



apidays
LIVE
JAKARTA
2020

From Code To Community: How Do You Run An Open Source Project?



Uli Hitzel
Developer Advocate
Axway

How do you run an Open Source Project?



1. Take some Source Code
2. Dump in on GitHub
3. Done!

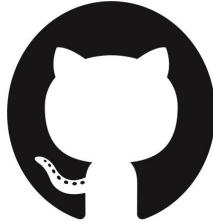




Developer Advocate



/in/uhitzel



/u1i/slides



IBM



11:15 AM

Projects

- [Yoisho Open Banking Project](#)
- [Bambleweeny](#): A lightweight key-value store and message broker based on HTTP/REST
- [keiju](#) - Minimalist API Gateway in a ~25 MB Container.
- [samsa](#) converts between OpenAPI v2 and v3 formats (YAML/JSON)
- [majime](#) – Dead Simple API Unit Testing

Speaking Engagements – Slides

- [Redis Meetup Singapore](#) (April 2018)
- [PyCon Bangkok](#) (June 2018)
- [Stripe - API for Humans](#) (March 2019)
- [RedisConf San Francisco](#) (April 2019)

<https://apigeek.net>



majime



bamble
weeny

- Run test cases on API endpoints
- Test cases are defined in YAML files
- majime can generate test cases from Swagger specs

The logo for majime features the word "majime" in a large, black, cursive font. The letters are interconnected, with the 'm' having a long tail that loops back to form the 'a', 'j', and 'i'. The 'e' has a short tail extending downwards.

/ma-ji-me/

noun

1. an earnest, reliable person who can get things done
2. dead simple API Unit Testing

python 3.7

license MIT

issues 0 open

Run Test Cases against Endpoints

```
PUT http://backend.yoisho.dob.jp/banking/v2/atm/5
HTTP 200
DELETE http://backend.yoisho.dob.jp/banking/v2/atm/3
HTTP 200 but expected 204
GET http://backend.yoisho.dob.jp/banking/v2/atm/1
HTTP 200
POST http://backend.yoisho.dob.jp/banking/v2/atm
HTTP 201
GET http://backend.yoisho.dob.jp/banking/v2/atm
HTTP 200
5 tests, 4 successful and 1 failed
$ █
```

JSON Output Also Can!

```
/mj # majime -j -f ATM_Locations-6971.yaml
{"tests": [{"method": "PUT", "url": "http://backend.yoisho.dob.jp/banking/v2/atm/{id}", "http_response": "404", "http_expected_response": "200", "result": "FAIL"}, {"method": "DELETE", "url": "http://backend.yoisho.dob.jp/banking/v2/atm/{id}", "http_response": "404", "http_expected_response": "204", "result": "FAIL"}, {"method": "GET", "url": "http://backend.yoisho.dob.jp/banking/v2/atm/{id}", "http_response": "404", "http_expected_response": "200", "result": "FAIL"}, {"method": "POST", "url": "http://backend.yoisho.dob.jp/banking/v2/atm", "http_response": "201", "result": "OK"}, {"method": "GET", "url": "http://backend.yoisho.dob.jp/banking/v2/atm", "http_response": "200", "result": "OK"}], "output": {"overall_result": "FAIL", "successful-tests": "2", "failed-tests": "3"}}
/mj #
```

