iPlant Collaborative
Foundation API Tutorial

iPlant Collaborative™

What is an API?

Application programming interface

From Wikipedia, the free encyclopedia

"API" redirects here. For other uses, see API (disambiguation).

An **application programming interface** (**API**) is a specification intended to be used as an interface by software components to communicate with each other. An API may include specifications for routines, data structures, object classes, and variables.

APIs provide:

- Utility
- Abstraction
- Interoperability
- Documentation

iPlant APIs are of a class known as Web APIs

- This doesn't mean you only do web things with them
- It means you interact with them via web actions

iPlant Collaborative™

What is an API?

Web APIs [edit]

Main article: web service

When used in the context of web development, an API is typically defined as a set of Hypertext Transfer Protocol (HTTP) request messages, along with a definition of the structure of response messages, which is usually in an Extensible Markup Language (XML) or JavaScript Object Notation (JSON) format. While "Web API" is virtually a synonym for web service, the recent trend (so-called Web 2.0) has been moving away from Simple Object Access Protocol (SOAP) based services towards more direct Representational State Transfer (REST) style communications.[5] Web APIs allow the combination of multiple services into new applications known as mashups.[6]

If the Discovery Environment is “User Interface,” the Foundation API is “Machine Interface”

iPlant Collaborative™

REST (REpresentational State Transfer)

1. REST uses URIs to refer to and to access resources.
2. REST is built on top of the stateless HTTP 1.1 protocol.
3. REST uses HTTP commands to define operations.

This last point is essential in REST architecture. HTTP commands have precise semantics:

1. **GET** lists or retrieves a resource at a given URI.
2. **PUT** replaces or updates a resource at a given URI.
3. **POST** creates a resources at a given URI.
4. **DELETE** removes the resources at a given URI.

iPlant Collaborative™

A REST Example (1)

I've built a library database in which I store information about my books, DVDs, music, etc. and exposed it as a RESTful service

`http://fonner.org/library`

Sub-URLs of this /library endpoint represent types of media

- `http://fonner.org/library/books`
- `http://fonner.org/library/dvds`
- `http://fonner.org/library/music`

I can add, update, remove, or fetch records from these via HTTP operations.

iPlant Collaborative™

A REST Example (2)

Let's go about storing information on a **dvd** that my 3 year old son loves:


```
{"title":"Mary Poppins",
 "genre":"Family",
 "release-date":"August, 1964",
 "notes":"Carson will sing Chim Chim
  Cher-ee for a week after watching"}
```

I encode this data into an HTML form and POST it to

`http://fonner.org/library/dvds`

And then…

iPlant Collaborative™

A REST Example (3)

I get back the following:

1) An HTTP status code of 200 OK, which means that the operation succeeded
2) A message body (in JSON format due to my preference)

```
{"id":101,
 "date":"May 18, 2012",
 "uri":"http://fonner.org/library/dvds/101"}
```

Now let's say I want to retrieve that new record because I have a short memory and can't recall what I just POSTed.

A REST Example (4)

I perform an HTTP GET operation on
`http://fonner.org/library/dvds/101`

I get back:

1) An HTTP status code of 200 OK, which means that the operation succeeded
2) A message body

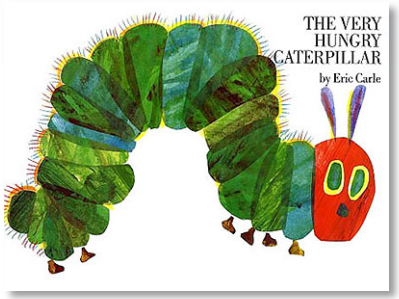
```
{"id":101,
 "date":"May 18, 2012",
 "title":”Mary Poppins”,
 "genre":"Family”,
 "release-date":"August, 1964”,
 "notes":"Carson will sing Chim Chim Cher-ee for
      a week after watching”}
```

A REST Example (5)

