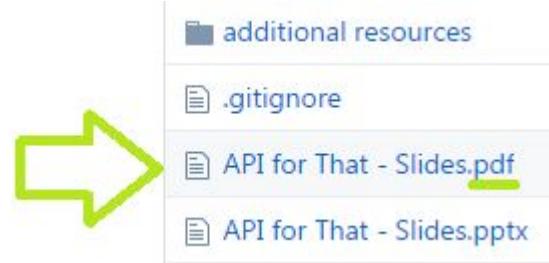


3. Select our “repo”

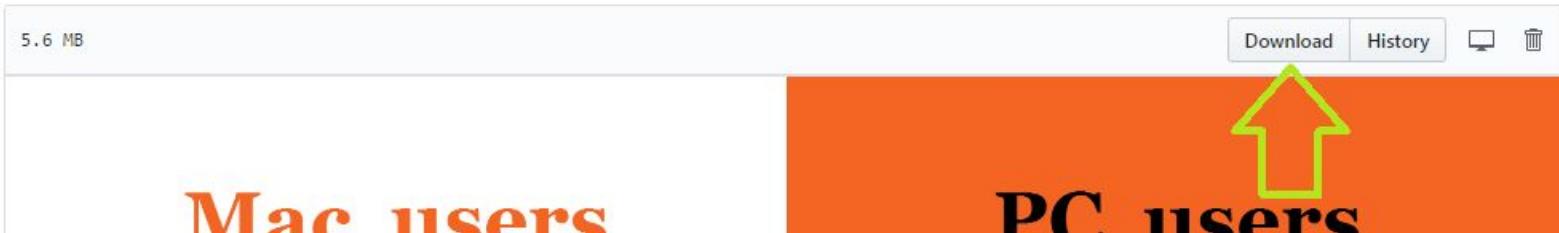


Resource Pitstop - Get these slides

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.



You can 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!

Understanding and Setting up your tech

(We promise you're in the right workshop)

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:

- Get some important packages **installed** on your machines
- **Clone our GitHub repository** (download some scripts to your computer)

Caveats:

- We **may 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:

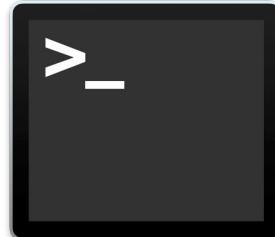
- We've got many different environments to troubleshoot and many different opportunities to fail - **please be patient!**
- If, at any point for the remainder of the workshop, you need **assistance**, please place a **red/hot pink post-it** on the back of your laptop screen

You should have done the
following already:

Technical Pitstop - Installing packages

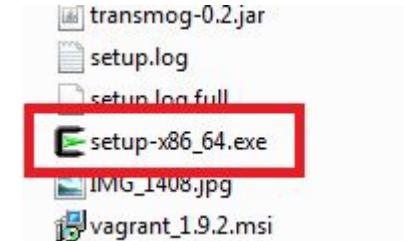
Mac

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. Bring up the Cygwin installer that you downloaded in the pre-workshop instructions



Technical Pitstop - Installing packages

Mac

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

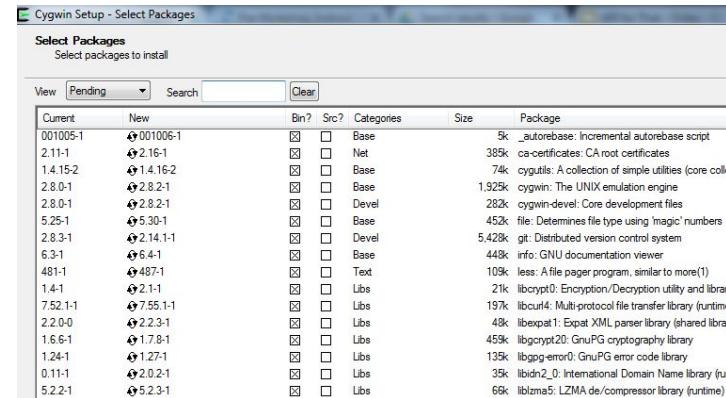
If you are prompted to install, hit Install!
Once complete, type `gcc --version` again.

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 installing. Say yes to everything
2. Pick any site to install from and continue
3. Stop when you get to the Select Packages screen:



Technical Pitstop - Installing packages

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

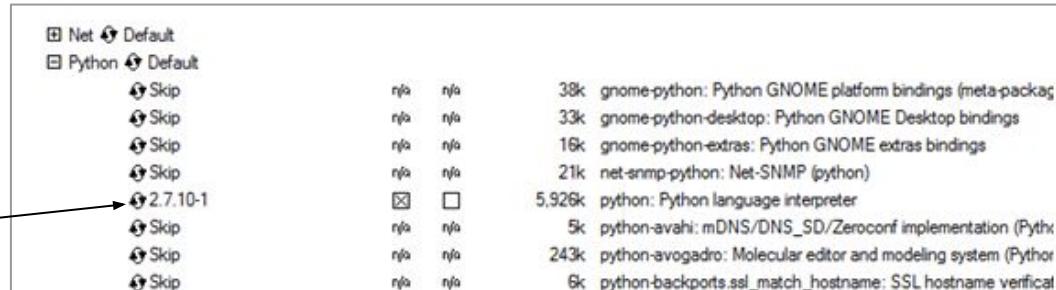
1. Look for View: at the top of the screen and select Category
2. Search for “python2” (without quotes) but do not hit enter; the search happens as soon as you type



Windows users only:

1. After you have searched for python2 locate and *unskip the following (the list is strictly alphabetical)*:
 - a. python2: Python 2 language interpreter
 - b. python2-requests: Python HTTP/1.1 request module
2. Now search for git (but don't hit enter), expand the Devel heading, unskip:
 - a. git: Distributed version control system
3. Finally, search for openssh (don't hit enter), expand the Net heading, unskip:
 - a. openssh: The OpenSSH server and client programs
4. Once all four have been unskipped, proceed with install: Next > Next

This screenshot shows what an “unskipped” line looks like



Net <input checked="" type="checkbox"/> Default			
Python <input checked="" type="checkbox"/> Default			
<input checked="" type="checkbox"/> Skip	n/a	n/a	38k gnome-python: Python GNOME platform bindings (meta-package)
<input checked="" type="checkbox"/> Skip	n/a	n/a	33k gnome-python-desktop: Python GNOME Desktop bindings
<input checked="" type="checkbox"/> Skip	n/a	n/a	16k gnome-python-extras: Python GNOME extras bindings
<input checked="" type="checkbox"/> Skip	n/a	n/a	21k net-snmp-python: Net-SNMP (python)
<input checked="" type="checkbox"/> 2.7.10-1	<input checked="" type="checkbox"/>	<input type="checkbox"/>	5,926k python: Python language interpreter
<input checked="" type="checkbox"/> Skip	n/a	n/a	5k python-avahi: mDNS/DNS_SD/Zeroconf implementation (Python)
<input checked="" type="checkbox"/> Skip	n/a	n/a	243k python-avogadro: Molecular editor and modeling system (Python)
<input checked="" type="checkbox"/> Skip	n/a	n/a	6k python-backports.ssl_match_hostname: SSL hostname verification

This is new:

Technical Pitstop - Installing packages

Mac

1. Open Terminal
2. Type `brew install python` and hit enter
2. Type `python --version` and hit enter

```
Loras-MacBook-Pro:~ lorajdavis$ python --version
Python 2.7.10
Loras-MacBook-Pro:~ lorajdavis$
```

3. Type `pip install requests`

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

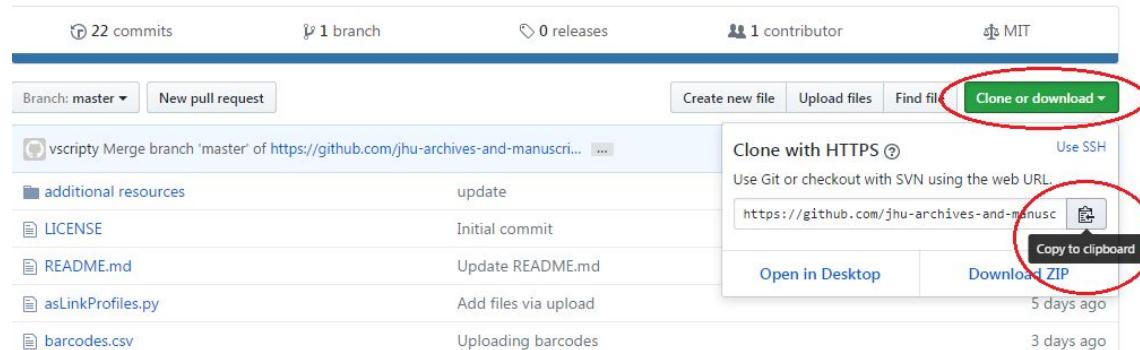
PC

You can proceed if you wish...

Technical Pitstop - Downloading scripts

Everyone

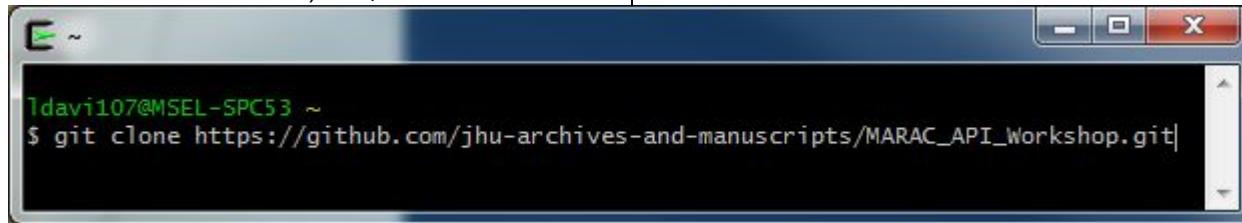
1. Go back to our [GitHub repo](#), which you bookmarked earlier.
2. Then click the green button, click the little clipboard icon, and **Copy to clipboard**



Technical Pitstop - Downloading scripts

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)
4. Hit enter



PC

1. Open Cygwin
2. Type `git clone` [right-click then paste]
↑
(don't type this, we mean an action)
3. Hit enter



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

(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

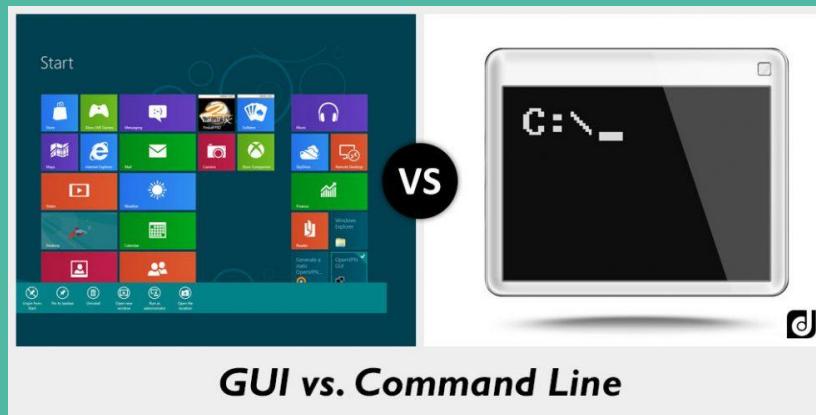


Image from <http://www.differencebtw.com/difference-between-gui-and-command-line/>

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 is a research-based journalism site that 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...



