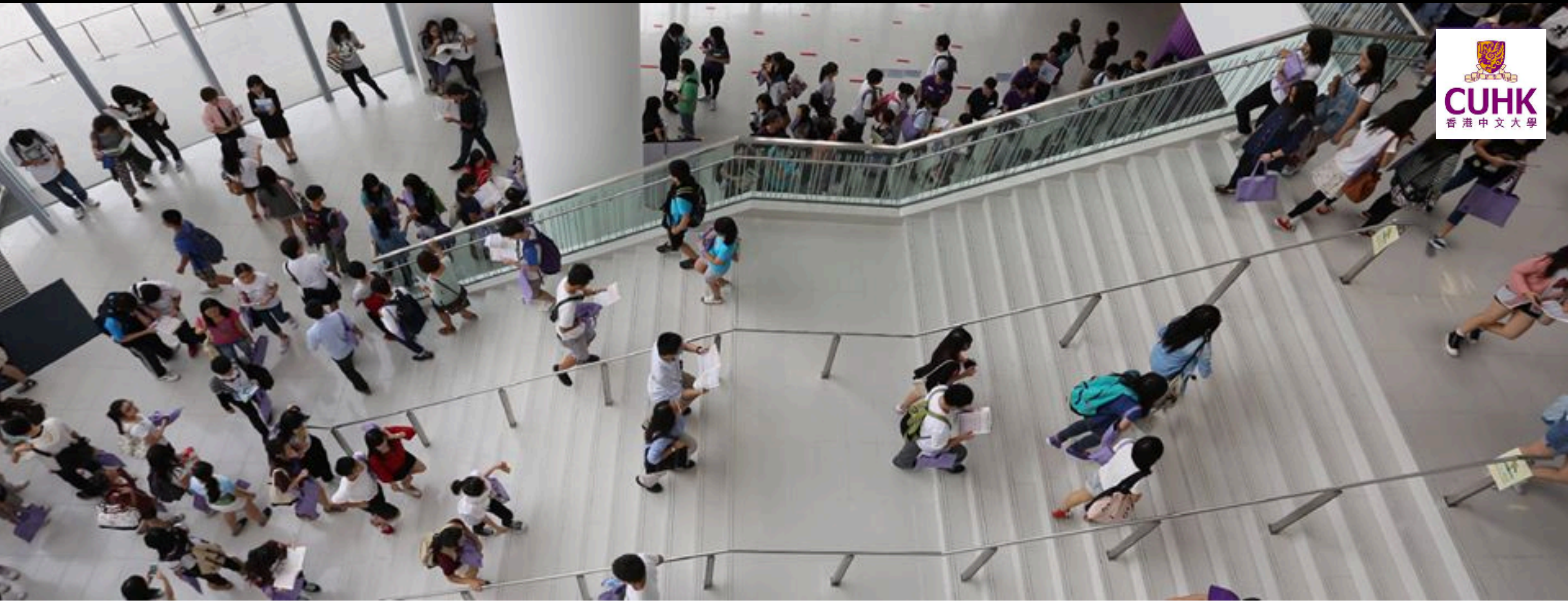


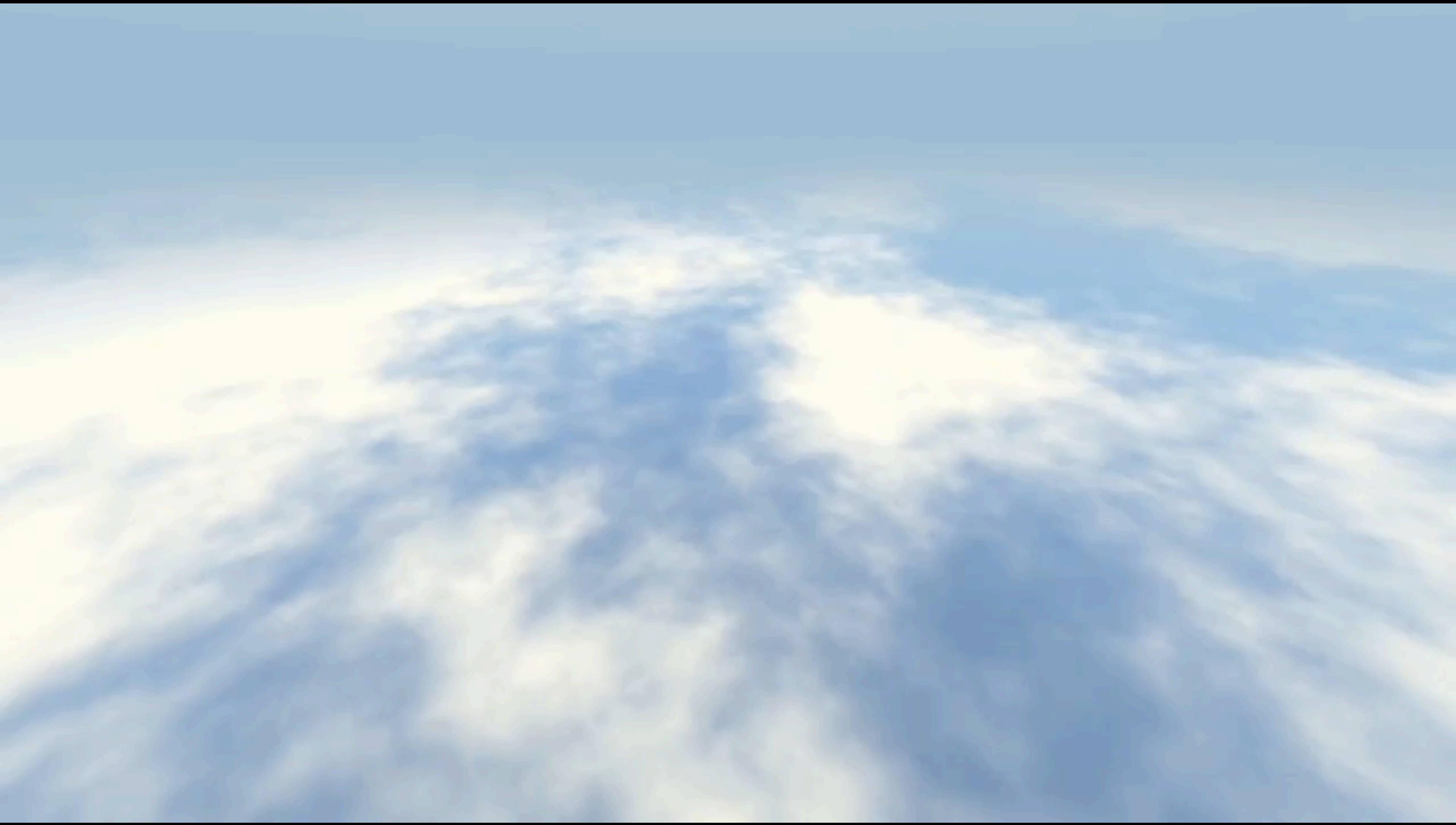
COM5940 NEW MEDIA BUSINESS MODEL & INNOVATION:
LESSON 1 - REST API, AUTHENTICATION & AUTHORIZATION

Bernard Suen
Center for Entrepreneurship
Chinese University of Hong Kong

Today's **agenda**.

1. Introduction to REST API and its related CRUD operations using Postman.
2. Understand the HTTP status codes.
3. Authentication and Authorisation through Basic Authentication, JWT, and Oauth2.
4. Introduction to localStorage.
5. LocalStorage of JWT token.

What is **REST**?



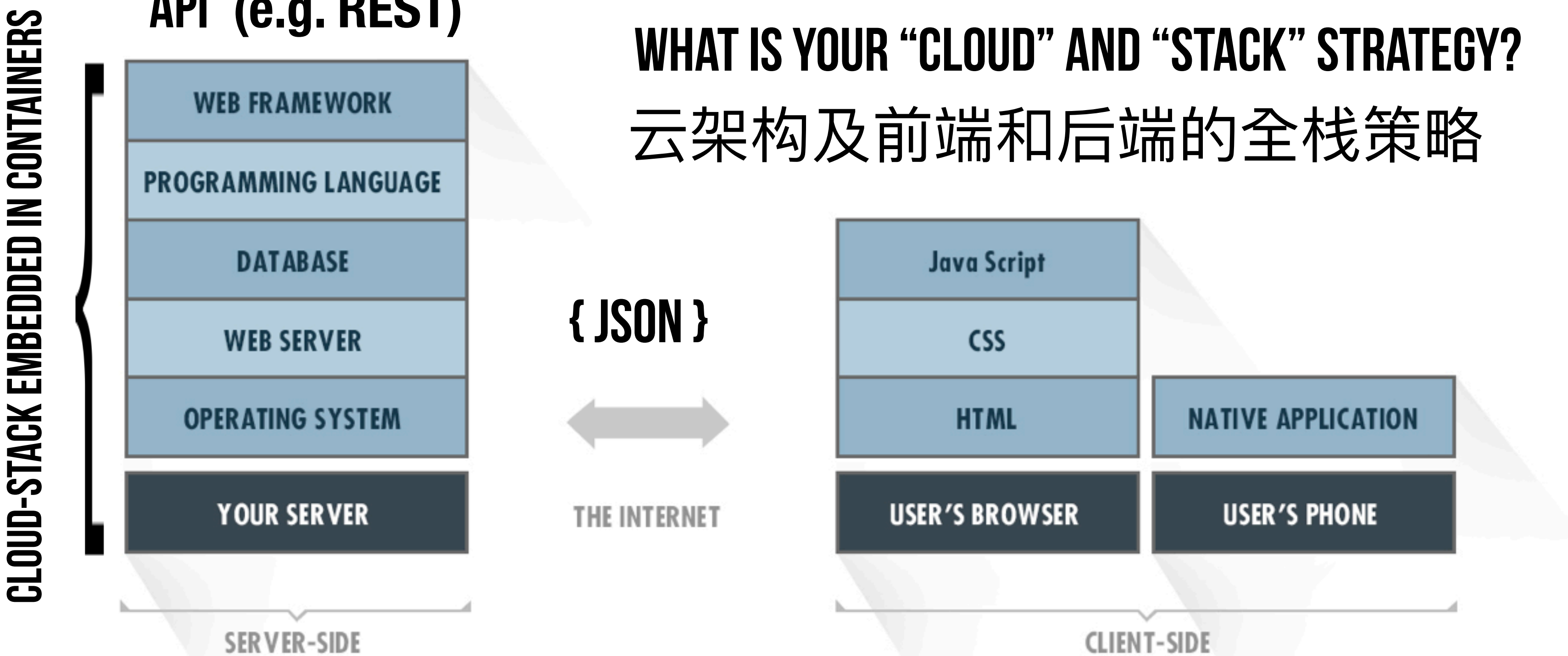
https://www.youtube.com/watch?v=u-RnFs9dby4

REST (Representational State Transfer)

API (e.g. REST)

WHAT IS YOUR “CLOUD” AND “STACK” STRATEGY?

云架构及前端和后端的全栈策略



BACK-END TECHNOLOGY
后台

FRONT-END TECHNOLOGY
前台

Create (**insert**)
Retrieve (**select**)
Update (**update**)
Delete (**delete**)

SQL commands

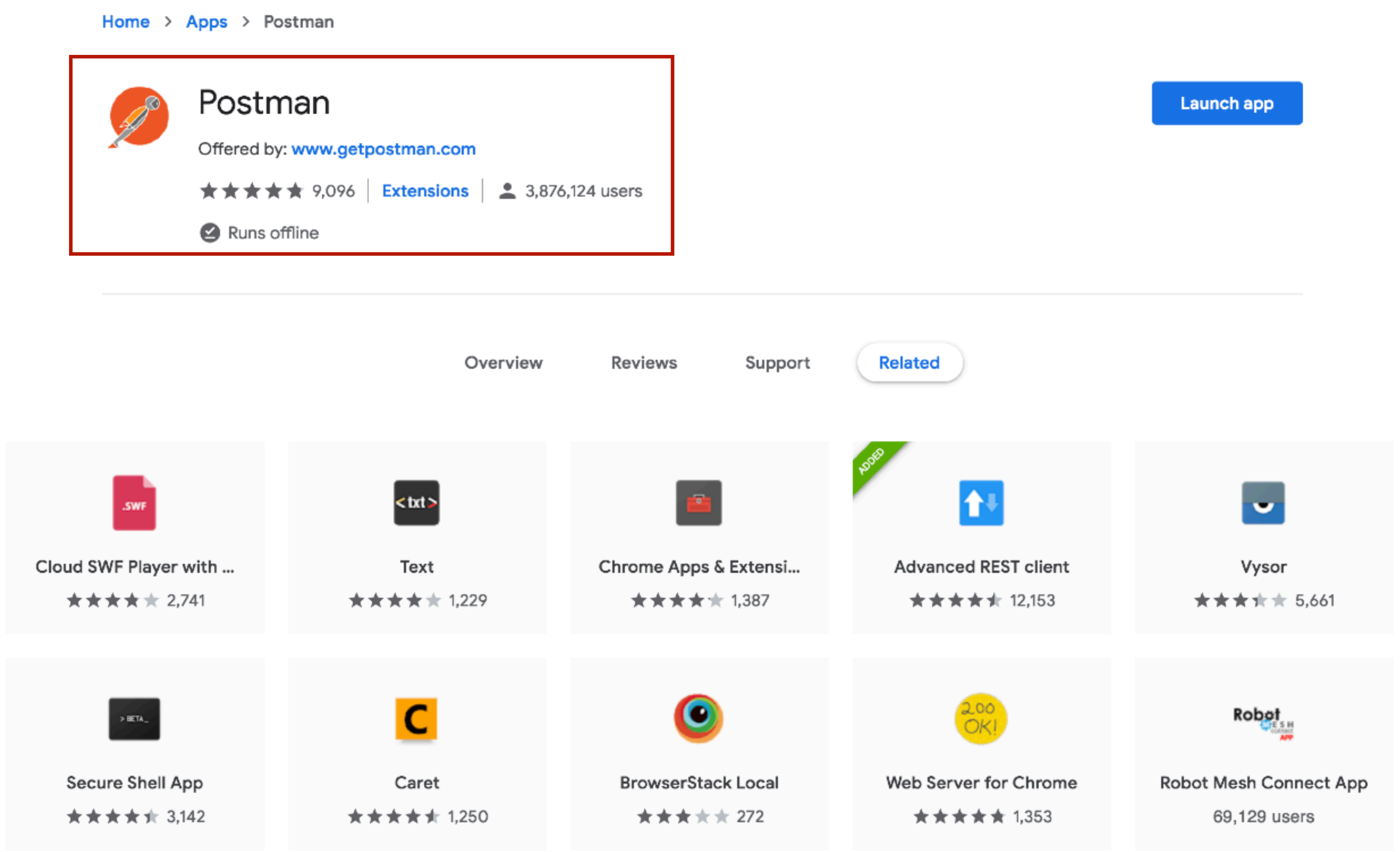
CRUD Operations in **REST API** via **Postman**

HTTP **Delete** method
HTTP **Put** method
HTTP **Get** method
HTTP **Post** method

CRUD & Authentication: Secured RESTful Operations

Before doing that, we need a special tool for testing the REST API. This tool is called Postman.

YouTube intro: https://www.youtube.com/watch?v=t5n07Ybz7yI



https://chrome.google.com/webstore/detail/postman/fhbjgbiflinjbdggehcddcbncdddomop?hl=en

REST API derives from **HTTP**. It is a HTTP based data **request** and **response** method.