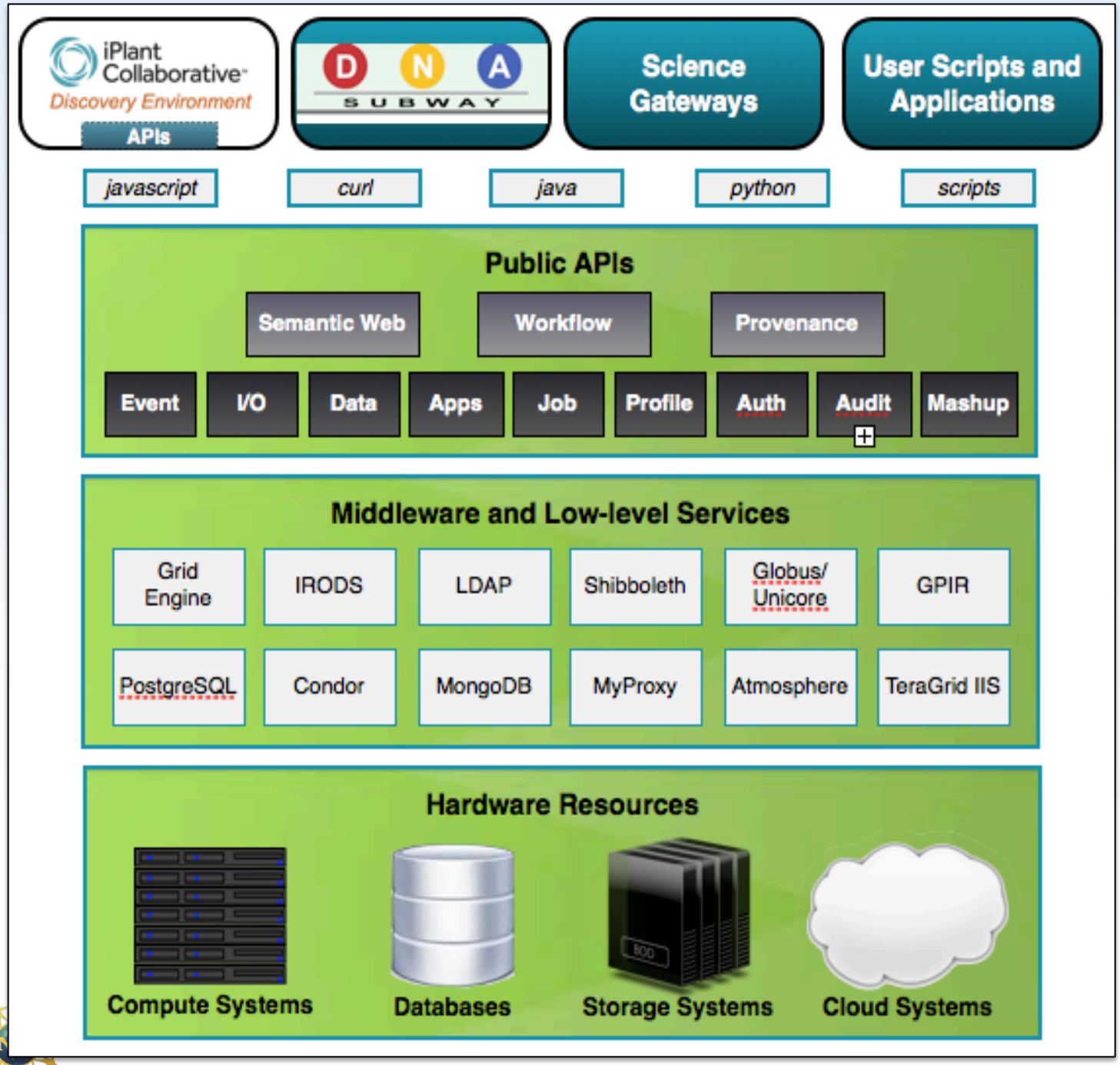
To update the record, say to change the notes to something less silly, I send an updated version of the original data object via HTTP PUT to http://fonner.org/library/dvds/101

To delete the record entirely, I perform an HTTP DELETE on http://fonner.org/library/dvds/101

That's a nutshell example of a simple REST API



Importantly, the paradigm can be extended to accommodate more than just database records! We can move beyond documenting a children's book collection to driving a national-scale cyberinfrastructure.



