API 101 (Python3)

DAT-A SMASH! #01:
"API with Python Fundamental"

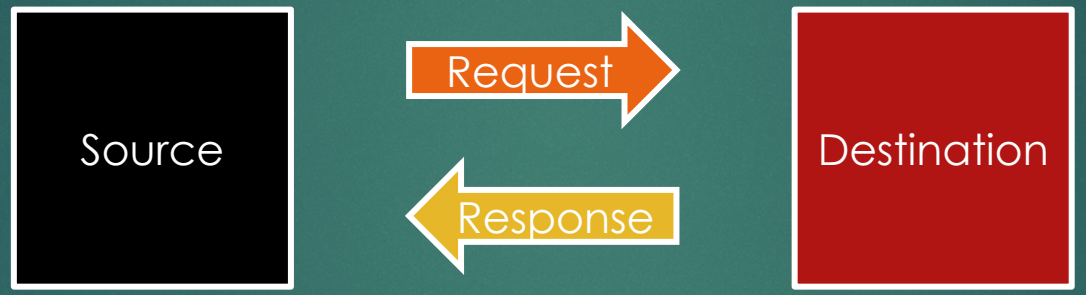
SIVAPHONG NIYOMPHANICH

Agenda

- API Overview
- Basic Network – IP Address (Public/Private), Protocol/Port, Payload
- Server & Infrastructure – Hardware, OS, (Hypervisor/Container) Middleware, Application
- Cloud – IaaS, PaaS, SaaS
- Web API
- HTTP – GET, POST, PUT, DELETE, PATCH
- Restful, JSON
- API with Python
- Flask

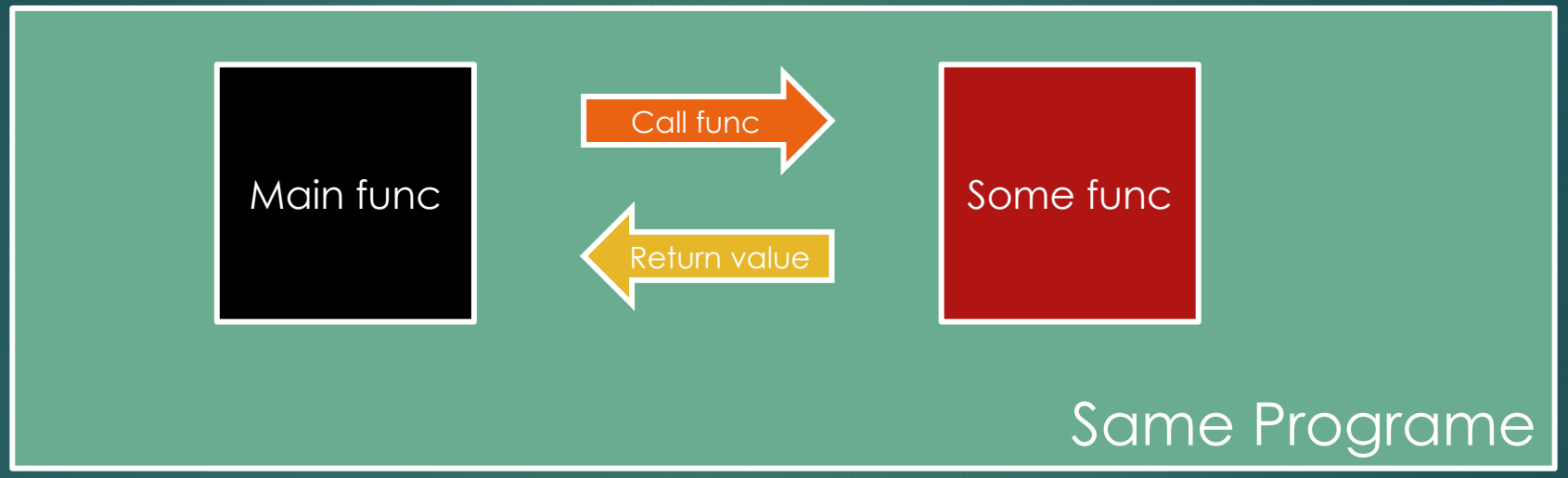
API Overview

- Connect between two or more object (and exchange data)