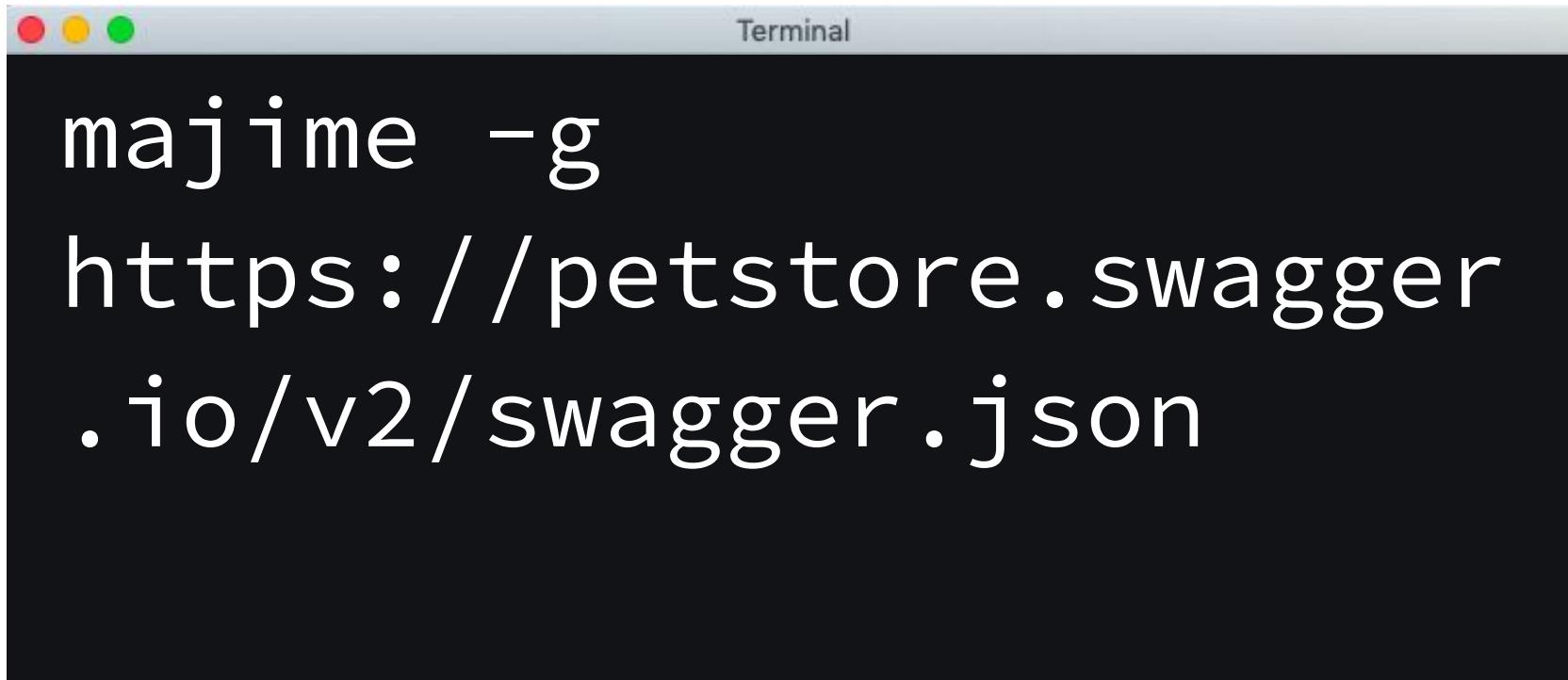
Test Cases Defined In YAML

Base: 'http://backend.yoisho.dob.jp/banking/v2'

Tests:

- path: '/atm/{id}'
method: GET
headers: ''
expect-response: '200'
expect-body: json
- path: '/atm/{id}'
method: PUT
headers: ''
content-type: application/json
body: {}
expect-response: '200'

Generate Test Cases from Swagger

A screenshot of a Mac OS X Terminal window. The window has the standard title bar with red, yellow, and green buttons on the left and the word "Terminal" centered. The main area of the window is a dark gray color where the command is displayed in white text.

