The Anatomy Of An API

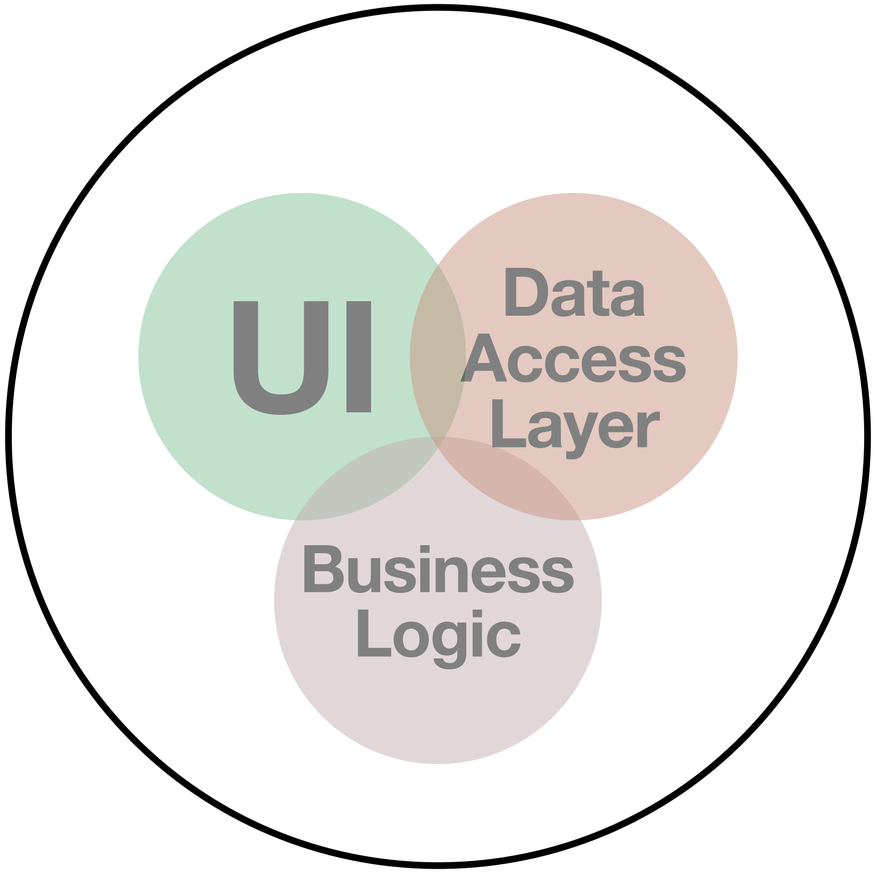
MacSysAdmin 2020

Charles Edge

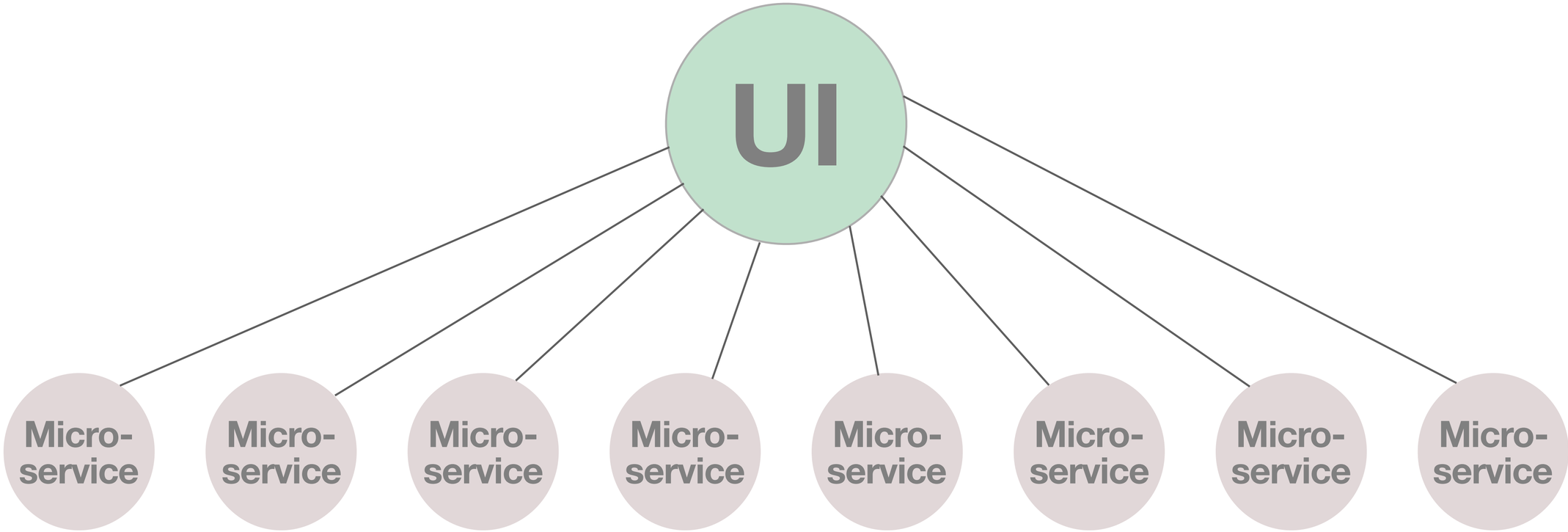
Software Is Just A Collection of Interconnected API Endpoints

Microservices

Monoliths



Microservices



Those Microservices Are Usually API Endpoints

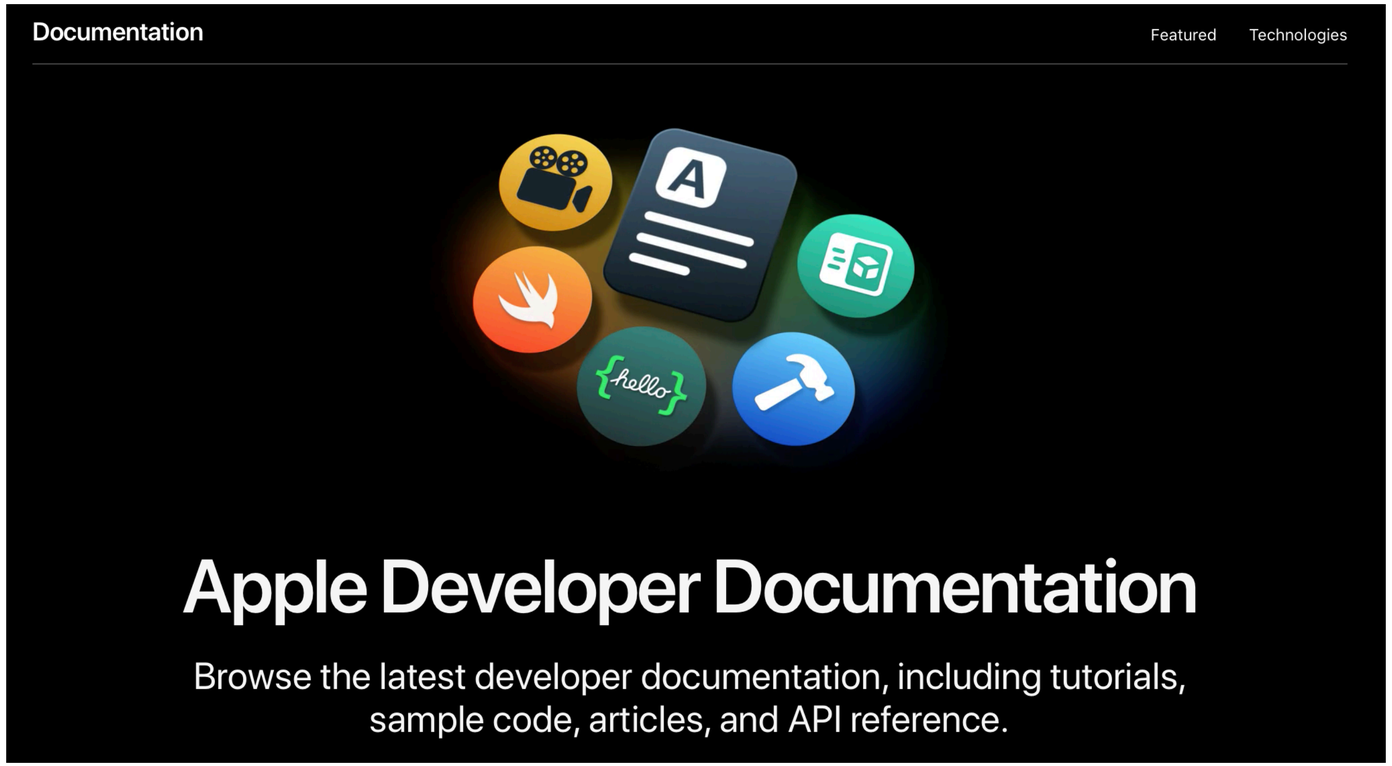
What’s an Endpoint?