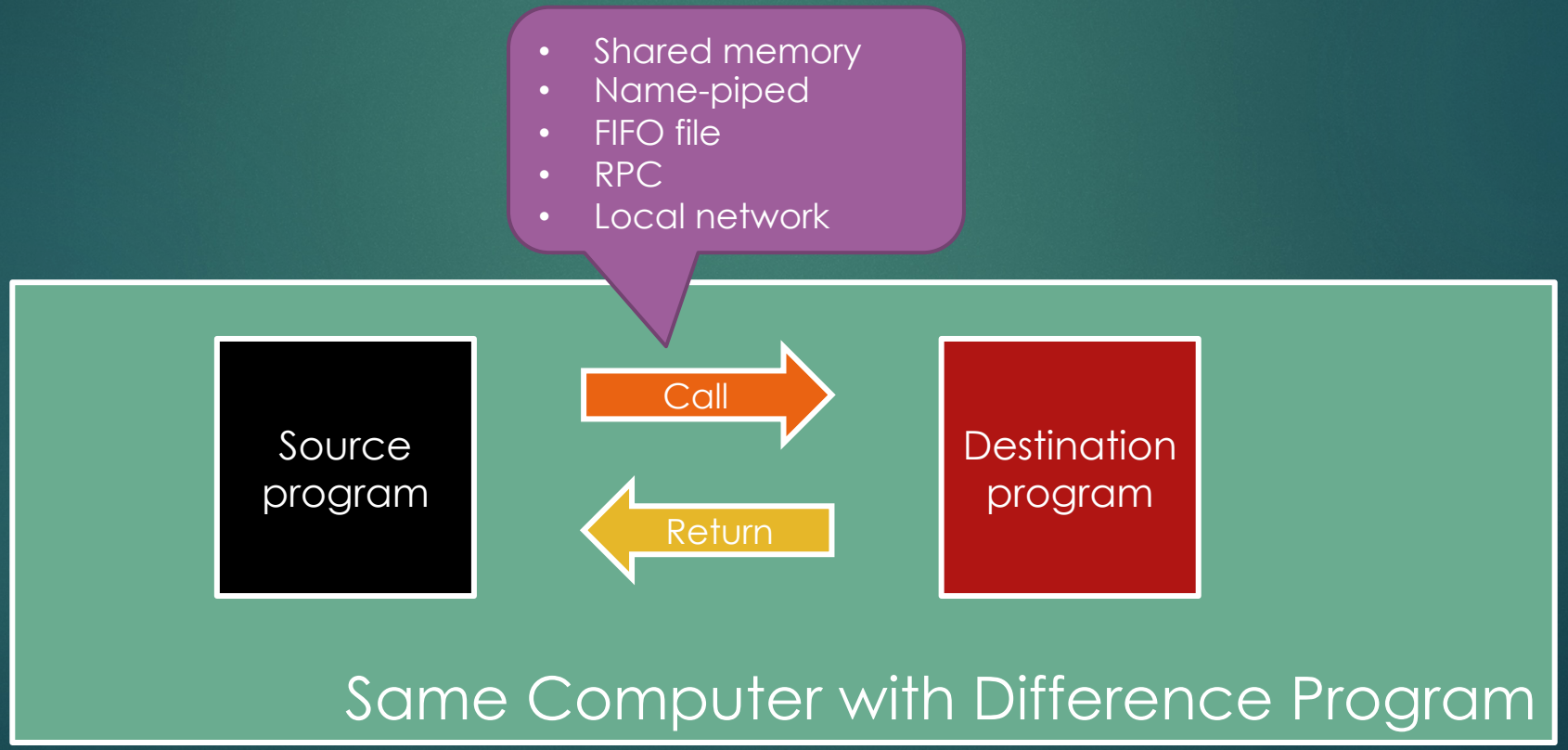
API Overview

- Connect between two or more object (and exchange data)



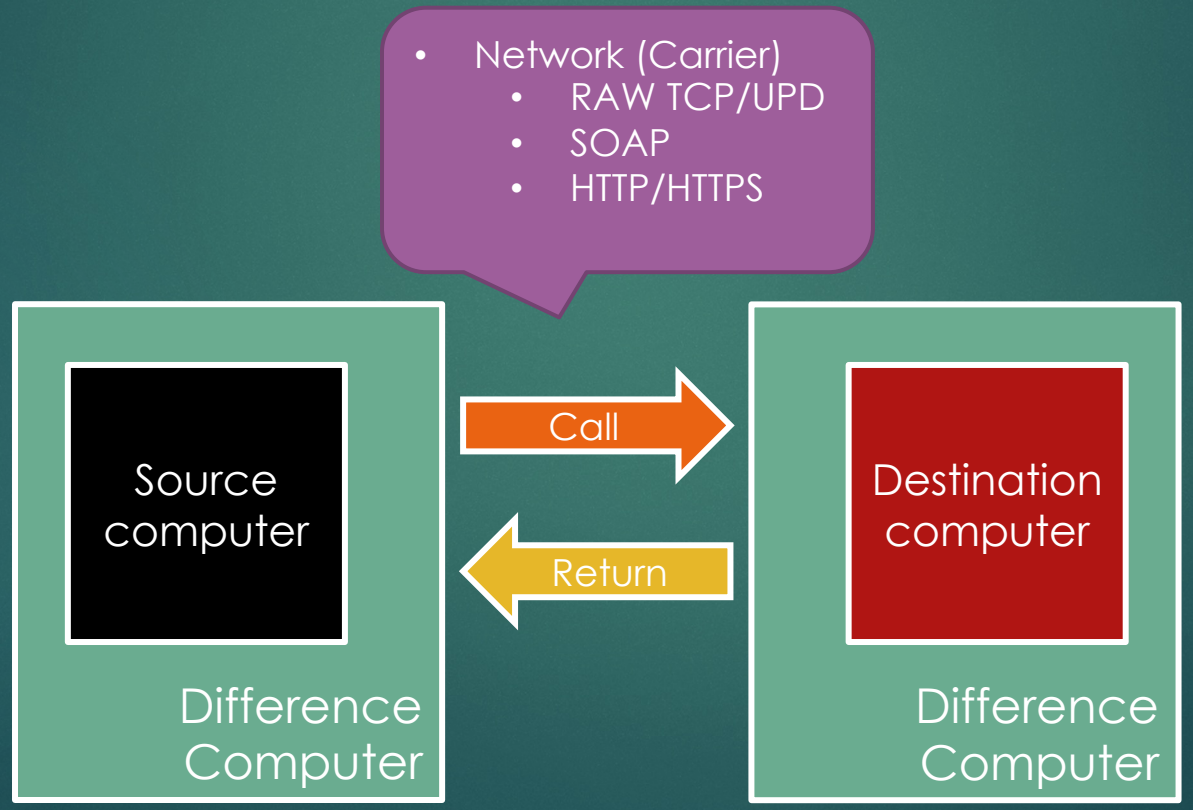
API Overview

- Connect between two or more object (and exchange data)



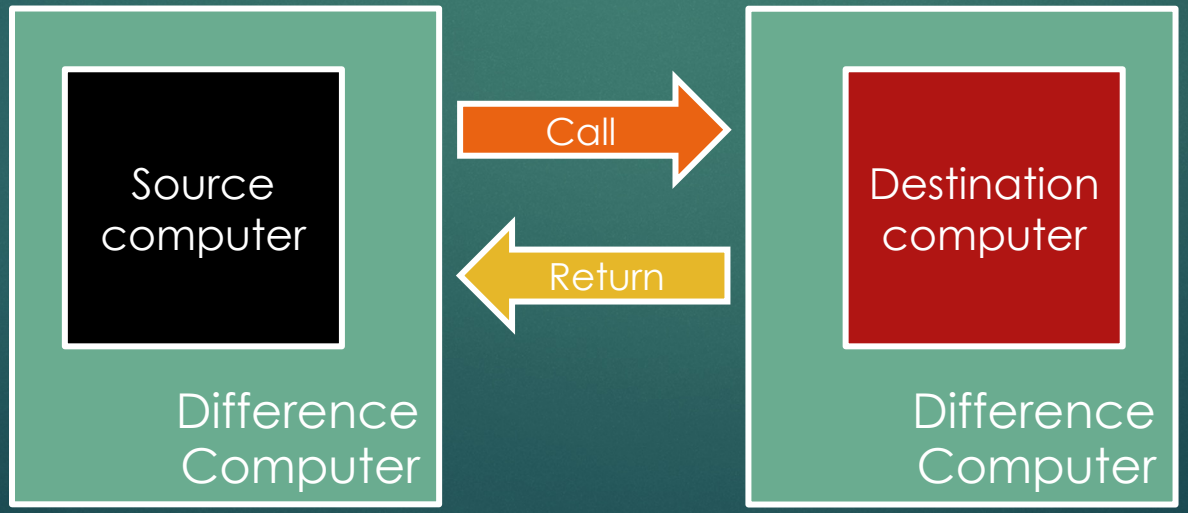
API Overview

- Connect between two or more object (and exchange data)