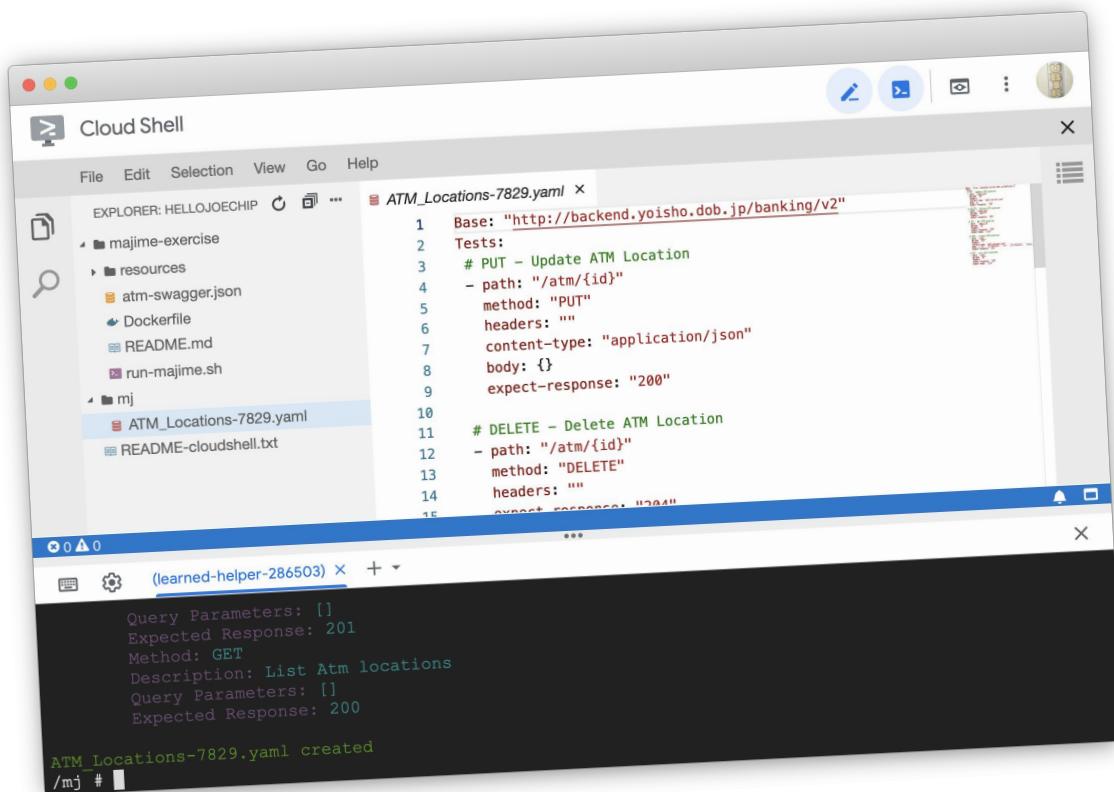
```
majime -g  
https://petstore.swagger.  
.io/v2/swagger.json
```

Generate Test Cases from Swagger

```
Terminal
Generate test suite from https://petstore.swagger.io/v2/swagger.json
Title: Swagger Petstore
Host: petstore.swagger.io
Base Path: /v2
Scheme: https
Path: /pet/{petId}/uploadImage
    Method: POST
    Description: uploads an image
    Query Parameters: []
    Expected Response: 200
Path: /pet
    Method: POST
    Description: Add a new pet to the store
    Query Parameters: []
    Expected Response: 405
    Method: PUT
    Description: Update an existing pet
```

<https://github.com/u1i/majime-exercise>

Let's Try It Out!



The screenshot shows a Cloud Shell interface with two main panes. The left pane is an Explorer view showing a directory structure for a project named 'majime-exercise'. The right pane contains a code editor and a terminal window.

Code Editor:

```
Cloud Shell
File Edit Selection View Go Help
EXPLORER: HELLOJOECHIP
majime-exercise
  resources
    atm-swagger.json
  Dockerfile
  README.md
  run-majime.sh
  mj
    ATM_Locations-7829.yaml
  README-cloudshell.txt

ATM_Locations-7829.yaml ×
1 Base: "http://backend.yoisho.dob.jp/banking/v2"
2 Tests:
3 # PUT - Update ATM Location
4 - path: "/atm/{id}"
5   method: "PUT"
6   headers: ""
7   content-type: "application/json"
8   body: {}
9   expect-response: "200"
10
11 # DELETE - Delete ATM Location
12 - path: "/atm/{id}"
13   method: "DELETE"
14   headers: ""
15   expect-response: "204"
```

Terminal:

```
(learned-helper-286503) × + 
Query Parameters: []
Expected Response: 201
Method: GET
Description: List Atm locations
Query Parameters: []
Expected Response: 200

ATM_Locations-7829.yaml created
/mj #
```

Open Banking - Live Endpoints

Webservices and REST APIs that expose bank related data services with dynamic content, use them for testing and demos. Available over [HTTP](#) and [HTTPS](#)

[Full Documentation](#)



ATM

Full CRUDL REST interface (Create, Read, Update, Delete, List), /v1 and /v2 endpoints with respective Swagger specs

[API Version 1 - Swagger](#)

[API Version 1 - Swagger Editor](#)

[API Version 2 - Swagger](#)

[API Version 2 - Swagger Editor](#)

[Sample Request](#)

Currency Exchange

REST/JSON, 1 parameter. Response is different each time.