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...

The screenshot shows the Nonprofit Explorer homepage. A green bracket highlights the title "Nonprofit Explorer" and the subtitle "Research Tax-Exempt Organizations". Below the title, it says "By Mike Tigas and Sisi Wei, ProPublica, Updated February 7, 2017." A green arrow points from the word "animal" in the search bar down to the search results. The search bar contains the text "animal". The results page has a heading "Search for Nonprofit Data" and a search bar with the placeholder "Enter a nonprofit's name, a keyword, or city". Below the search bar are filters for "State" (set to "Any State"), "Major nonprofit categories" (set to "Any Category"), "Org. Type" (set to "Any Type"), and a "SEARCH" button.

Nonprofit Explorer
Research Tax-Exempt Organizations

By Mike Tigas and Sisi Wei, ProPublica, Updated February 7, 2017.

Use this database to view summaries of 2.2 million tax returns from tax-exempt organizations and see financial revenue and expenses. You can browse raw IRS data released since 2013 and access over 9.4 million tax returns.

Search for Nonprofit Data

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

animal

Example: ProPublica, Research or Minneapolis

Advanced Search

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

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



The screenshot shows the Postman application interface. At the top, there is a search bar with the URL "http://projects.propublica.org/nonprofits/api/v2/search.json?q=animal". Below the search bar, the method is set to "GET". To the right of the URL, there is a "Send" button, which is circled in blue. Other buttons visible include "Params", "Save", and "Cookies". Below the main toolbar, there are tabs for "Authorization", "Headers", "Body", "Pre-request Script", and "Tests", with "Authorization" being the active tab. At the bottom left, there is a "Type" dropdown set to "No Auth".

2. Type this next to the GET command:

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

This is a breakdown of the endpoint we're about to use:

<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 <i>animal</i> , which will output as JSON.

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

Web search versus API

Nonprofit Explorer

Research Tax-Exempt Organizations

Search for Nonprofit Data

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

 State: Any State Major nonprofit categories Any Category Org. Type Any Type SEARCH Advanced Search Examples: ProPublica, Research or Minneapolis

2718 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 2012-2015 calendar years.

1 2 3 4 ... 26 27 28 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	HERMITAGE	PA	Boys Clubs ↳ Youth Development	501(c)(3)

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

```
{
  "total_results": 2718,
  "organizations": [
    {
      "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_pdfts": true,
      "score": 441.1612
    },
    ...
  ]
}
```

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”
```

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.

Google search? Look at the address bar.

Amazon search? Address bar.

Your bank account?? Address bar!

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” without the quotes

The screenshot shows the Chronicling America homepage. At the top, there are links for "Search Pages", "Advanced Search", and "All Digitized Newspapers 1789-1924". Below these are dropdown menus for "All states", "from 1789 to 1924", and a search bar containing the query "the times dispatch". A green oval highlights the search bar. To the right of the search bar is a link to the "US Newspaper Directory, 1690-Present". Below the search bar, it says "Pages Available: 11,845,995". On the left, there's a sidebar with links to "About Chronicling America", "About the Site and API" (which has a green arrow pointing to it), "Recommended Topics", and "Help". Under "More Resources", there are links to the "National Digital Newspaper Program", "NDNP Award Recipients", "Newspaper and Current Periodicals Reading Room", "Ask LC Newspaper & Current Periodicals Librarian", and "Historic Newspapers on Flickr" (part of the LC Flickr Commons photostream). The main content area displays three newspaper front pages from different dates: "The Ogden Standard" (12pp.) from Ogden City, Utah, dated 4/18/1917; "The Brattleboro daily reformer" (8pp.) from Brattleboro, Vt., dated 4/18/1917; and "HICKORY DAILY RECORD" (4pp.) from Hickory, N.C., dated 4/18/1917. Each newspaper is labeled with its name, page count, location, and date.

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 it's 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 ▾						
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
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
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	31	
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					
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
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
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
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
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 the 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.

The screenshot shows a REST client interface with the following details:

- URL: http://chroniclingamerica
- Method: GET (highlighted with a green oval)
- Request URL: http://chroniclingamerica.loc.gov/lccn/sn85038615.json (highlighted with a green rectangle)
- Environment: No Environment
- Buttons: Params, Send (highlighted with a green oval)
- Status: 200 OK
- Time: 1425 ms
- Tab navigation: Body (selected), Cookies, Headers (15), Tests

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.

The screenshot shows a POSTMAN interface with the following details:

- Method:** GET
- URL:** <http://chroniclingamerica.loc.gov/lccn/sn85038615.json>
- Authorization:** No Auth
- Status:** 200 OK
- Time:** 1228 ms
- Body:** (Pretty, Raw, Preview, JSON) - The JSON response is displayed below.

```
1 {  
2   "place_of_publication": "Richmond, Va.",  
3   "lccn": "sn85038615",  
4   "start_year": "1903",  
5   "place": [  
6     "Virginia--Richmond"  
7   ],  
8   "name": "The times dispatch.",  
9   "publisher": "Times-Dispatch Co.",  
10  "url": "http://chroniclingamerica.loc.gov/lccn/sn85038615.json",  
11  "end_year": "1914",  
12  "issues": [  
13    {  
14      "url": "http://chroniclingamerica.loc.gov/lccn/sn85038615/1903-01-27/ed-1.json",  
15      "date_issued": "1903-01-27"  
16    },  
17    {  
18      "url": "http://chroniclingamerica.loc.gov/lccn/sn85038615/1903-01-28/ed-1.json",  
19      "date_issued": "1903-01-28"  
20    }  
21  ]  
22}
```

Re-purposing API data

