

# Bambleweeny

Redis with  
HTTP & OAuth



Uli Hitzel  
Developer Advocate  
Axway



redisconf19



# Developer Advocate



/in/uhitzel



/u1i/slides



# Singapore Meetup



- **Talks** from real practitioners
- **Exciting venues**
- **Connect** with peers in the real world



**redis.sg**

# Agenda

1 – What does it do?

2 – Use Cases

3 – Get involved!



# bamblweeny

release v0.31 Docker Pulls 1451 license MIT issues 4 open

GET /keys/{id} Read Key



PUT /keys/{id} Write Key



DELETE /keys/{id} Delete Key

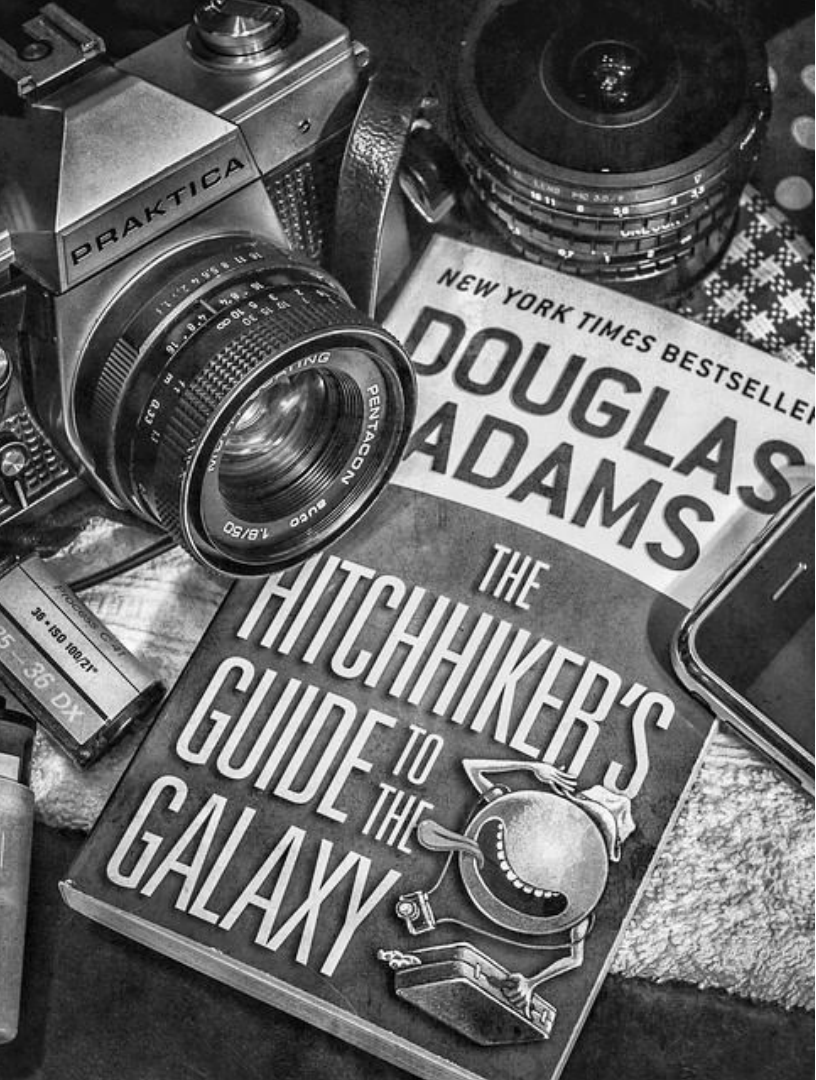


GET /keys/ List Keys



GET /incr/{id} Increase Key

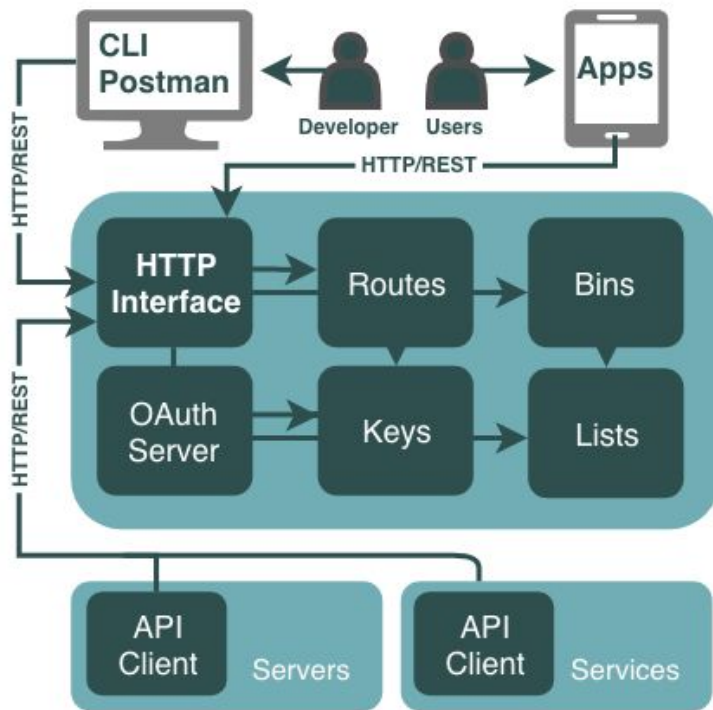




The Bambleweeny  
57 Submeson  
Brain is part of the  
Finite  
Improbability  
Generator.

# 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



# In Simple Terms

Client

HTTP(s)