Language/OS-independent application development

Complexity is abstracted behind API layer

Reuse and integration of Open Source Middleware

National-scale physical resources

Foundation API Overview

Endpoint	Activities
IO	File data storage/retrieval/management. Database interoperability. File metadata management
DATA	Transparent file-format conversion
PROFILE	User information discovery and management
APPS	Registration and discovery of public or private HPC applications
JOBS	Submission and management of computing jobs on XSEDE systems
AUTH	Token-based highly secure authentication with proxy capability
SYSTEMS	Returns availability and other information about XSEDE JOB hosts
POSTIT	A URL –shortener and limited-used URL generator

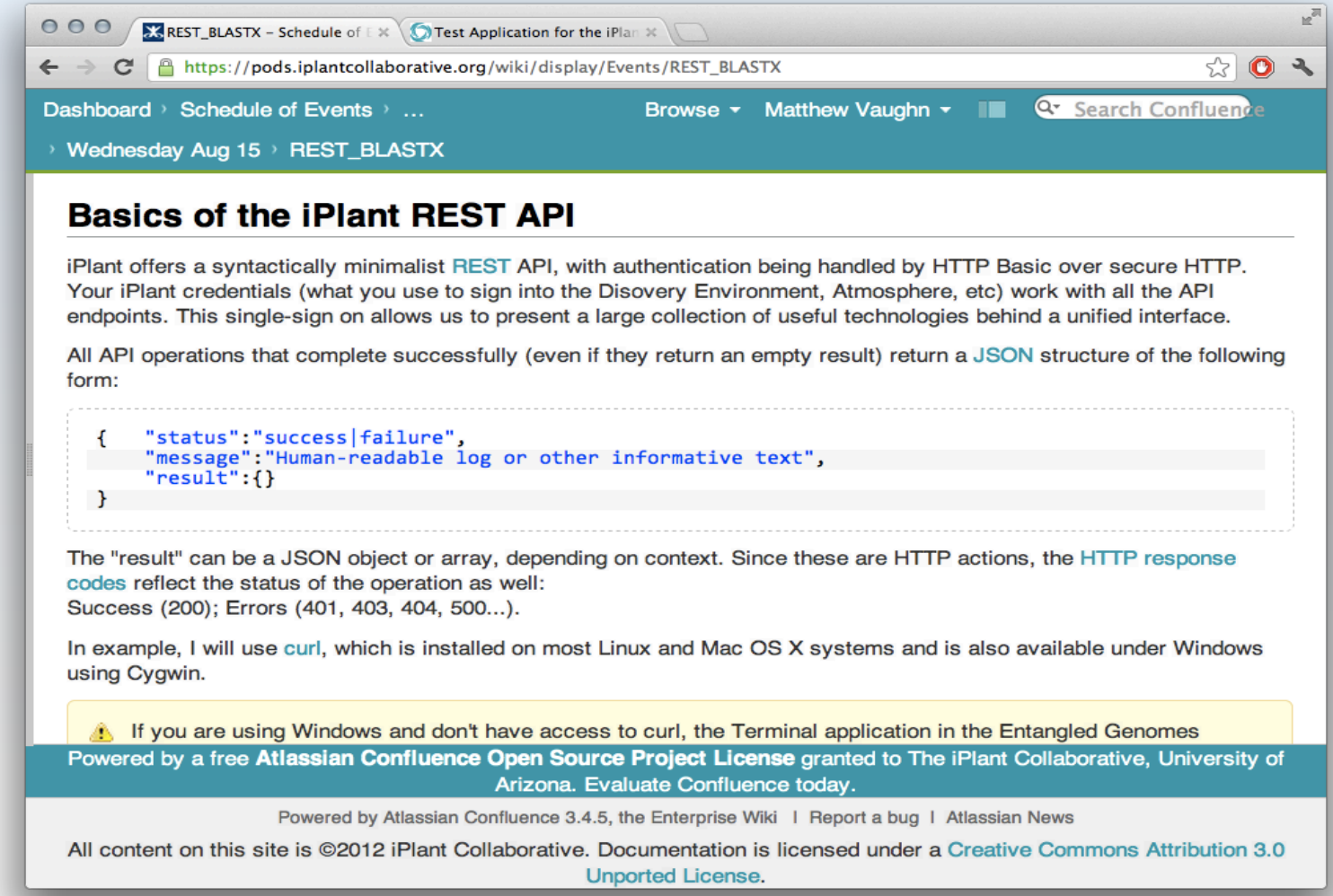
iPlant Collaborative™

Foundation API Overview

Endpoint	Activities
IO	File data storage/retrieval/management. Database interoperability. File metadata management
DATA	Transparent file-format conversion
PROFILE	User information discovery and management
APPS	Registration and discovery of public or private HPC applications
JOBS	Submission and management of computing jobs on XSEDE systems
AUTH	Token-based highly secure authentication with proxy capability
SYSTEMS	Returns availability and other information about XSEDE JOB hosts
POSTIT	A URL –shortener and limited-used URL generator