Computer Network Concept

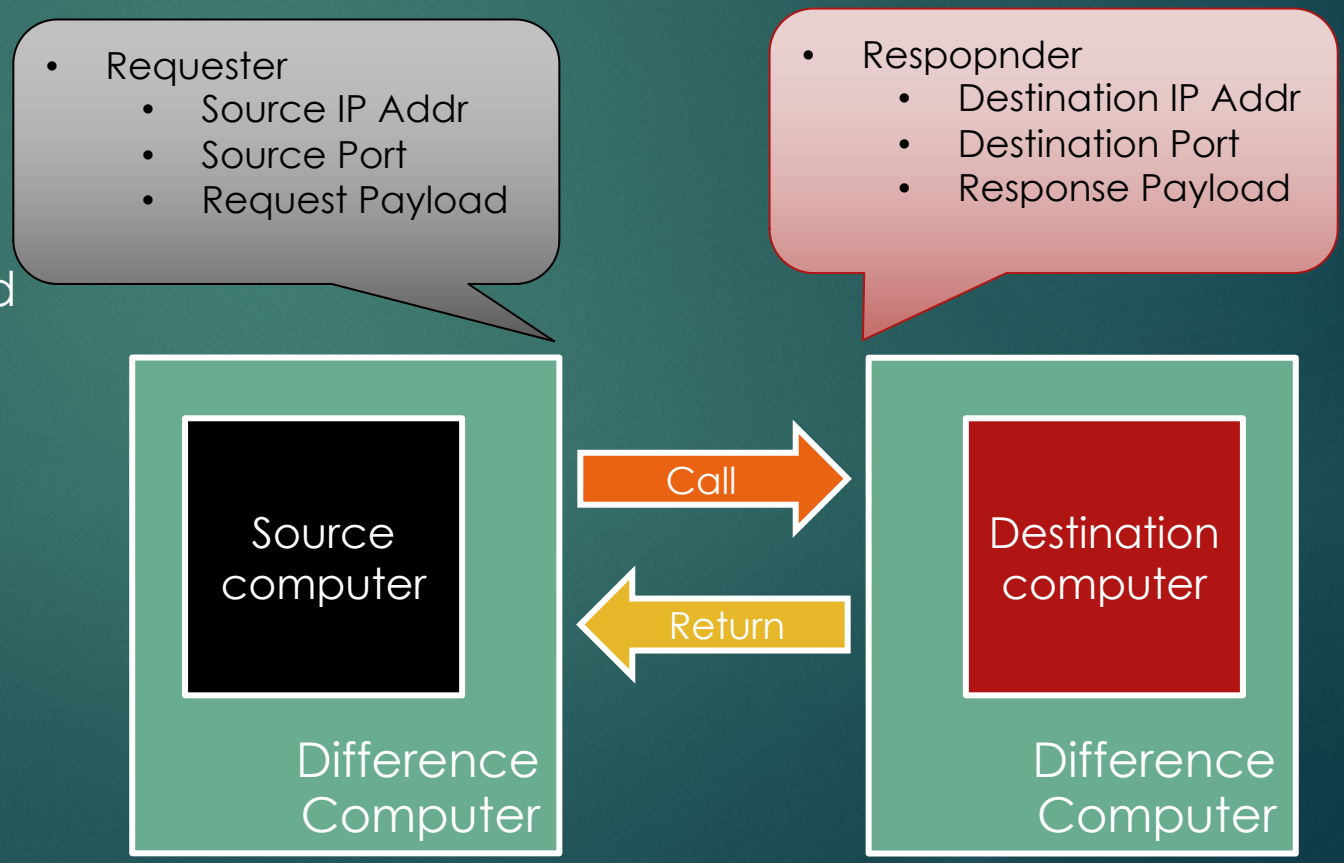
- Connect between two or more object (and exchange data)
- Concept
  - IP Address
  - Protocol/Port
  - Protocol/ Data Payload