**B9y**

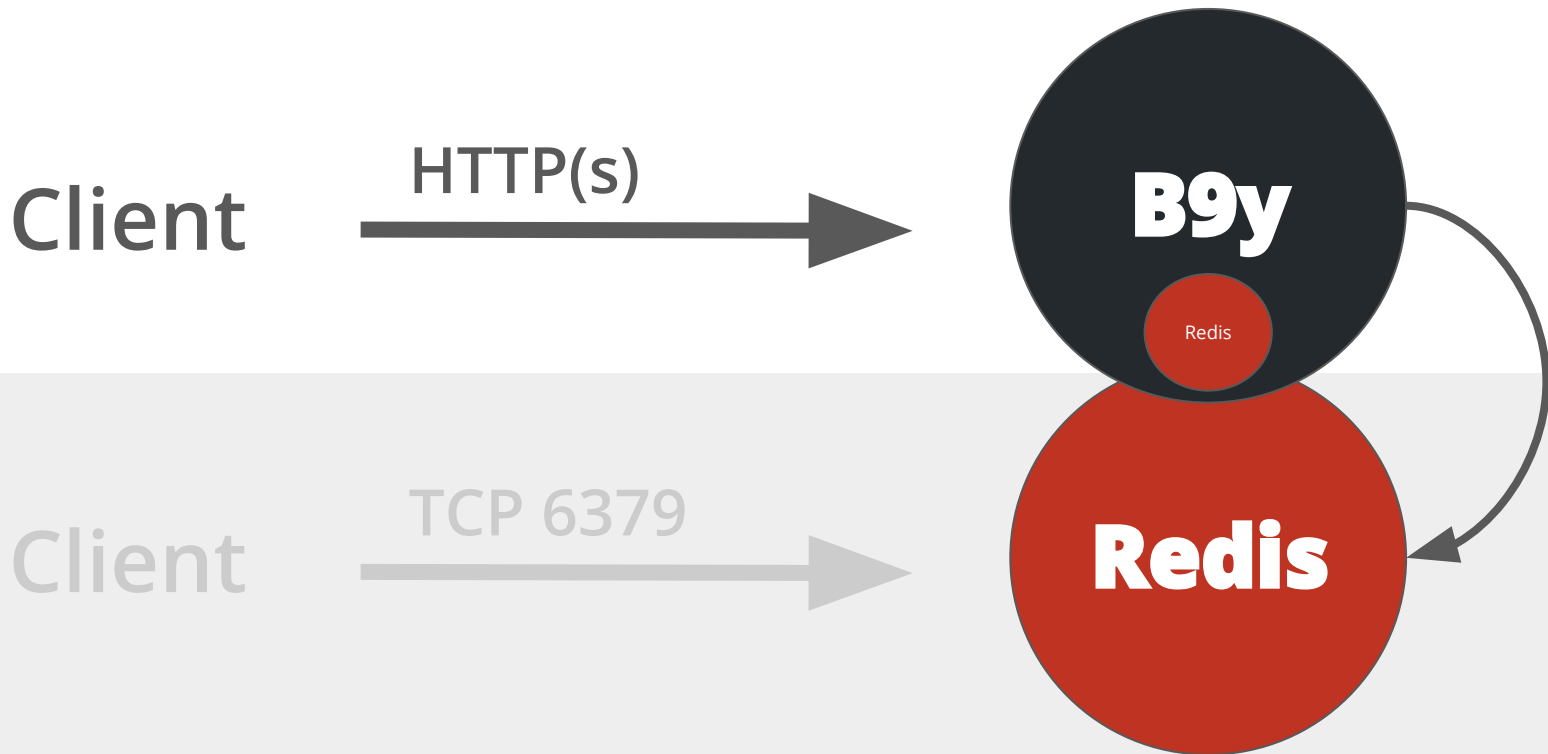
Client

TCP 6379

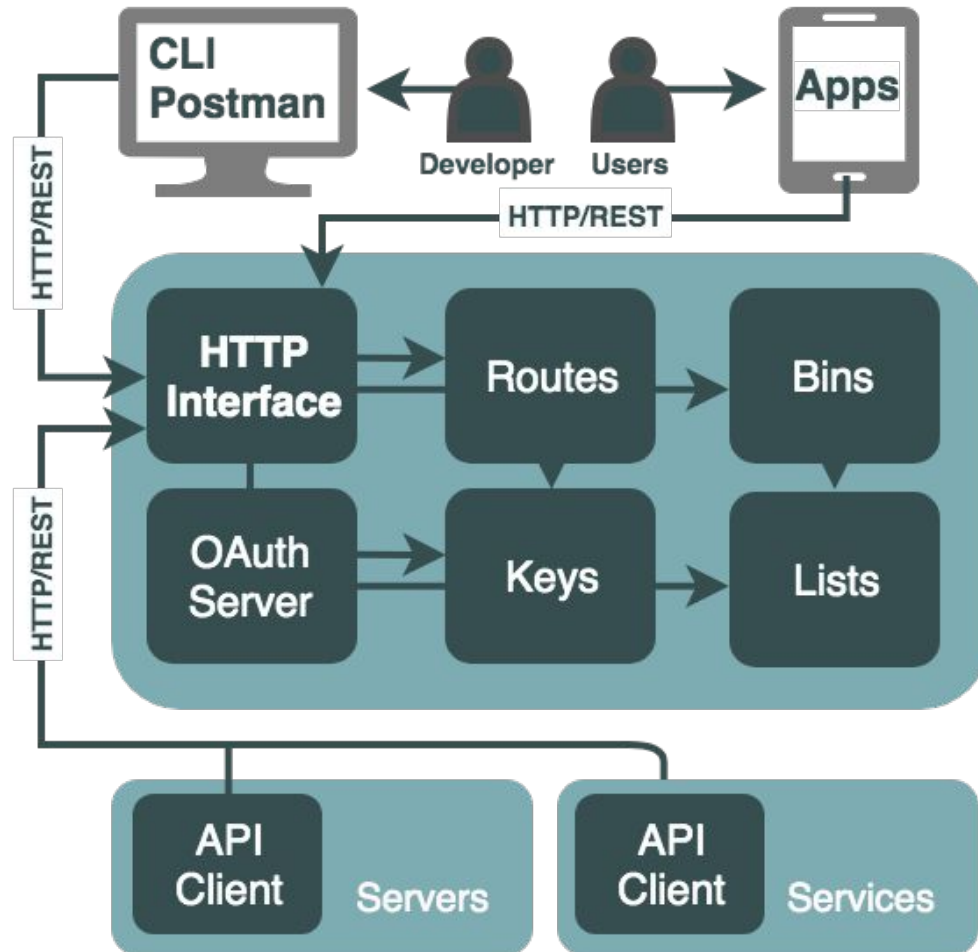


**Redis**

# Self Contained or External Redis

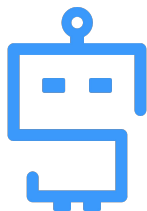






# REST API

- CRUD Interfaces for Keys & Lists
- Design First, Code Later
- API Specs at /swagger
- Use to render Documentation & SDKs