A purpose-built communication channel to expose programmatic access to a resource

Oversimplification: It’s like a purpose-built web page

Buuuuuut…

That assumes it’s a web app…



A Super-Brief History of APIs

The Macintosh Toolbox

- 1949 EDSAC gives us program libraries
- 1960: Ivan Sutherland’s Sketchpad (object and instance)
- 1962: Norwegian Kristen Nygaard starts Simula (classes and data bindings)
- 1966: Alan Kay uses “object oriented programming” term
- 1968: “Data structures and techniques for remote computer graphics” uses the term API
- 1980: Kay, et al write Smalltalk at Xerox PARC
- 1984: The original Mac was primarily written in PASCAL (Kay joins Apple)
  - Macintosh Toolbox allowed for procedural calls
- 1987: Windows 1 used DOS as an API of sorts
- 1988: NeXT licenses Objective-C
- 1996: Apple buys NeXT, carbon, cocoa
- 2000: REST
- 2014: Swift

Scraping: Bad

```
curl -s 'https://apps.apple.com/us/app/coursera-learn-new-skills/id736535961?mt8' \
| awk '/meta name="description"/{;print }'
```

https://krypted.com/iphone/get-title-app-apple-app-store-urls/

Web Services

Why Scraping Is Bad

- Inefficient
- No authentication
- Pages can change
- You might get blocked
- Developers throw things at you and call you names
- But it works… Same as shelling out from Swift…

REST: Good

```
curl -X GET \
 https://api.appstoreconnect.apple.com/v1/appInfos/id736535961 \
 -H 'Authorization: orgId=<OrgID>' \
 -H 'Content-Type: application/json' \
 --cert-type p12 \
 --cert <FILENAME>.p12 \
 --pass <PASSWORD>
```

https://developer.apple.com/documentation/appstoreconnectapi/read_app_info_information

Most APIs are RESTful

Representational State Transfer (REST)

REST

- Designed in 2000 by Roy Fielding
- Built on top of http
- See https://standards.rest for a list of the standards
- Used for inter and intra-site communication
- Most developers think endpoints anyone else built are crap

REST

- The Endpoint
- The Method
- The Headers
- The Data

REST

The Endpoint

```
curl https://api.github.com

{
  "current_user_url": "https://api.github.com/user",
  "current_user_authorizations_html_url": "https://github.com/settings/connections/applications{/client_id}",
  "authorizations_url": "https://api.github.com/authorizations",
  "code_search_url": "https://api.github.com/search/code?q={query}{&page,per_page,sort,order}",
  "commit_search_url": "https://api.github.com/search/commits?q={query}{&page,per_page,sort,order}",
  "emails_url": "https://api.github.com/user/emails",
  "emojis_url": "https://api.github.com/emojis",
  "events_url": "https://api.github.com/events",
  "feeds_url": "https://api.github.com/feeds",
  "followers_url": "https://api.github.com/user/followers",
  "following_url": "https://api.github.com/user/following{/target}",
  "gists_url": "https://api.github.com/gists{/gist_id}",
  "hub_url": "https://api.github.com/hub",
  "issue_search_url": "https://api.github.com/search/issues?q={query}{&page,per_page,sort,order}",
  "issues_url": "https://api.github.com/issues",
  "keys_url": "https://api.github.com/user/keys",
  "label_search_url": "https://api.github.com/search/labels?q={query}&repository_id={repository_id}{&page,per_page}",
  "notifications_url": "https://api.github.com/notifications",
  "organization_url": "https://api.github.com/orgs/{org}",
  "organization_repositories_url": "https://api.github.com/orgs/{org}/repos{?type,page,per_page,sort}",
  "organization_teams_url": "https://api.github.com/orgs/{org}/teams",
  "public_gists_url": "https://api.github.com/gists/public",
  "rate_limit_url": "https://api.github.com/rate_limit",
  "repository_url": "https://api.github.com/repos/{owner}/{repo}",
  "repository_search_url": "https://api.github.com/search/repositories?q={query}{&page,per_page,sort,order}",
  "current_user_repositories_url": "https://api.github.com/user/repos{?type,page,per_page,sort}",
  "starred_url": "https://api.github.com/user/starred{/owner}{/repo}",
```

REST

The Methods

- GET: Read
- POST: Create
- PUT: Update/Replace
- DELETE: Delete
- PATCH: Modify

REST

The Header

```
curl -X GET -head http://google.com

HTTP/1.1 301 Moved Permanently
Location: http://www.google.com/
Content-Type: text/html; charset=UTF-8
Date: Fri, 21 Aug 2020 18:40:49 GMT
Expires: Sun, 20 Sep 2020 18:40:49 GMT
Cache-Control: public, max-age=2592000
Server: gws
Content-Length: 219
X-XSS-Protection: 0
X-Frame-Options: SAMEORIGIN
```

https://mobile-jon.com/2019/11/30/a-foray-into-working-with-the-vmware-workspace-one-apis/

REST

The Header

```
curl -X POST \
  --header 'Content-Type: application/json' \
  --header 'Accept: application/json' \
  --header 'Authorization: Basic krypted' \
  --header 'aw-tenant-code: mypassword’ \
  -d '{ \
      "deviceWipe": { \
      "disallowProximitySetup": true, \
      "RequestRequiresNetworkTether": false, \
      "preserveDataPlan": true, \
      "RequestType": "EraseDevice", \
      "PIN": “0000" \
      } \
      }' ‘https://as0000.awmdm.com/API/mdm/devices/commands/DeviceWipe/device/SerialNumber/
      serialnumberhere'
```

https://mobile-jon.com/2019/11/30/a-foray-into-working-with-the-vmware-workspace-one-apis/

-d: The Data in JSON

REST

The Data

```
curl -X POST \

--header 'Content-Type: application/json' \
--header 'Accept: application/json' \
--header 'Authorization: Basic krypted' \
--header 'aw-tenant-code: mypassword’ \

-d '{ \
    "deviceWipe": { \
    "disallowProximitySetup": true, \
    "RequestRequiresNetworkTether": false, \
    "preserveDataPlan": true, \
    "RequestType": "EraseDevice", \
    "PIN": “0000” \
    } \
    }' ‘https://as0000.awmdm.com/API/mdm/devices/commands/DeviceWipe/device/SerialNumber/
    serialnumberhere'
```

https://mobile-jon.com/2019/11/30/a-foray-into-working-with-the-vmware-workspace-one-apis/

JSON

JSON

- Object
- Whitespace
- Separator
- Value
  - String
  - Number
  - Objects
  - Array
  - Boolean
  - Null