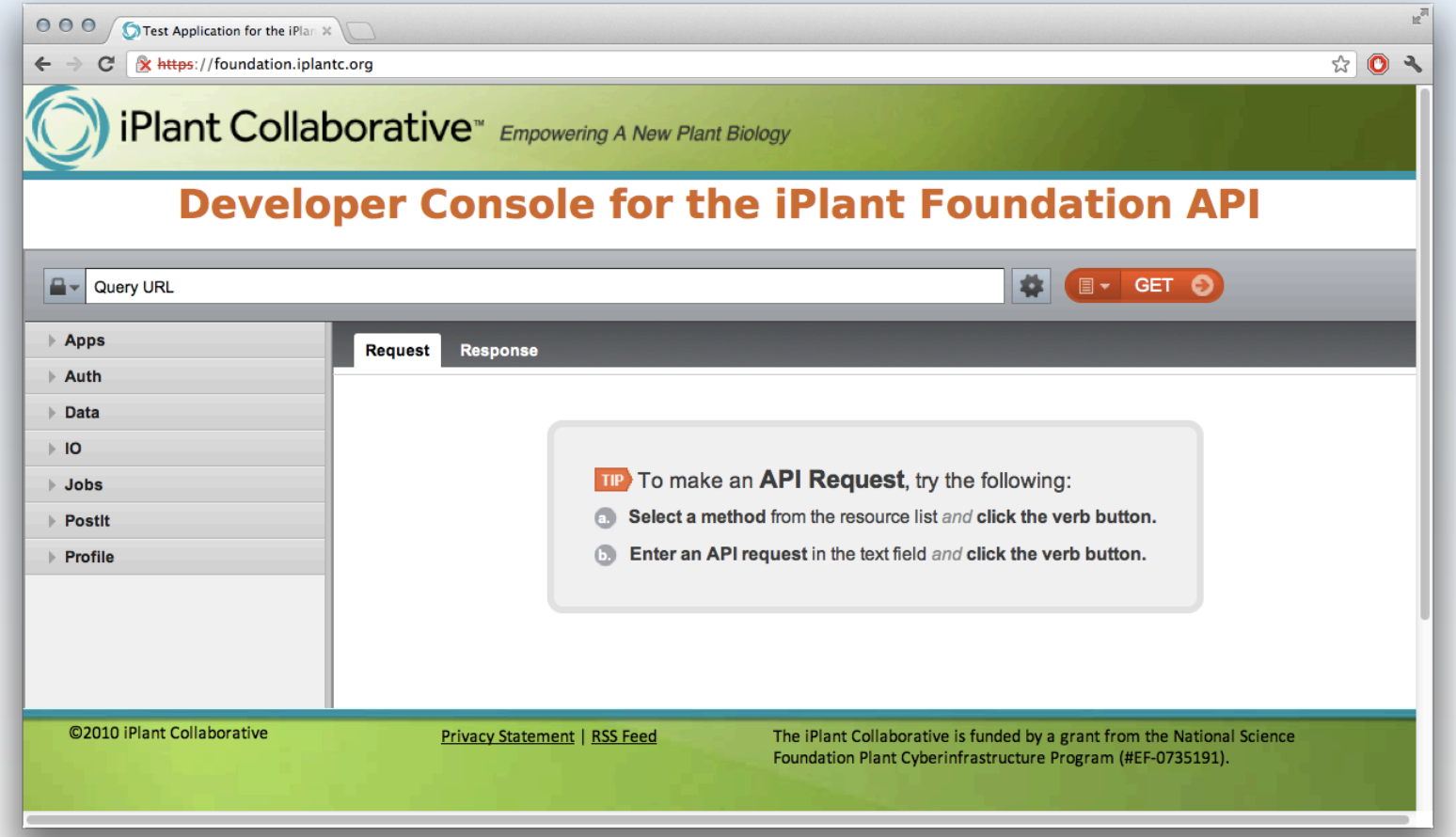
We'll focus on using these four today.