## Keys

**GET** /keys/{id} Read Key

**PUT** /keys/{id} Write Key

**DELETE** /keys/{id} Delete Key

**GET** /keys/ List Keys

**GET** /incr/{id} Increase Key

## Routes

**GET** /routes/{id} Read Key

**POST** /routes Create Route

## Lists

**POST** /lists/{id} Add Item to List

**GET** /lists/{id} Get Item from List

**DELETE** /lists/{id} Delete List

**GET** /lists Get All Lists



▼ bambleweeny  
22 requests

► config

► routes

► bins

► auth

▼ keys

GET Read Key

PUT Write Key

DEL Delete Key

GET List Keys

► save

► info

► incr

► lists

► users

POST Get Access Token

PUT Write Key

No Environment

► Write Key

Examples (0)

PUT

http://localhost:8080/keys/redisconf19

Send

Save

Params

Authorization

Headers (3)

Body

Pre-request Script

Tests

Cookies

Code

Comments (0)

none

form-data

x-www-form-urlencoded

raw

binary

JSON (application/json)

Beautify

1 hello world!

Body

Cookies (4)

Headers (5)

Test Results

Status: 200 OK

Time: 10 ms

Size: 170 B

Save

Download

Pretty

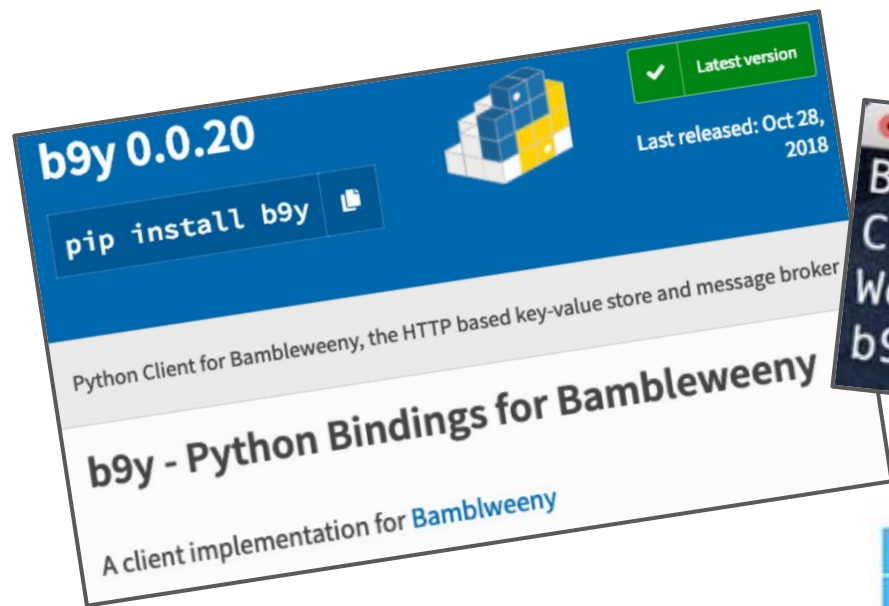
Raw

Preview

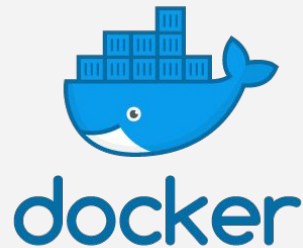
JSON

```
1 {  
2   "info": "ok"  
3 }
```