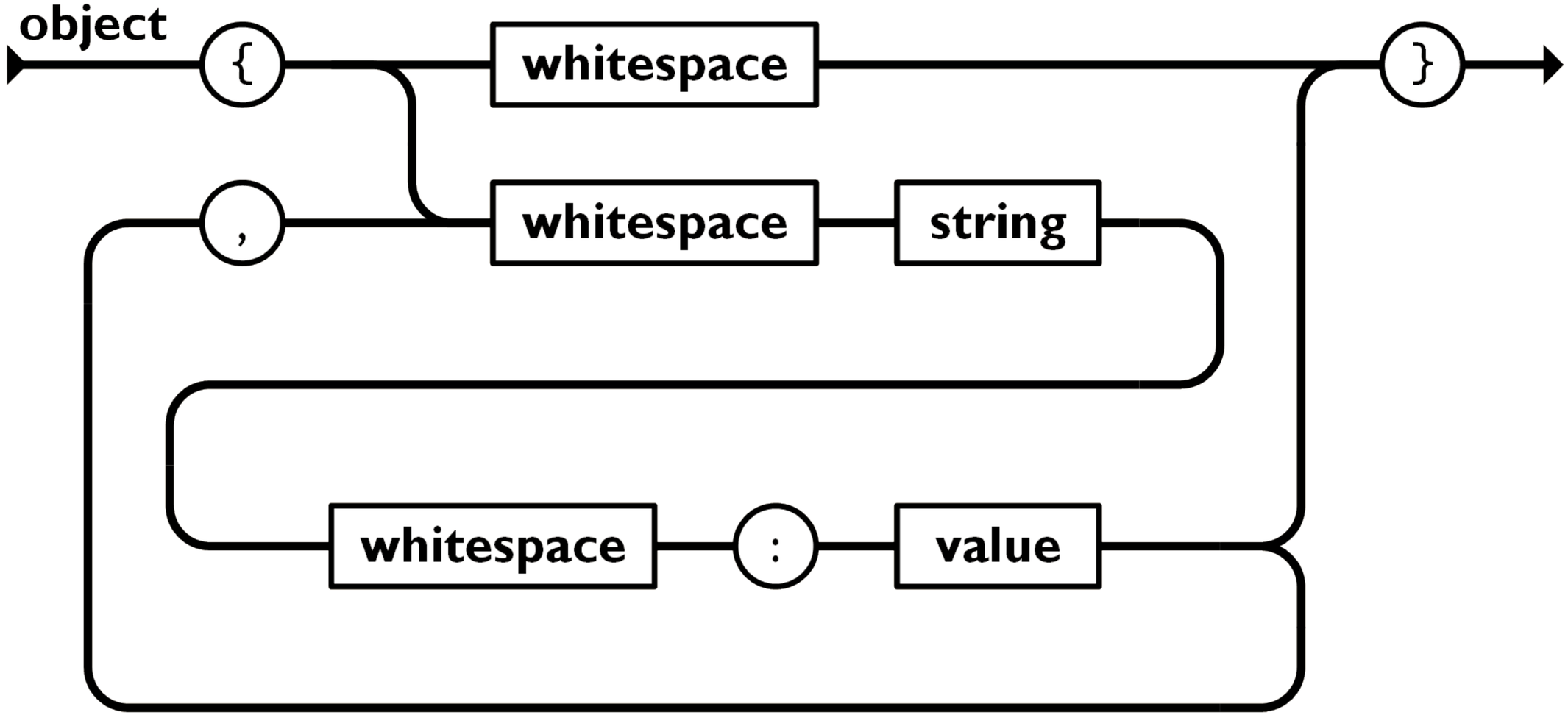
JSON

Examples

```
{Computer: “Performa”}

{Computer: “Performa”, Identifier: “757”}

{“Computers":[
 {Computer: “Performa", Identifier: “575”}
 {Computer: “Macintosh SE", Identifier: “SuperDrive”}
 {Computer: “PowerMac”, Identifier: “G4”}
 ]}
```

JSON

Visually



https://www.json.org/json-en.html

REST

The Data Represented in JSON

```
curl -X POST \

--header 'Content-Type: application/json' \
--header 'Accept: application/json' \
--header 'Authorization: Basic krypted' \
--header 'aw-tenant-code: mypassword’ \

-d '{ \
    "deviceWipe": { \
    "disallowProximitySetup": true, \
    "RequestRequiresNetworkTether": false, \
    "preserveDataPlan": true, \
    "RequestType": "EraseDevice", \
    "PIN": “111111" \
    } \

    }' ‘https://as0000.awmdm.com/API/mdm/devices/commands/DeviceWipe/device/SerialNumber/
    serialnumberhere'
```

https://mobile-jon.com/2019/11/30/a-foray-into-working-with-the-vmware-workspace-one-apis/

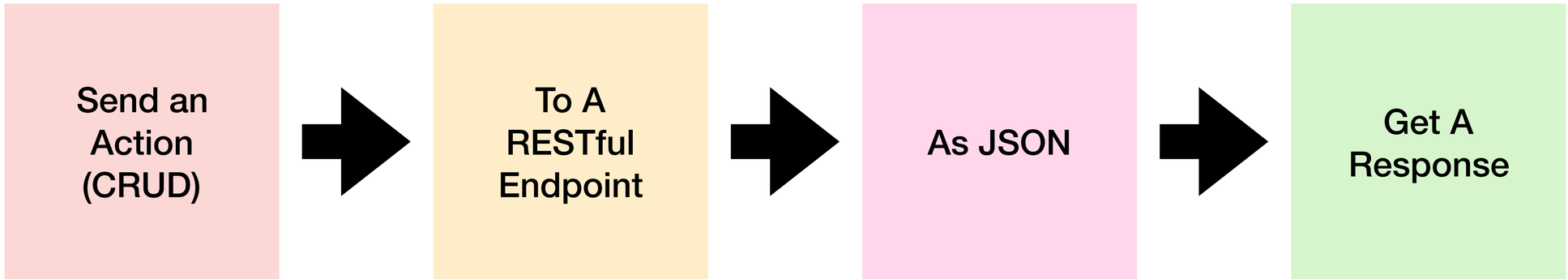
REST

The Data Represented in JSON

```
import requests
import json
import sys

access_token_url = 'https://login.salesforce.com/services/oauth2/token'

data = {
    'grant_type': 'password',
    'client_id': 'INSERTYOURCLIENTIDHERE',
    'client_secret': 'INSERTYOURSECRETHERE',
    'username': sys.argv[1],
    'password': sys.argv[2]
}

headers = {
    'content-type': 'application/x-www-form-urlencoded'

}

req = requests.post(access_token_url, data=data, headers=headers)
response = req.json()

print("Completed Response ==> ")
print(json.dumps(response, indent=4,))
print("")
print("Access Token ==> " + response['access_token'])
print("")
print("Script Completed...")
```

https://gist.github.com/krypted/3bc28b55623b29d880d9996a58b0f035

Oversimplified Transaction

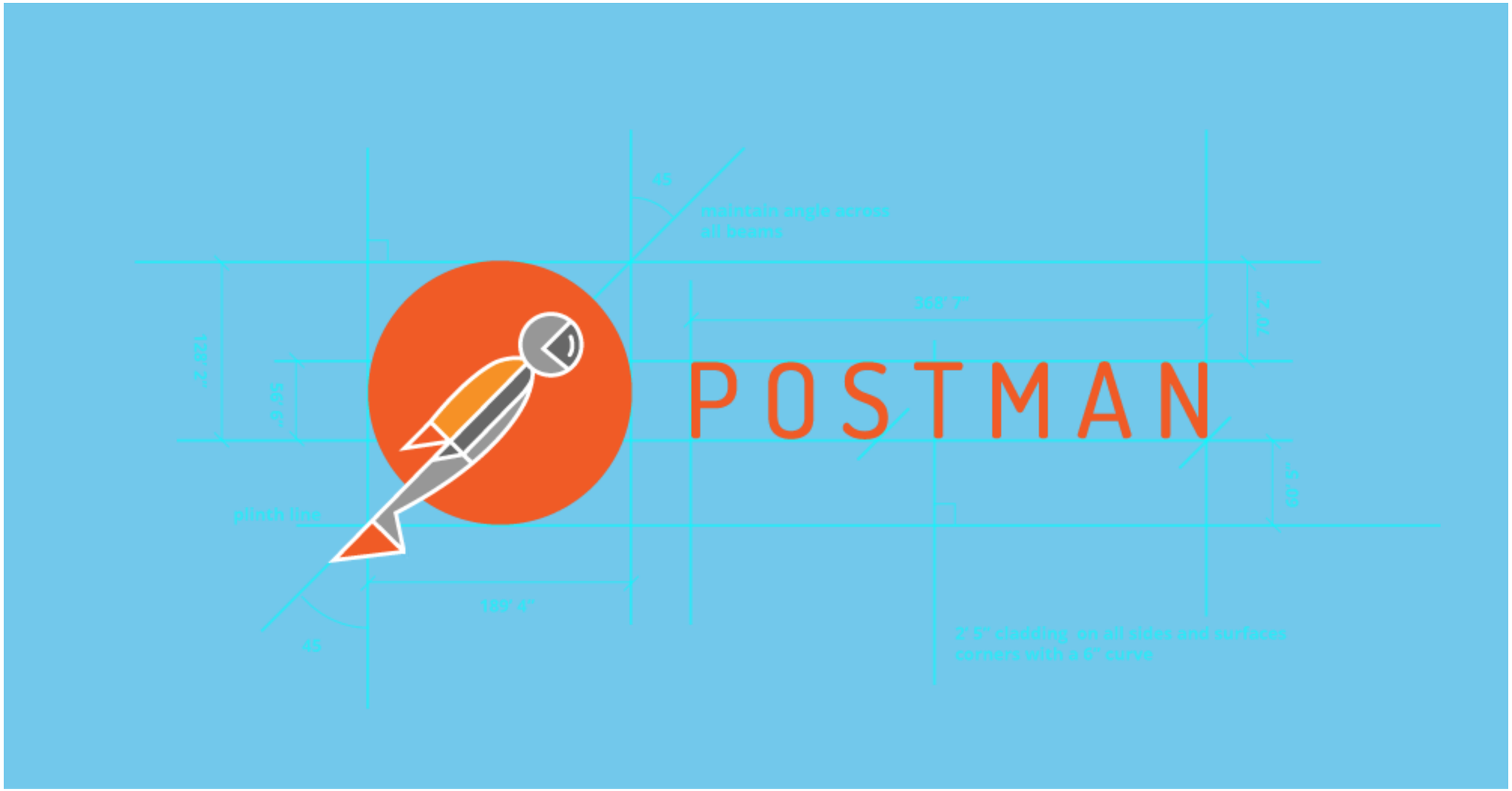


The Other Side

```
import flask
from flask import request, jsonify

app = flask.Flask(__name__)
app.config["DEBUG"] = True

# Create static response
Computer = [
    {'id': 0,
     'name': ‘Krypted Macbook',
     'user': ‘Charles Edge',
     'model': 'MacBook10,1',
     ‘purchase_data': '01011975'},
    {'id': 1,
     'name': ‘Krypted Macbook 1',
     'user': ‘Charles Edge',
     'model': 'MacBook10,1',
     ‘purchase_data': '01011975'},
    {'id': 2,
     'name': ‘Krypted Macbook 2',
     'user': ‘Charles Edge',
     'model': 'MacBook10,1',
     ‘purchase_data': '01011975'}
]

@app.route('/', methods=['GET'])
def home():
    return ‘''<h1>Computer Data</h1>
<p>Just messing around.</p>’''


# A route to return all computers.
@app.route('/api/v1/computers/all', methods=['GET'])
def api_all():
    return jsonify(computers)

app.run()
```