[Swagger](#)

[Swagger Editor](#)

[Sample Request](#)

Assets (SOAP)

Bank Assets & Debt- SOAP/XML, 2 methods, dynamic output.

[WSDL](#)

ATM Locations 2.0

[Base URL: backend.yoisho.dob.jp/banking/v2]

<https://backend.yoisho.dob.jp/banking/v2/swagger>

List of ATM locations for Yoisho Banking Corporation

PUT `/atm/{id}` Update ATM Location

DELETE `/atm/{id}` Delete ATM Location

GET `/atm/{id}` Get ATM Location

POST `/atm` Create ATM Location

GET `/atm` List Atm locations

How do you run an Open Source Project?



Open Source



Community
powered
Innovation

Open Source

People will **see** my
Code

People will **use** my
Stuff





Contribute

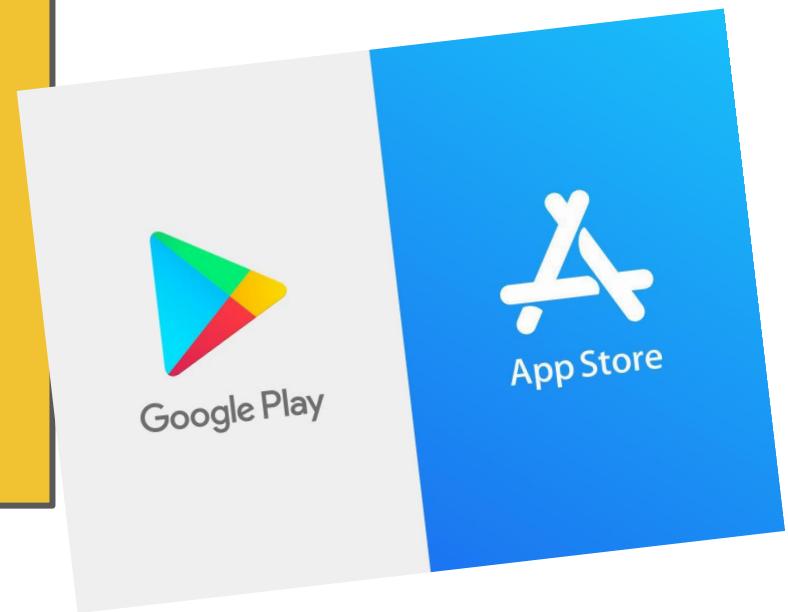
- Code
- Documentation & Translation
- Using it
 - Opening Tickets
 - Spreading the Word

Why do it?

- You get inspired & grow
- Make new connections
- Give back to the
Community
- It's rewarding & fun



You're on the 'App Store'!



Your RedisConf19 session "Bambleweeny: Redis with HTTP & OAuth" has been accepted!

ⓘ You replied on Mon 2/25/2019 18:06

DN

Dave Nielsen <dave@redislabs.com>

Sat 2/23/2019 01:55



Uli Hitzel ▾

Dear Uli,

Thank you for submitting your talk to RedisConf19.

On behalf of the Speaker Selection Committee, I'm delighted to let you know that your talk has been accepted and we'd be honored to have you speak at RedisConf.



How do you run an Open Source Project?



You'll have to be:

A Developer
+ Marketing Person
+ Project Manager



I'm probably here



Solve a real Problem

1



Build something
you're using
yourself

Turn it into a 'Living Thing'

- GitHub
- Markdown
- Issues, Versions, Releases
- Contributions & Pull Requests

2





Market it!

3

- Branding: Naming, Logo
- Superb Documentation
- Great 'First 20 Minutes' Experience
- Promote it



bambleweeny

Bambleweeny is lightweight key-value store and message broker based on HTTP/REST.