Demonstration#1:
Basic CRUD Request in REST API

Basic RESTful Methods and Structure

HTTP
Methods (verbs)

Resource
Identifier (URI)

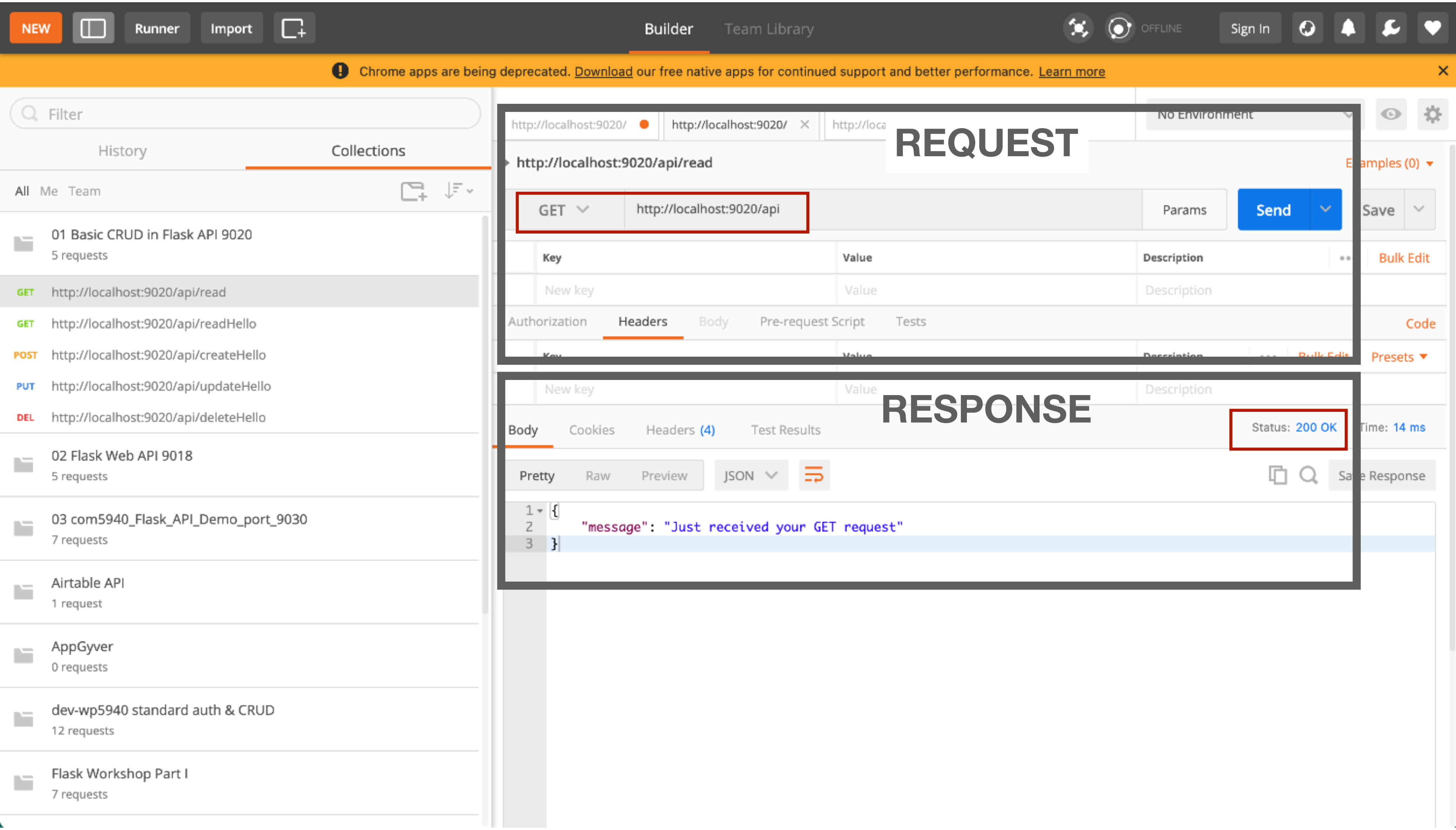
- Endpoint
接口

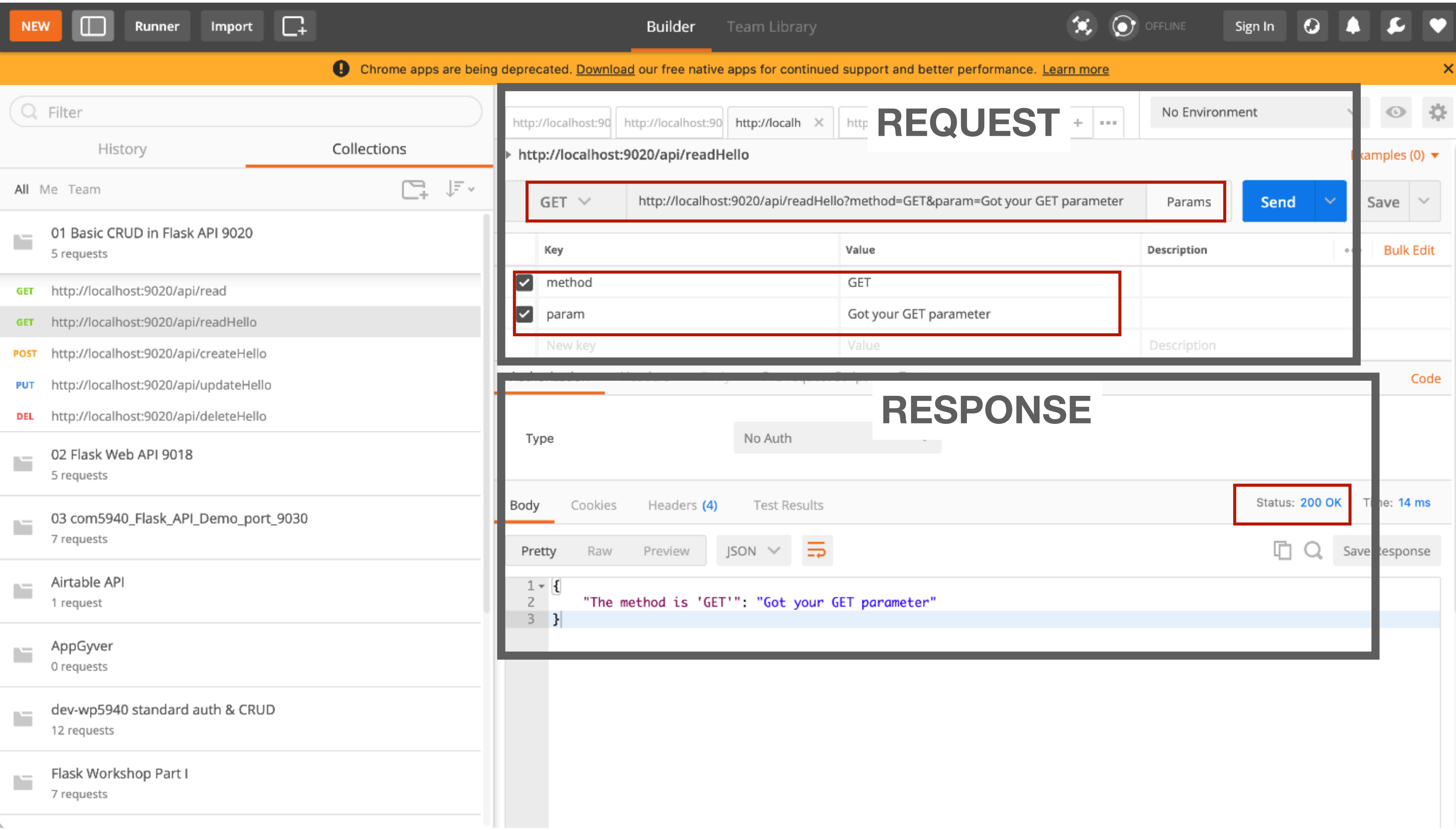
1. Create (Post): http://<url path>/<resource_name>
2. Retrieve (Get): http://<url path>/<resource_name>/<id>
3. Update (Put): http://<url path>/<resource_name>/<id>
4. Delete (Delete): http://<url path>/<resource_name>/<id>

e.g. http://localhost:9020/api/note/3

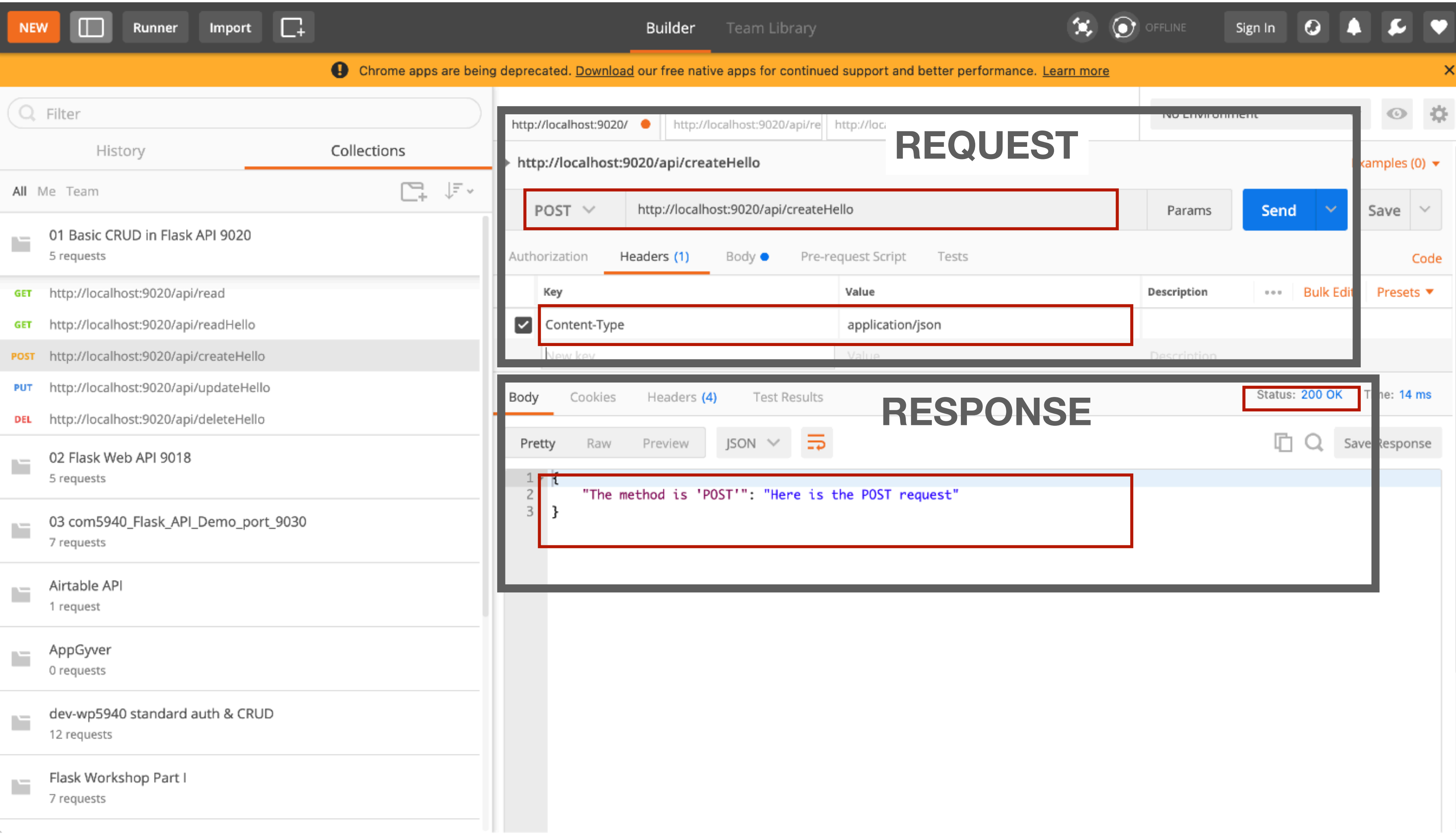
https://suentze2020.pythonanywhere.com/api/note/3

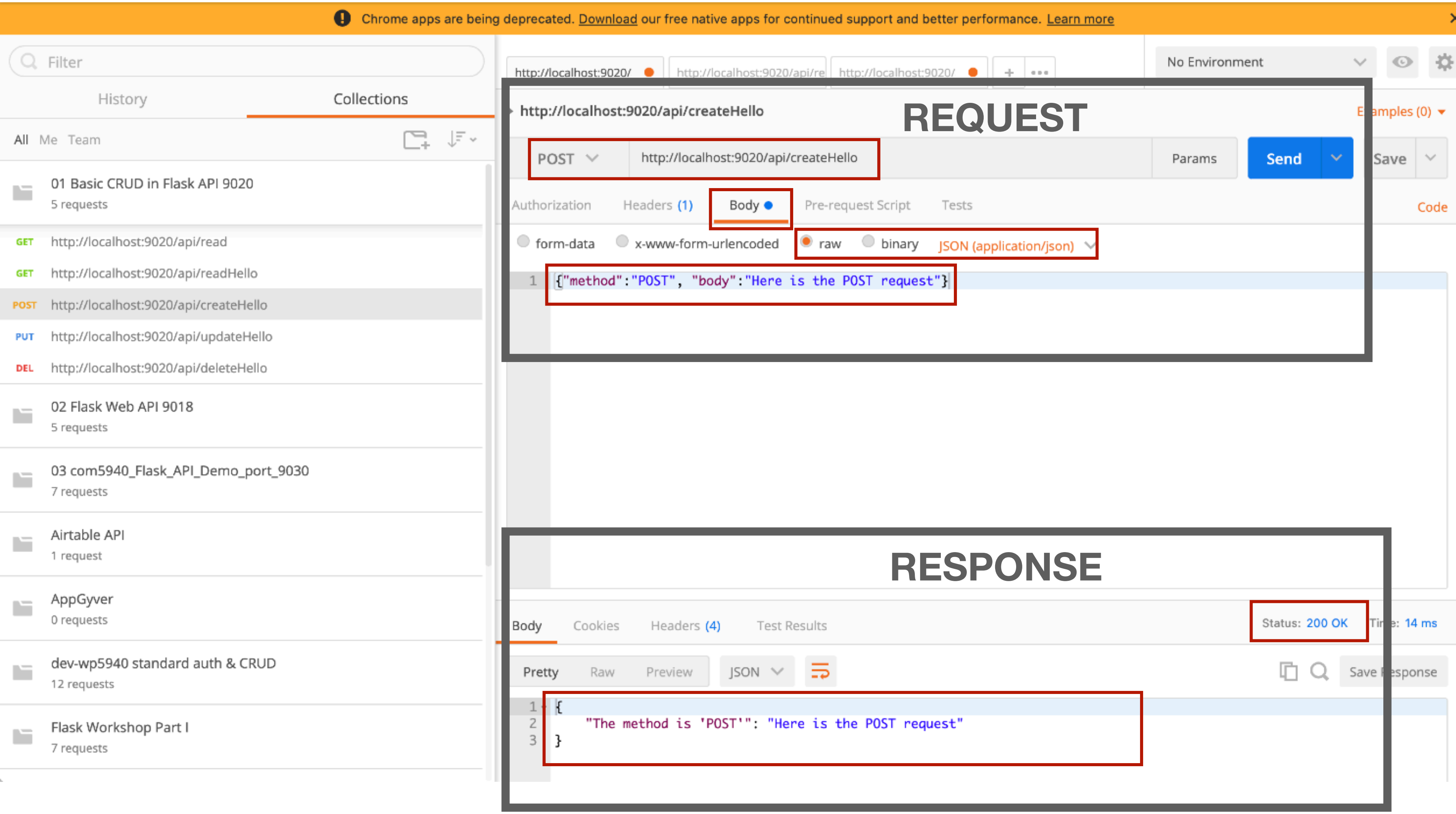

jupyter
Quit
Logout
Files
Running
Clusters
Select items to perform actions on them.
Upload
New
0
/ < your folder path > / lesson_1
Name
Last Modified
File size
..
seconds ago
Intro_to_CRUD_REST_API.ipynb
Running 12 hours ago 3.09 kB
Intro_to_CRUD_REST_API_with_Auth.ipynb
Running 11 minutes ago 10.8 kB
run_Init_db.ipynb
8 months ago 613 B
localStorage.html
7 days ago 4.13 kB
todo.db
10 hours ago 4.1 kB