Computer Network Concept

Pongier
8 May 2021

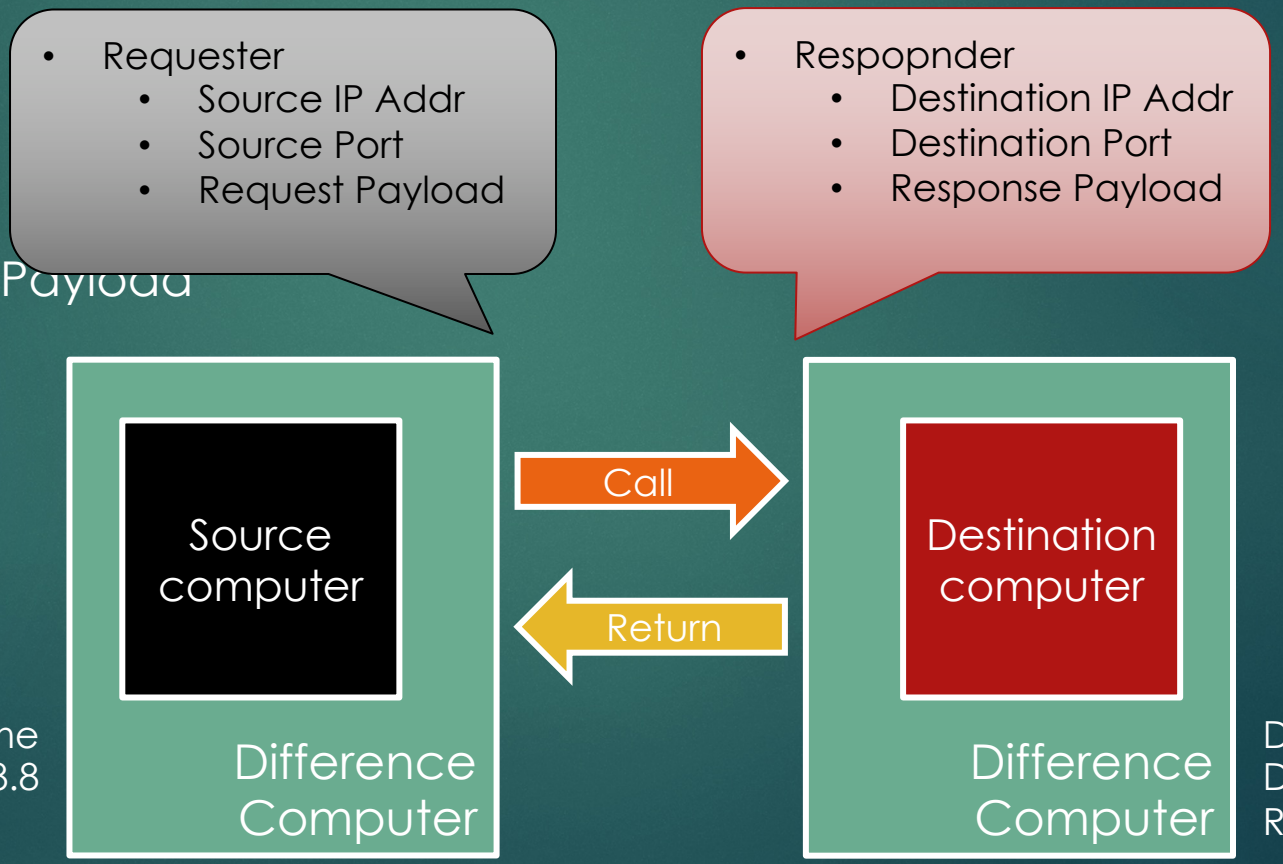
- Connect between two or more object (and exchange data)
- Concept
  - IP Address
  - Protocol/Port
  - Protocol/ Data Payload



Computer Network Concept

Pongier
8 May 2021

- Connect between two or more object (and exchange data)
- Concept
  - IP Address
  - Protocol/Port
  - Protocol/Data Payload



Open Chrome
URL: 8.8.8.8

Dst IP: 8.8.8.8
Dst Port: HTTPS/443
Res Payload: HTTP Code 202

Computer Network Concept

Pongier
8 May 2021

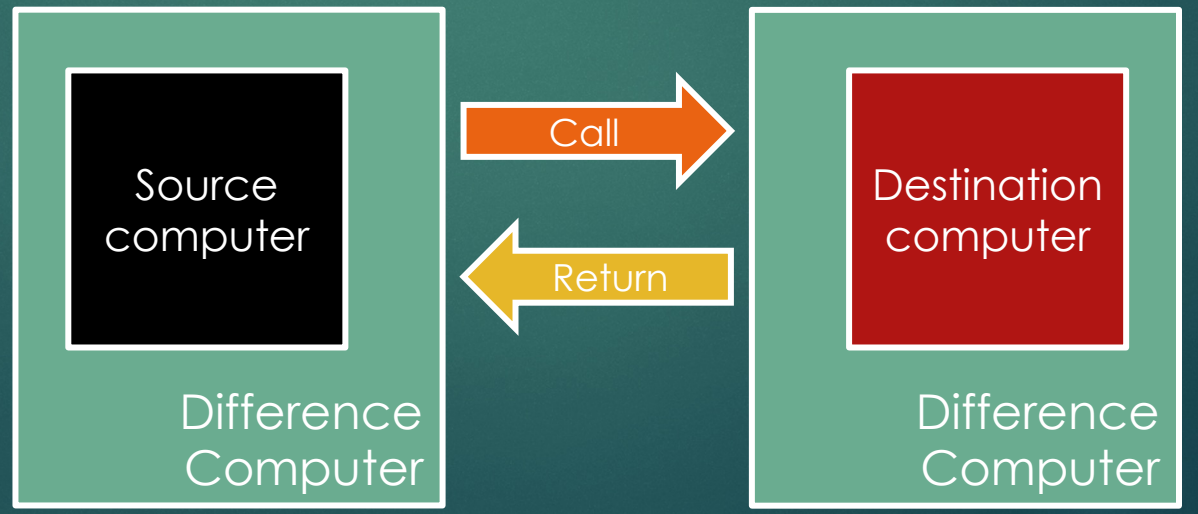
- Connect between two or more object (and exchange data)
- Concept
  - IP Address
  - Protocol/Port
  - Protocol/Data Payload

- Requester
  - Source IP Addr
  - Source Port
  - Request Payload

- Respopnder
  - Destination IP Addr
  - Destination Port
  - Response Payload

DNS
(Domain Name Service)