# Python Client Library & CLI



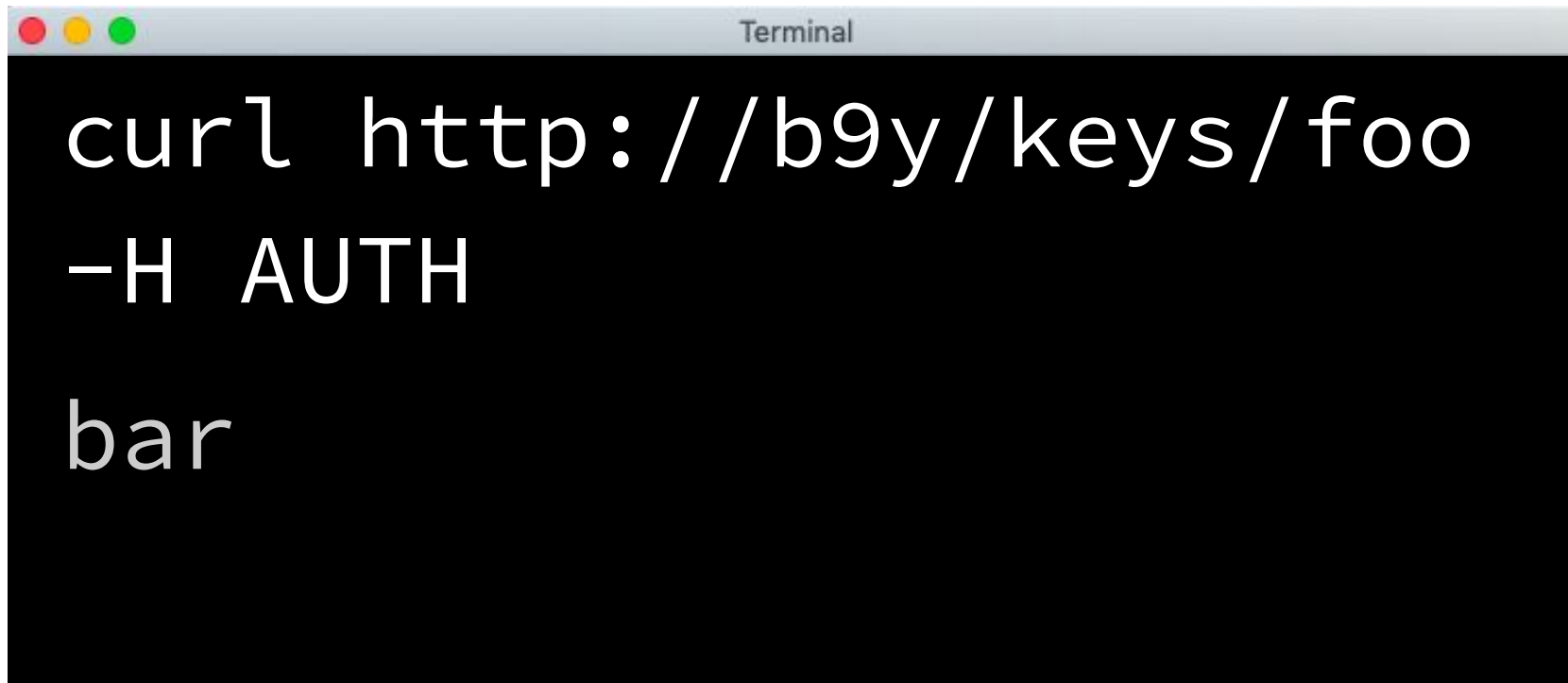
# Easy to Run



```
Terminal
docker run -d -P
u1ih/bamblweeny
```