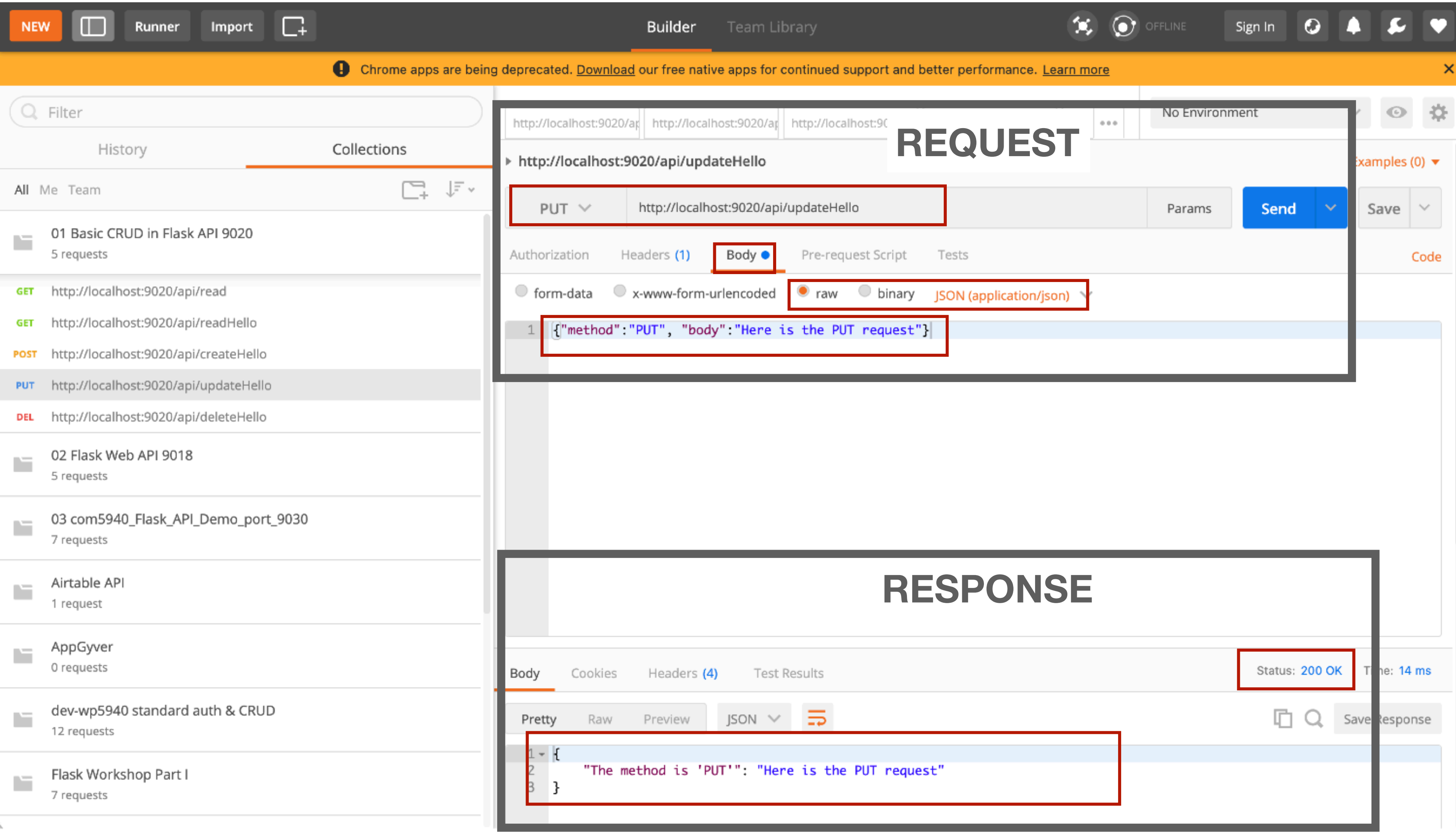

NEW
Runner
Import
Builder
Team Library
OFFLINE
Sign In
Chrome apps are being deprecated. Download our free native apps for continued support and better performance. Learn more
Filter
History
Collections
All Me Team
01 Basic CRUD in Flask API 9020
5 requests
GET http://localhost:9020/api/read
GET http://localhost:9020/api/readHello
POST http://localhost:9020/api/createHello
PUT http://localhost:9020/api/updateHello
DEL http://localhost:9020/api/deleteHello
02 Flask Web API 9018
5 requests
03 com5940_Flask_API_Demo_port_9030
7 requests
Airtable API
1 request
AppGyver
0 requests
dev-wp5940 standard auth & CRUD
12 requests
Flask Workshop Part I
7 requests
REQUEST
http://localhost:9020/api/read
GET
http://localhost:9020/api
Params
Send
Save
Key
Value
Description
Bulk Edit
Authorization
Headers
Body
Pre-request Script
Tests
Code
Presets
RESPONSE
Body
Cookies
Headers (4)
Test Results
Status: 200 OK
Time: 14 ms
Pretty
Raw
Preview
JSON
{
"message": "Just received your GET request"
}

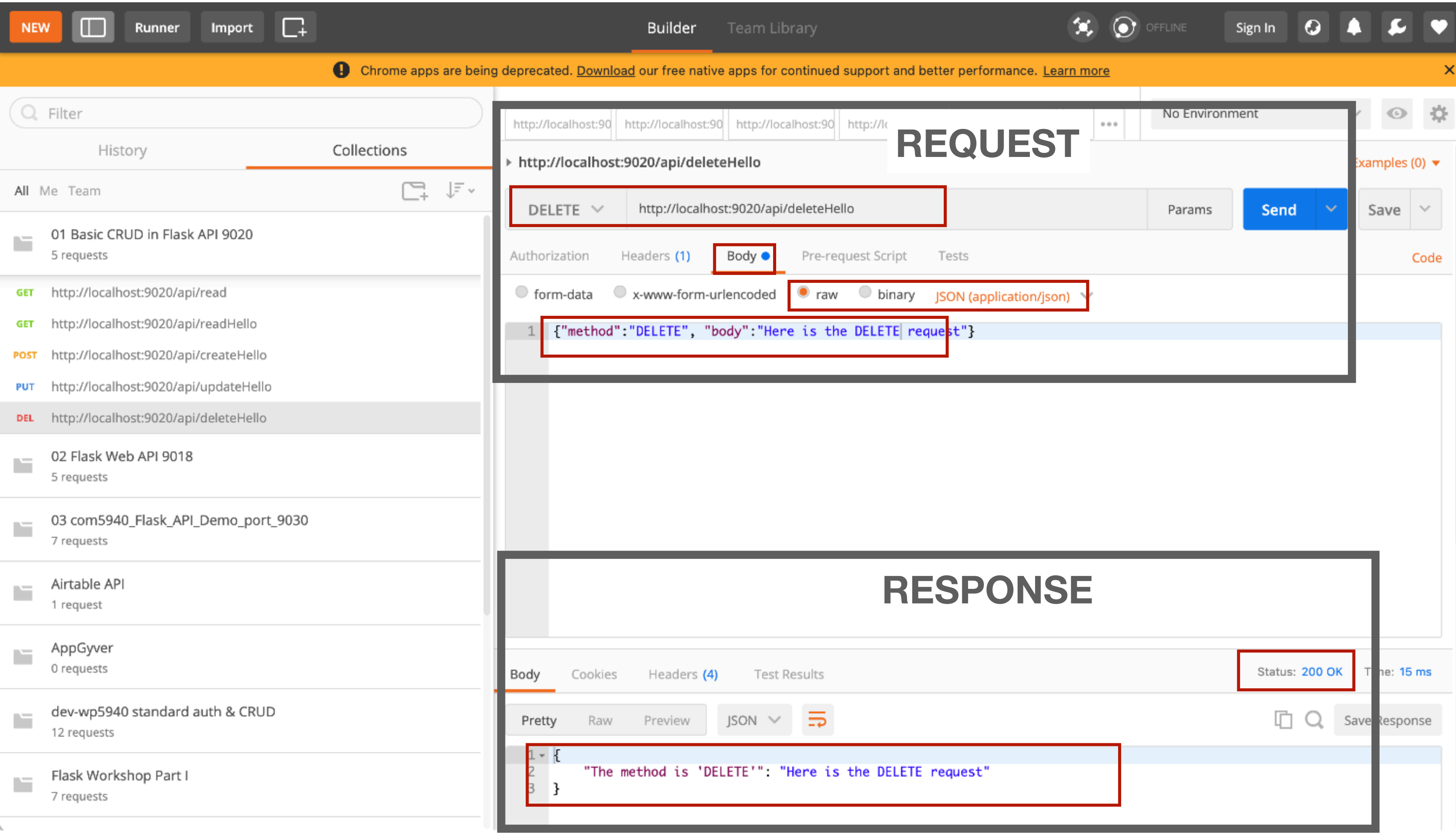


NEW
Runner
Import
Builder
Team Library
OFFLINE
Sign In
Chrome apps are being deprecated. Download our free native apps for continued support and better performance. Learn more
Filter
History
Collections
All Me Team
01 Basic CRUD in Flask API 9020
5 requests
GET http://localhost:9020/api/read
GET http://localhost:9020/api/readHello
POST http://localhost:9020/api/createHello
PUT http://localhost:9020/api/updateHello
DEL http://localhost:9020/api/deleteHello
02 Flask Web API 9018
5 requests
03 com5940_Flask_API_Demo_port_9030
7 requests
Airtable API
1 request
AppGyver
0 requests
dev-wp5940 standard auth & CRUD
12 requests
Flask Workshop Part I
7 requests
REQUEST
No Environment
http://localhost:9020/api/readHello
GET
http://localhost:9020/api/readHello?method=GET&param=Got your GET parameter
Params
Send
Save
Key
Value
Description
Bulk Edit
method
GET
param
Got your GET parameter
New key
Value
Description
Code
RESPONSE
Type
No Auth
Body
Cookies
Headers (4)
Test Results
Status: 200 OK
14 ms
Pretty
Raw
Preview
JSON
{
"The method is 'GET'": "Got your GET parameter"
}

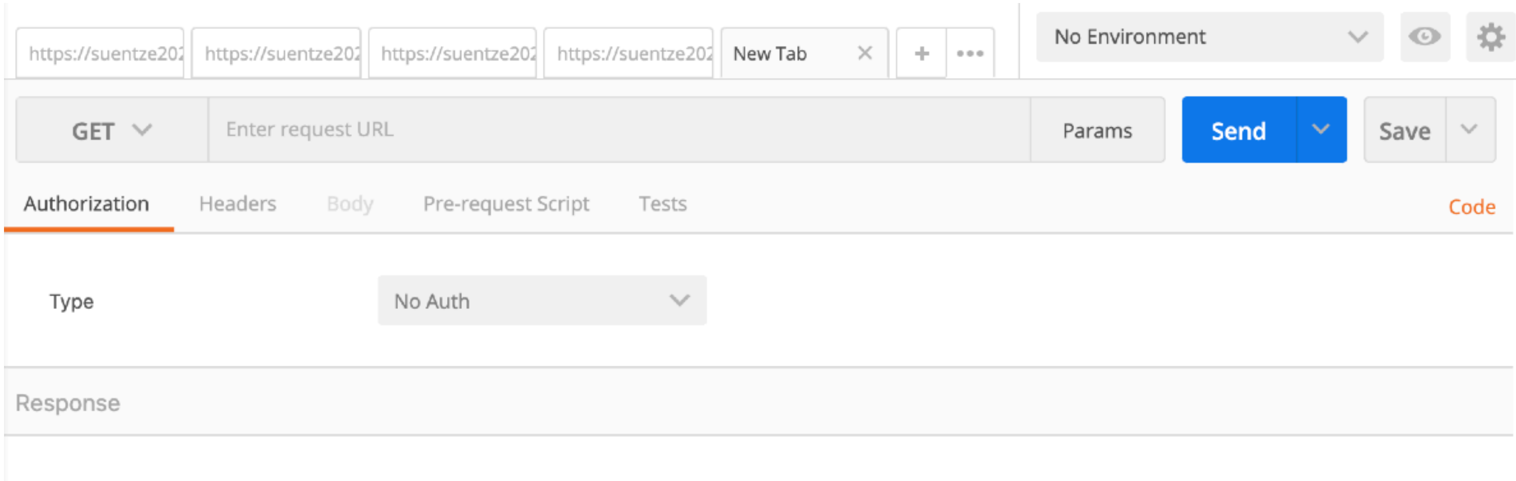

NEW
Runner
Import
Builder
Team Library
OFFLINE
Sign In
Chrome apps are being deprecated. Download our free native apps for continued support and better performance. Learn more
Filter
History
Collections
All Me Team
01 Basic CRUD in Flask API 9020
5 requests
GET http://localhost:9020/api/read
GET http://localhost:9020/api/readHello
POST http://localhost:9020/api/createHello
PUT http://localhost:9020/api/updateHello
DEL http://localhost:9020/api/deleteHello
02 Flask Web API 9018
5 requests
03 com5940_Flask_API_Demo_port_9030
7 requests
Airtable API
1 request
AppGyver
0 requests
dev-wp5940 standard auth & CRUD
12 requests
Flask Workshop Part I
7 requests
REQUEST
http://localhost:9020/api/createHello
POST
http://localhost:9020/api/createHello
Params
Send
Save
Authorization
Headers (1)
Body
Pre-request Script
Tests
Code
Key
Value
Description
Bulk Edit
Presets
Content-Type
application/json
RESPONSE
Body
Cookies
Headers (4)
Test Results
Status: 200 OK
Time: 14 ms
Pretty
Raw
Preview
JSON
Save Response
{
"The method is 'POST'": "Here is the POST request"
}


Chrome apps are being deprecated. Download our free native apps for continued support and better performance. Learn more
Filter
History
Collections
All Me Team
01 Basic CRUD in Flask API 9020
5 requests
GET http://localhost:9020/api/read
GET http://localhost:9020/api/readHello
POST http://localhost:9020/api/createHello
PUT http://localhost:9020/api/updateHello
DEL http://localhost:9020/api/deleteHello
02 Flask Web API 9018
5 requests
03 com5940_Flask_API_Demo_port_9030
7 requests
Airtable API
1 request
AppGyver
0 requests
dev-wp5940 standard auth & CRUD
12 requests
Flask Workshop Part I
7 requests
No Environment
http://localhost:9020/api/createHello
REQUEST
POST
http://localhost:9020/api/createHello
Params
Send
Save
Authorization
Headers (1)
Body
Pre-request Script
Tests
Code
form-data
x-www-form-urlencoded
raw
binary
JSON (application/json)
1 {"method":"POST", "body":"Here is the POST request"}
RESPONSE
Body
Cookies
Headers (4)
Test Results
Status: 200 OK
14 ms
Pretty
Raw
Preview
JSON
1 {
2 "The method is 'POST'": "Here is the POST request"
3 }


NEW
Runner
Import
Builder
Team Library
OFFLINE
Sign In
Chrome apps are being deprecated. Download our free native apps for continued support and better performance. Learn more
Filter
History
Collections
All Me Team
01 Basic CRUD in Flask API 9020
5 requests
GET http://localhost:9020/api/read
GET http://localhost:9020/api/readHello
POST http://localhost:9020/api/createHello
PUT http://localhost:9020/api/updateHello
DEL http://localhost:9020/api/deleteHello
02 Flask Web API 9018
5 requests
03 com5940_Flask_API_Demo_port_9030
7 requests
Airtable API
1 request
AppGyver
0 requests
dev-wp5940 standard auth & CRUD
12 requests
Flask Workshop Part I
7 requests
REQUEST
No Environment
http://localhost:9020/api/updateHello
Examples (0)
PUT
http://localhost:9020/api/updateHello
Params
Send
Save
Authorization
Headers (1)
Body
Pre-request Script
Tests
Code
form-data
x-www-form-urlencoded
raw
binary
JSON (application/json)
{"method":"PUT", "body":"Here is the PUT request"}
RESPONSE
Body
Cookies
Headers (4)
Test Results
Status: 200 OK
Time: 14 ms
Pretty
Raw
Preview
JSON
Save Response
{
"The method is 'PUT'": "Here is the PUT request"
}



NEW
Runner
Import
Builder
Team Library
OFFLINE
Sign In
Chrome apps are being deprecated. Download our free native apps for continued support and better performance. Learn more
Filter
History
Collections
All Me Team
01 Basic CRUD in Flask API 9020
5 requests
GET http://localhost:9020/api/read
GET http://localhost:9020/api/readHello
POST http://localhost:9020/api/createHello
PUT http://localhost:9020/api/updateHello
DEL http://localhost:9020/api/deleteHello
02 Flask Web API 9018
5 requests
03 com5940_Flask_API_Demo_port_9030
7 requests
Airtable API
1 request
AppGyver
0 requests
dev-wp5940 standard auth & CRUD
12 requests
Flask Workshop Part I
7 requests
REQUEST
No Environment
http://localhost:9020/api/deleteHello
DELETE
http://localhost:9020/api/deleteHello
Params
Send
Save
Authorization
Headers (1)
Body
Pre-request Script
Tests
Code
form-data
x-www-form-urlencoded
raw
binary
JSON (application/json)
{"method":"DELETE", "body":"Here is the DELETE request"}
RESPONSE
Body
Cookies
Headers (4)
Test Results
Status: 200 OK
15 ms
Pretty
Raw
Preview
JSON
{
"The method is 'DELETE'": "Here is the DELETE request"
}



Can you use <u>Postman</u> to GET the blog posts?