The Other Side

https://github.com/micromdm/micromdm/blob/main/mdm/enroll/endpoint.go


45
maintain angle across
all beams
368' 7"
POSTMAN
plinth line
189' 4"
45
2' 5" cladding on all sides and surfaces
corners with a 6" curve

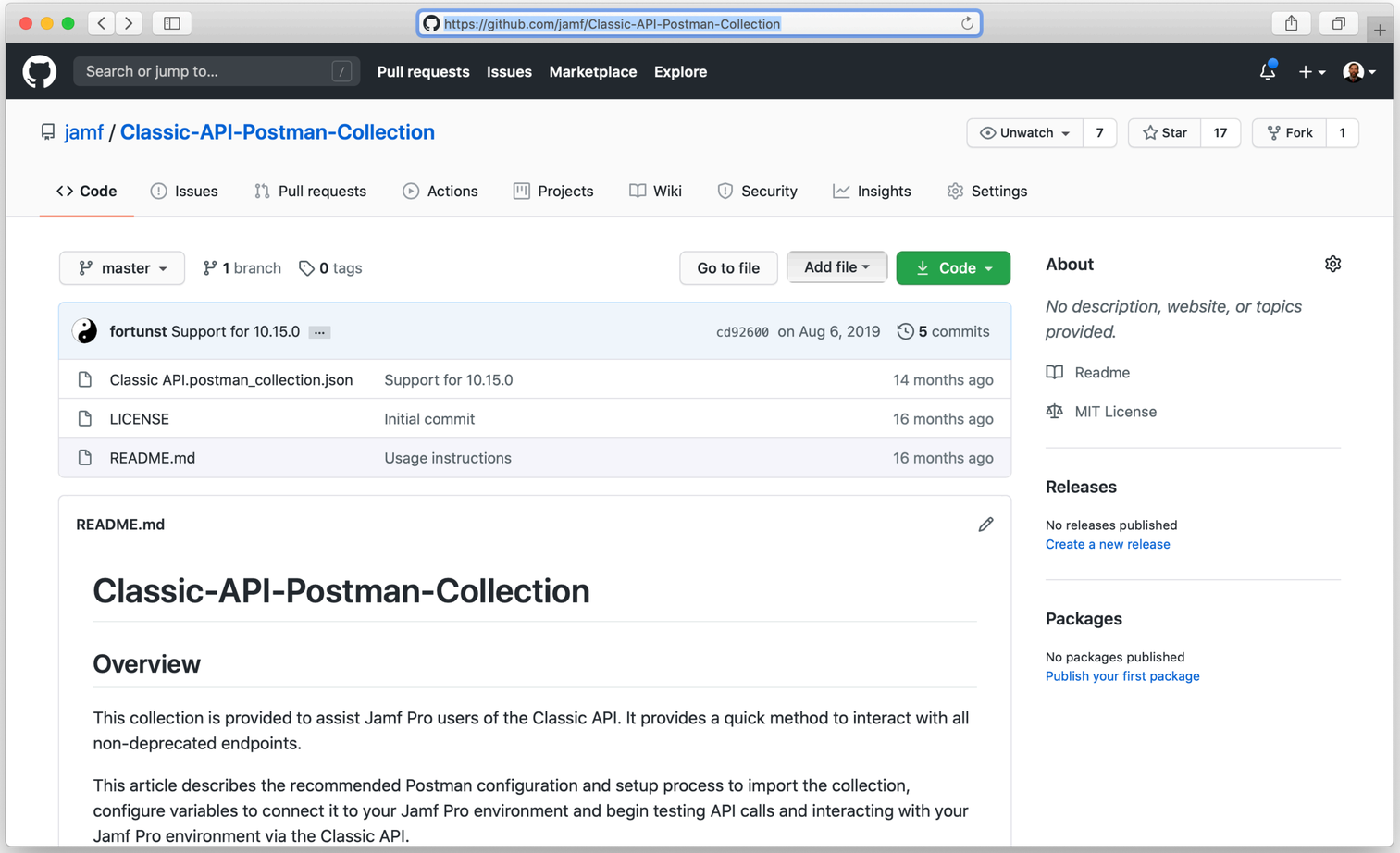
Postman

Why Postman

- Easy visualization of API interactions
- Save information like keys
- Variables
- Run routine scriptable operations
- Collections
- Generate Code: https://learning.postman.com/docs/sending-requests/generate-code-snippets/
- Examples: https://geekygordo.com/2020/04/22/using-postman-for-jamf-pro-api-testing-part-2-creating-and-updating-policies/