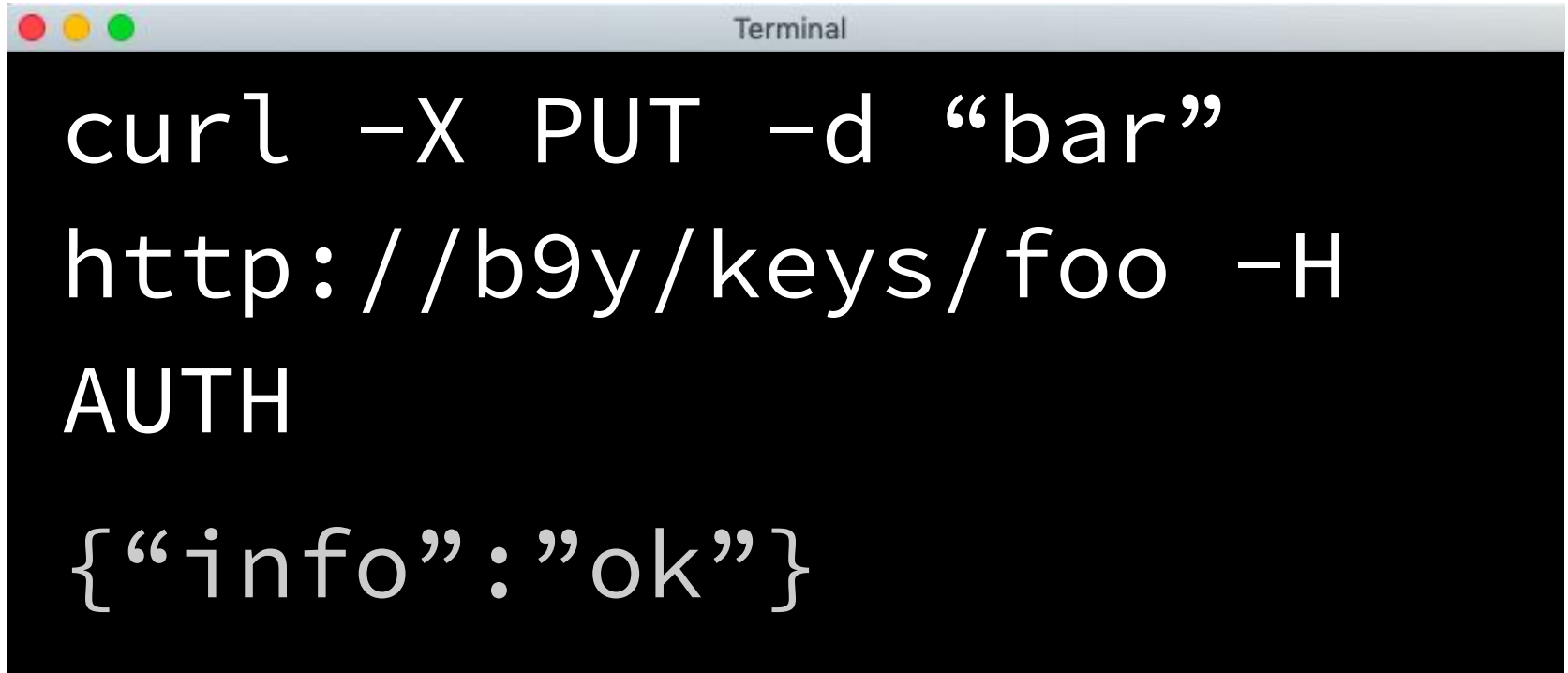


# cURL: Get 'foo'

A screenshot of a macOS Terminal window. The window has a title bar with three colored window control buttons (red, yellow, green) on the left and the word "Terminal" in the center. The background of the terminal is black, and the text is white. The command entered is `curl http://b9y/keys/foo` on the first line, `-H AUTH` on the second line, and `bar` on the third line.

```
Terminal  
curl http://b9y/keys/foo  
-H AUTH  
bar
```

# cURL: Set 'foo' = 'bar'

A terminal window with a title bar labeled "Terminal" and three colored window control buttons (red, yellow, green) on the left. The terminal has a black background with white text. It displays a cURL command to perform a PUT request and the resulting JSON response.

```
curl -X PUT -d "bar"  
http://b9y/keys/foo -H  
AUTH  
{ "info": "ok" }
```

# cURL: Allow public access to 'foo'



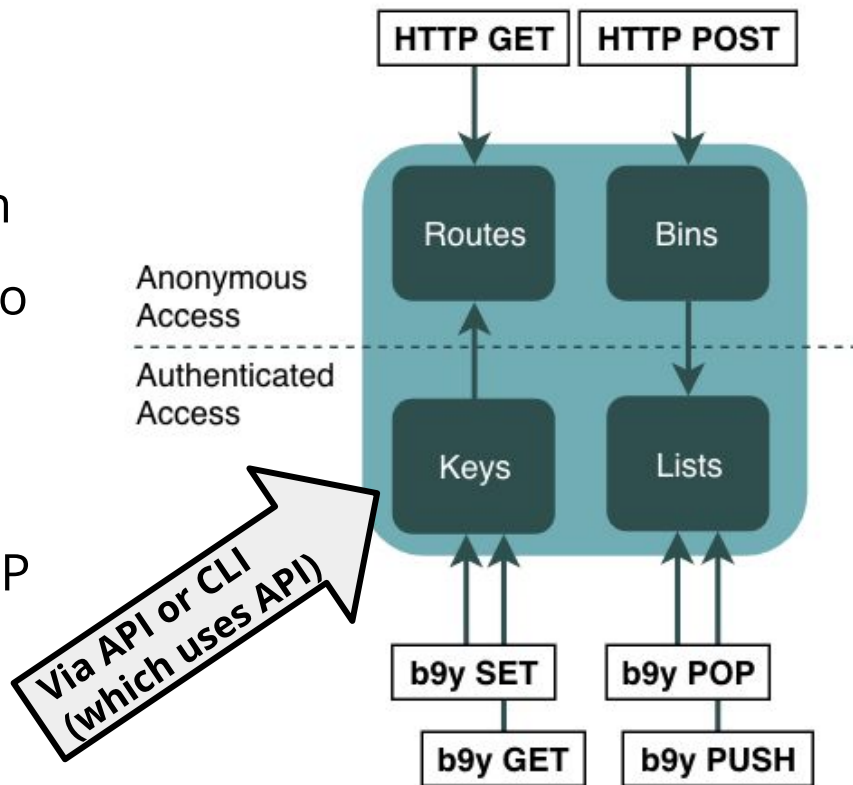
```
Terminal
echo '{"key":"foo",
"content_type":"text/plain"}' | curl -X
POST -d @- http://b9y/routes -H AUTH
/routes/125e6a6f-c3f3-403b-b096-8997871
curl http://b9y/routes/125e6a6f-c3f3-403
bar
```

No Auth!