HTTP **Status Codes**

Reference video: https://www.youtube.com/watch?v=VLH3FMQ5BIQ

HTTP response status codes

HTTP response status codes indicate whether a specific HTTP request has been successfully completed. Responses are grouped in five classes:

1. Informational responses (`100` – `199`)
2. Successful responses (`200` – `299`)
3. Redirection messages (`300` – `399`)
4. Client error responses (`400` – `499`)
5. Server error responses (`500` – `599`)

Source: MDN Wed Docs (https://developer.mozilla.org/en-US/docs/Web/HTTP/Status)

HTTP Status Codes

This page is created from HTTP status code information found at ietf.org and Wikipedia. Click on the **category heading** or the **status code** link to read more.

1xx Informational

100 Continue
101 Switching Protocols
102 Processing (WebDAV)

2xx Success

★ 200 OK
★ 201 Created
202 Accepted
203 Non-Authoritative Information
★ 204 No Content
205 Reset Content
206 Partial Content
207 Multi-Status (WebDAV)
208 Already Reported (WebDAV)
226 IM Used

3xx Redirection

300 Multiple Choices
301 Moved Permanently
302 Found
303 See Other
★ 304 Not Modified
305 Use Proxy
306 (Unused)
307 Temporary Redirect
308 Permanent Redirect (experimental)

4xx Client Error

★ 400 Bad Request
★ 401 Unauthorized
402 Payment Required
★ 403 Forbidden
★ 404 Not Found
405 Method Not Allowed
406 Not Acceptable
407 Proxy Authentication Required
408 Request Timeout
★ 409 Conflict
410 Gone
411 Length Required
412 Precondition Failed
413 Request Entity Too Large
414 Request-URI Too Long
415 Unsupported Media Type
416 Requested Range Not Satisfiable
417 Expectation Failed
418 I'm a teapot (RFC 2324)
420 Enhance Your Calm (Twitter)
422 Unprocessable Entity (WebDAV)
423 Locked (WebDAV)
424 Failed Dependency (WebDAV)
425 Reserved for WebDAV
426 Upgrade Required
428 Precondition Required
429 Too Many Requests
431 Request Header Fields Too Large
444 No Response (Nginx)
449 Retry With (Microsoft)
450 Blocked by Windows Parental Controls (Microsoft)
451 Unavailable For Legal Reasons
499 Client Closed Request (Nginx)

5xx Server Error

★ 500 Internal Server Error
501 Not Implemented
502 Bad Gateway
503 Service Unavailable
504 Gateway Timeout
505 HTTP Version Not Supported
506 Variant Also Negotiates (Experimental)
507 Insufficient Storage (WebDAV)
508 Loop Detected (WebDAV)
509 Bandwidth Limit Exceeded (Apache)
510 Not Extended
511 Network Authentication Required
598 Network read timeout error
599 Network connect timeout error

★ **"Top 10"** HTTP Status Code. More REST service-specific information is contained in the entry.

Source: Web API Tutorial (https://www.restapitutorial.com/httpstatuscodes.html)

How can CRUD operations be secured?

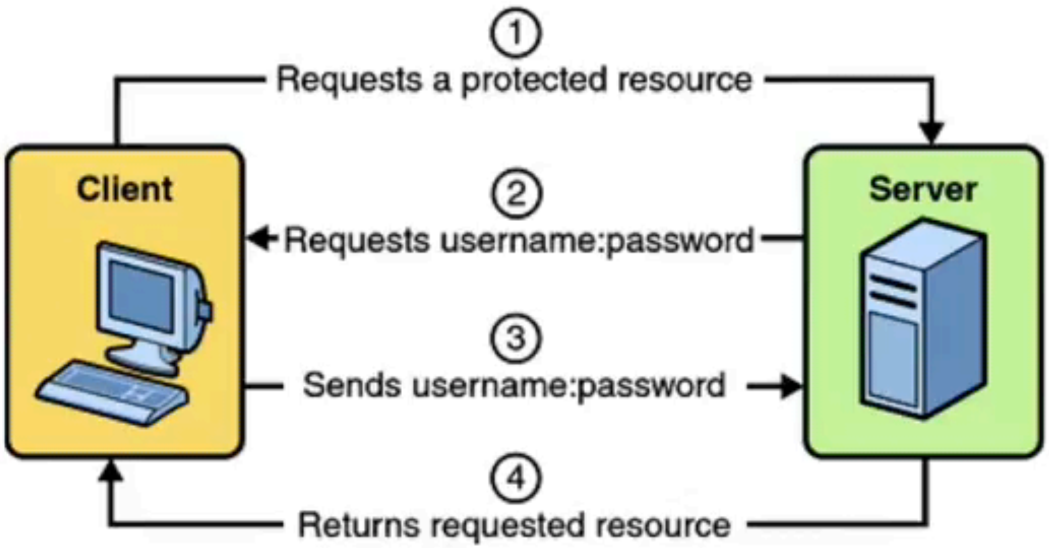
Introduction to Basic Auth, Oauth2, and Access Token (e.g. JWT)

How to limit **API** access to **authenticated** and **authorised** people?

To grab data through REST API, you need:

- URI (Resources Identifier)
- HTTP method
- Headers (Authorization, Content-Type)
- Body (JSON, form-urlencoded)
- Authentication methods (e.g. Basic, OAuth 2, JWT)

Basic Authentication



Reference video: https://www.youtube.com/watch?v=Us1fNcynZ5o

ScienceDirect Journals & Books Register Sign in

Basic Authentication

Related terms:

Authentication Method, Digest Authentication

View all Topics >

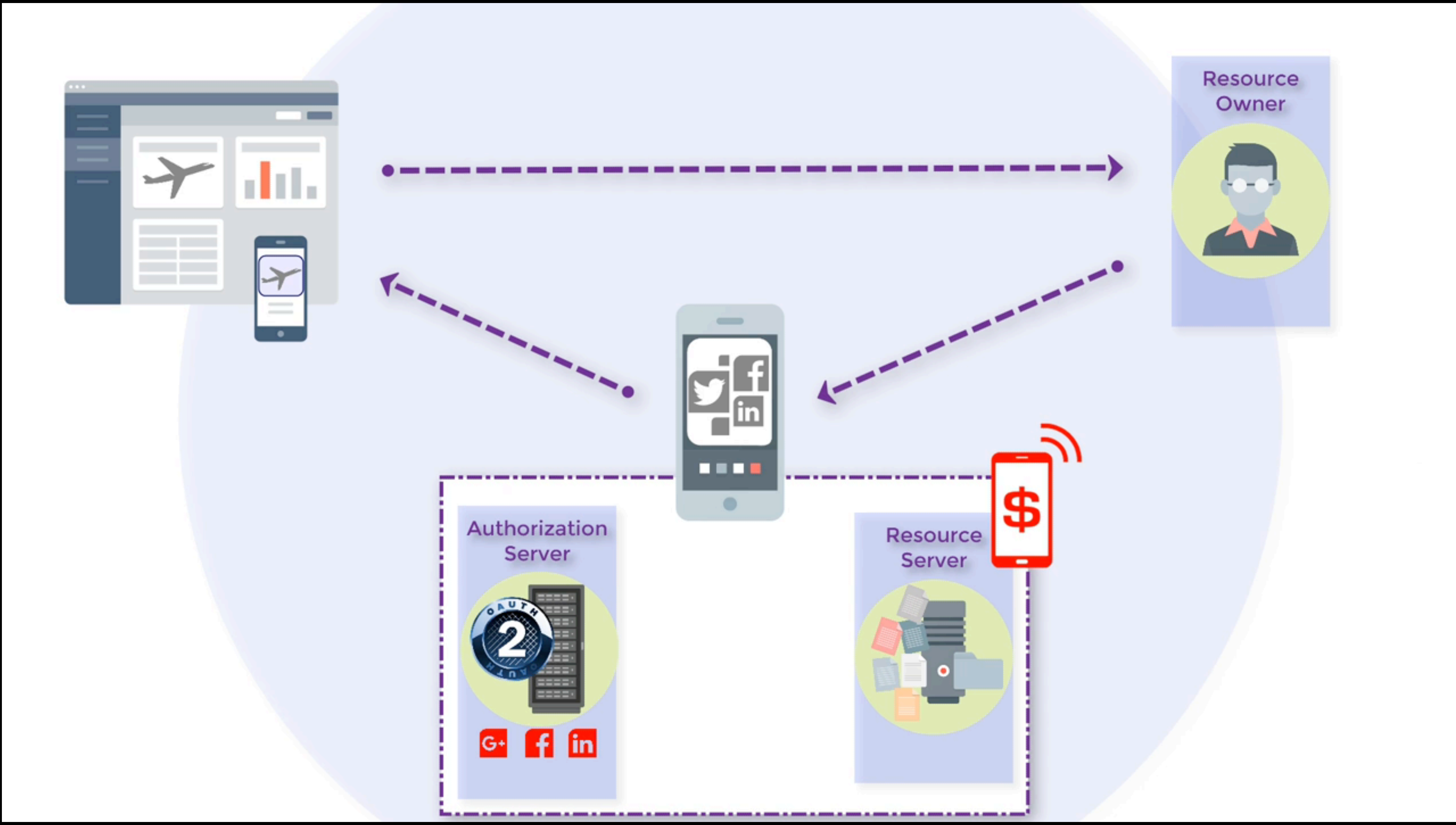
Download as PDF Set alert About this page

Authentication and Granular Access

Administration of an IIS 7.0 Web Server

https://www.sciencedirect.com/topics/computer-science/basic-authentication

OAuth 2.0



https://www.youtube.com/watch?v=zEysfgIbqlg

Community   Tutorials   Questions   Get Involved   Sign Up

Language: EN

Contents

An Introduction to OAuth 2

Posted July 21, 2014   1.6m   SECURITY   API   CONCEPTUAL

By Mitchell Anicas
Become an author

Introduction

OAuth 2 is an authorization framework that enables applications to obtain limited access to user accounts on an HTTP service, such as Facebook, GitHub, and DigitalOcean. It works by delegating user authentication to the service that hosts the user account, and authorizing third-party applications to access the user account. OAuth 2 provides authorization flows for web and desktop applications, and mobile devices.

This informational guide is geared towards application developers, and provides an overview of OAuth 2 roles, authorization grant types, use cases, and flows.

Let's get started with OAuth Roles!