Postman

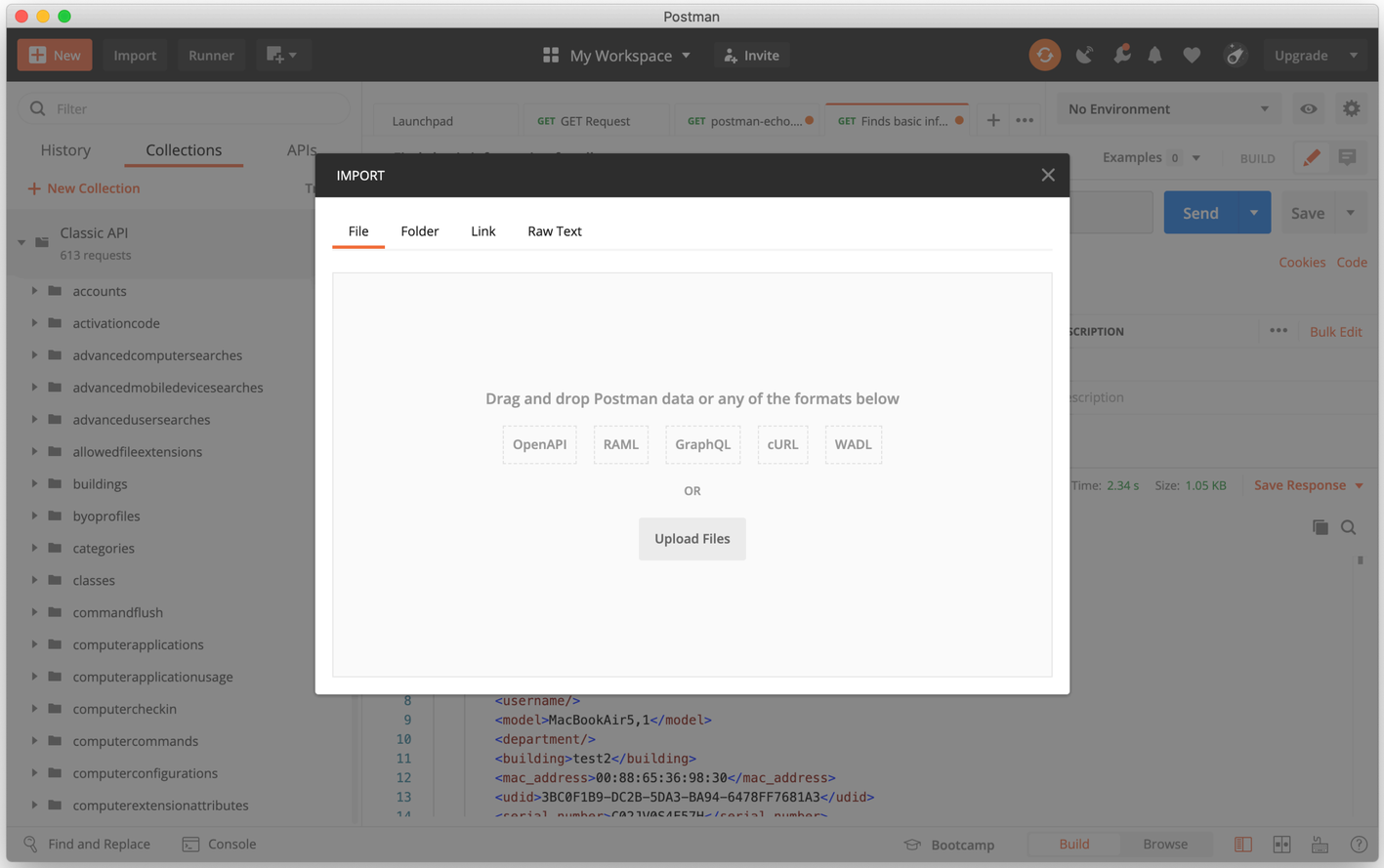
Collections



https://github.com/jamf/Classic-API-Postman-Collection

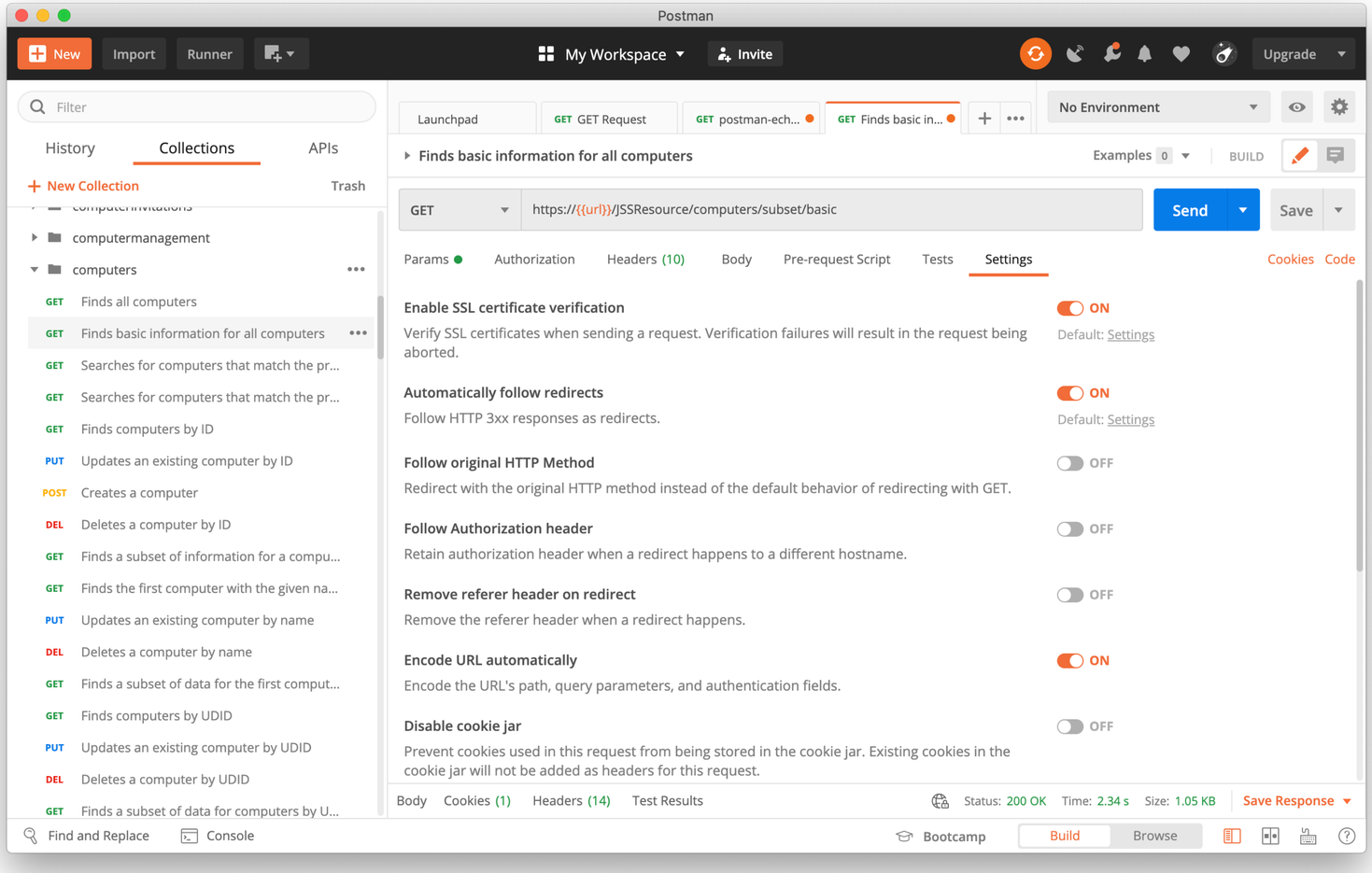
Postman

Importing a Collection



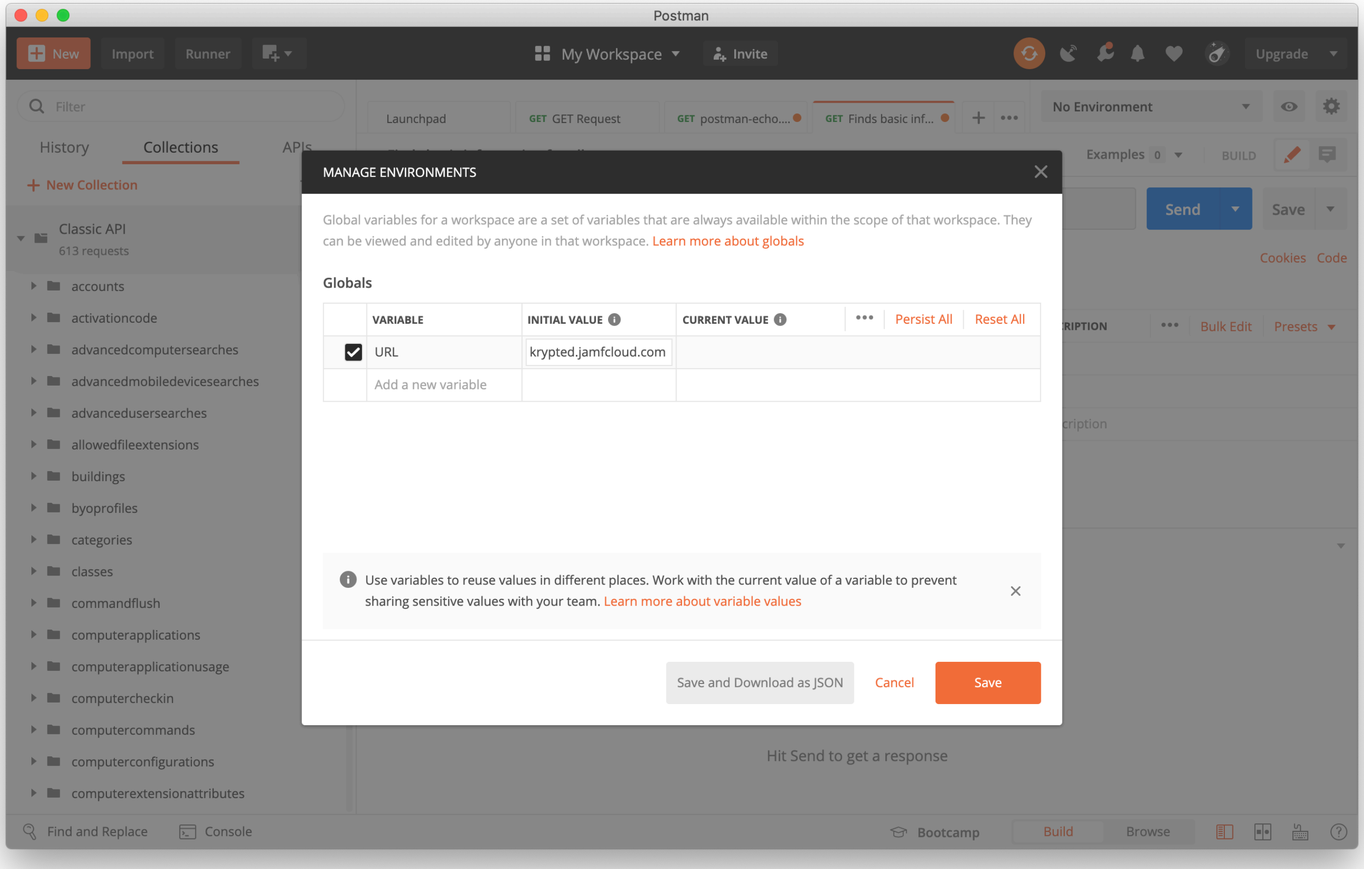
Postman

Find The Endpoint



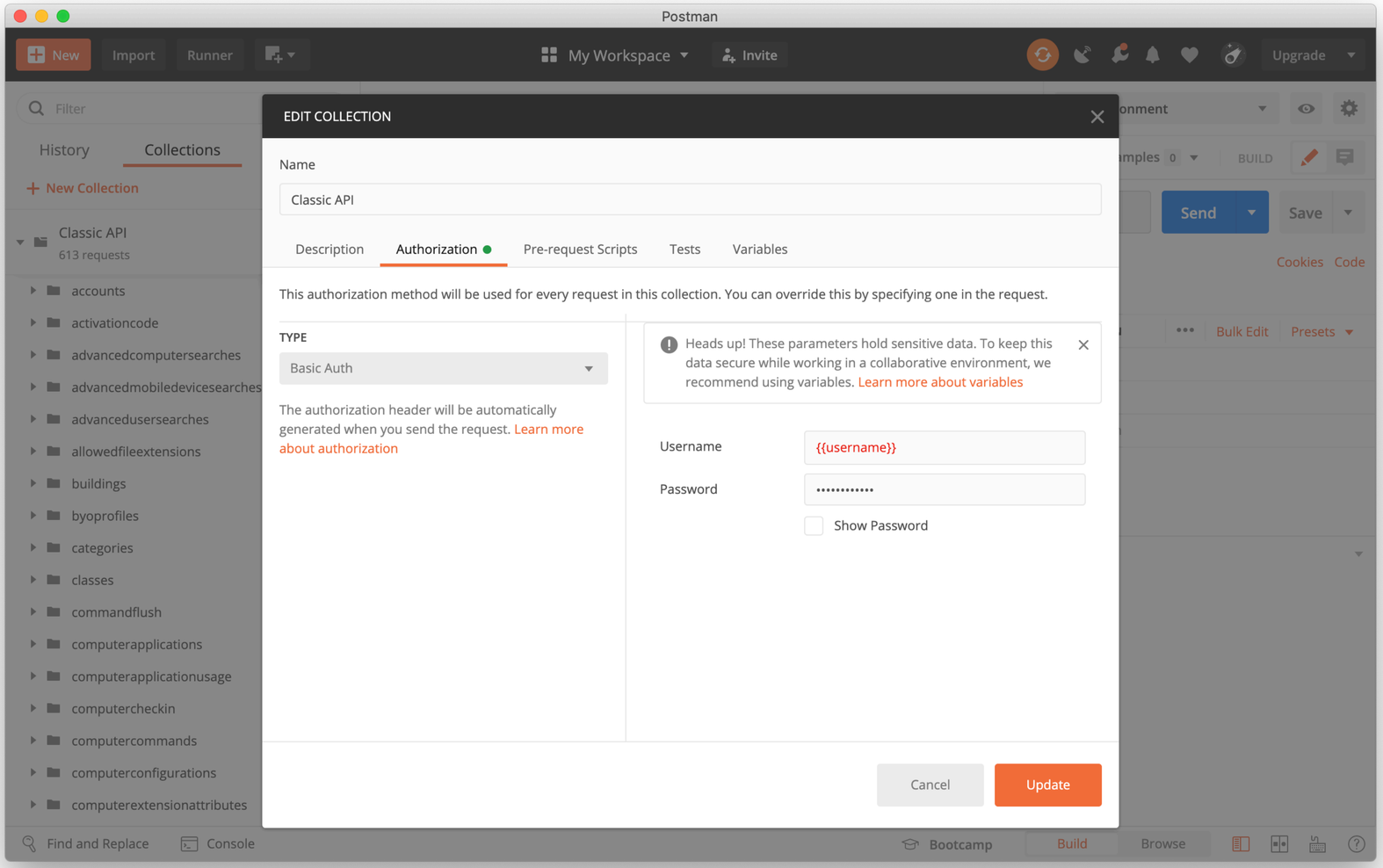
Postman

Configure Globals



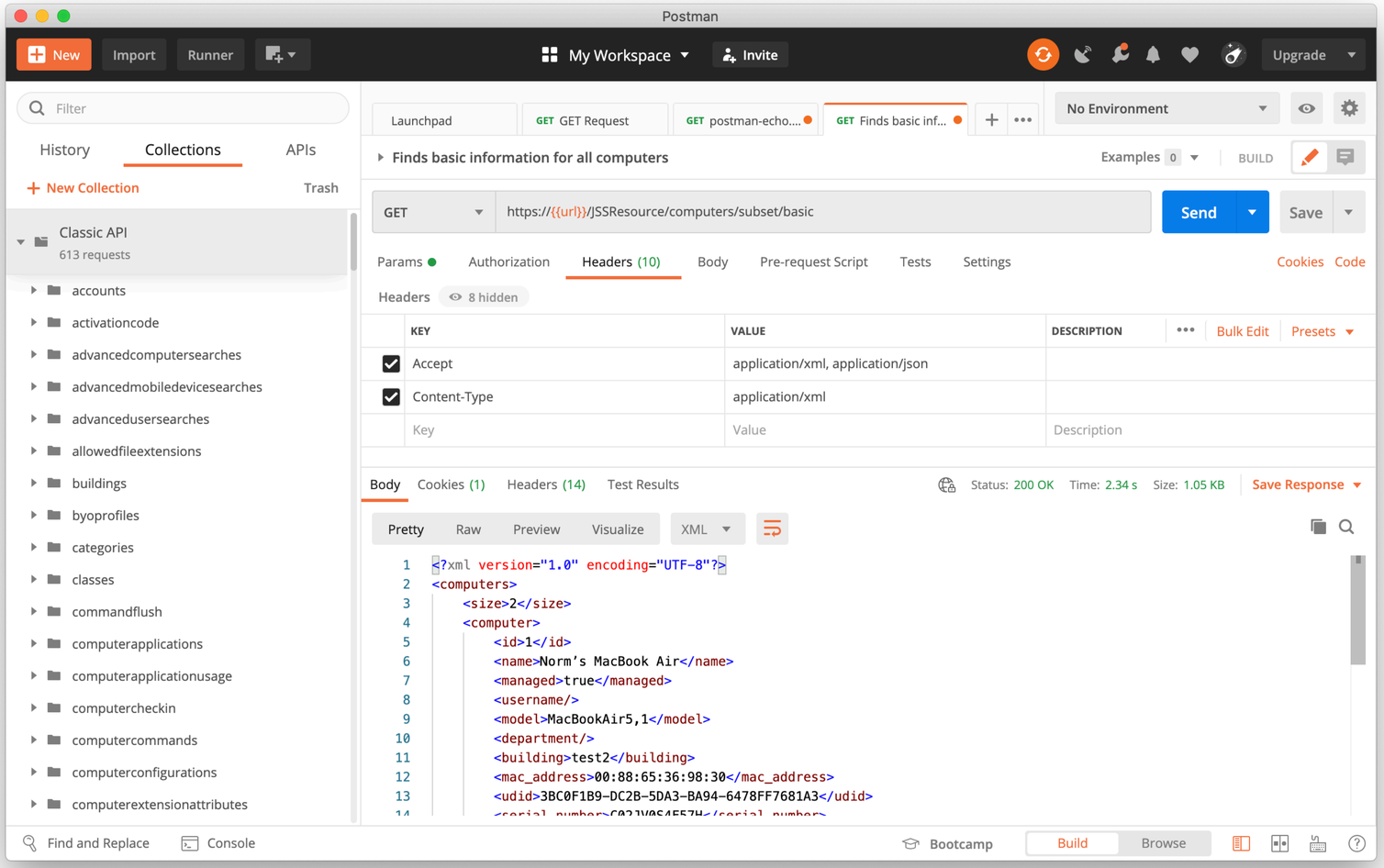
Postman

Configure Authentication