```
1  [
2    {
3      "place_of_publication": "Richmond, Va.",
4      "lccn": "sn85038615",
5      "start_year": "1903",
6      "place": [
7        {
8          "name": "The times dispatch",
9          "publisher": "times-Dispatch Co.",
10         "url": "http://chroniclingamerica.loc.gov/lccn/sn85038615.json",
11         "end_year": "1914",
12         "issues": [
13           {
14             "url": "http://chroniclingamerica.loc.gov/lccn/sn85038615/1903-01-27/ed-1.json",
15             "date_issued": "1903-01-27"
16           },
17           {
18             "url": "http://chroniclingamerica.loc.gov/lccn/sn85038615/1903-01-28/ed-1.json",
19             "date_issued": "1903-01-28"
20           },
21           {
22             "url": "http://chroniclingamerica.loc.gov/lccn/sn85038615/1903-01-29/ed-1.json",
23             "date_issued": "1903-01-29"
24           },
25           {
26             "url": "http://chroniclingamerica.loc.gov/lccn/sn85038615/1903-01-30/ed-1.json",
27             "date_issued": "1903-01-30"
28           },
29           {
30             "url": "http://chroniclingamerica.loc.gov/lccn/sn85038615/1903-01-31/ed-1.json",
31             "date_issued": "1903-01-31"
32           },
33           {
34             "url": "http://chroniclingamerica.loc.gov/lccn/sn85038615/1903-02-01/ed-1.json",
35             "date_issued": "1903-02-01"
36           },
37           {
38             "url": "http://chroniclingamerica.loc.gov/lccn/sn85038615/1903-02-02/ed-1.json",
39             "date_issued": "1903-02-02"
40           },
41           {
42             "url": "http://chroniclingamerica.loc.gov/lccn/sn85038615/1903-02-04/ed-1.json",
43             "date_issued": "1903-02-04"
44           },
45           {
46             "url": "http://chroniclingamerica.loc.gov/lccn/sn85038615/1903-02-05/ed-1.json",
47             "date_issued": "1903-02-05"
48           },
49           {
50             "url": "http://chroniclingamerica.loc.gov/lccn/sn85038615/1903-02-06/ed-1.json",
51             "date_issued": "1903-02-06"
52           },
53           {
54             "url": "http://chroniclingamerica.loc.gov/lccn/sn85038615/1903-02-07/ed-1.json",
55             "date_issued": "1903-02-07"
56           },
57           {
58             "url": "http://chroniclingamerica.loc.gov/lccn/sn85038615/1903-02-08/ed-1.json",
59             "date_issued": "1903-02-08"
60           }
61         ]
62       }
63     ]
64   ]
```

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)

GET with GUI - Chronicling America

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

The screenshot shows a web-based interface for making HTTP requests. At the top, there's a URL bar with 'http://chroniclingamerica' and a dropdown menu set to 'No Environment'. Below the URL bar, there are several tabs: 'GET' (which is highlighted with a green oval), 'Body' (which is active and underlined), 'Cookies', 'Headers (15)', and 'Tests'. The main area contains the request line 'http://chroniclingamerica.loc.gov/lccn/sn85038615.json' with a red 'X' over it, indicating it's invalid or has been cleared. To the right of the request line are buttons for 'Params' and 'Send' (which is also highlighted with a green oval). Below these buttons, the status is shown as 'Status: 200 OK' and the time taken as 'Time: 1425 ms'. At the bottom of the interface, there are tabs for 'Body', 'Cookies', 'Headers (15)', and 'Tests'.

One last thing:



<http://chroniclingamerica.loc.gov/lccn/sn85038615.xml>

Questions?

GET with GUI - Twitter

Using an API to collect records

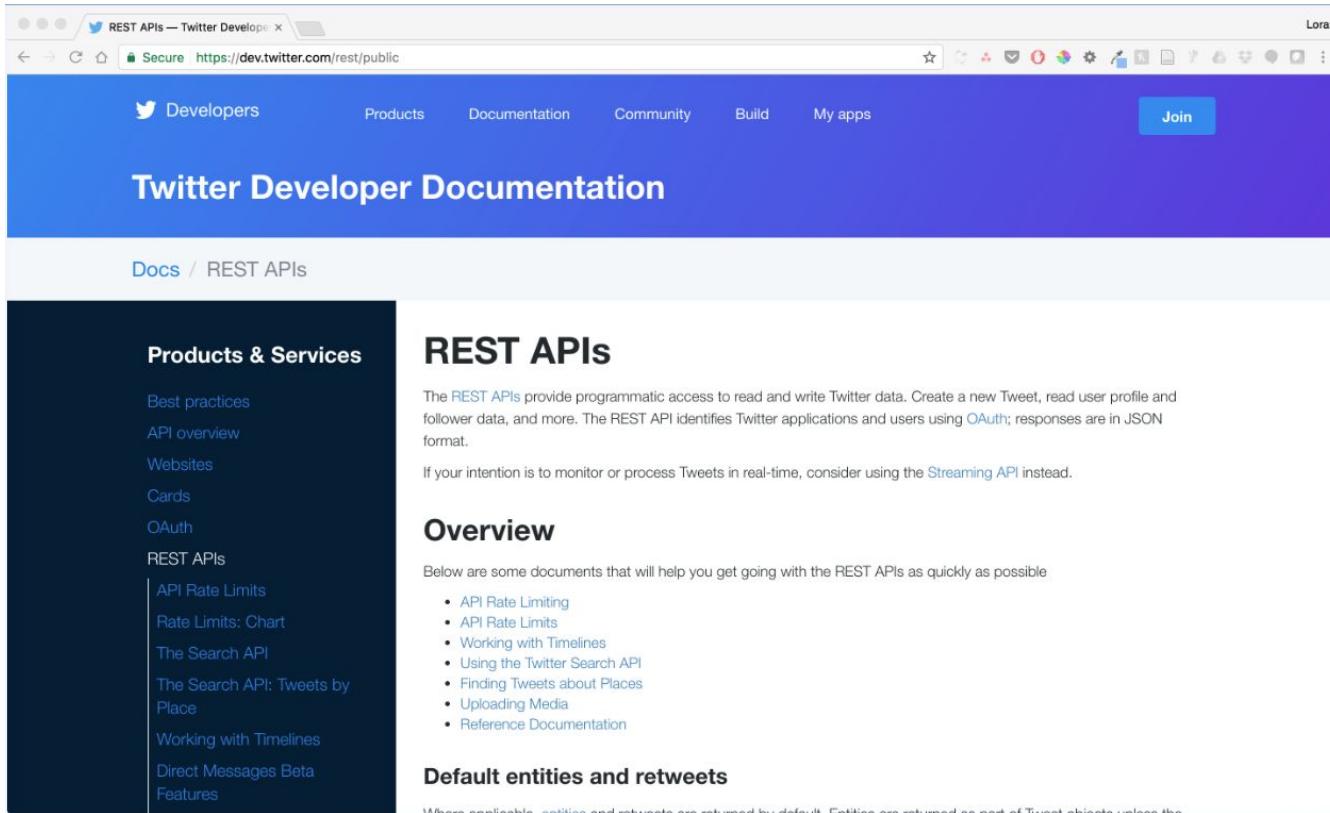
@JohnsHopkins

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

The screenshot shows a Twitter search interface with the query 'johnshopkins' in the search bar. The 'LATEST' tab is selected. The results list several tweets from various users:

- March For ScienceCLE** (@ScienceMarchCLE) - 28m ago: Great recognition for a world-class scientist from #CLE! hub.jhu.edu/2017/04/17/rac... **@JohnsHopkins** (This tweet is circled in red)
- Johns Hopkins biologist, geneticist Rachel Green...** (@JohnsHopkins) - 3 hours ago: Johns Hopkins biologist and geneticist has been researching the cellular structure for nearly three decades hub.jhu.edu
- GalenusDixit** (@GalenOfPergamum) - 35m ago: Today, I am giving a paper on so-called "folk" and "popular" medicine in the ancient world in the history of medicine dept. **@JohnsHopkins** (This tweet is circled in red)
- Hopkins Engineering** (@HopkinsEngineer) - 1h ago: #ICYMI: Photographer captures the complex work materials scientists bit.ly/2nGivds via @HubJHU **@JohnsHopkins** (This tweet is circled in red)
- Going to extremes: Photographer captures the co...** (@JohnsHopkins) - 1h ago: MICA photographer Jay Gould creates arts based on his HEMI artist-in-residence experience hub.jhu.edu

Using an API to collect records



The screenshot shows a web browser displaying the Twitter Developer Documentation for the REST APIs. The page has a blue header with the Twitter logo and navigation links for Developers, Products, Documentation, Community, Build, My apps, and Join. Below the header is a large blue banner with the text "Twitter Developer Documentation". The main content area has a white background. On the left, there's a sidebar titled "Products & Services" with links to Best practices, API overview, Websites, Cards, OAuth, REST APIs, API Rate Limits, Rate Limits: Chart, The Search API, The Search API: Tweets by Place, Working with Timelines, Direct Messages Beta, and Features. The main content area starts with a section titled "REST APIs" which describes the programmatic access provided by the REST APIs. It mentions creating new Tweets, reading user profiles, and follower data, and identifies Twitter applications and users using OAuth. It also notes that responses are in JSON format and suggests the Streaming API for real-time monitoring or processing. Below this is an "Overview" section with a list of documents to help get started, including API Rate Limiting, Working with Timelines, Using the Twitter Search API, Finding Tweets about Places, Uploading Media, and Reference Documentation. At the bottom, there's a section titled "Default entities and retweets" with a note about entities being available in certain regions and not supported by default.

REST APIs — Twitter Developers

Secure <https://dev.twitter.com/rest/public>

Lora

Join

Twitter Developer Documentation

Docs / REST APIs

Products & Services

- Best practices
- API overview
- Websites
- Cards
- OAuth
- REST APIs
- API Rate Limits
- Rate Limits: Chart
- The Search API
- The Search API: Tweets by Place
- Working with Timelines
- Direct Messages Beta
- Features

REST APIs

The REST APIs provide programmatic access to read and write Twitter data. Create a new Tweet, read user profile and follower data, and more. The REST API identifies Twitter applications and users using OAuth; responses are in JSON format.

If your intention is to monitor or process Tweets in real-time, consider using the Streaming API instead.

Overview

Below are some documents that will help you get going with the REST APIs as quickly as possible

- API Rate Limiting
- API Rate Limits
- Working with Timelines
- Using the Twitter Search API
- Finding Tweets about Places
- Uploading Media
- Reference Documentation

Default entities and retweets

Where applicable, entities and retweets are returned by default. Entities are returned as part of Tweet objects, unless otherwise specified.

Using an API to collect records

The screenshot shows a web browser displaying the Twitter Developer Documentation at <https://dev.twitter.com/rest/public/search>. The page title is "The Search API — Twitter Dev". The main content area is titled "The Search API". It describes the API as part of Twitter's REST API, allowing queries against recent or popular Tweets. It notes that the API is focused on relevance and not completeness, and provides a detailed reference for the `GET search/tweets` endpoint. A section titled "How to build a query" explains how to test a query by visiting `twitter.com/search` and copying the URL. A red circle highlights the URL `https://api.twitter.com/1.1/search/tweets.json?q=%40twitterapi`.

The page includes a sidebar with links to "Products & Services" such as Best practices, API overview, Websites, Cards, OAuth, REST APIs, API Rate Limits, Rate Limits: Chart, The Search API, The Search API: Tweets by Place, Working with Timelines, Direct Messages Beta, Features, and Collections.

The Search API

The Twitter Search API is part of Twitter's REST API. It allows queries against the indices of recent or popular Tweets and behaves similarly to, but not exactly like the Search feature available in Twitter mobile or web clients, such as [Twitter.com search](#). The Twitter Search API searches against a sampling of recent Tweets published in the past 7 days.

Before getting involved, it's important to know that the Search API is focused on relevance and not completeness. This means that some Tweets and users may be missing from search results. If you want to match for completeness you should consider using a [Streaming API](#) instead.

A detailed reference on this API endpoint can be found at [GET search/tweets](#).

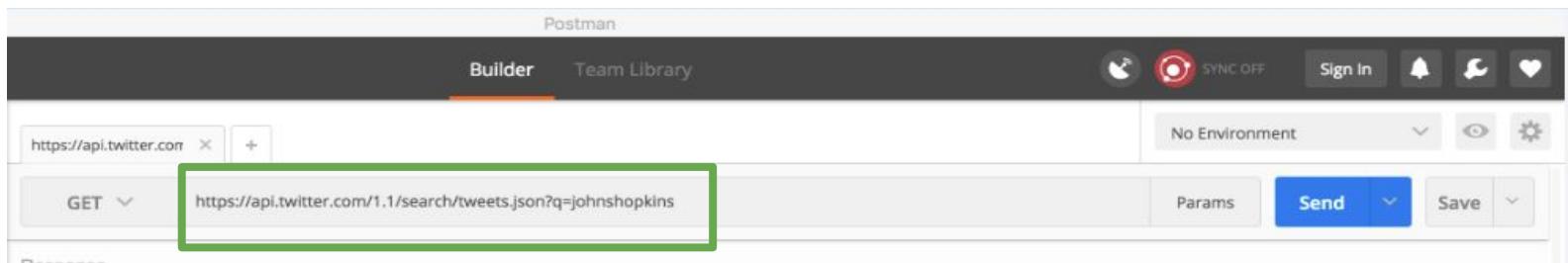
How to build a query

The best way to build a query and test if it's valid and will return matched Tweets is to first try it at [twitter.com/search](#). As you get a satisfactory result set, the URL loaded in the browser will contain the proper query syntax that can be reused in the API endpoint. Here's an example:

1. We want to search for Tweets referencing @twitterapi account. First, we run the search on [twitter.com/search](#)
2. Check and copy the URL loaded. In this case, we got: <https://twitter.com/search?q=%40twitterapi>
3. Replace "<https://twitter.com/search>" with "<https://api.twitter.com/1.1/search/tweets.json>" and you will get:

<https://api.twitter.com/1.1/search/tweets.json?q=%40twitterapi>

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

The screenshot shows the Postman application interface. At the top, the title "Postman" is visible, followed by tabs for "Builder" and "Team Library". Below the tabs, the URL "https://api.twitter.com" is entered in the address bar, which is highlighted with a green box. To the right of the address bar are buttons for "Sign In", "SYNC OFF", and various notification and settings icons.

In the main workspace, a "GET" request is selected, and the URL "https://api.twitter.com/1.1/search/tweets.json?q=johnshopkins" is displayed in the request field. To the right of the URL are buttons for "Params", "Send", and "Save".

The "Headers" tab is active, showing a single header entry: "Key" (New key) and "Value" (value). Below the headers, the "Body" tab is selected, showing the response body in JSON format. The response status is "Status: 400 Bad Request", "Time: 29 ms", and "Size: 360 B".

The response body is a JSON object:

```
1+ {  
2+   "errors": [  
3+     {  
4+       "code": 215,  
5+       "message": "Bad Authentication data."  
6+     }  
7+   ]  
8+ }
```

The message "Bad Authentication data." is highlighted with a red box.

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!

The screenshot shows the Postman application interface. At the top, the URL is set to `https://api.twitter.com`. Below the URL, there is a dropdown menu under the "Authorization" tab labeled "Type". The "OAuth 1.0" option is highlighted. The "Body" tab is selected in the main content area, displaying a JSON response from the Twitter API. The response indicates a "Bad Authentication" error (code 215). The status bar at the bottom right shows "Status: 400 Bad Request" and "Time: 29 ms".