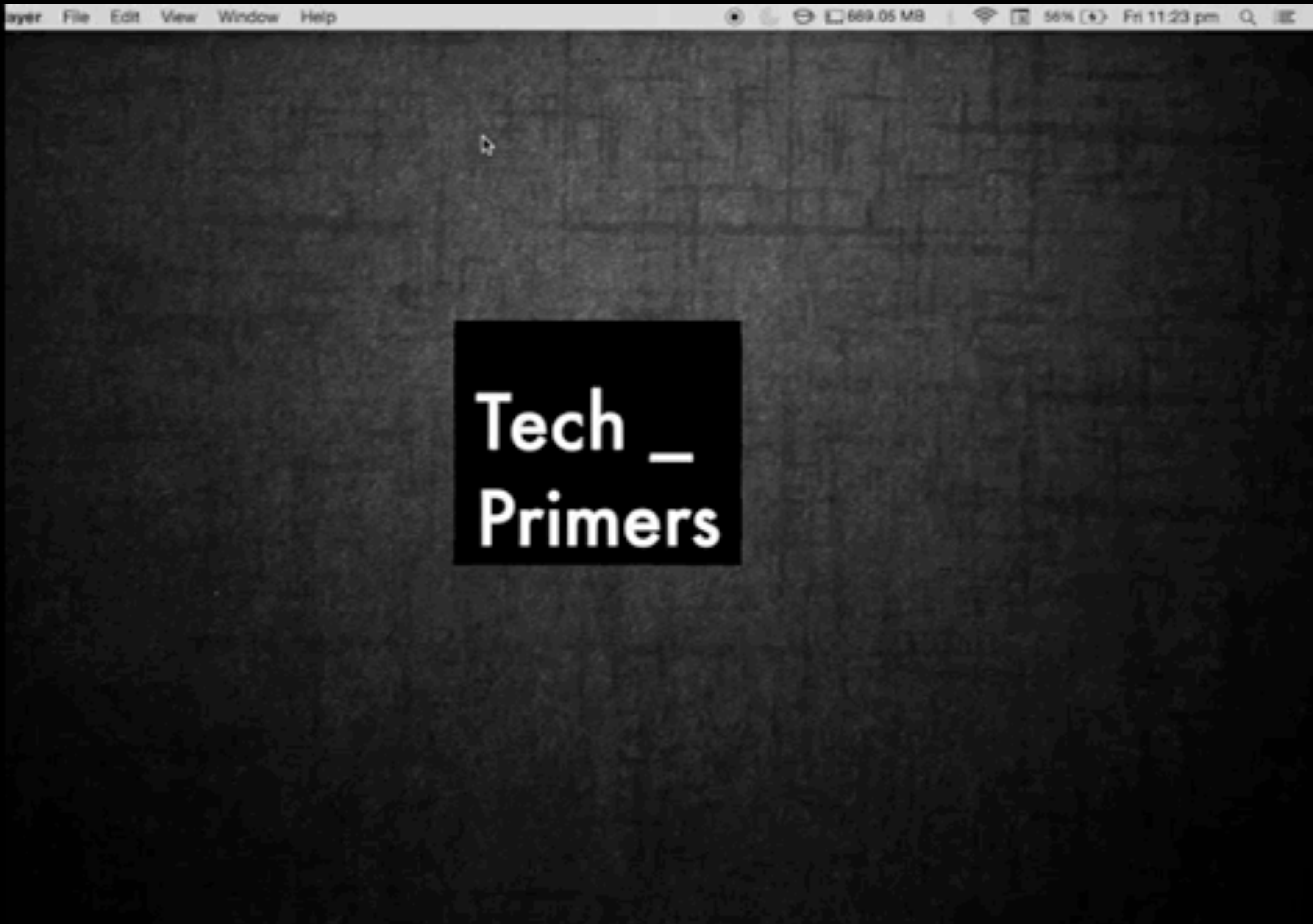
Related

Recommended Steps To Harden Apache HTTP on FreeBSD 12.0
Tutorial

How To Set Up SSH Keys on CentOS 8
Tutorial

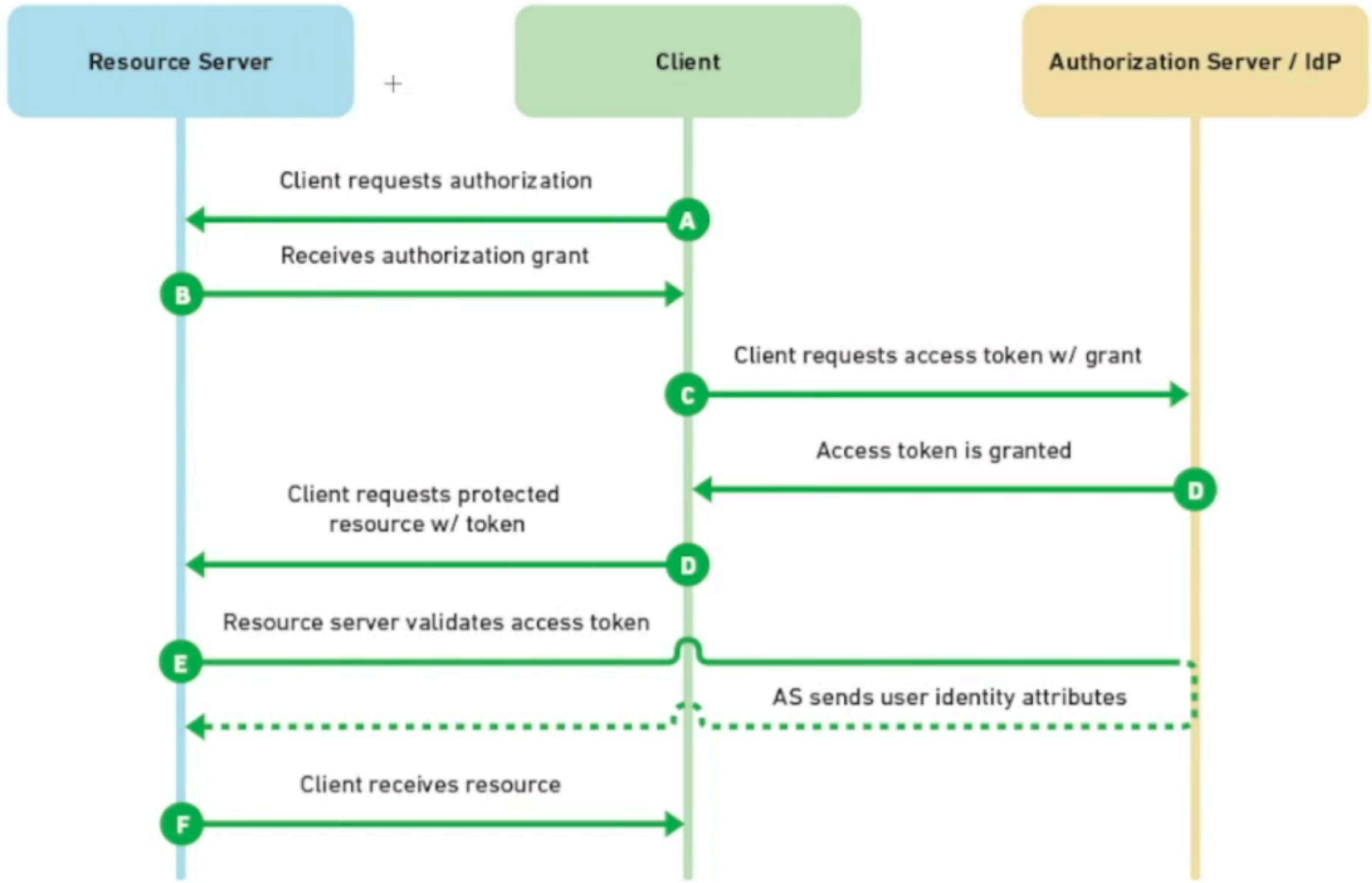
https://www.digitalocean.com/community/tutorials/an-introduction-to-oauth-2

JWT vs. OAuth 2

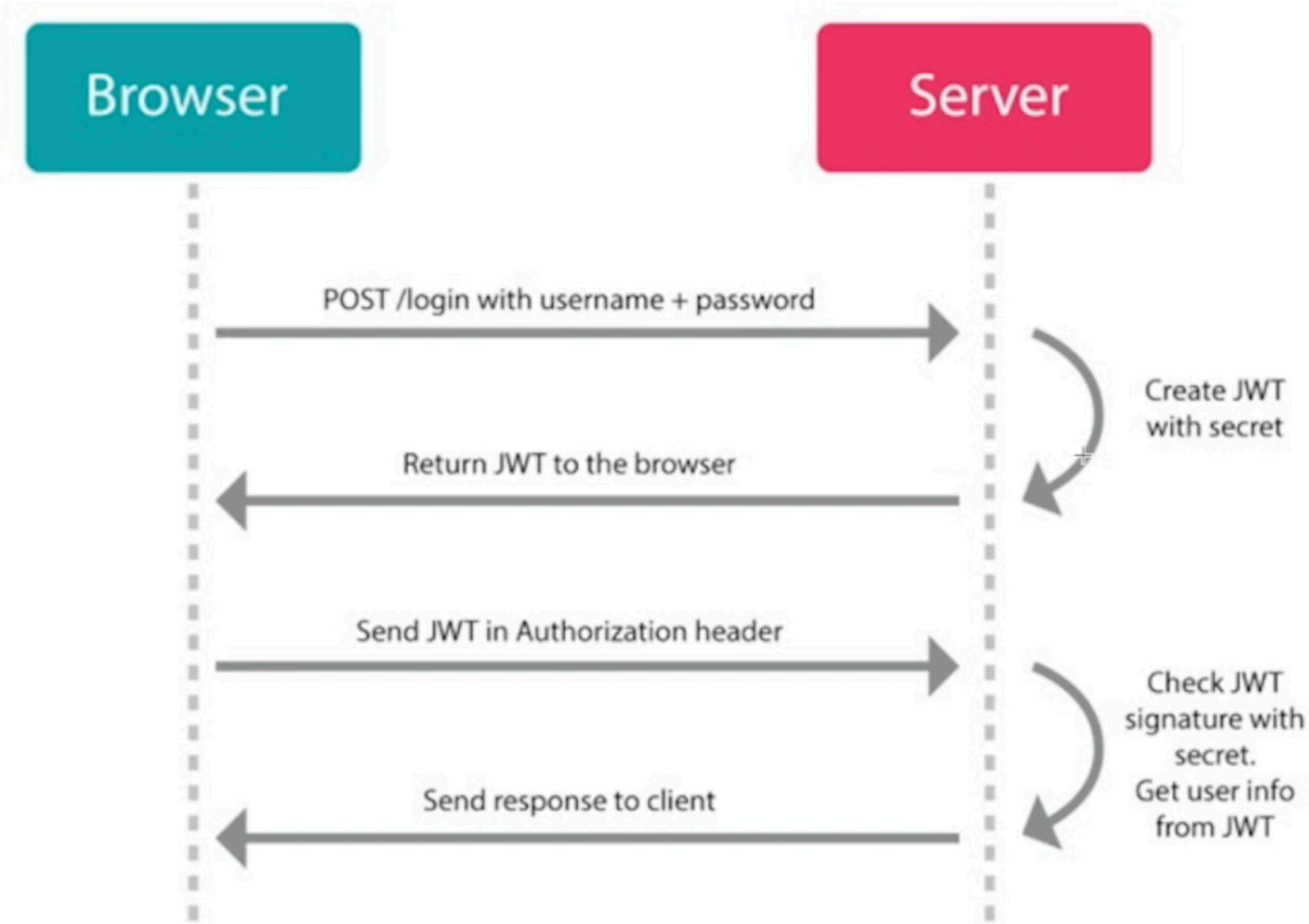


https://www.youtube.com/watch?time_continue=55&v=a9R3Gq1BKxI&feature=emb_logo

JWT vs. OAuth 2



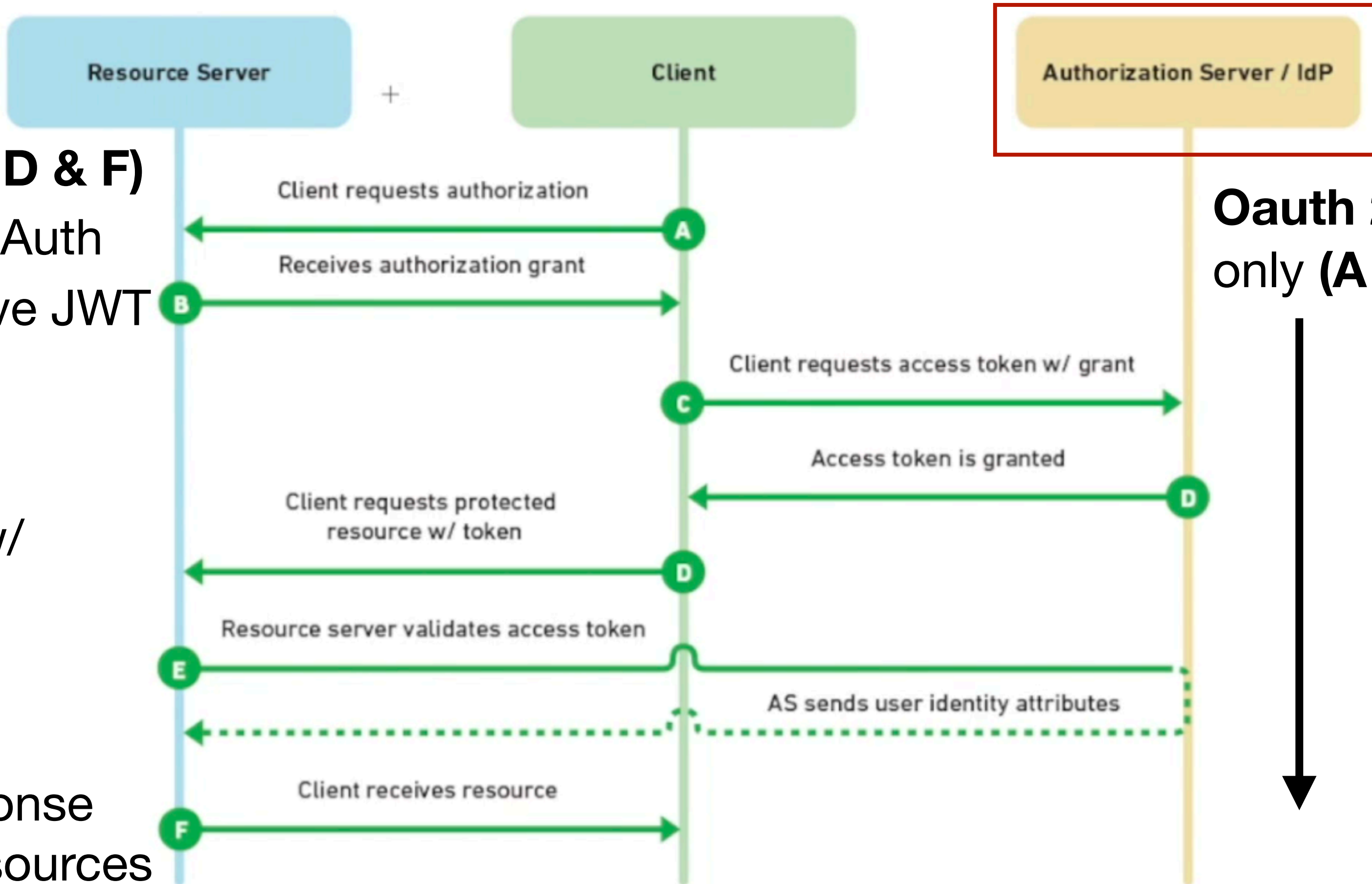
JWT vs. OAuth 2



JWT

(A, B, D & F)

1. Basic Auth
2. Receive JWT
3. Req w/ token
4. Response w/ resources



Oauth 2

only **(A to F)**

Blog

Introduction to JWT

Avi Aryan - July 2, 2016 - Event Management / Open Event

In this post, I will try to explain what is JWT, what are its advantages and why you should be using it.

JWT stands for JSON Web Tokens. Let me explain what each word means.

1. Tokens – Token is in tech terms a piece of data (claim) which gives access to certain piece of information and allows certain actions.
2. Web – Web here means that it was designed to be used on the web i.e. web projects.
3. JSON – JSON means that the token can contain json data. In JWT, the json is first serialized and then Base64 encoded.

A JWT looks like a random sequence of strings separated by 2 dots. The `yyyyy` part which you see below has the Base64 encoded form of json data mentioned earlier.

https://blog.fossasia.org/introduction-to-jwt/

3 Common API Authentication/Authorization Methods

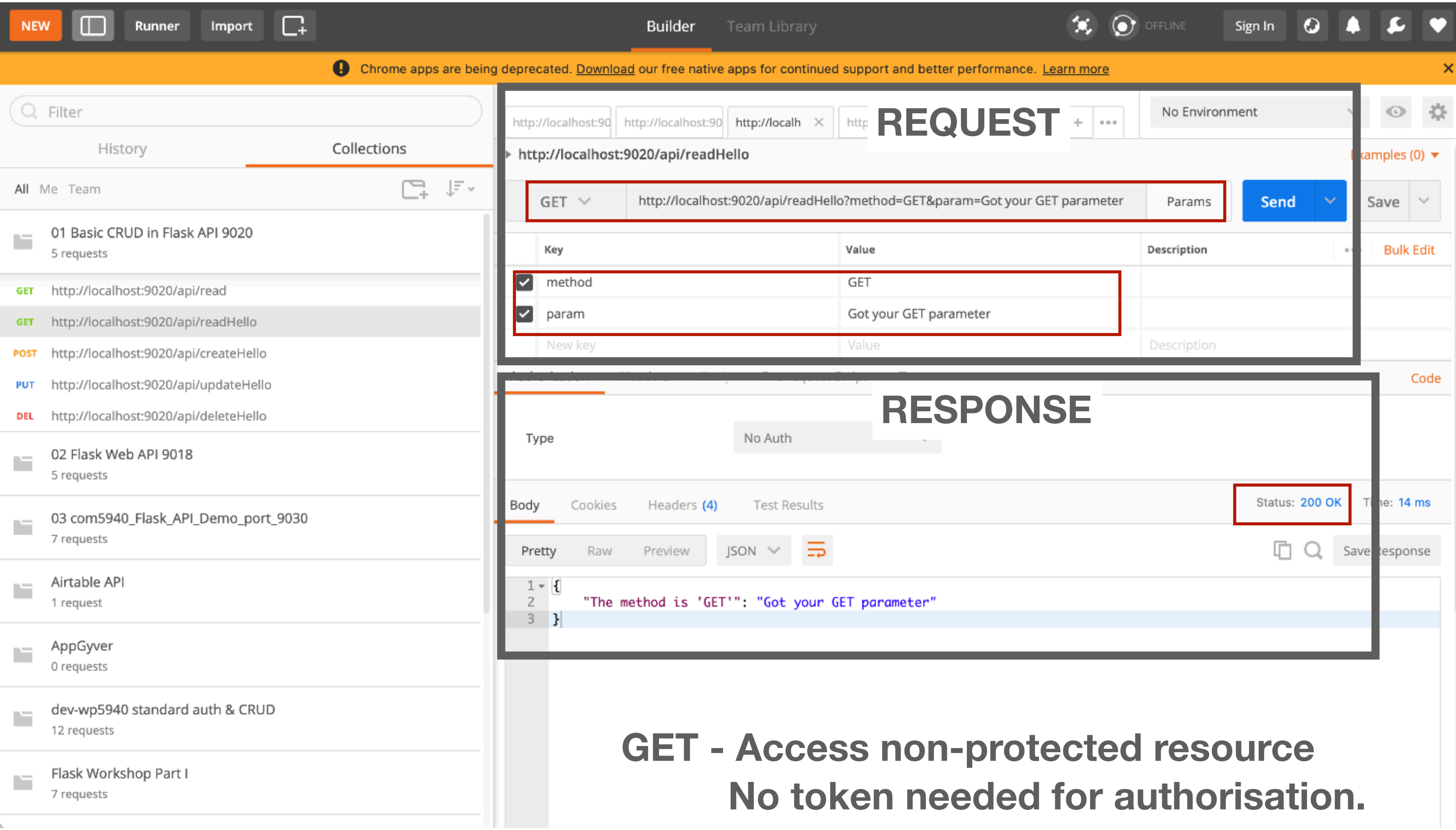
	BASIC AUTH	**OAUTH 2**	**Access Token (e.g. JWT)**
Encryption	No (Rely on https)	Yes	Yes
Authentication	User Name/ Password	Token (from resource provider)	Token (stored locally)
Authorization	NA Rely on server-side programs.	Yes	Yes
Multiple Parties	NA	Yes	NA