Real World Example: BLASTX



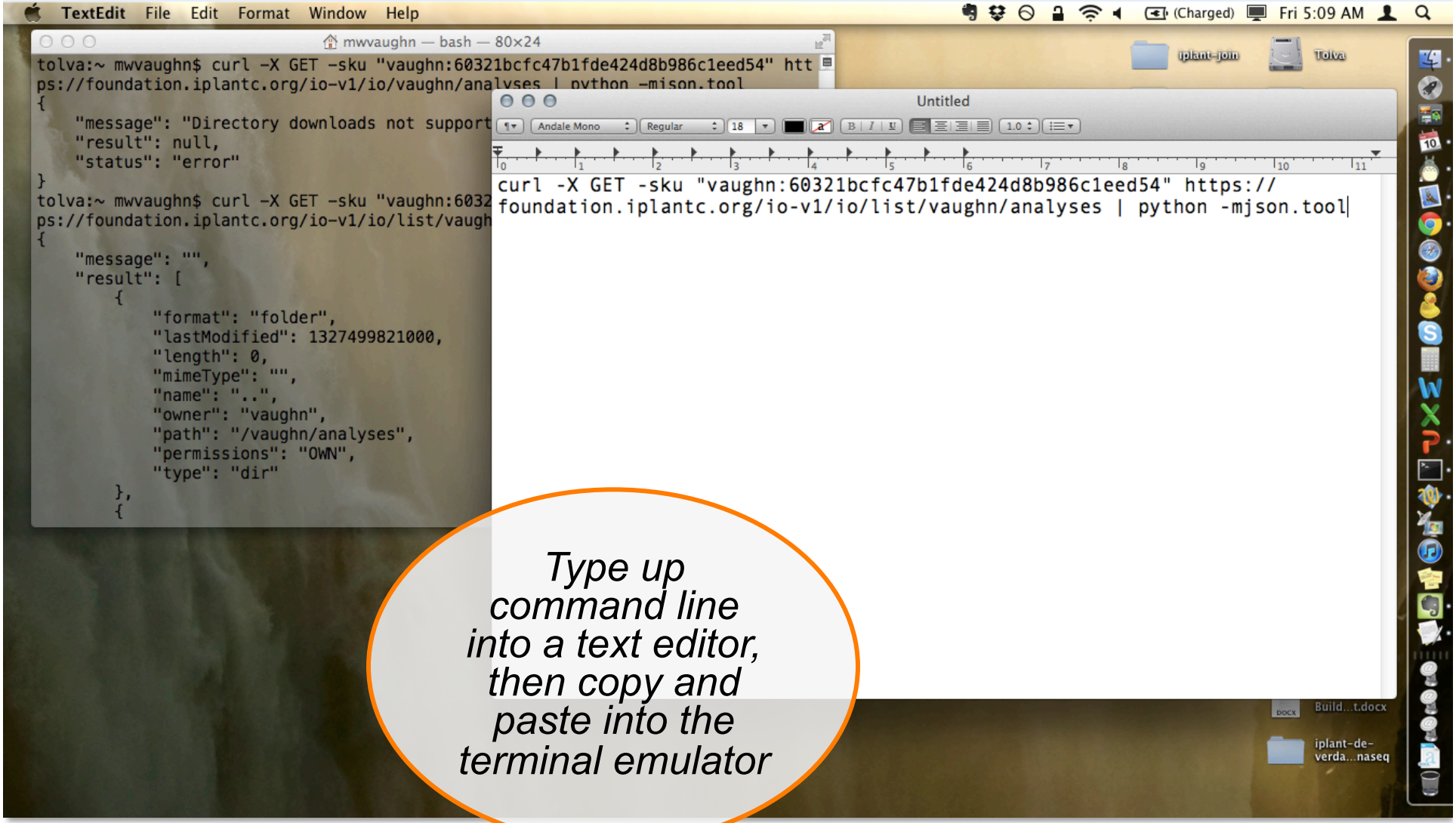
https://pods.iplantcollaborative.org/wiki/display/Events/REST_BLASTX