19



6

python

2.7

release

v0.36

Docker Pulls

2497

issues

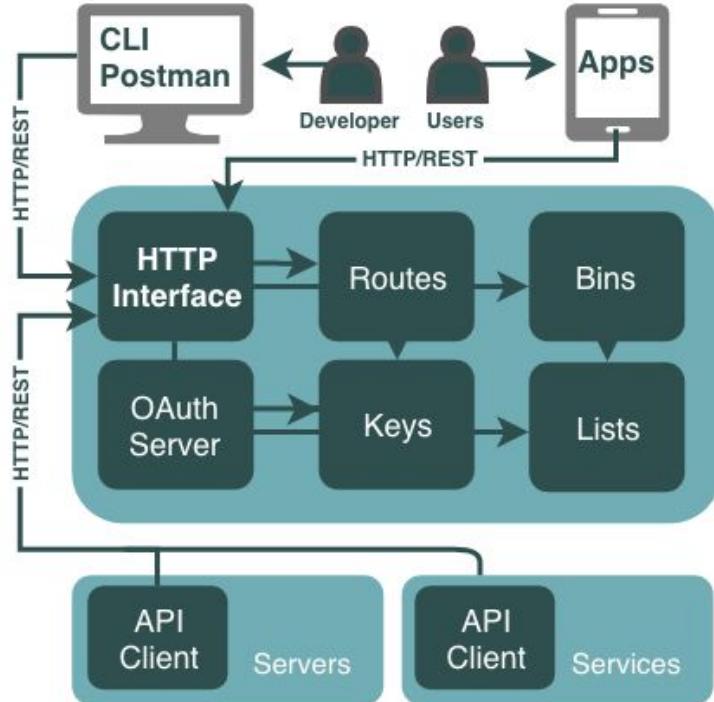
5 open