Convert "Name" to "IP Addr"
e.g. www,google.com = 8.8.8.8



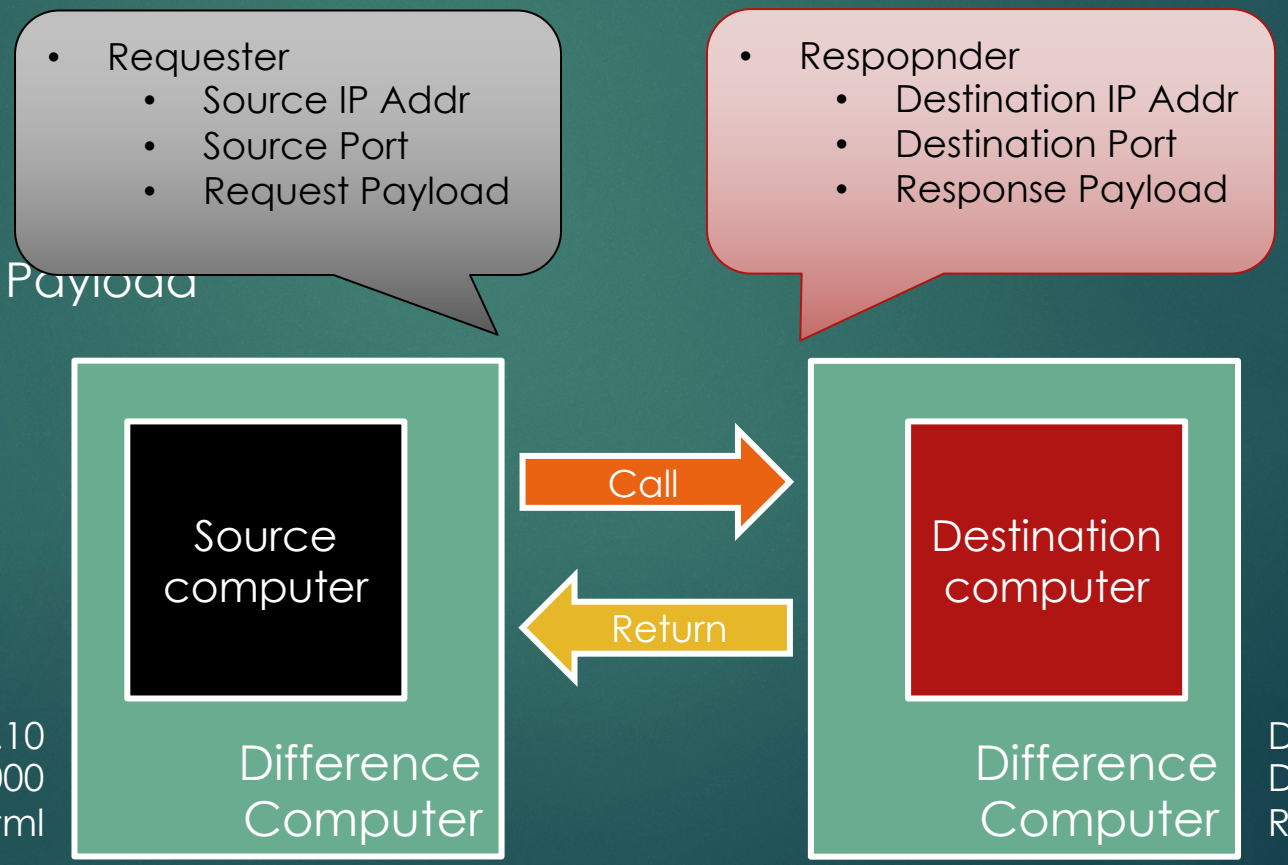
Open Chrome
URL: www.google.com

Dst IP: 8.8.8.8
Dst Port: HTTPS/443
Res Payload: HTTP Code 202

Computer Network Concept

Pongier
8 May 2021

- Connect between two or more object (and exchange data)
- Concept
  - IP Address
  - Protocol/Port
  - Protocol/Data Payload