Demonstration#2:

CRUD Request in REST API with Authentication and Authorisation Using Basic Auth and JWT

Reference video: https://www.youtube.com/watch?v=WxGBoY5iNXY&t=1473s

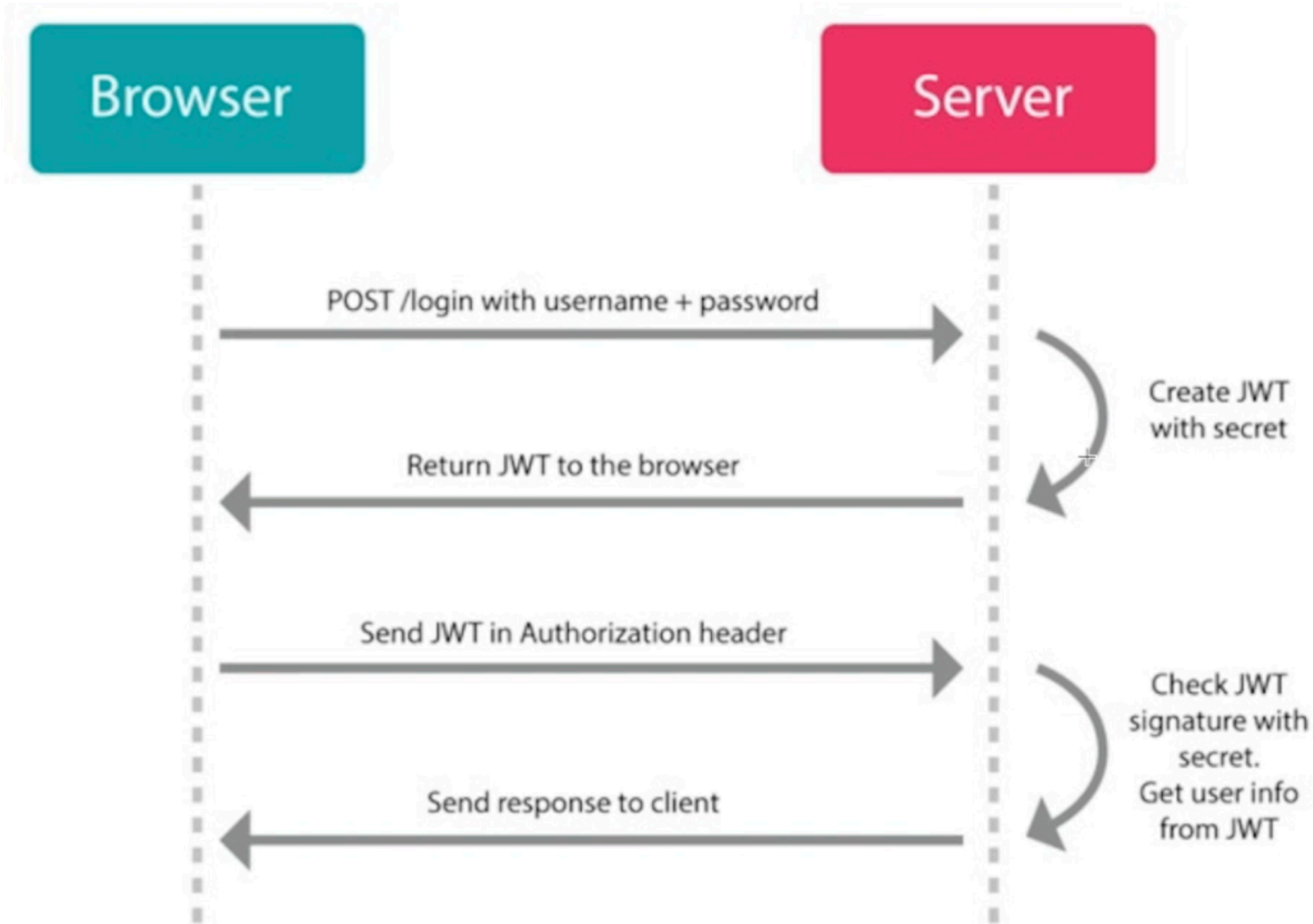

jupyter
Quit
Logout
Files
Running
Clusters
Select items to perform actions on them.
Upload
New
0
/ < your folder path > / lesson_1
Name
Last Modified
File size
..
seconds ago
Intro_to_CRUD_REST_API.ipynb
Running 12 hours ago 3.09 kB
Intro_to_CRUD_REST_API_with_Auth.ipynb
Running 11 minutes ago 10.8 kB
run_Init_db.ipynb
8 months ago 613 B
localStorage.html
7 days ago 4.13 kB
todo.db
10 hours ago 4.1 kB



GET - Access non-protected resource

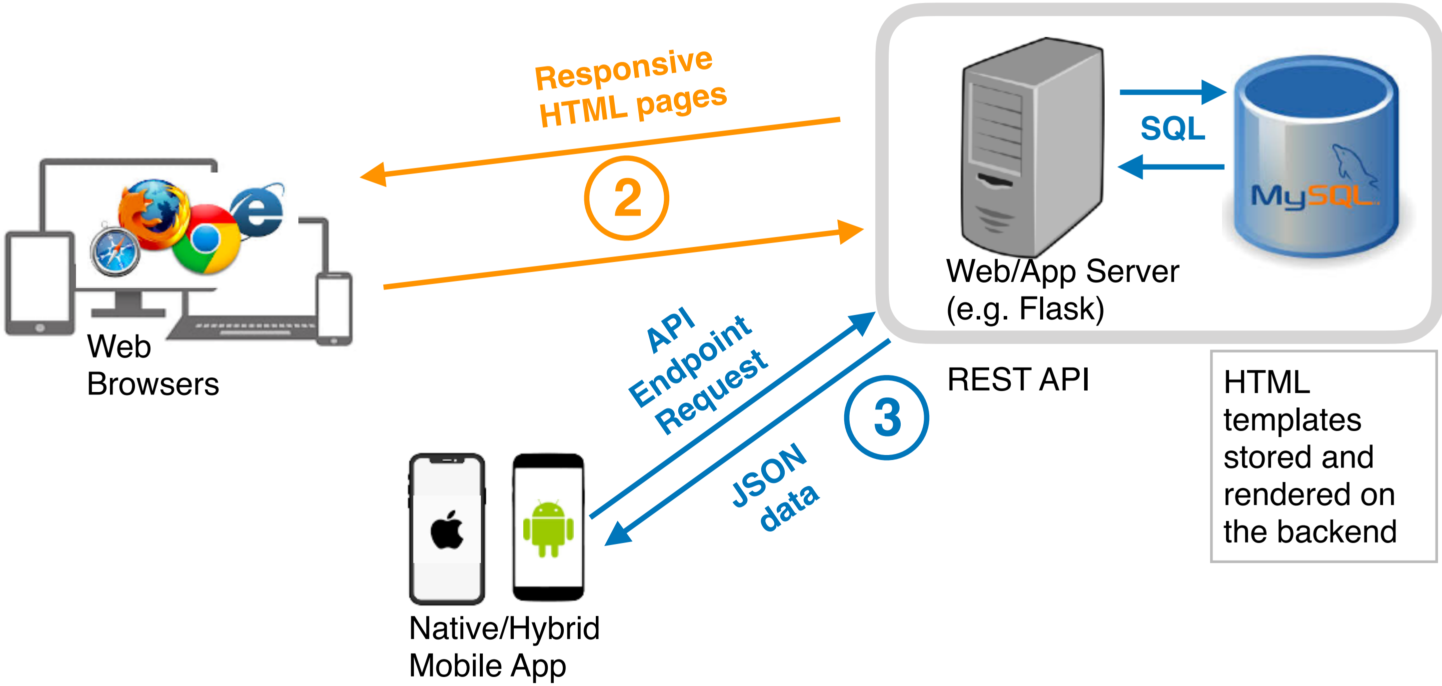
No token needed for authorisation.

JWT vs. OAuth 2



Authenticate through Basic Auth to obtain JWT token.

JWT Token Decorator (Used for API authorisation)



HTML templates stored and rendered on the frontend

The templates come bundled with the mobile app when we download it from the app store.

DATA FLOW

<HTML>/{ JSON }

FRONT-END PROCESSING

HTML/CSS/JS

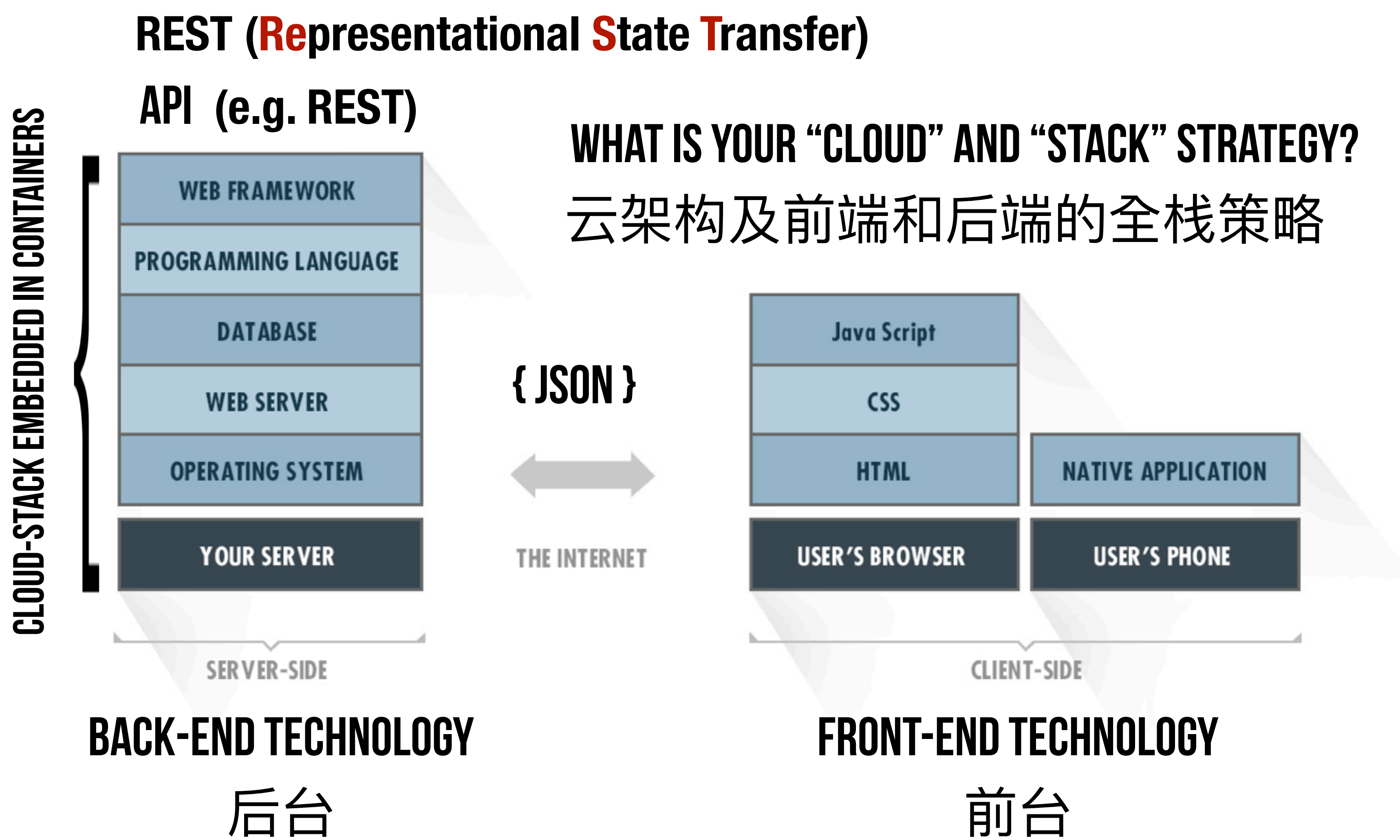
React/Angula/Vue
PWA, Native IOS and
Android Mobile App

WEB APP FRAMEWORKS
For Front-end & Back-end

BACK-END PROCESSING

PYTHON/PHP,
NODE.JS/C# etc

FLASK/Django
Laravel
EXPRESS.JS/.NET


REST (Representational State Transfer)
API (e.g. REST)
WHAT IS YOUR "CLOUD" AND "STACK" STRATEGY?
云架构及前端和后端的全栈策略
CLOUD-STACK EMBEDDED IN CONTAINERS
WEB FRAMEWORK
PROGRAMMING LANGUAGE
DATABASE
WEB SERVER
OPERATING SYSTEM
YOUR SERVER
SERVER-SIDE
{ JSON }
THE INTERNET
Java Script
CSS
HTML
NATIVE APPLICATION
USER'S BROWSER
USER'S PHONE
CLIENT-SIDE
BACK-END TECHNOLOGY
后台
FRONT-END TECHNOLOGY
前台

```python
import os
import secrets
from PIL import Image
from flask import Flask, flash, render_template, request, redirect, url_for, session, jsonify, make_response
# For pythonanywhere development, use the following code
# from flask_session import Session
# and remove from flask_session.__init__ import Session
from flask_session.__init__ import Session
from flask_bcrypt import Bcrypt
from functools import wraps
from flask_sqlalchemy import SQLAlchemy
from datetime import datetime, timedelta
import jwt
from flask_cors import CORS
```

Request form or JSON data

Turn object into JSON format.

Decorator

JWT token

CORS - Cross Origin Resources Sharing

```python
def token_required(f):
    @wraps(f)
    def decorated(*args, **kwargs):
        token = None

        if 'x-access-token' in request.headers:
            token = request.headers['x-access-token']

        if not token:
            return jsonify({'message' : 'Token is missing!'}), 401

        try:
            # data = jwt.decode(token, app.config['SECRET_KEY'])
            data = jwt.decode(token, app.config['SECRET_KEY'], algorithms=['HS256'])
            current_user = User.query.filter_by(id=data['id']).first()
            print("current_user = ", current_user)
        except:
            return jsonify({'message' : 'Token is invalid!'}), 401

        return f(current_user, *args, **kwargs)

    return decorated
```

Decorator

```python
def login_required(func):
    @wraps(func)
    def wrap(*args, **kwargs):
        if 'logged_in' in session:
            return func(*args, **kwargs)
        else:
            flash('You need to login first.','danger')
            return redirect(url_for('login'))
    return wrap
```

Decorator

Comparing Web and API in handling registration.

Web

```python
@app.route("/register", methods=["GET","POST"])
def register():
    if request.method == "POST":
        username = request.form['username']
        email = request.form['email']
        bio = request.form['bio']
        password = request.form['password']
        password = bcrypt.generate_password_hash(password)
        admin = 0
        user = User(username=username,email=email,bio=bio,password=password,admin=admin)
        db.session.add(user)
        db.session.commit()
        print("You have been registered!")

    return render_template('register.html')
```

Request form data.

Return html file.