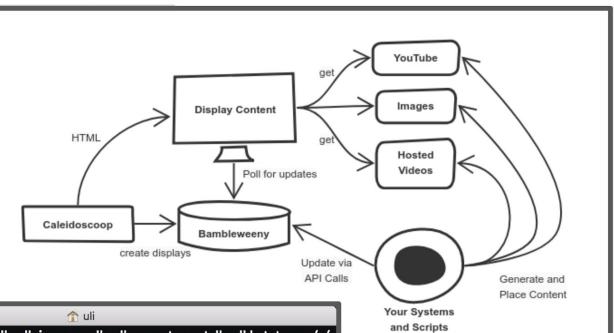
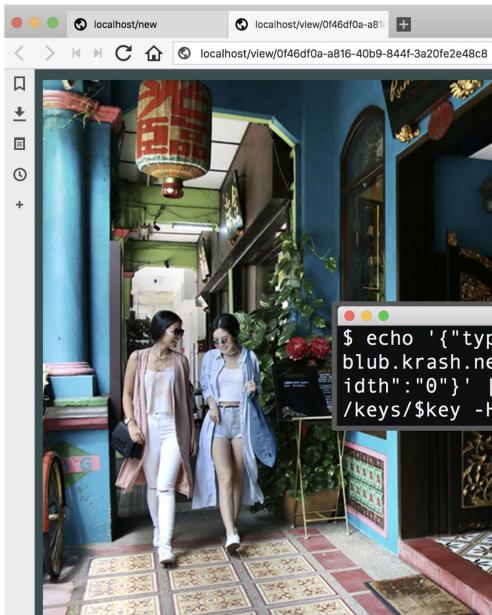
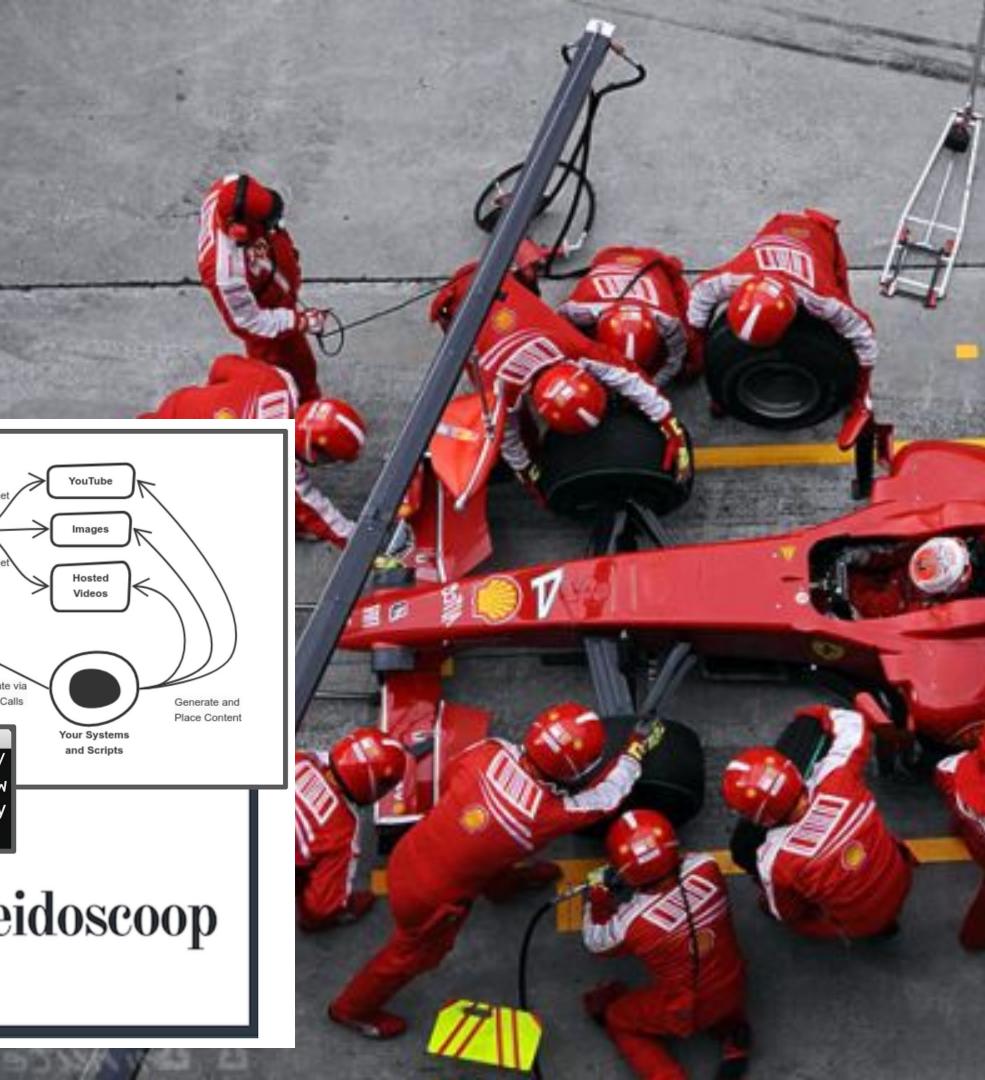
bamble
weeny