```
1 - {  
2 -   "errors": [  
3 -     {  
4 -       "code": 215,  
5 -       "message": "Bad Authentication"  
6 -     }  
7 -   ]  
8 - }
```

Using an API to collect records

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

```
1 {  
2   "statuses": [  
3     {  
4       "created_at": "Tue Apr 18 14:16:03 +0000 2017",  
5       "id": 854337735129092096,  
6       "id_str": "854337735129092096",  
7       "text": "RT @ScienceMarchCLE: Great recognition for  
8       \"truncated\": false,  
9       "entities": {  
10         "hashtags": [  
11           {  
12             "text": "CLE",  
13             "indices": [  
14               72,  
15               76  
16             ]  
17           },  
18           {  
19             "text": "ScienceMarchCLE",  
20             "indices": [  
21               102,  
22               118  
23             ]  
24           }  
25         ],  
26         "symbols": [],  
27         "user_mentions": [  
28           {  
29             "screen_name": "ScienceMarchCLE",  
30             "name": "March For ScienceCLE",  
31             "id": 824422865206472706,  
32             "id_str": "824422865206472706",  
33             "indices": [  
34               3,  
35               19  
36             ]  
37           },  
38           {  
39             "screen_name": "JohnsHopkins",  
40             "name": "Johns Hopkins U.",  
41             "id": 14441010,  
42             "id_str": "14441010",  
43             "indices": [  
44               119,  
45               132  
46             ]  
47           }  
48         ]  
49       }  
50     ]  
51   }  
52 }
```

F. Be a Good Partner to Twitter

1. Follow the [guidelines](#) for using Tweets in broadcast if you display Tweets offline and the [guidelines](#) for using Periscope Broadcasts in a broadcast if you display Periscope Broadcasts 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, Direct Message 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 Tweet Objects and/or User Objects per user of your Service, per day.
 - b. Any Content provided to third parties remains subject to this Policy, and those third parties must agree to the Twitter [Terms of Service](#), [Privacy Policy](#), [Developer Agreement](#), and [Developer Policy](#) before receiving such downloads.
 - a. You may not distribute more than 1,500,000 Tweet IDs to any entity (inclusive of multiple individual users associated with a single entity) within any given 30 day period, without the express written permission of Twitter.
 - b. You may not distribute Tweet IDs for the purposes of (a) enabling any entity to store and analyze Tweets for a period exceeding 30 days without the express written permission of Twitter, or (b) enabling any entity to circumvent any other limitations or restrictions on the distribution of Twitter Content as contained in this Policy, the Twitter Developer Agreement, or any other agreement with Twitter.

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

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.

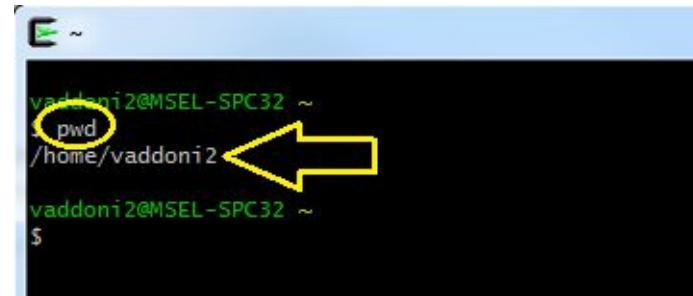
Mac

Mac users should see something like this:

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

PC

PC users should see something like this:



```
vaddoni2@MSEL-SPC32 ~  
pwd  
/home/vaddoni2 ←  
vaddoni2@MSEL-SPC32 ~  
$
```

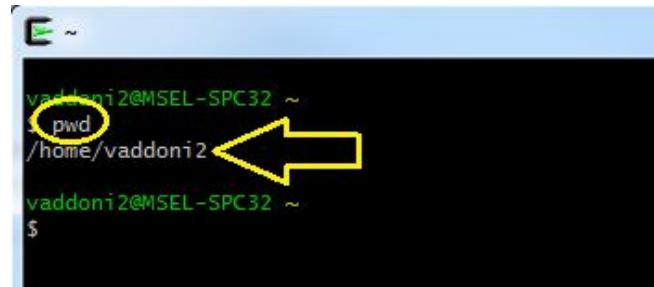
Note: There will be more screenshots for Windows users than Macs for the next few slides as we help PC users determine where they are. If your work computer is Windows, this will eventually matter to you.

Command line bootcamp: *Where are you?*

PC

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

This location:



```
vaddoni2@MSEL-SPC32 ~
$ pwd
/home/vaddoni2
vaddoni2@MSEL-SPC32 ~
$
```

A screenshot of a Cygwin terminal window. The title bar says 'vaddoni2@MSEL-SPC32 ~'. The command 'pwd' is entered, and the output '/home/vaddoni2' is displayed. A yellow arrow points from the text 'This location:' to the output line.

Is this location:



Unix commands for Mac and Cygwin

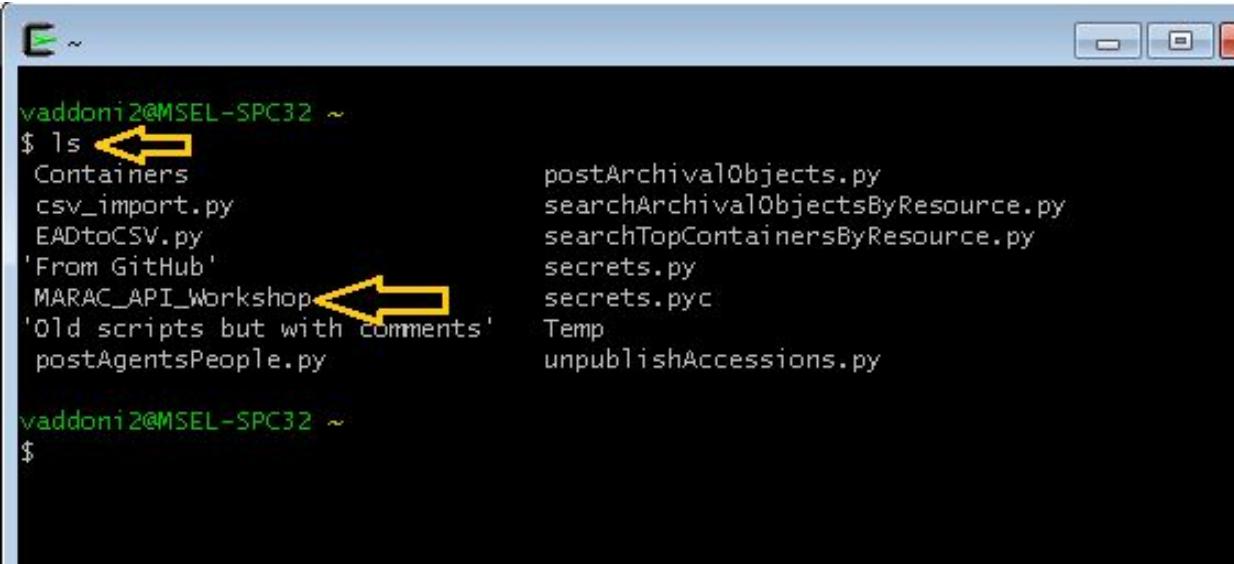
Where am I?

“print working directory”

`pwd`

Command line bootcamp: *What is here?*

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



```
vaddoni2@MSEL-SPC32 ~
$ ls
Containers
csv_import.py
EADtoCSV.py
'From GitHub'
'MARAC_API_Workshop' <-- Yellow arrow points here
'Old scripts but with comments' <-- Yellow arrow points here
postAgentsPeople.py

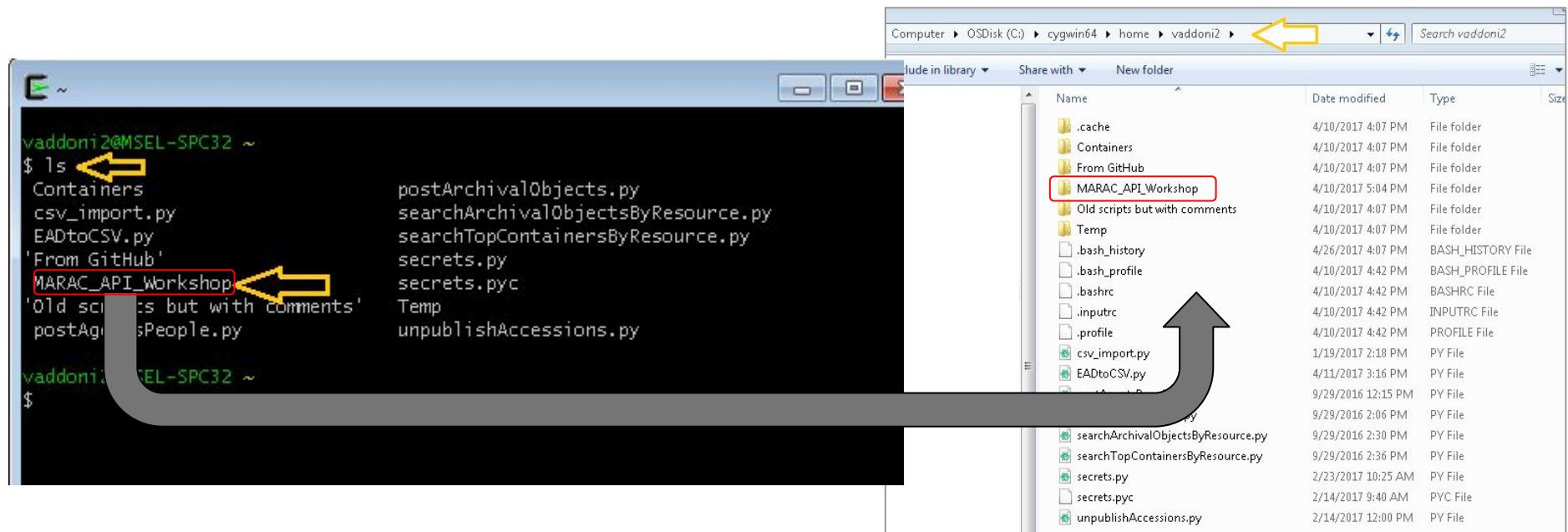
postArchivalObjects.py
searchArchivalObjectsByResource.py
searchTopContainersByResource.py
secrets.py
secrets.pyc
Temp
unpublishAccessions.py