(

API

```python
@app.route('/api/register', methods=['POST'])
def register():
    data = request.get_json()
    hashed_password = bcrypt.generate_password_hash(data['password'])
    new_user = User(name=data['name'], password=hashed_password, admin=False)
    db.session.add(new_user)
    db.session.commit()
    return jsonify({'message' : 'Registration completed!'})
```

Request JSON

Return JSON

API

```python
@app.route('/api/login')
def api_login():
    auth = request.authorization

    if not auth or not auth.username or not auth.password:
        return make_response('Could not verify', 401, {'WWW-Authenticate' : 'Basic realm="Login required!"'})

    user = User.query.filter_by(username=auth.username).first()

    if not user:
        return make_response('Could not verify', 401, {'WWW-Authenticate' : 'Basic realm="Login required!"'})

    if bcrypt.check_password_hash(user.password, auth.password):
        session['logged_in'] = True
        session['username'] = user.username
        session['admin'] = user.admin
        #token = jwt.encode({'id' : user.id, 'exp' : datetime.datetime.utcnow() + datetime.timedelta(minutes=30)}, ap
        token = jwt.encode({'id' : user.id, 'exp' : datetime.now() + timedelta(minutes=30)}, app.config['SECRET_KEY']

        #return jsonify({'token' : token.decode('UTF-8')})
        return jsonify({'token' : token})

    return make_response('Could not verify', 401, {'WWW-Authenticate' : 'Basic realm="Login required!"'})
```

Request JSON

Return JSON

WEB

```python
@app.route("/login", methods=["GET","POST"])
def login():
    msg = None
    if 'username' in session:
        msg = 'You are logged in as ' + session['username'] + '.'

    if request.method == "POST":
        username = request.form['username']
        password = request.form['password']
        user = User.query.filter_by(username=username).first()
        if not user:
            print("Account does not exist!")
            msg = "Account does not exist!"
        else:
            if bcrypt.check_password_hash(user.password, password):
                session['logged_in'] = True
                session['username'] = user.username
                session['admin'] = user.admin
                print("Welcome!")
                msg = "Welcome!"
                return redirect(url_for('profile', username=username))
            msg = "Wrong password!"

    return render_template('login.html', msg = msg)
```

Request form data.

Return HTML

```python
@app.route("/login", methods=["GET","POST"])
def login():
    msg = None
    if 'username' in session:
        msg = 'You are logged in as ' + session['username'] + '.'

    if request.method == "POST":
        username = request.form['username']
        password = request.form['password']
        user = User.query.filter_by(username=username).first()
        if not user:
            print("Account does not exist!")
            msg = "Account does not exist!"
        else:
            if bcrypt.check_password_hash(user.password, password):
                session['logged_in'] = True
                session['username'] = user.username
                session['admin'] = user.admin
                print("Welcome!")
                msg = "Welcome!"
                return redirect(url_for('profile', username=username))
            msg = "Wrong password!"

    return render_template('login.html', msg = msg)
```

Request form data.

Server-side session variable saved for later validation.

Return HTML

```python
@app.route('/api/login')
def api_login():
    auth = request.authorization
```

Request JSON Basic Authentication JSON data.

```python
    if not auth or not auth.username or not auth.password:
        return make_response('Could not verify', 401, {'WWW-Authenticate' : 'Basic realm="Login required!"'})

    user = User.query.filter_by(username=auth.username).first()

    if not user:
        return make_response('Could not verify', 401, {'WWW-Authenticate' : 'Basic realm="Login required!"'})

    if bcrypt.check_password_hash(user.password, auth.password):
        session['logged_in'] = True
        session['username'] = user.username
        session['admin'] = user.admin
        #token = jwt.encode({'id' : user.id, 'exp' : datetime.datetime.utcnow() + datetime.timedelta(minutes=30)}, ap
        token = jwt.encode({'id' : user.id, 'exp' : datetime.now() + timedelta(minutes=30)}, app.config['SECRET_KEY']]
```

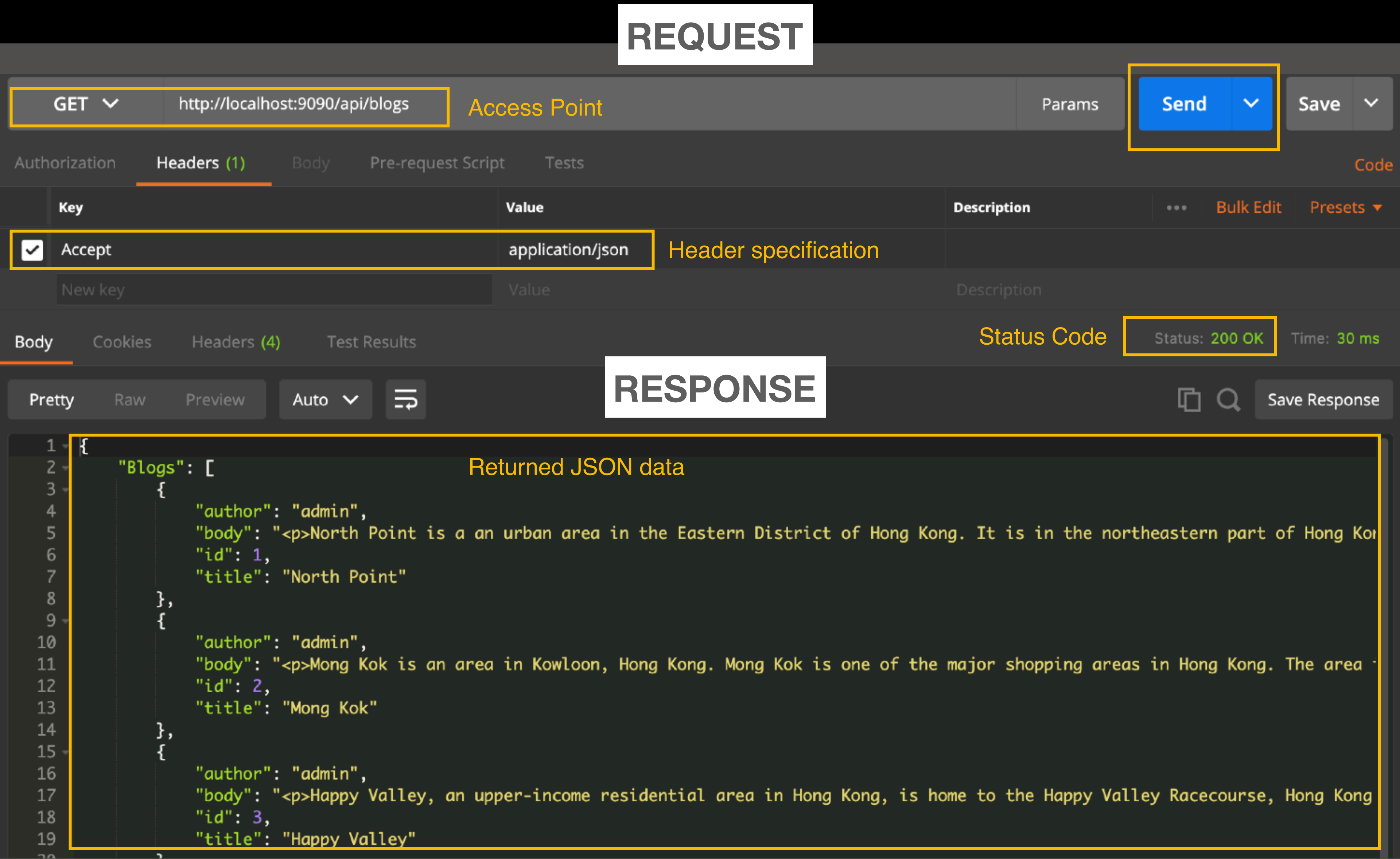
Server-side session variable and client-side JWT token saved for later validation.

```python
        #return jsonify({'token' : token.decode('UTF-8')})
        return jsonify({'token' : token})
```

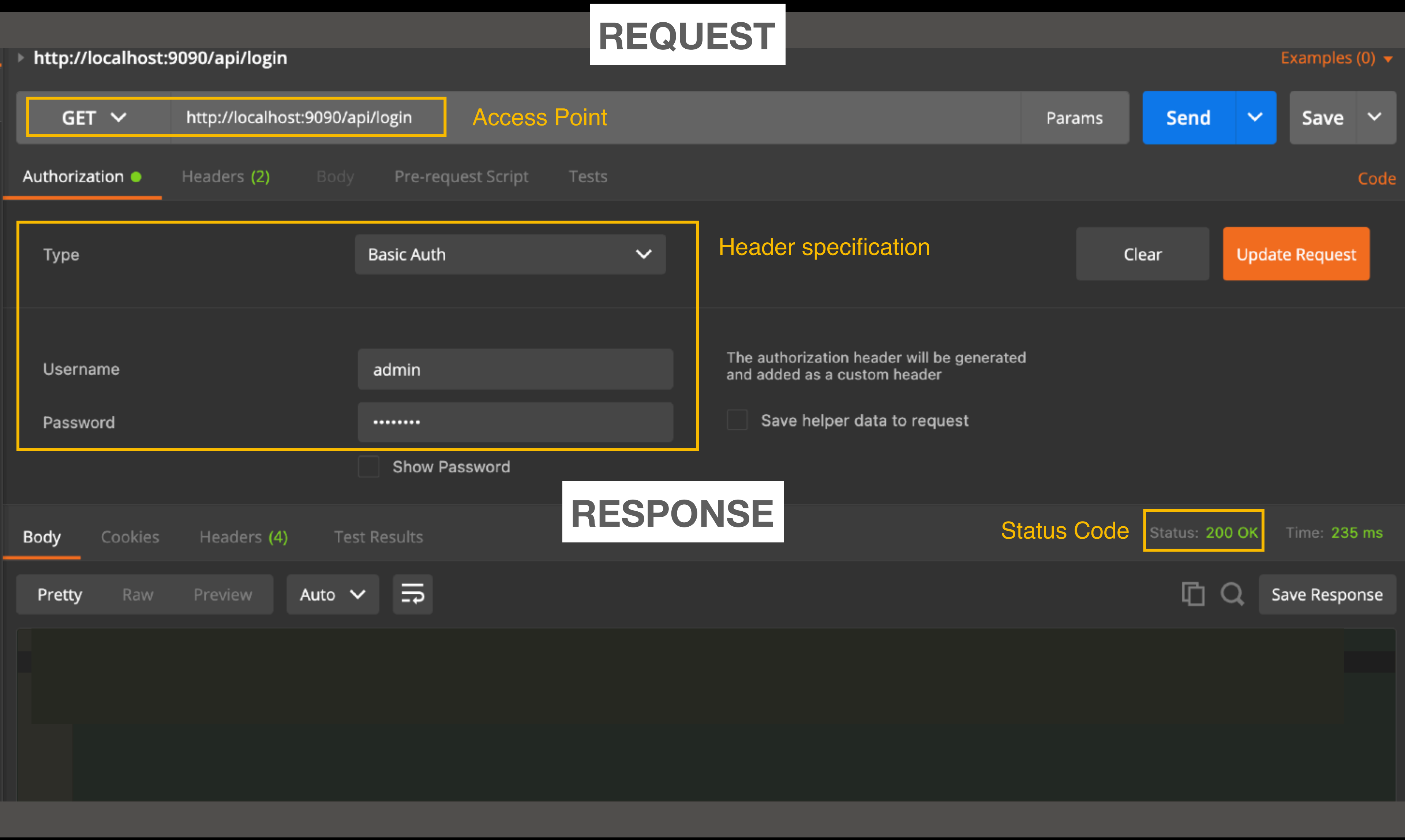
Return JSON token.

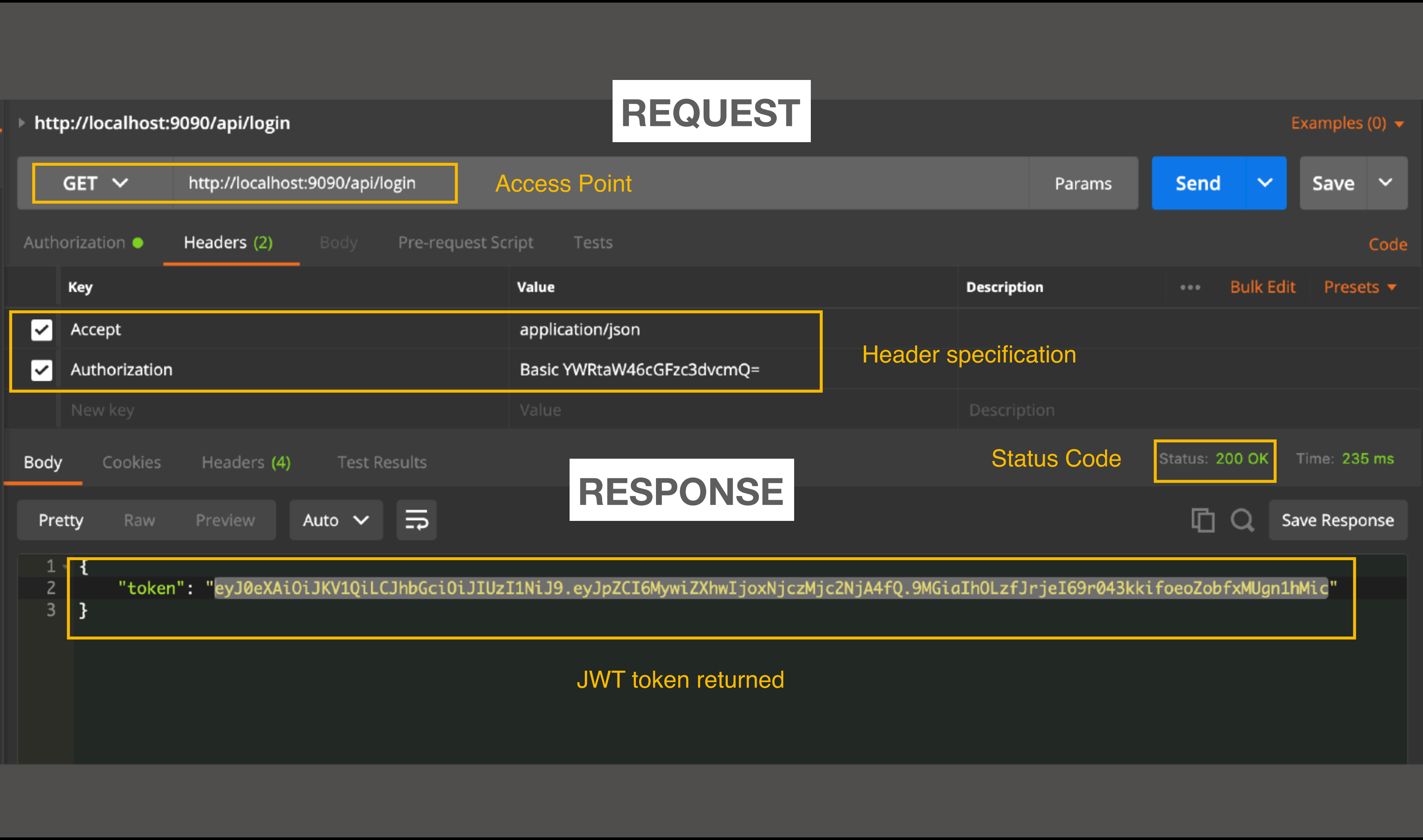
```python
    return make_response('Could not verify', 401, {'WWW-Authenticate' : 'Basic realm="Login required!"'})
```

Demonstration of Introducing Hong Kong Example.