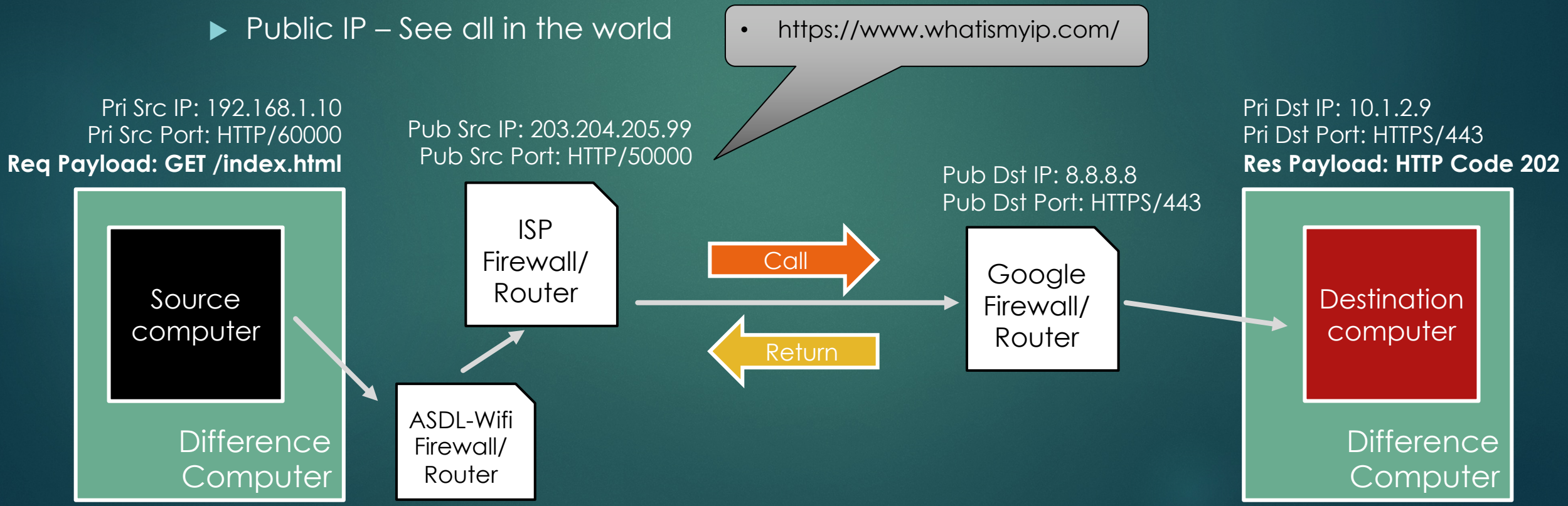


Src IP: 192.168.1.10
Src Port: HTTP/60000
Req Payload: GET /index.html

Dst IP: 192.168.1.20
Dst Port: HTTP/80
Res Payload: HTTP Code 202

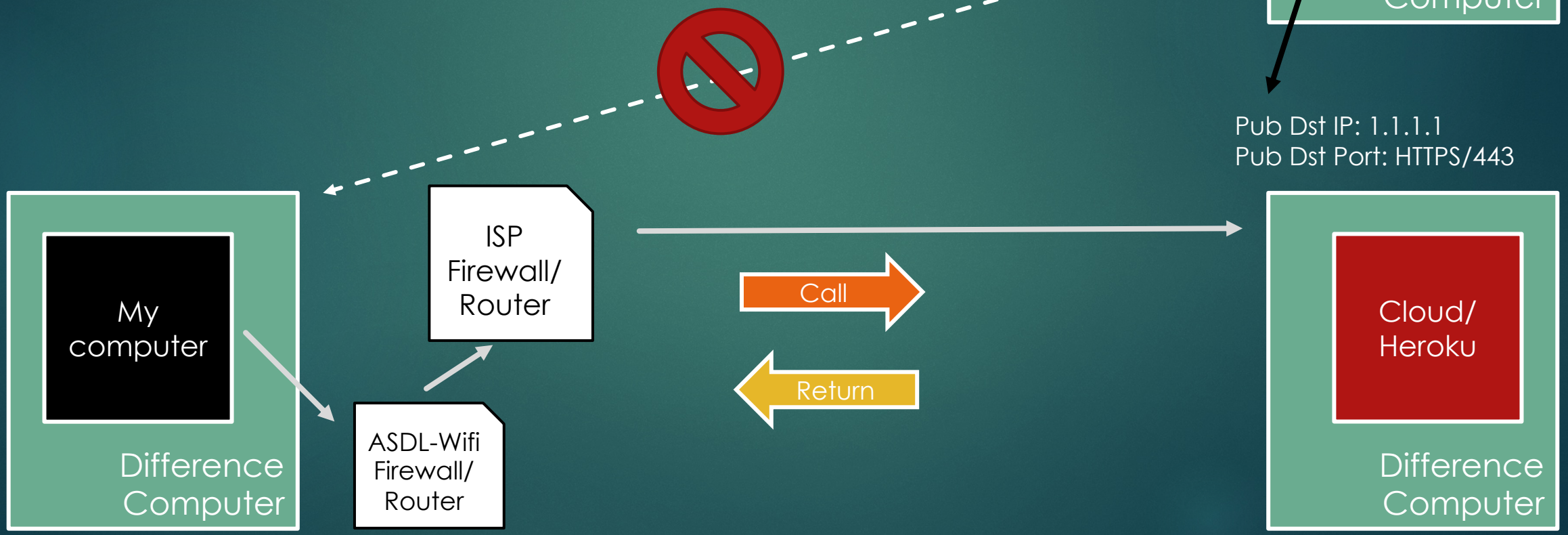
Computer Network Concept

- In Real world
  - Private IP – See only in our home, office, organization
  - Public IP – See all in the world



Computer Network Concept

- If no Public IP Address
  - Rent a cloud or heroku

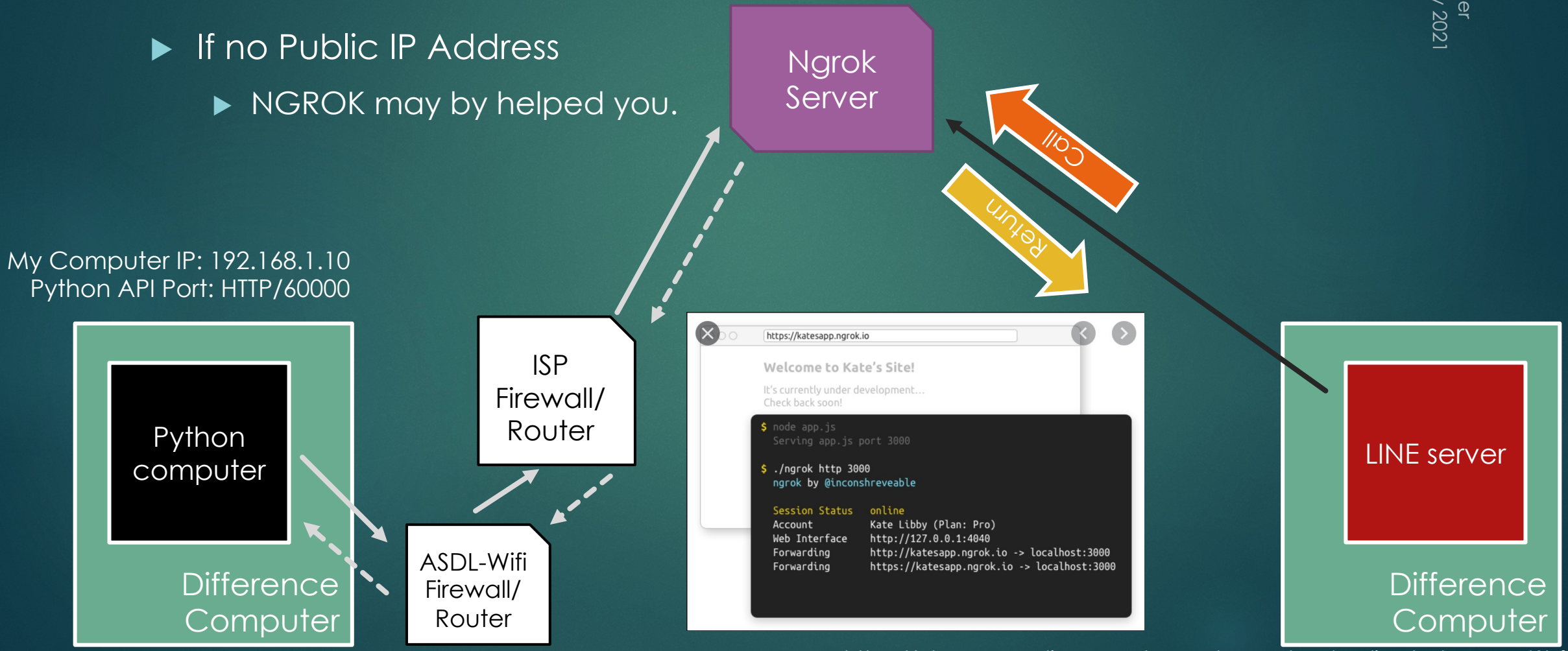


Computer Network Concept

Pongier
8 May 2021

Ngrok Pub Src IP: 203.204.205.99 (Assign with name)
Ngrok Pub Src Port: HTTP/10000 (Assigh per source computer)

- If no Public IP Address
  - NGROK may by helped you.