vaddoni2@MSEL-SPC32 ~
$
```

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):



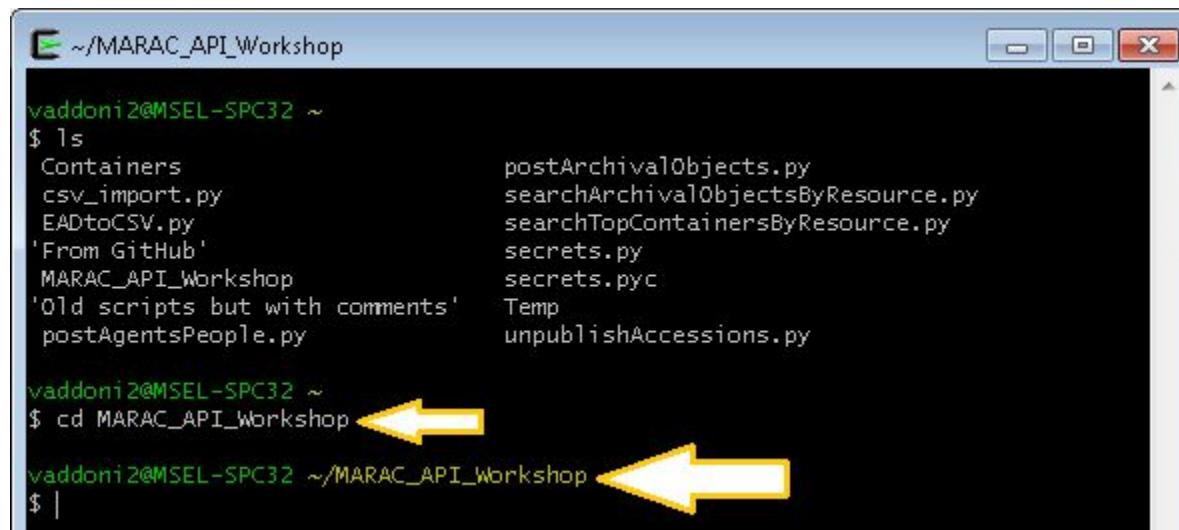
Unix commands for Mac and Cygwin

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 MARAC API Workshop 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 MARAC_API_Workshop`



A screenshot of a terminal window titled 'vaddoni2@MSEL-SPC32 ~'. The window shows the user's home directory (~) and lists several files and directories. The user has typed the command '\$ cd MARAC_API_Workshop' and is awaiting the system's response. Two yellow arrows point from the text above the terminal window to the command line itself, highlighting the 'cd' command and its target directory.

```
vaddoni2@MSEL-SPC32 ~
$ ls
Containers          postArchivalObjects.py
csv_import.py       searchArchivalObjectsByResource.py
EADtoCSV.py        searchTopContainersByResource.py
'From GitHub'
MARAC_API_Workshop secrets.py
'Old scripts but with comments' secrets.pyc
postAgentsPeople.py Temp
unpublishAccessions.py

vaddoni2@MSEL-SPC32 ~
$ cd MARAC_API_Workshop
vaddoni2@MSEL-SPC32 ~/MARAC_API_Workshop
$ |
```

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



```
vaddoni2@MSEL-SPC32: ~/MARAC_API_Workshop
```

A screenshot of a terminal window on a Linux system. The prompt shows the user's name 'vaddoni2' followed by the host name 'MSEL-SPC32' and the current directory '/~/MARAC_API_Workshop'. A large yellow arrow points from the left towards the directory path, highlighting it. The terminal has a black background with white text and a blue cursor.

Unix commands for Mac and Cygwin

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

Command line bootcamp: *Move up*

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”

```
vaddoni2@MSEL-SPC32 ~
$ cd MARAC_API_Workshop You were here
vaddoni2@MSEL-SPC32 ~/MARAC_API_Workshop You went here
$ cd ..
vaddoni2@MSEL-SPC32 ~      And you went back
$
```

Unix commands for Mac and Cygwin

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

Command line bootcamp: *Repeat command*

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”	<code>pwd</code>
What is here?	“list” (remember L as in List)	<code>ls</code>
How do I move from here to there?	“change directory”	<code>cd [type the name of the directory]</code> ↑ (don't type this, we mean an action)
Move up one level		<code>cd ..</code>
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.

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!

(Note to self: Lora should vagrant up)

Load, GET, and compare - VIAF

API possibilities

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



We are
here

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. Open “organizations.csv” in Excel/Numbers (or your preferred spreadsheet program) from the MARAC_API_Workshop folder
2. Add a few additional corporate names of your choosing (universities, businesses, etc.) to column A
3. Save the file (if prompted, save as a .csv) and close
4. 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 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
```

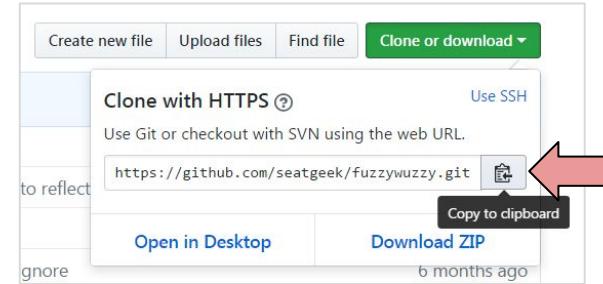
Compare that script to what we're running now:

```
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 MARAC_API_Workshop 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`”
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 or up-arrow `python viafReconciliationCorporate.py`
2. Success!
3. Go back to your spreadsheet program and open the newly created “viafCorporateResults.csv” file from the MARAC_API_Workshop directory

Mac users: Desktop/MARAC_API_Workshop

Windows users: C:\cygwin64\home\[username]\MARAC_API_Workshop

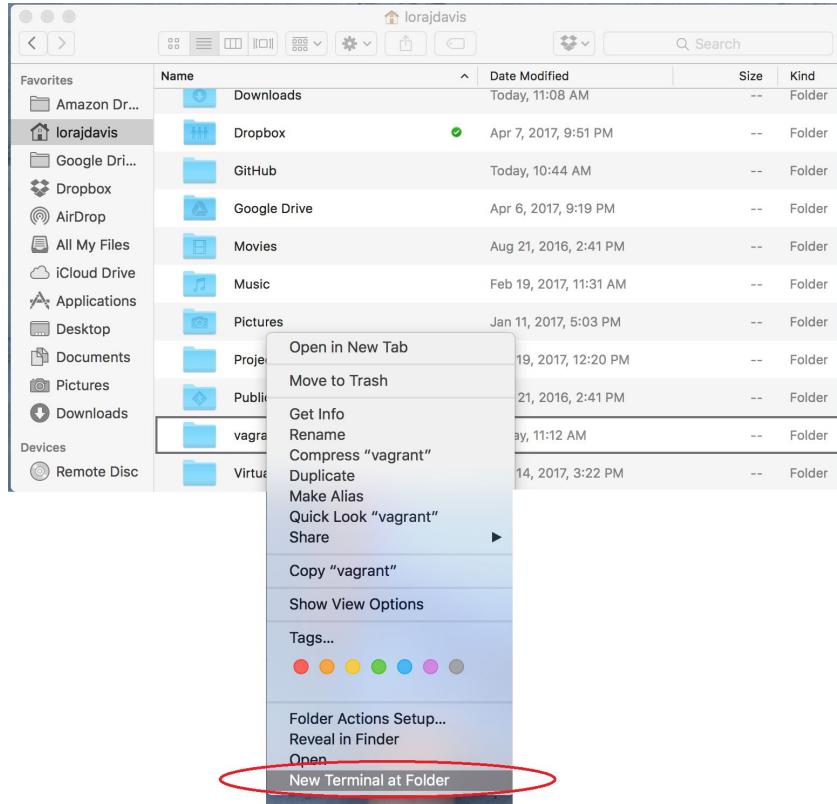
4. The script created this new file for you. Let's inspect it!

Questions?

Technical Pitstop: vagrant install and vagrant up

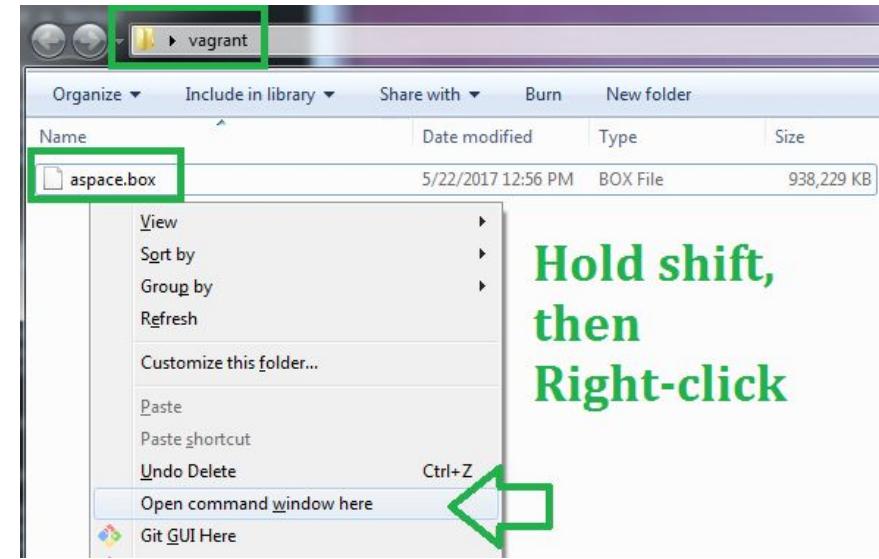
(this is super awesome)

Mac



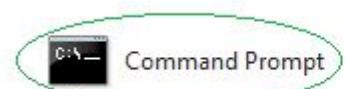
PC

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



**Hold shift,
then
Right-click**

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



Cygwin64 Terminal

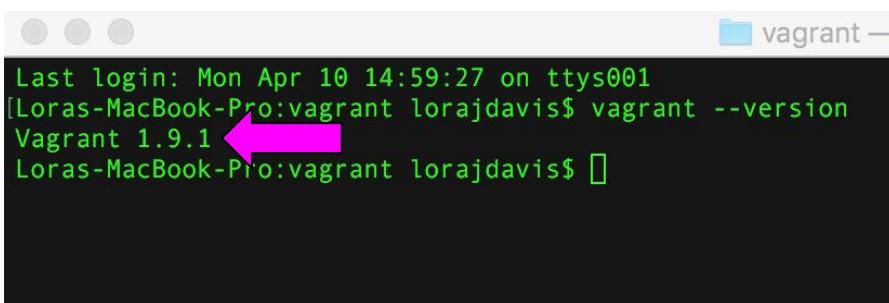
Technical Pitstop - Vagrant

Mac

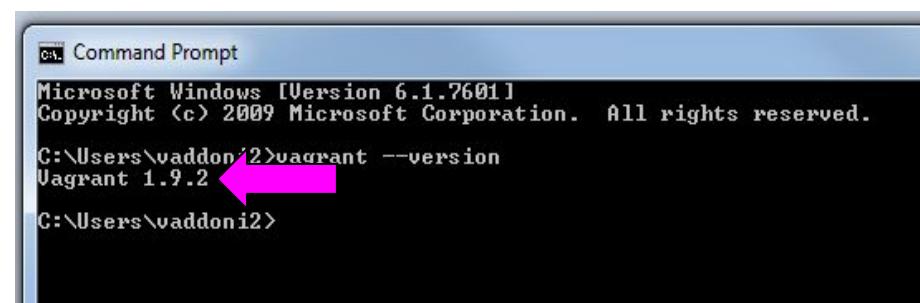
PC

Type `vagrant --version` and hit Enter

Does your screen match the pink arrow below?