Bambleweeny

- Lightweight HTTP/REST based Key-Value Store & Message Broker
- Multi-Tenancy → Manage Identities, Access & Quotas
- Fast, Easy to Use & Well-Documented
- Written in Python, using Redis, deployable in a tiny Container



Use Case: Fast Prototyping

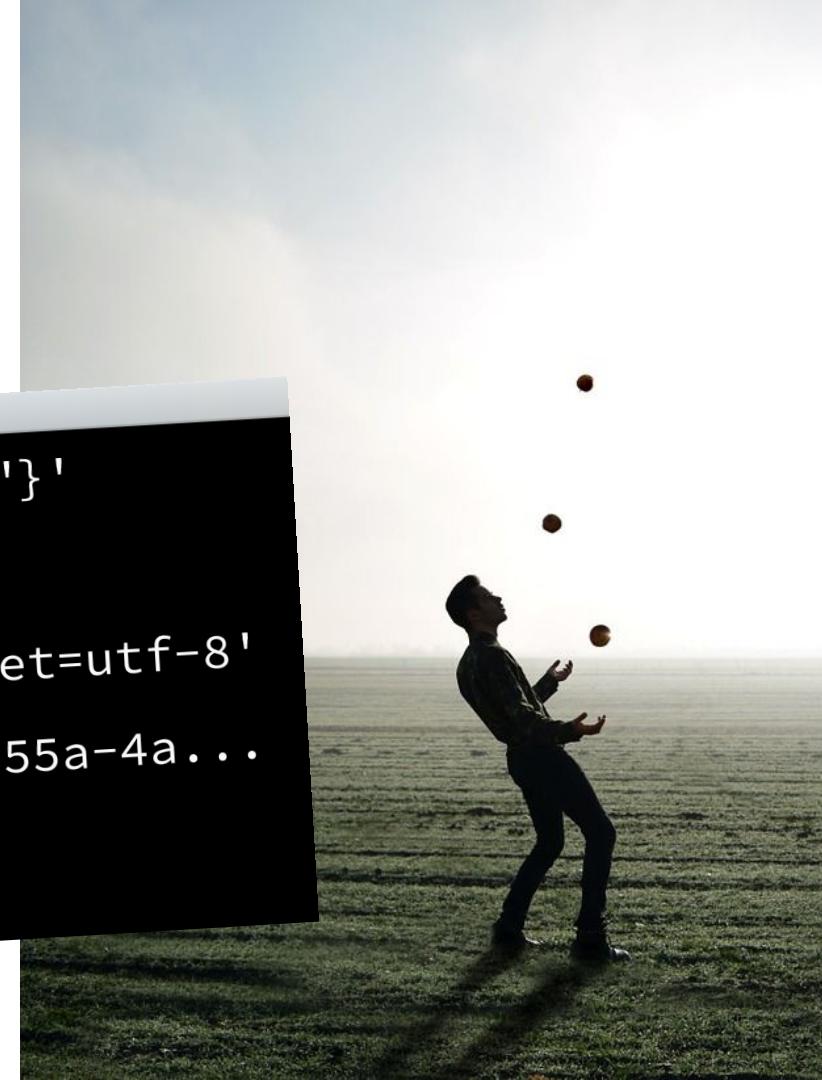


```
$ echo '{"type": "image", "content": "http://blub.krash.net/katong.jpg", "height": "0", "width": "0"}' | curl -X PUT -d @- http://b9y/keys/$key -H $AUTH
```

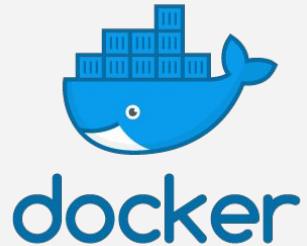
caleidoscoop

Use Case: POCs & API Mocking

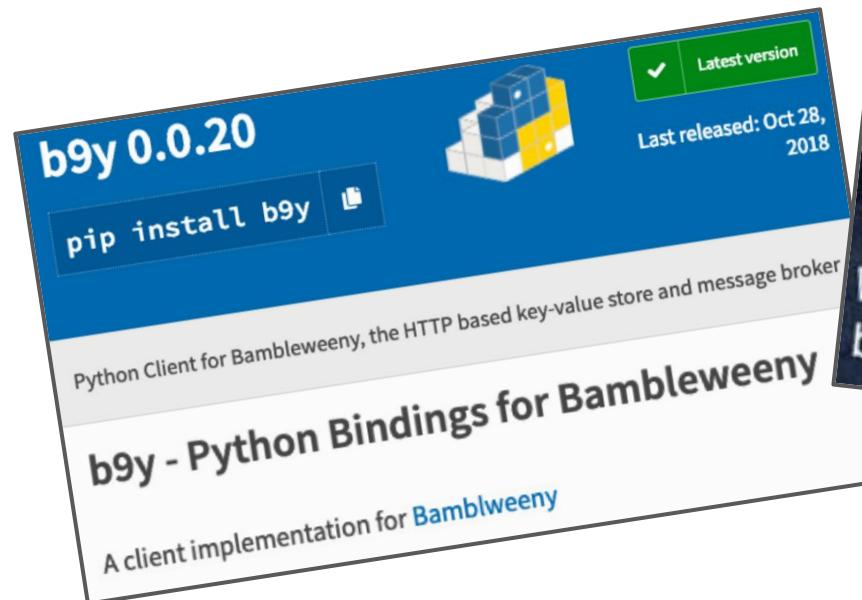
```
Terminal
set api '{"message": "!@[message]"}'
set message 'cool stuff!'
route api 'application/json;charset=utf-8'
curl http://b9y/routes/cf670f2b-755a-4a...
{"message": "cool stuff!"}
```



Easy to Run

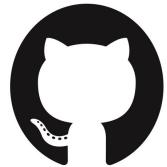


Python Client Library & CLI



Get involved

- Try out Majime or Bambleweeny!
- Help make it better → open GitHub issues for bugs & new features
- Spread the Word!



/u1i