REQUEST
GET
http://localhost:9090/api/blogs
Access Point
Params
Send
Save
Authorization
Headers (1)
Body
Pre-request Script
Tests
Code
Key
Value
Description
Bulk Edit
Presets
Accept
application/json
Header specification
New key
Value
Description
Body
Cookies
Headers (4)
Test Results
Status Code
Status: 200 OK
Time: 30 ms
RESPONSE
Pretty
Raw
Preview
Auto
Save Response
Returned JSON data
{
"Blogs": [
{
"author": "admin",
"body": "<p>North Point is a an urban area in the Eastern District of Hong Kong. It is in the northeastern part of Hong Ko
"id": 1,
"title": "North Point"
},
{
"author": "admin",
"body": "<p>Mong Kok is an area in Kowloon, Hong Kong. Mong Kok is one of the major shopping areas in Hong Kong. The area
"id": 2,
"title": "Mong Kok"
},
{
"author": "admin",
"body": "<p>Happy Valley, an upper-income residential area in Hong Kong, is home to the Happy Valley Racecourse, Hong Kong
"id": 3,
"title": "Happy Valley"

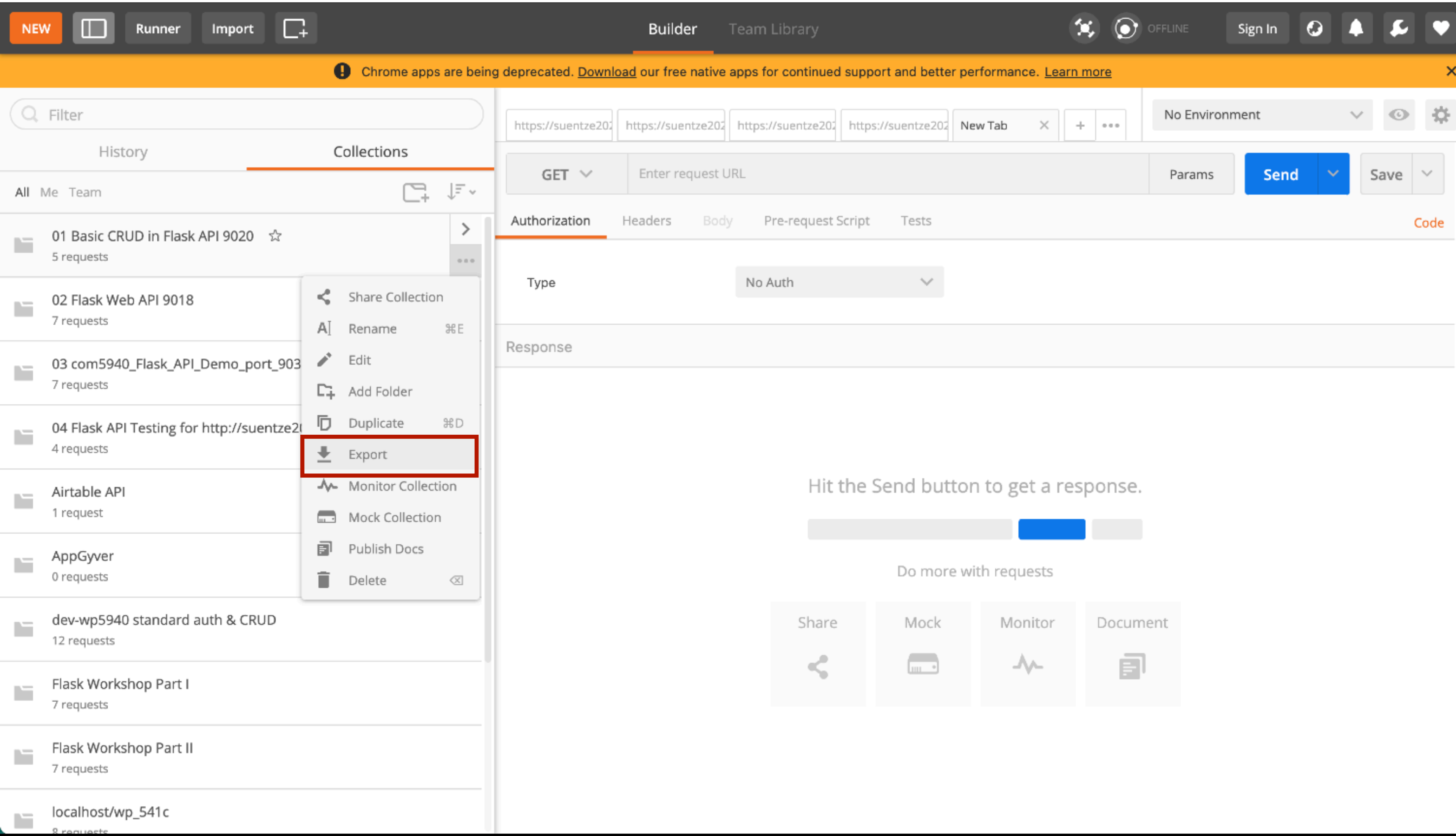

REQUEST
http://localhost:9090/api/login
Examples (0)
GET
http://localhost:9090/api/login
Access Point
Params
Send
Save
Authorization
Headers (2)
Body
Pre-request Script
Tests
Code
Type
Basic Auth
Header specification
Clear
Update Request
Username
admin
Password
The authorization header will be generated and added as a custom header
Save helper data to request
Show Password
RESPONSE
Body
Cookies
Headers (4)
Test Results
Status Code
Status: 200 OK
Time: 235 ms
Pretty
Raw
Preview
Auto
Save Response

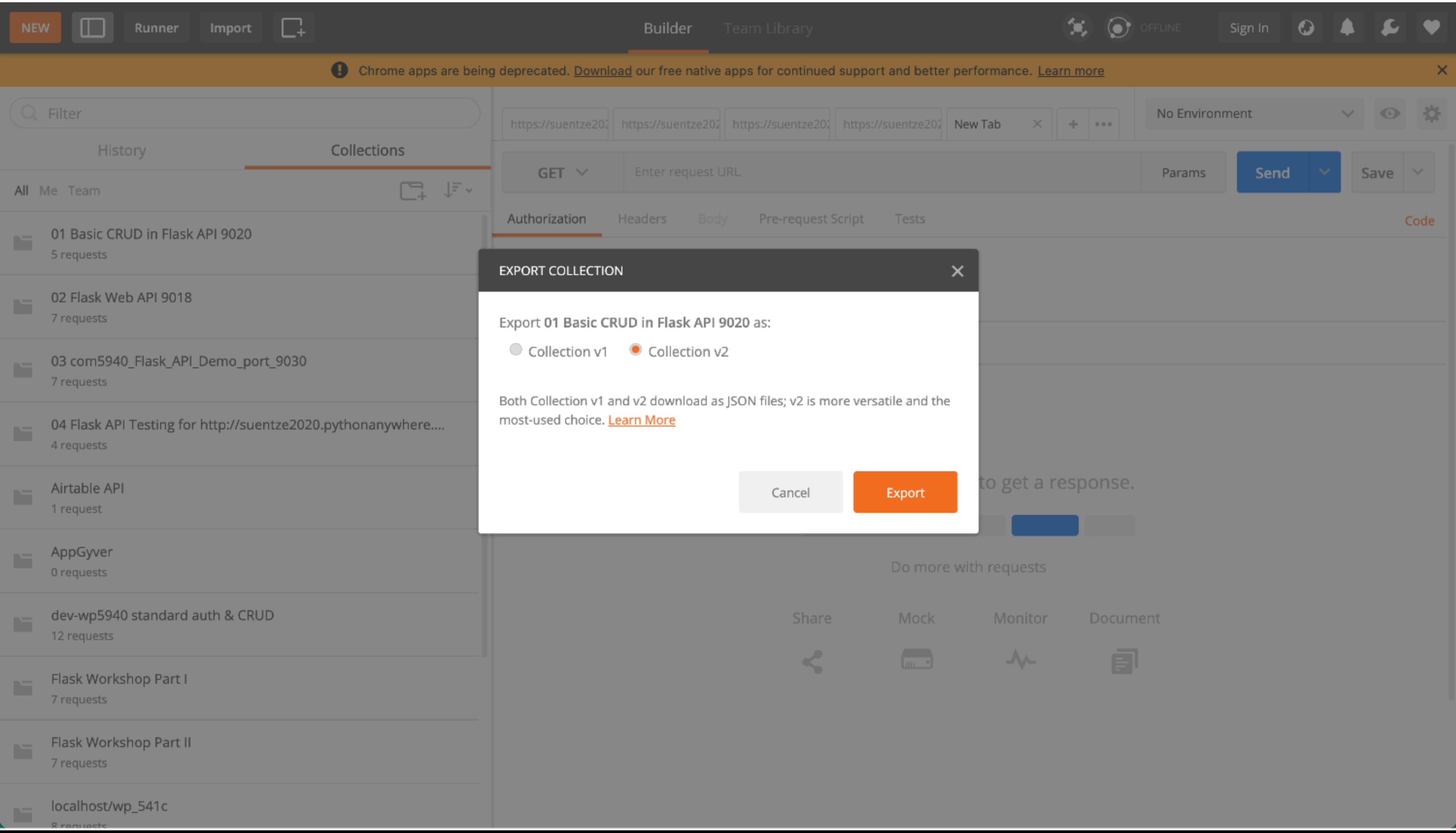


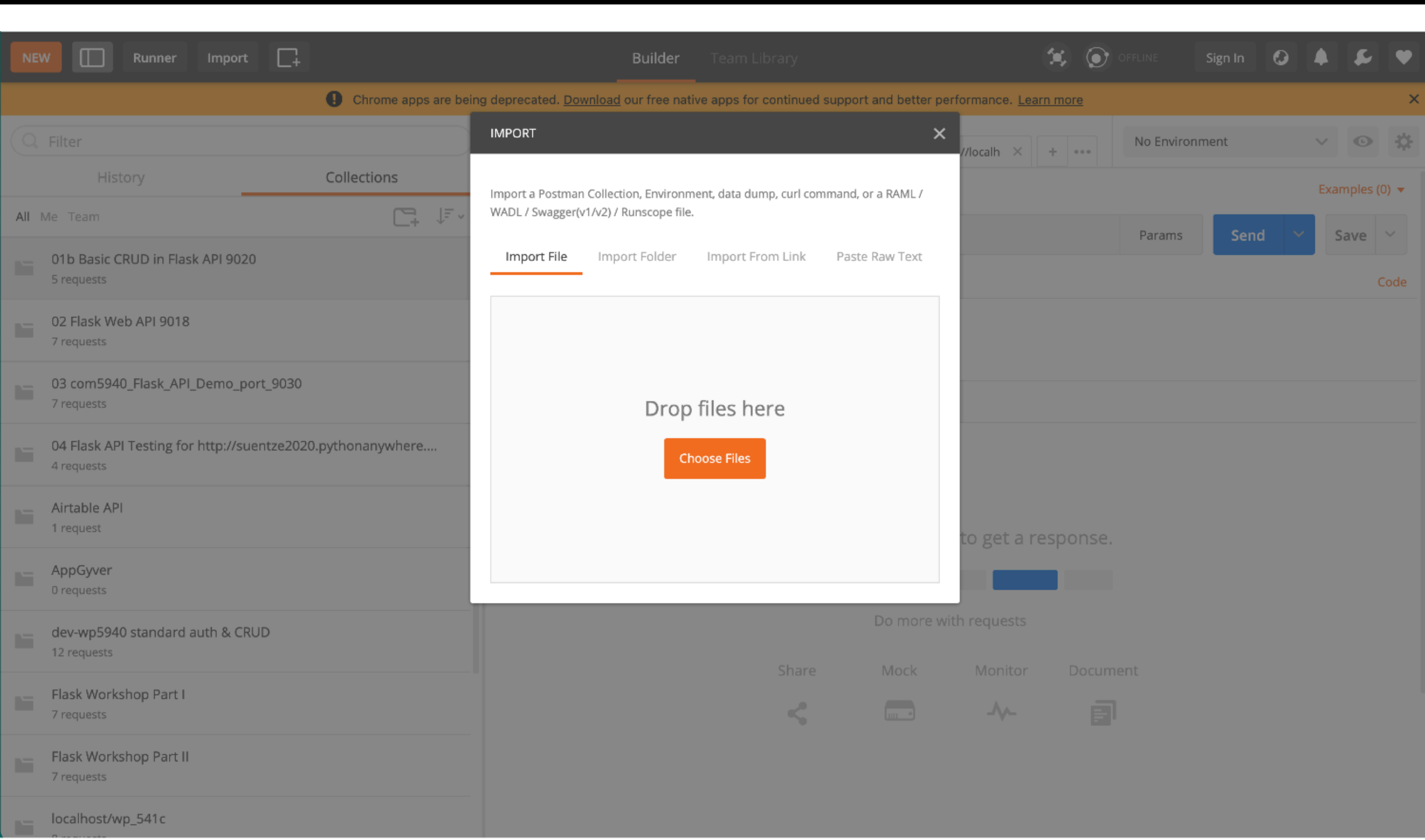
REQUEST
http://localhost:9090/api/login
Examples (0)
GET
http://localhost:9090/api/login
Access Point
Params
Send
Save
Authorization
Headers (2)
Body
Pre-request Script
Tests
Code
Key
Value
Description
Bulk Edit
Presets
Accept
application/json
Authorization
Basic YWRtaW46cGFzc3dvcmQ=
Header specification
New key
Value
Description
Body
Cookies
Headers (4)
Test Results
Status Code
Status: 200 OK
Time: 235 ms
RESPONSE
Pretty
Raw
Preview
Auto
Save Response
{
"token": "eyJ0eXAiOiJKV1QiLCJhbGciOiJIUzI1NiJ9.eyJpZCI6MywiZXhwIjoxNjczMjc2NjA4fQ.9MGiaIhOLzfJrjeI69r043kkifoeoZobfxMUgn1hMic"
}
JWT token returned

Problem Set #1

1. Create CRUD API for your COM5961 project in Jupyter Notebook with authentication.
2. Add JWT token required to protect against unauthorised access.
3. Add login authentication to release access token.
4. Submit your Jupyter Notebook and Postman JSON export to github.
5. Create a Github assignment page to document your work.


NEW
Runner
Import
Builder
Team Library
OFFLINE
Sign In
Chrome apps are being deprecated. Download our free native apps for continued support and better performance. Learn more
Filter
History
Collections
All Me Team
01 Basic CRUD in Flask API 9020
5 requests
02 Flask Web API 9018
7 requests
03 com5940_Flask_API_Demo_port_903
7 requests
04 Flask API Testing for http://suentze2
4 requests
Airtable API
1 request
AppGyver
0 requests
dev-wp5940 standard auth & CRUD
12 requests
Flask Workshop Part I
7 requests
Flask Workshop Part II
7 requests
localhost/wp_541c
Share Collection
Rename ⌘E
Edit
Add Folder
Duplicate ⌘D
Export
Monitor Collection
Mock Collection
Publish Docs
Delete
https://suentze202
New Tab
No Environment
GET
Enter request URL
Params
Send
Save
Authorization
Headers
Body
Pre-request Script
Tests
Code
Type
No Auth
Response
Hit the Send button to get a response.
Do more with requests
Share
Mock
Monitor
Document


NEW
Runner
Import
Builder
Team Library
OFFLINE
Sign In
Chrome apps are being deprecated. Download our free native apps for continued support and better performance. Learn more
Filter
History
Collections
All Me Team
01 Basic CRUD in Flask API 9020
5 requests
02 Flask Web API 9018
7 requests
03 com5940_Flask_API_Demo_port_9030
7 requests
04 Flask API Testing for http://suentze2020.pythonanywhere....
4 requests
Airtable API
1 request
AppGyver
0 requests
dev-wp5940 standard auth & CRUD
12 requests
Flask Workshop Part I
7 requests
Flask Workshop Part II
7 requests
localhost/wp_541c
New Tab
No Environment
GET
Enter request URL
Params
Send
Save
Authorization
Headers
Body
Pre-request Script
Tests
Code
EXPORT COLLECTION
Export 01 Basic CRUD in Flask API 9020 as:
Collection v1
Collection v2
Both Collection v1 and v2 download as JSON files; v2 is more versatile and the most-used choice. Learn More
Cancel
Export
to get a response.
Do more with requests
Share
Mock
Monitor
Document


NEW
Runner
Import
Builder
Team Library
OFFLINE
Sign In
Chrome apps are being deprecated. Download our free native apps for continued support and better performance. Learn more
Filter
History
Collections
All Me Team
01b Basic CRUD in Flask API 9020
5 requests
02 Flask Web API 9018
7 requests
03 com5940_Flask_API_Demo_port_9030
7 requests
04 Flask API Testing for http://suentze2020.pythonanywhere....
4 requests
Airtable API
1 request
AppGyver
0 requests
dev-wp5940 standard auth & CRUD
12 requests
Flask Workshop Part I
7 requests
Flask Workshop Part II
7 requests
localhost/wp_541c
IMPORT
Import a Postman Collection, Environment, data dump, curl command, or a RAML / WADL / Swagger(v1/v2) / Runscope file.
Import File
Import Folder
Import From Link
Paste Raw Text
Drop files here
Choose Files
No Environment
Examples (0)
Params
Send
Save
Code
to get a response.
Do more with requests
Share
Mock
Monitor
Document

Thank you for your time!