```
Last login: Mon Apr 10 14:59:27 on ttys001  
[Loras-MacBook-Pro:vagrant lorajdavis$ vagrant --version  
Vagrant 1.9.1 ←  
Loras-MacBook-Pro:vagrant lorajdavis$ ]
```



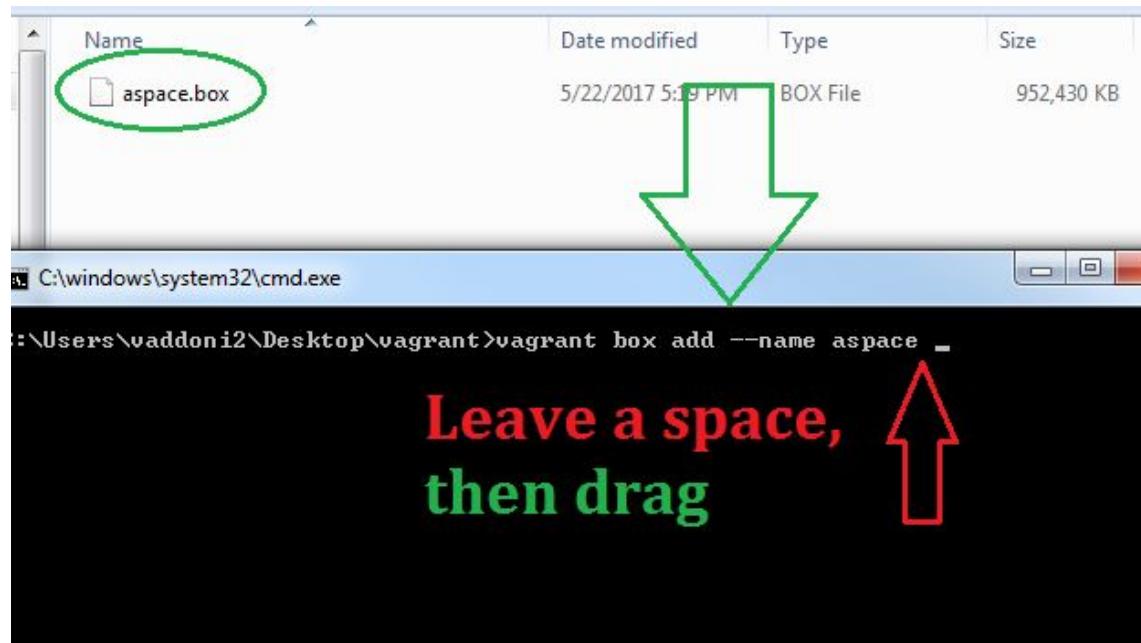
```
Command Prompt  
Microsoft Windows [Version 6.1.7601]  
Copyright (c) 2009 Microsoft Corporation. All rights reserved.  
C:\Users\waddon\2>vagrant --version  
Vagrant 1.9.2 ←  
C:\Users\waddon\2>
```

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

Read this whole slide before doing anything:

Type **vagrant box add --name aspace** and leave a space at the end,
then drag aspace.box from its folder to the prompt:
(see next slide to confirm before hitting enter)



Everyone

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



vagrant — -bash — 120x40

```
Last login: Mon Apr 10 11:19:13 on ttys000
[Loras-MacBook-Pro:vagrant lorajdavis$ vagrant init
A `Vagrantfile` has been placed in this directory. You are now
ready to `vagrant up` your first virtual environment! Please read
the comments in the Vagrantfile as well as documentation on
`vagrantup.com` for more information on using Vagrant.
[Loras-MacBook-Pro:vagrant lorajdavis$ ls
Vagrantfile      aspace.box
Loras-MacBook-Pro:vagrant lorajdavis$ vagrant box add --name aspace /Users/lorajdavis/vagrant/aspace.box]
```

Technical Pitstop - Initialize Vagrant

Mac

PC

You will see text as it runs. Once you have the prompt again, type the following and hit enter:

vagrant init aspace

```
Last login: Mon Apr 17 21:30:39 on ttys002
(Loras-MacBook-Pro:vagrant lorajdavis$ vagrant --version
Vagrant 1.9.1
[Loras-MacBook-Pro:vagrant lorajdavis$ vagrant box add --name aspace /Users/lorajdavis/vagrant/aspace.box
=> box: Box file was not detected as metadata. Adding it directly...
=> box: Adding box 'aspace' (v0) for provider:
  box: Unpacking necessary files from: file:///Users/lorajdavis/vagrant/aspace.box
=> box: Successfully added box 'aspace' (v0) for 'virtualbox'!
Loras-MacBook-Pro:vagrant lorajdavis$ vagrant init aspace[]
```

```
C:\Users\vaddoni2\Desktop\vagrant>vagrant init aspace
A 'Vagrantfile' has been placed in this directory. You are now
ready to 'vagrant up' your first virtual environment! Please read
the comments in the Vagrantfile as well as documentation on
'verbose.com' for more information on using Vagrant.
```

Technical Pitstop - Vagrant up

Mac

```
Last login: Mon Apr 10 11:19:13 on ttys000
[Loras-MacBook-Pro:~]lorajdavis$ vagrant init
A 'Vagrantfile' has been placed in this directory. You are now
ready to 'vagrant up' your first virtual environment! Please read
the comments in the Vagrantfile as well as documentation on
'vagrantup.com' for more information on using Vagrant.
[Loras-MacBook-Pro:~]lorajdavis$ ls
Vagrantfile  aspace.box
[Loras-MacBook-Pro:~]lorajdavis$ vagrant box add --name aspace /Users/lorajdavis/vagrant/aspace.box
==> box: Box file was not detected as metadata. Adding it directly...
==> box: Adding box 'aspace' (v0) for provider:
    box: Unpacking necessary files from: file:///Users/lorajdavis/vagrant/aspace.box
==> box: Successfully added box 'aspace' (v0) for 'virtualbox'!
[Loras-MacBook-Pro:~]lorajdavis$ vagrant up
```

PC

vagrant up

```
C:\Windows\system32\cmd.exe

C:\Users\vaddoni2\Desktop\vagrant>vagrant init
A 'Vagrantfile' has been placed in this directory. You are now
ready to 'vagrant up' your first virtual environment! Please read
the comments in the Vagrantfile as well as documentation on
'vagrantup.com' for more information on using Vagrant.
C:\Users\vaddoni2\Desktop\vagrant>vagrant box add --name aspace C:\Users\vaddoni2\Desktop\vagrant\aspace.box
==> box: Box file was not detected as metadata. Adding it directly...
==> box: Adding box 'aspace' (<v0>) for provider:
    box: Unpacking necessary files from: file:///C:/Users/vaddoni2/Desktop/vagrant/aspace.box
    box: Progress: 100% <Rate: 21.0M/s, Estimated time remaining: ---:-->
==> box: Successfully added box 'aspace' (<v0>) for 'virtualbox'!
C:\Users\vaddoni2\Desktop\vagrant>vagrant up
```

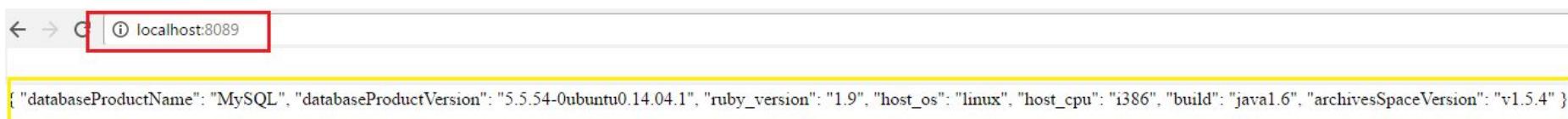
Technical Pitstop - Check Vagrant backend

While you're waiting, open your browser and navigate to:

<http://localhost:8089>

This may take awhile for some of you; stretch and get some water.

Once everything installs, refresh until you see this:

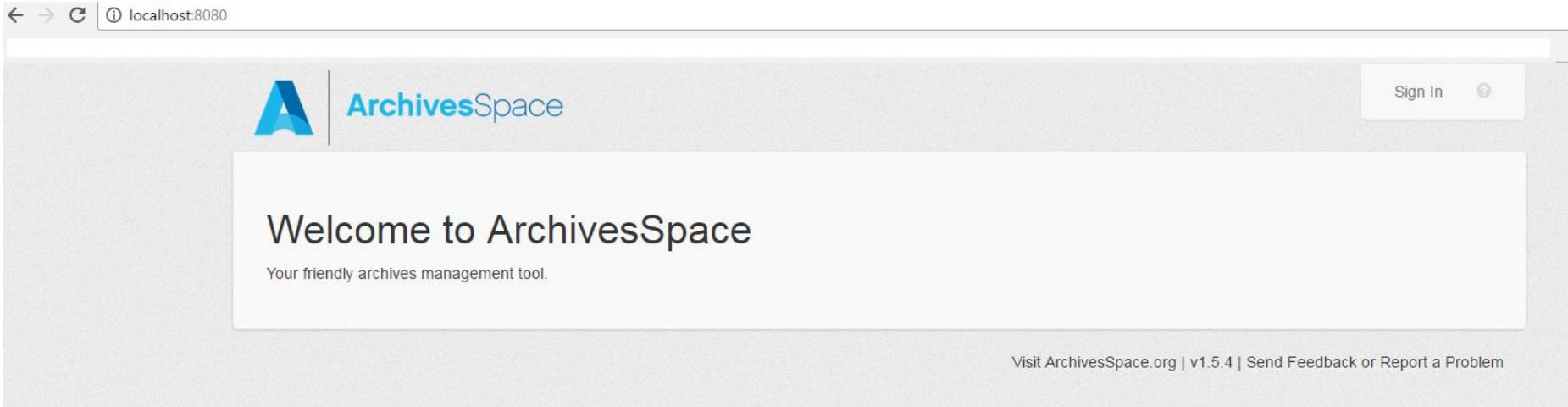


A screenshot of a web browser window. The address bar shows the URL `localhost:8089`. The main content area displays a JSON object:

```
{"databaseProductName": "MySQL", "databaseProductVersion": "5.5.54-0ubuntu0.14.04.1", "ruby_version": "1.9", "host_os": "linux", "host_cpu": "i386", "build": "java1.6", "archivesSpaceVersion": "v1.5.4"}
```

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 archivistry 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 - Authentication

Before we start posting to AS, we need to authenticate, so let's do that and try a GET first. Make your screen look like this:

The screenshot shows the Postman interface with the following configuration:

- Method:** POST
- Endpoint:** http://localhost:8089/users/admin/login
- Body:** (selected tab)
- Content Type:** x-www-form-urlencoded (selected)
- Key:** password
- Value:** admin

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

password: admin

POST to AS with GUI - Authentication

You should see this:

The screenshot shows the Postman interface with the following details:

- Method:** GET (highlighted with a green circle)
- URL:** localhost:8089/users/admin/login
- Authorization:** No Auth
- Body:** JSON (Pretty) - Contains session and user data.
- Session Data:** session: '525f7eb786396736a7722347e87d9a98b3bb2918addae5bacdd7916b7b4f5077', user: { lock_version: 3, username: 'admin', name: 'Administrator', 'is system user': true. }

Copy just the value and switch to GET

GET from AS with GUI

Make your screen look like this:

GET <http://localhost:8089/repositories/2/resources/1>

Headers (2)

Key	Value
Content-Type	application/x-www-form-urlencoded
X-ArchivesSpace-Session	ed9936142c2bd42139ba4daa8d9a1d20dae04fd16abc5c8fcc38accf382b0bbb

Type this

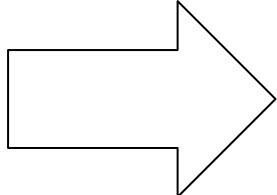
Paste this

Key: X-ArchivesSpace-Session

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

GET from AS with GUI

Success! Your first GET from AS!

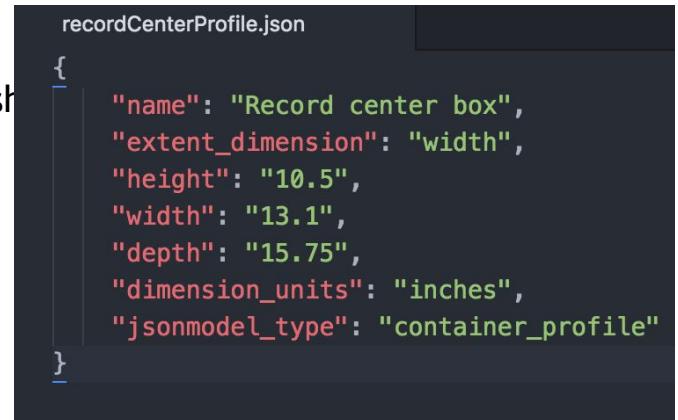


The screenshot shows a POSTMAN API request configuration for a GET operation. The URL is set to `localhost:8089/repositories/2/resources/1`. The 'Headers' tab is selected, containing a single header entry: `X-ArchivesSpace-Session` with value `d27d48823126077f682d1176347d891fe4440cba5e633f9c16...`. The 'Body' tab is also visible below the headers. The response pane displays a JSON object representing an archival resource record:

```
1  {
2      "lock_version": 0,
3      "title": "G\u00e9rard Defaux papers",
4      "publish": true,
5      "restrictions": false,
6      "ead_id": "MS.0584",
7      "finding_aid_title": "G\u00e9rard Defaux papers",
8      "created_by": "admin",
9      "last_modified_by": "admin",
10     "create_time": "2017-05-22T16:46:17Z",
11     "system_mtime": "2017-05-22T16:46:17Z",
12     "user_mtime": "2017-05-22T16:46:17Z",
13     "suppressed": false,
14     "id_0": "MS",
15     "id_1": "0584",
16     "language": "eng",
17     "level": "collection",
18     "finding_aid_status": "in_progress",
19     "jsonmodel_type": "resource",
20     "external_ids": [],
21     "subjects": [],
22     "linked_events": [],
23     "extents": [
```

POST to AS with GUI - Container profiles

1. Open Atom
2. Open “recordCenterProfile.json” from the directory with our cloned GitHub repo
Mac users: Desktop
Windows users: C:\cygwin64\home\[username]\MARAC_API_Workshop
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



A screenshot of the Atom code editor showing a JSON file named "recordCenterProfile.json". The file contains the following JSON object:

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

POST to AS with GUI - Container profiles

The screenshot shows the Postman application interface. At the top, there is a header bar with 'POST' dropdown, URL 'http://localhost:8089/container_profiles', 'Params' button, 'Send' button, and 'Save' dropdown. Below the header are tabs: 'Authorization', 'Headers (1)', 'Body' (selected), 'Pre-request Script', and 'Tests'. Under 'Body' tab, there are radio buttons for 'form-data', 'x-www-form-urlencoded', 'raw' (selected), and 'binary'. The 'Text' dropdown is also visible. The main body area contains a JSON object:

```
1 {
2     "name": "Record center box",
3     "extent_dimension": "width",
4     "height": "10.5",
5     "width": "13.1",
6     "depth": "15.75",
7     "dimension_units": "inches",
8     "jsonmodel_type": "container_profile"
9 }
```

A green arrow points from the text 'Pasted from recordCenterProfile.json' to the JSON object in the body field.

Below the body field is a 'Response' section which is currently empty.

Pasted from
recordCenterProfile.json

Endpoint: http://localhost:8089/container_profiles

POST to AS with GUI - Container profiles

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

Hurray! You should now have a record carton profile in AS!

Go check:

http://localhost:8080/container_profiles

POST to AS with GUI - Container profiles

1. Back to Atom
2. Open “containerProfiles.json”

Mac users: Desktop

Windows users: C:\cygwin64\home\[username]\MARAC_API_Workshop

3. Packages > Atom Beautify > Beautify
4. Here are *all* the profiles, ready to go
5. 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

The screenshot shows the Postman interface with the following details:

- Method:** POST (highlighted with a green circle)
- URL:** localhost:8089/container_profiles (highlighted with a green circle)
- Body Type:** raw (highlighted with a green circle)
- Body Content:** A JSON array containing three container profile objects. The third object is highlighted with a green circle.

```
1 [ {  
2   "name": "Flat box01",  
3   "extent_dimension": "width",  
4   "height": "3",  
5   "width": "12",  
6   "depth": "16",  
7   "dimension_units": "inches",  
8   "jsonmodel_type": "container_profile"  
9 },  
10 {  
11   "name": "Flat box02",  
12   "extent_dimension": "width",  
13   "height": "3",  
14   "width": "21",  
15   "depth": "25",  
16   "dimension_units": "inches",  
17   "jsonmodel_type": "container_profile"  
18 },  
19 {  
20   "name": "Flat box03",  
21   "extent_dimension": "width",  
22   "height": "3",  
23 }]
```

Paste JSON

Endpoint: http://localhost:8089/container_profiles

POST to AS with GUI - Container profiles

The screenshot shows a POST request to `localhost:8089/container_profiles`. The request body contains two JSON objects representing container profiles:

```
89
90
91
92
93
94
95
96
97
98
99
100
101
102
103
104
105
106
107
108
109
110
  },
  "name": "Postcard box",
  "extent_dimension": "width",
  "height": "5.5",
  "width": "3.5",
  "depth": "2",
  "dimension_units": "inches",
  "jsonmodel_type": "container_profile"
},
{
  "name": "Foto corner box",
  "extent_dimension": "width",
  "height": "10.5",
  "width": "13.1",
  "depth": "15.75",
  "dimension_units": "inches",
  "jsonmodel_type": "container_profile"
}
]
```

The response body shows an error message:

```
1 [
  {
    "error": "can't convert String into Integer"
  }
]
```

A large red arrow points from the error message in the response to the JSON input in the Body tab.

Don't hate us: you cannot post multiple records through the GUI.
This frustrating exercise will save you a month.

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



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?

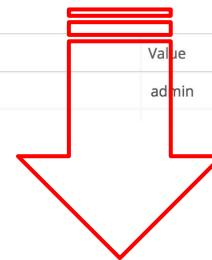


POST http://localhost:8089/users/admin/login

Authorization Headers (1) **Body** ● Pre-request Script Tests Cookies Code

form-data x-www-form-urlencoded raw binary

Key	Value	Bulk Edit
<input checked="" type="checkbox"/> password	admin	



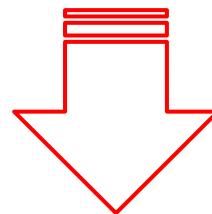
```
secrets.py
```

```
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- 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!) and hit enter
5. Navigate back to AS in your browser (http://localhost:8080/container_profiles)

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!) and hit enter
5. Navigate back to AS in your browser (http://localhost:8080/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*

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.

Container 1: [31151030080422]		Top Container
Container Profile	Record center box	<input checked="" type="checkbox"/>
Indicator	1	
Barcode	31151030080422	
Exported to ILS	Not exported	
Legacy Restricted?	False	

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 Defaux papers (<http://localhost:8080/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 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!) and hit enter
5. Navigate back to AS in your browser (<http://localhost:8080/resources/1>)

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!) and hit enter
5. Navigate back to AS in your browser (<http://localhost:8080/resources/1>)

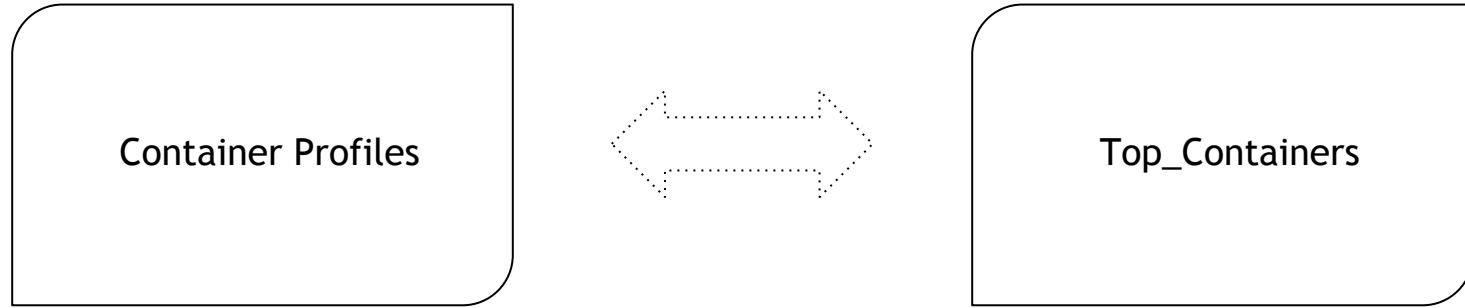
ArchivesSpace!

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*

POST to AS with script- Link profiles

Linking profiles to containers



POST to AS with script - Linking profiles

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:

A screenshot of the ArchivesSpace interface. The title bar says "ArchivesSpace | Resource". The address bar shows the URL "localhost:8080/resources/1#tree::resource_1". A red circle highlights this URL. The main content area displays the "ArchivesSpace" logo and navigation links for "Browse", "Create", and "Search All Records". Below this, the breadcrumb trail shows "Home / Resources / Gérard Defaux papers". A large green number "1" is on the left. At the bottom, there's a tree view with "Gérard Defaux papers" expanded, showing "Research Materials".