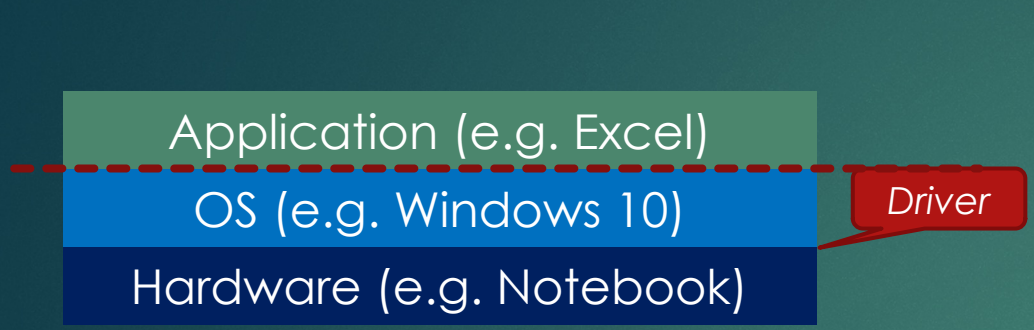


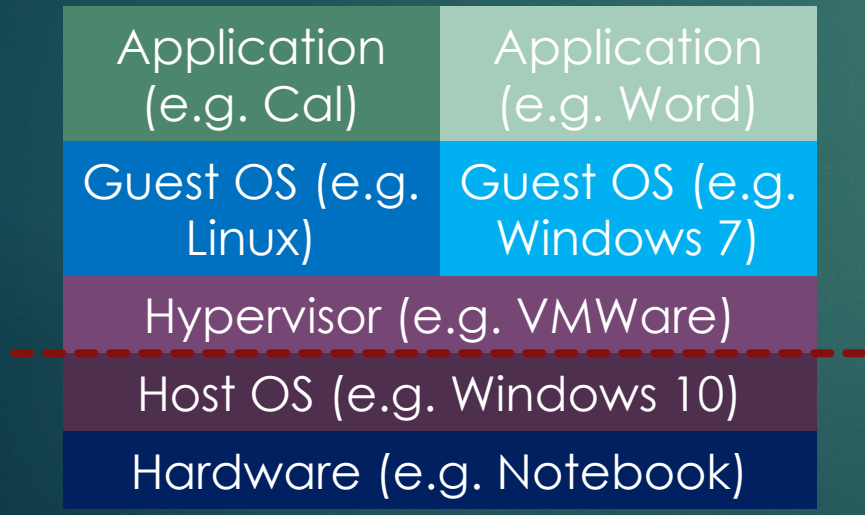
https://phayao.medium.com/ใช้-ngrok-กับการ-develop-line-bot-อย่างง่าย-5d8bf12e968f

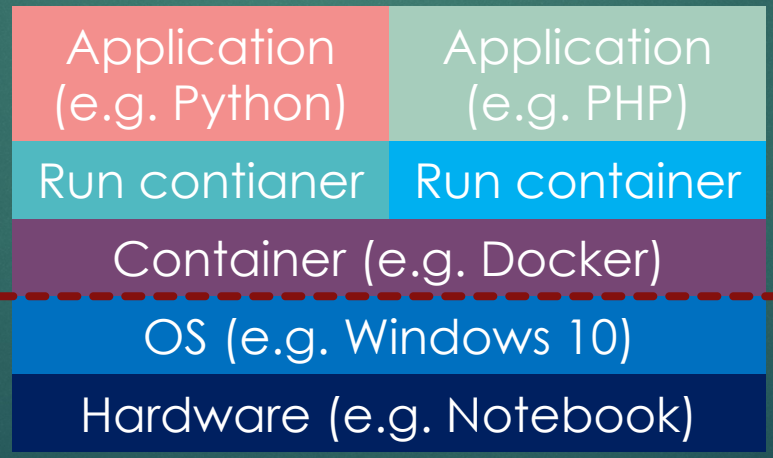
Computer Network Concept

- Connect between two or more object (and exchange data)
- Concept
  - IP Address
  - Protocol/Port (carrier)
    - HTTP (TCP/80), HTTPS (TCP/443), SSH (TCP/22), RDP (TCP/3389), DNS (UDP/53)
  - Protocol/ Data Payload
    - HTTP Method
    - JSON
    - RAW
- Protocol payload vs Data payload