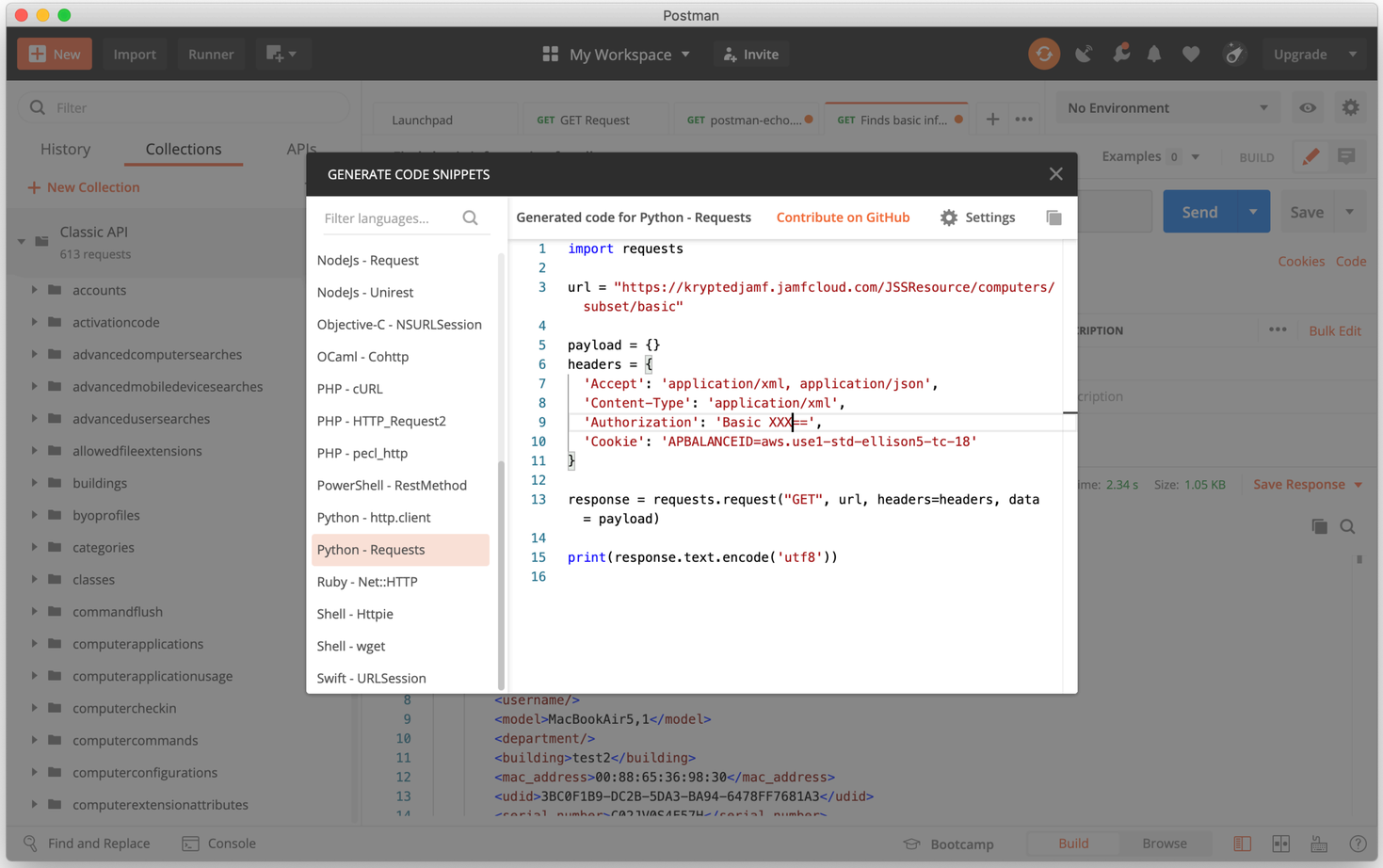
Postman

Send A Request



Postman

Export Code



GraphQL

Query Language for APIs

GraphQL By Example

```
curl -i \
     -H 'Content-Type: application/json' \
     -H "Authorization: bearer myGithubAccessToken" \
     -X POST -d \
        '{"query": \
           "query \
             {repository(owner: \"wso2\", \
             name: \"product-is\") {description}}"}' \