# OAuth & Access Concepts

- Keys & Lists are Private
- Bearer Token via /auth/token
- Routes: Public READ Access to Keys via plain HTTP
- Bins: Public WRITE Access (“push”) to Lists via plain HTTP



# HTTP(s) is expensive

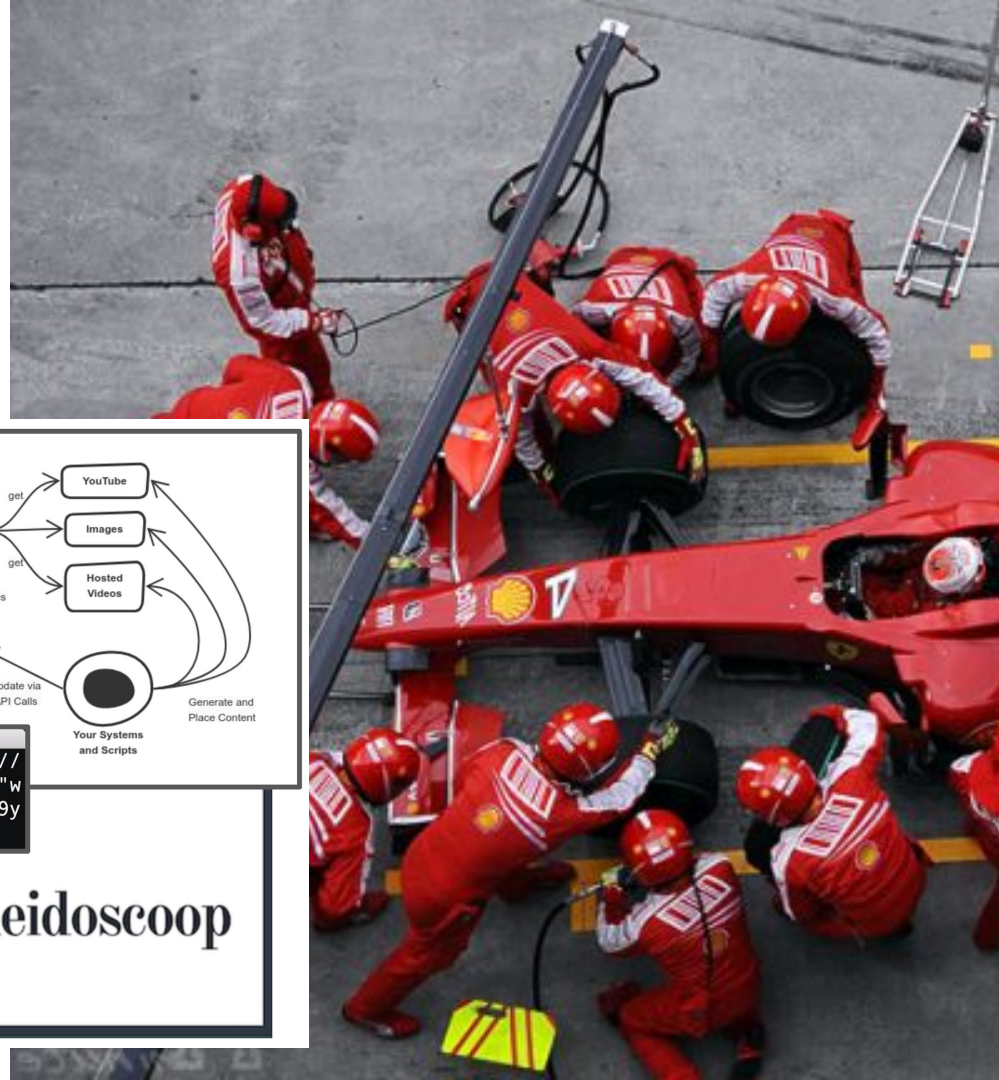
1x vCPU 1 GB RAM (AWS t2.micro)	Reads / second	Writes / second
<b>Redis 4.01</b>	<b>56k</b>	<b>55k</b>
<b>Bambleweeny</b>	<b>540</b>	<b>400</b>