View a container profile for its profile number:

A screenshot of the ArchivesSpace interface. The title bar says "ArchivesSpace | Container Profile". The address bar shows the URL "localhost:8080/container_profiles/12". A red circle highlights this URL. The main content area displays the "ArchivesSpace" logo and navigation links for "Browse", "Create", and "Search All Records". Below this, the breadcrumb trail shows "Home / Container Profiles / Record center box". A large orange number "1" is on the left. On the right, there are two sections: "Record center box" and "Basic Information".

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~~

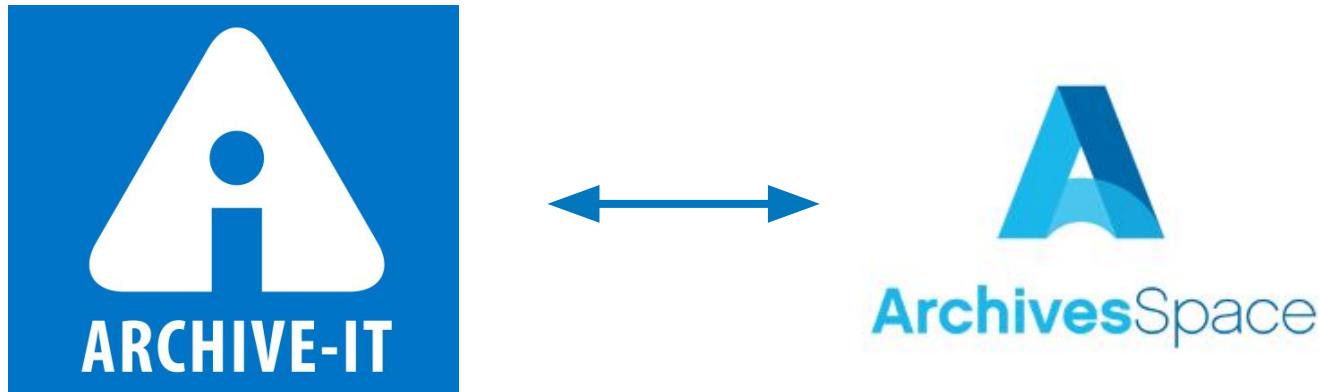


15 minute break!

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



ArchivesSpace

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

Code snippet from `archivelt.py`

App-to-app Communication



The screenshot shows the Archive-It homepage. At the top left is the logo "ARCHIVE-IT" with a stylized letter "A". To its right are four social media icons: Facebook, Twitter, and two others. To the right of the icons is a "Login" button. Below the header, there are four main navigation links: "HOME", "EXPLORE", "LEARN MORE", and "CONTACT US". To the right of these links is a large text block: "The leading web archiving service for collecting and accessing cultural heritage on the web" followed by "Built at the Internet Archive". Below this text is a small icon of a classical building.

Narrow Your Results

Type of Collecting Organization

Sort By: Count | (A-Z)

Colleges & Universities

Collecting Organization

Sort By: Count | (A-Z)

Johns Hopkins University (11)

Explore All Archives

Items in the archive are listed below. Narrow your results at left, or enter a search query below to find a collecting organization, collection, site, specific URL or to search the text of archived webpages.

Collection Name : Johns Hopkins University web collection 

The following results were found for the term(s): library.jhu.edu

- 11 Sites were found.
- Additional results for library.jhu.edu may be found by searching [within the page text](#).

Sites

Search Page Text

Page 1 of 1 (11 Total Results)

Sort By: Best Match | Title (A-Z) | Title (Z-A) | URL (A-Z) | URL (Z-A)

URL: <http://library.jhu.edu>

Collection: Johns Hopkins University web collection

Organization: Johns Hopkins University

Captured 43 times between Aug 20, 2010 and Feb 28, 2017



App-to-app Communication

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



YES!



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 currently 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 + '/' +  
69         crawl['timestamp'] + '/' + crawl['original']  
70         query = '/search?page=1&filter_term[]={"primary_type":"digital_object"}&q=' + doid  
71         existingdOID = requests.get(baseURL + query, headers=headers).json()  
72         if len(existingdOID['results']) > 0:
```

Code snippet from archivelt.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 import legacy 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>

APIs and bulk uploads?

- Hopkins is currently developing workflows for bulk uploads to DSpace, but we're not the ones doing it
- Eric Hanson's GitHub repo for DSpace work:
<https://github.com/ehanson8/dspace-editing>
- His script takes a folder of files and a CSV of user-created metadata and pushes everything in together (but you'll have to talk to him!)

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

```
1  [
2    {
3      "place_of_publication": "Richmond, Va.",
4      "lccn": "sn85038615",
5      "start_year": "1903",
6      "place": [
7        {
8          "name": "The times dispatch.",
9          "publisher": "times-Dispatch Co.",
10         "url": "http://chroniclingamerica.loc.gov/lccn/sn85038615.json",
11         "end_year": "1914",
12         "issues": [
13           {
14             "issue": [
15               {
16                 "url": "http://chroniclingamerica.loc.gov/lccn/sn85038615/1903-01-27/ed-1.json",
17                 "date_issued": "1903-01-27"
18               },
19               {
20                 "url": "http://chroniclingamerica.loc.gov/lccn/sn85038615/1903-01-28/ed-1.json",
21                 "date_issued": "1903-01-28"
22               },
23               {
24                 "url": "http://chroniclingamerica.loc.gov/lccn/sn85038615/1903-01-29/ed-1.json",
25                 "date_issued": "1903-01-29"
26               },
27               {
28                 "url": "http://chroniclingamerica.loc.gov/lccn/sn85038615/1903-01-30/ed-1.json",
29                 "date_issued": "1903-01-30"
30               },
31               {
32                 "url": "http://chroniclingamerica.loc.gov/lccn/sn85038615/1903-01-31/ed-1.json",
33                 "date_issued": "1903-01-31"
34               },
35               {
36                 "url": "http://chroniclingamerica.loc.gov/lccn/sn85038615/1903-02-01/ed-1.json",
37                 "date_issued": "1903-02-01"
38               },
39               {
40                 "url": "http://chroniclingamerica.loc.gov/lccn/sn85038615/1903-02-02/ed-1.json",
41                 "date_issued": "1903-02-02"
42               },
43               {
44                 "url": "http://chroniclingamerica.loc.gov/lccn/sn85038615/1903-02-04/ed-1.json",
45                 "date_issued": "1903-02-04"
46               },
47               {
48                 "url": "http://chroniclingamerica.loc.gov/lccn/sn85038615/1903-02-05/ed-1.json",
49                 "date_issued": "1903-02-05"
50               },
51               {
52                 "url": "http://chroniclingamerica.loc.gov/lccn/sn85038615/1903-02-06/ed-1.json",
53                 "date_issued": "1903-02-06"
54               },
55               {
56                 "url": "http://chroniclingamerica.loc.gov/lccn/sn85038615/1903-02-07/ed-1.json",
57                 "date_issued": "1903-02-07"
58               }
59             ],
60             "date_founded": "1903-01-01"
61           }
62         ]
63       }
64     ]
65   ]
```

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

	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	731665130	73-1665130	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		880099106	88-0099106	ANIMAL FAIR	ANIMAL FAIR	SEALDALE	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		18462882	01-8462882	ANIMAL ORPHANAGE	ANIMAL ORPHANAGE	ORONO	ME	D200	D200	3
011		454902341	45-4902341	ANIMAL PAD	ANIMAL PAD	SAN DIEGO	CA	D20	D20	3
012		818679304	81-0679304	ANIMAL ADVOCATES	ANIMAL ADVOCATES	BARNWELL	SC	D11	D11	3
013		20469900	82-0469900	ANIMAL ALLIES	ANIMAL ALLIES	MANCHESTER	NH	D20	D20	3
014		680630714	68-0630714	ANIMAL BALANCE	ANIMAL BALANCE	EL CERRITO	CA	Z99	Z99	3
015		955967863	95-5967863	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		680208668	68-0208668	ANIMAL PLACE	ANIMAL PLACE	GRASS VALLEY	CA	D200	D200	3
019		880610723	88-0610723	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	MEDIALPOLIS	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		890530102	89-0530102	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 SANITARIANS	ANIMAL SANITARIANS	THOUSAND PLMS	CA	D200	D200	3

```
- "organizations": [Array[100]
  -0: {
    "ein": 731665130,
    "strein": "73-1665130",
    "name": "ANIMAL ALLIANCE",
    "sub_name": "ANIMAL ALLIANCE",
    "city": "WOODLAND HLS",
    "state": "CA",
    "ntee_code": "D20",
    "raw_ntee_code": "D20",
    "subeccd": 3,
    "has_subeccd": 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,
    "subeccd": 3,
    "has_subeccd": 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",
    "subeccd": 3,
    "has_subeccd": 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 “vaf”
 - 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!

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](#) and [Sierra](#);
- 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: Phase 1

(We still have more, but this is important)

Shutting down your Vagrant box - **super important!**

Do the following in Terminal/the command prompt (not Cygwin):

1. Type `vagrant destroy` and confirm the command by typing `y` when prompted
 - a. “Destroying” 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.
2. Type `vagrant box remove aspace` to completely remove the box
 - a. This is the equivalent of returning your computer to the store. It will be removed from your computer (but as long as you keep that `aspace.box` file around, you can always redo this again in the future).

** If you DON'T want to reset the box to it's blank slate, you can use `vagrant halt` instead of `vagrant destroy`

Wrap up: Phase 2

Icing and Advice

Icing: Interpreting (ASpace) API endpoints

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

Non-Official: <https://gist.github.com/jgpawletko/18a1982ec91b290039a968fe4eb924e8>

GET /repositories/:repo_id/resources/:id

Description

Get a Resource

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

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)

Icing: Interpreting (ASpace) API endpoints

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

Determining the repository number:

http://archivesspace.fakeu.edu/repositories/3

Browse Create Search All Records

Home / Repositories / Special Collections

Repository Fields Contact Details

Special Collections

Repository is Currently Selected

Determining an agent number:

archivesspace.fakeu.edu/agents/agent_person/87

Browse Create Search All Records

Home / Agents / Abbe, Cleveland, 1838-1916

Basic Information Dates of Existence Names Contact Details Linked Records

Edit Agent

Abbe, Cleveland, 1838-1916

Basic Information

Agent Type Person
Publish True
Created by admin 2016-05-13 1

Dates of Existence

Existence 1838 – 1916

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
<code>/repositories</code>	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
<code>:repo_id</code>	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
<code>/resources</code>	Other examples are /accessions or /top_containers. EX. http://archivesspace.fakeu.edu:8089/repositories/3/accessions
<code>/1</code>	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

īdē' rāSH(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.

```
"lock_version": 5,  
"title": "university history scrapbook collection",  
"publish": true,
```

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

Thanks!

The Vagrant developed for this workshop was adapted by Lora Woodford from **Dallas Pallen'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 ArchivIt-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!).