             https://api.github.com/graphql
```

GraphQL Using Graphene

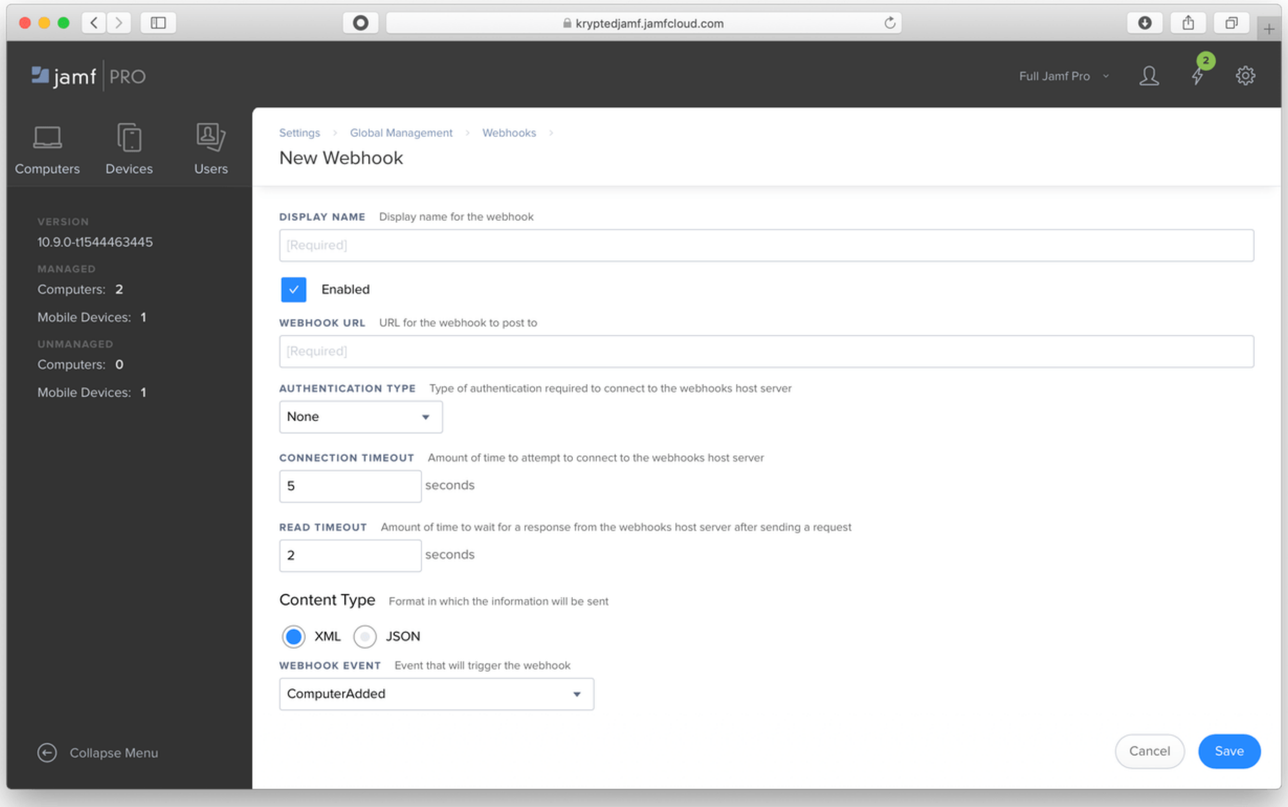
```
import graphene

class Query(graphene.ObjectType):
  hello = graphene.String(name=graphene.String(default_value="World"))

  def resolve_hello(self, info, name):
    return 'Hello ' + name

schema = graphene.Schema(query=Query)
result = schema.execute('{ hello }')
print(result.data['hello']) # "Hello World"
```