Kubernetes  
Microservices  
Kafka  
Elastic  
Zookeeper  
MongoDB

# Use Case: Fast Prototyping



localhost/new    localhost/view/0f46df0a-a816-40b9-844f-3a20fe2e48c8

localhost/view/0f46df0a-a816-40b9-844f-3a20fe2e48c8



```
$ 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

Diagram illustrating the system architecture:

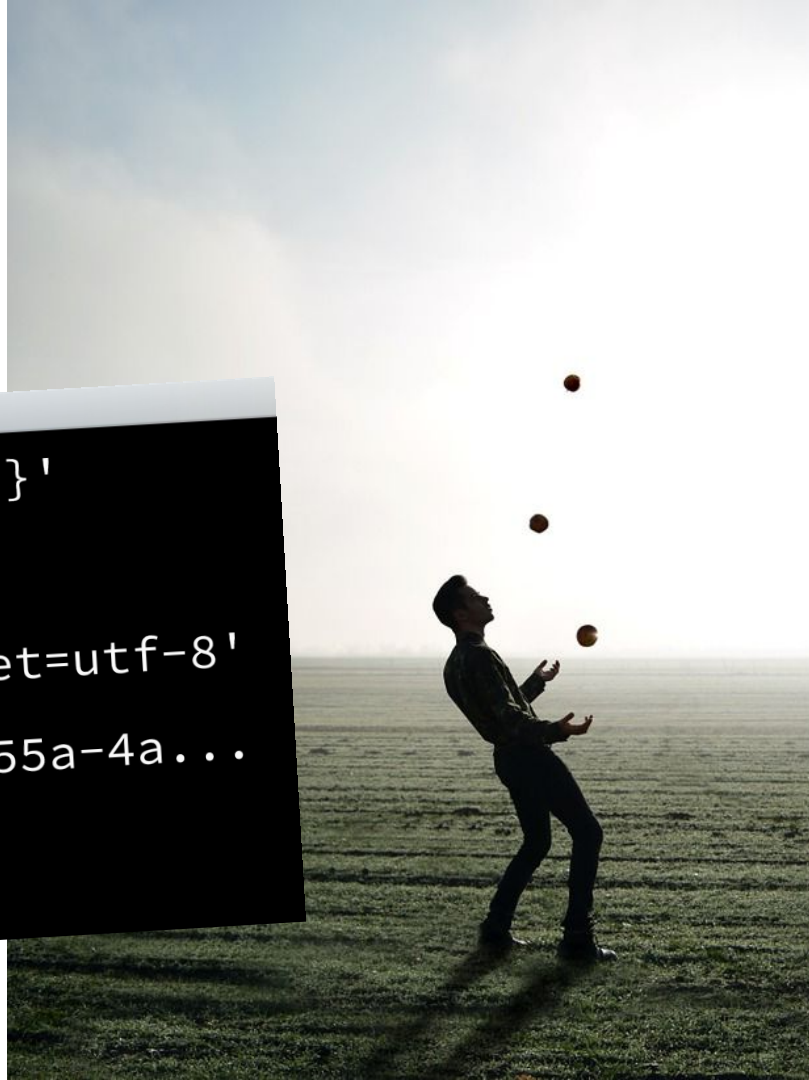
- Display Content** (Box) receives **HTML** input and sends **get** requests to **YouTube**, **Images**, and **Hosted Videos** (Boxes).
- Display Content** sends **Poll for updates** to **Bambleweeny** (Box).
- Caleidoscoop** (Box) sends **create displays** to **Bambleweeny**.
- Bambleweeny** sends **Update via API Calls** to **Your Systems and Scripts** (Circle).
- Your Systems and Scripts** sends **Generate and Place Content** to **YouTube**, **Images**, and **Hosted Videos**.



# 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!"}
```



# Get involved

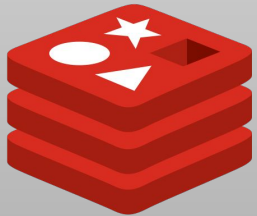
- Try it out!
- Help make it better → open GitHub issues for bugs & new features
- Spread the Word!



/u1i/bambleweeny



# Bambleweeny | April 2019



redis