The apigee API console

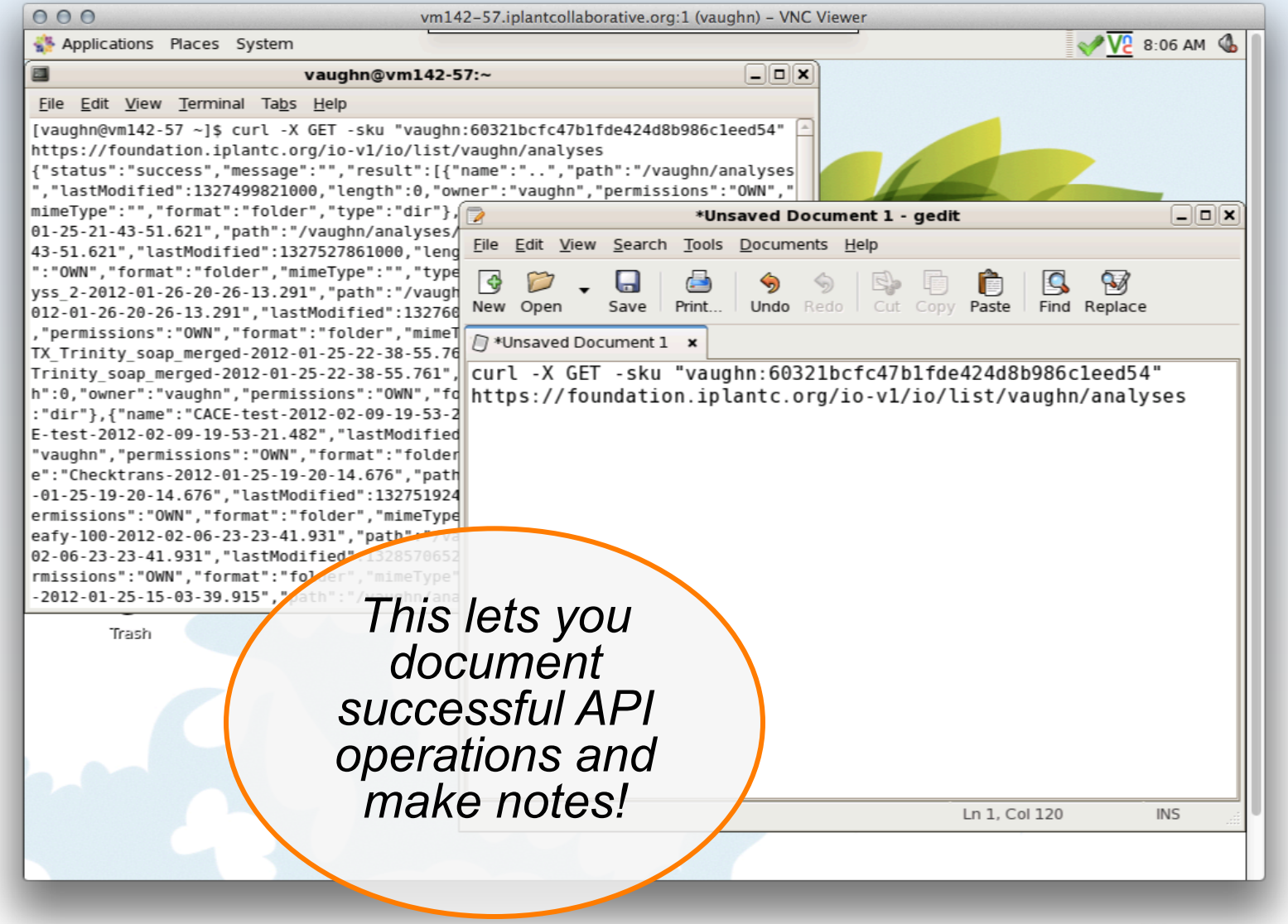


Powered by apigee

Working with cURL



Working with cURL



Interactive Session

iPlant Collaborative™

Additional topics

- Querying and interpreting the APPS catalog
- Callback URLs
- Using POSTIT effectively
- Deploying your own application in the APPS catalog
- Writing a simple HTML/JS application to interact with the iPlant API