https://graphql.org/code/#python

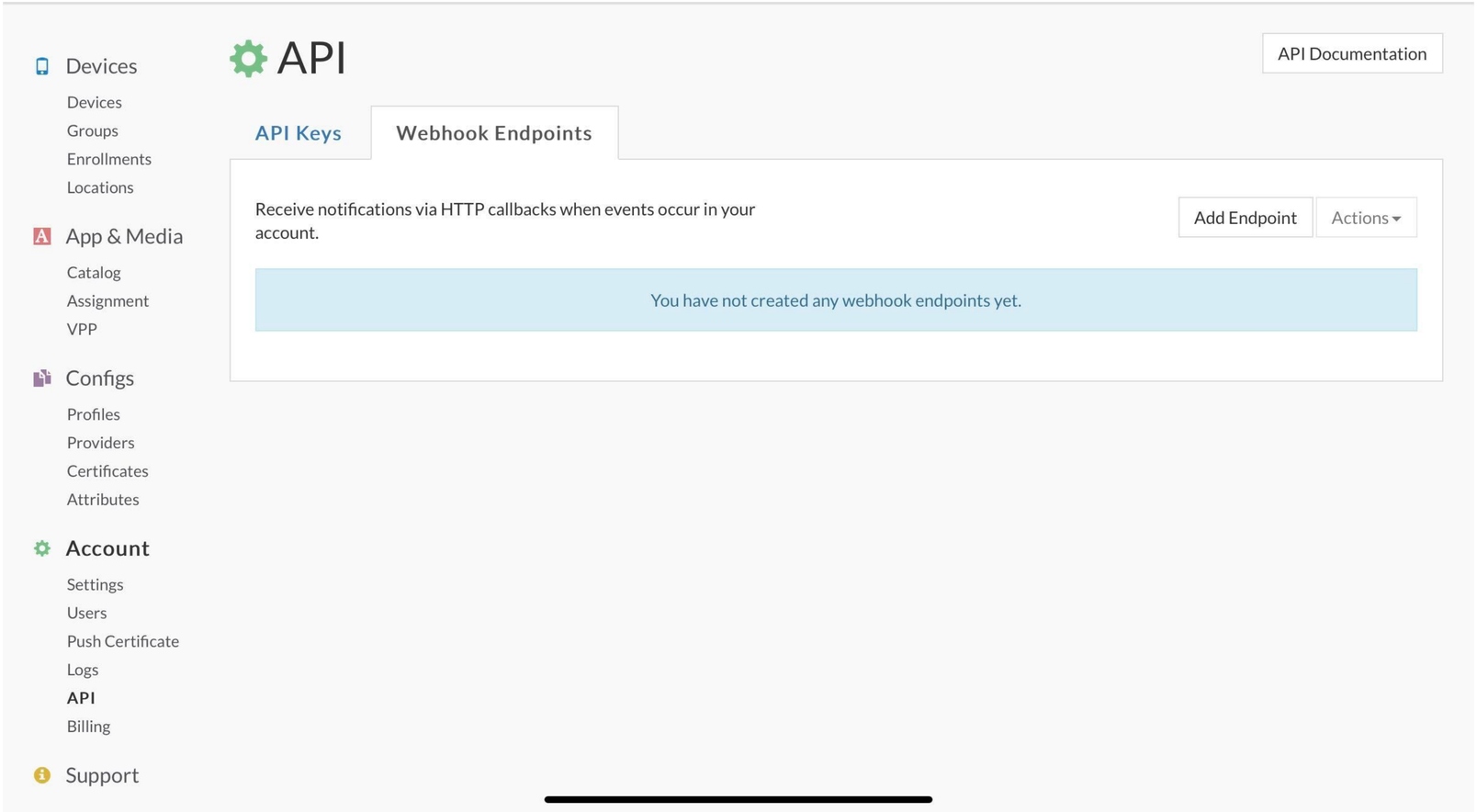
Webhooks

The Triggers of the Internets

Jamf Webhooks

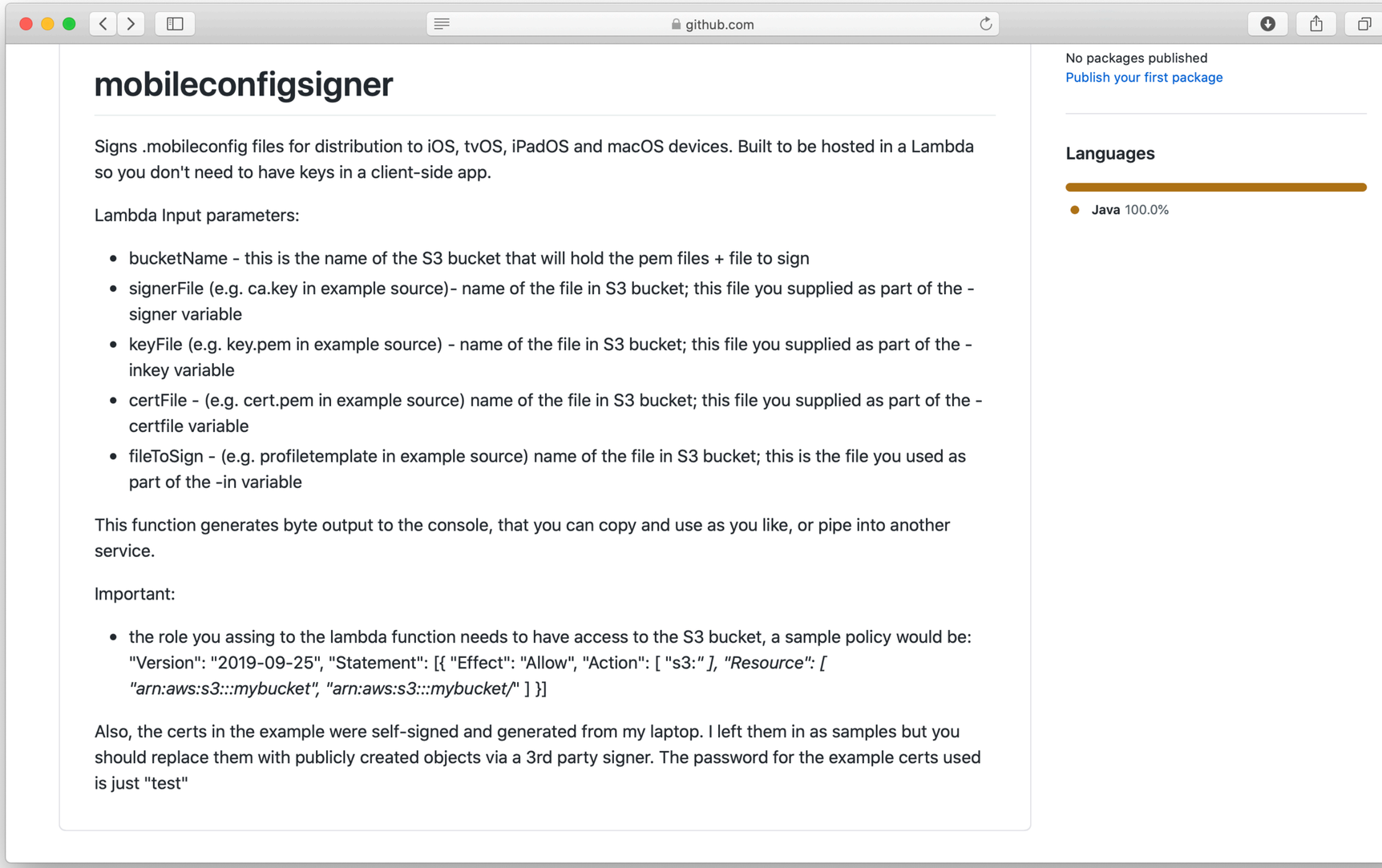


SimpleMDM Webhooks



https://simplemdm.com/docs/api/#webhooks

Lambda



https://github.com/krypted/mobileconfigsigner

IFTTT

krypted

Guides

Commands

Reference

Apple Community

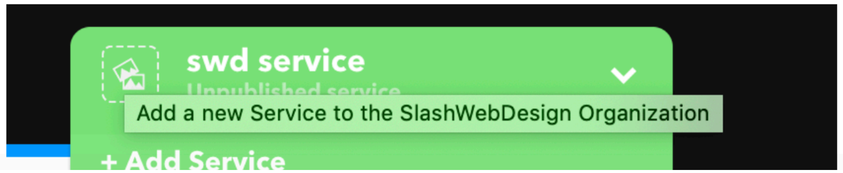
Whoami

Search …

Send Smart Group Changes Information From Jamf To IFTTT

Jamf allows you to register a webhook, IFTTT can trigger an action based on a webhook and then has a number of awesome services you can link the ingredients, or variables, that are caught by that webhook listener into other services. One of the easiest ways to see this kind of action is to just journal the response of a registered webhook into a Google Doc. This allows you to see what's happening on the fly.

Next, you'll need an IFTTT Platform account. This allows you to create new IFTTT services. Then, create a new service, using the Add Service icon in the top nav bar.



https://krypted.com/jamf/send-smart-group-changes-information-from-jamf-to-ifttt/

Zapier

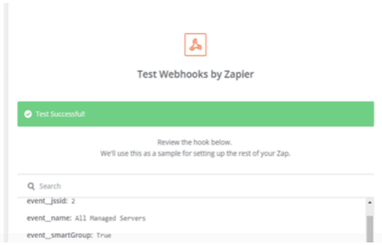
krypted

- Guides
- Commands
- Reference
- Apple Community
- Whoami

Search ...

Catching Smart Group Membership Change Zap (continued)

1. After completing setup of Jamf webhook. Click “Ok, I did this” to begin a test and pull in a test sample. Then return to your Jamf window and trigger the webhook. The webhook can be triggered in two ways.
    1. Trigger the webhook by going to Computers – > Search Inventory -> (Select a computer) -> Edit -> Update site to a new site that is in a different computer group and Save
    2. Trigger the webhook by going to Computers -> Smart Computer Groups – > (Select a Group) -> Update site to a new site that is in a different computer group and Save.
2. If the event is triggered correctly you will see a test result/message like below:



https://krypted.com/mac-os-x/add-jamf-pro-smart-group-google-doc-using-zapier/

Not Just Buzzwords

Other Stuff To Know

- Versioning
- JWT, bearer tokens, and other keys
- OAuth (and OAuth2 and OIDC) and SAML
- SOAP
- Private API
- Swagger (and Swagger Codegen)
- API First

Questions?

Hit me up at krypted@me.com