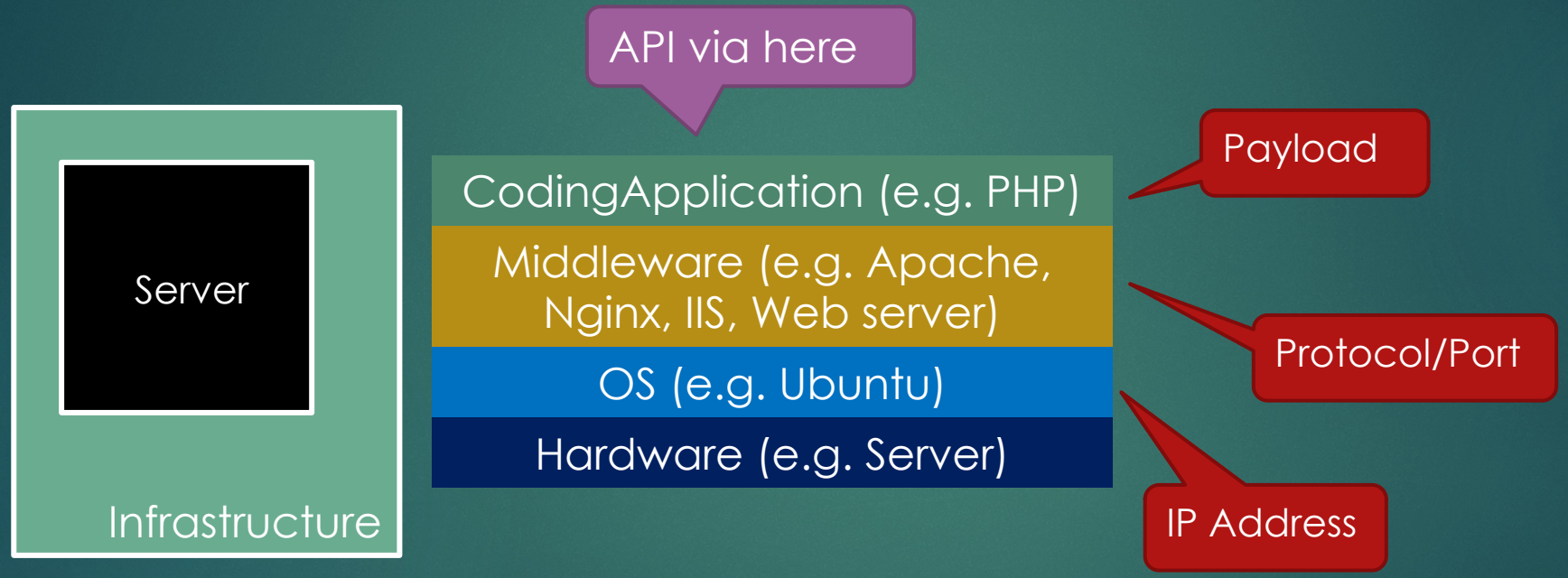
Server & Infrastructure



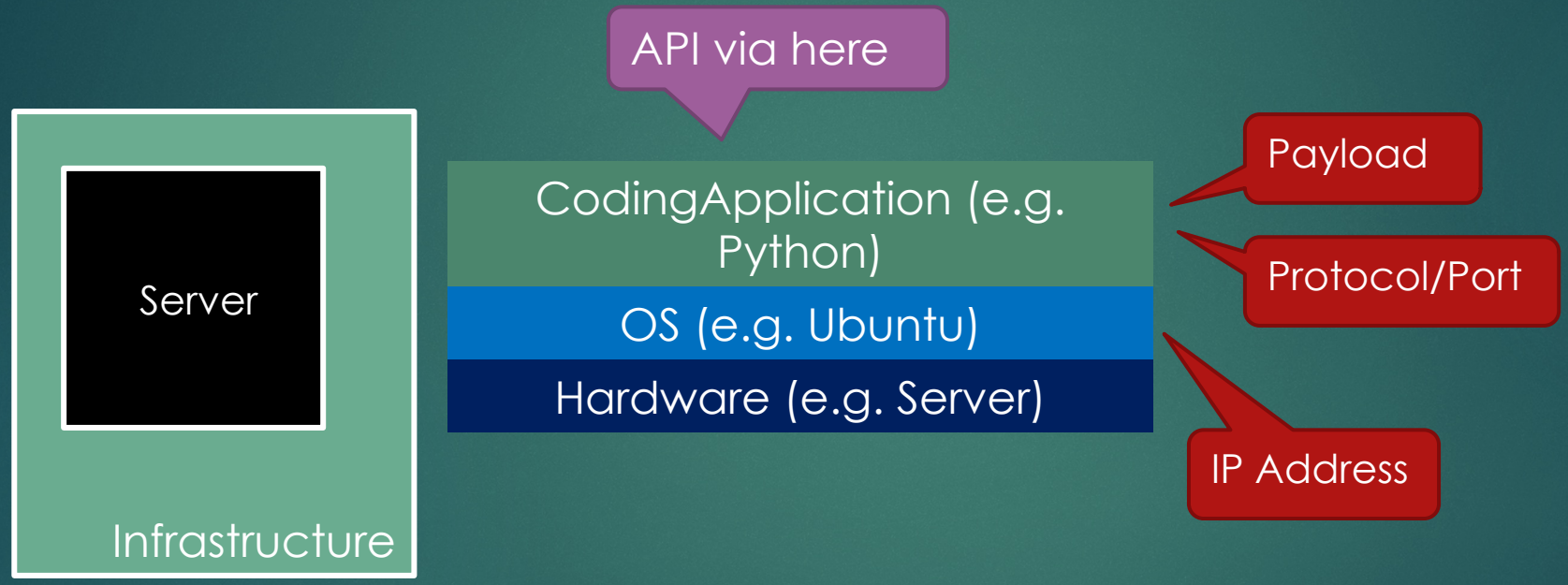




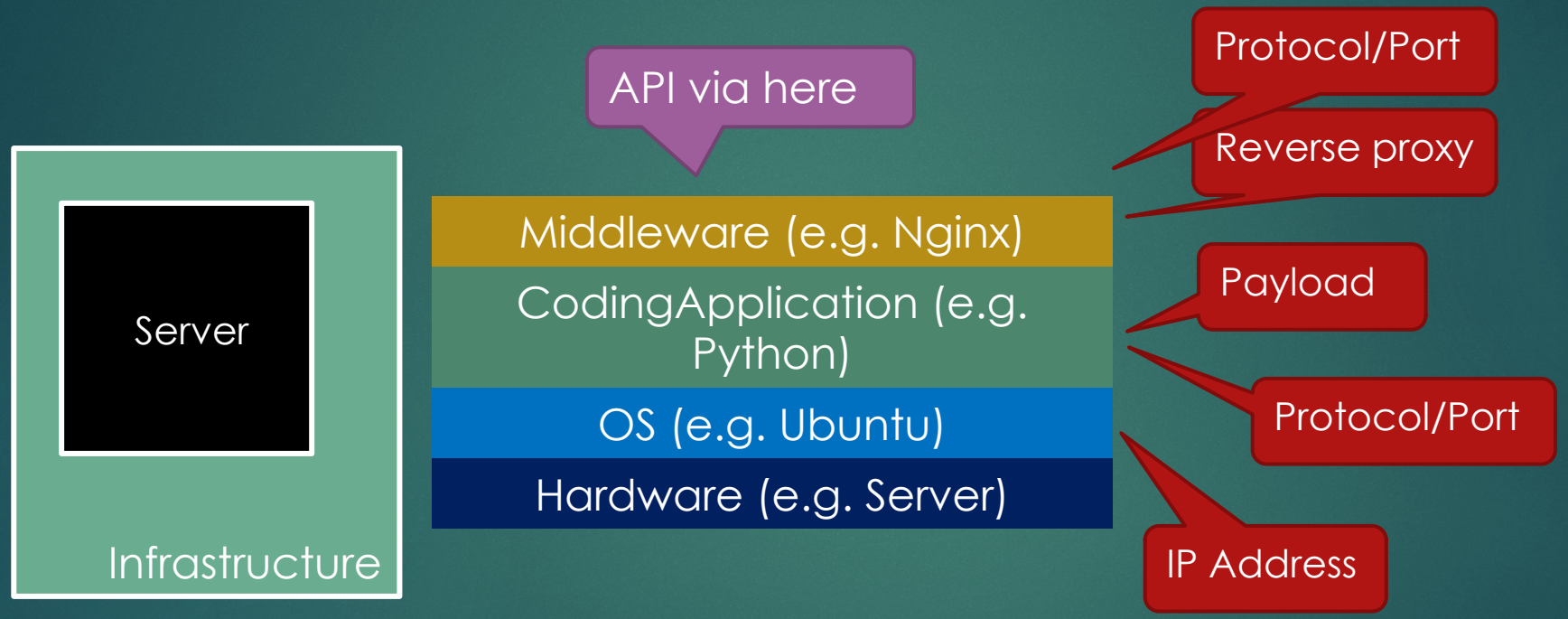
Hierarchy Example I



Hierarchy Example II



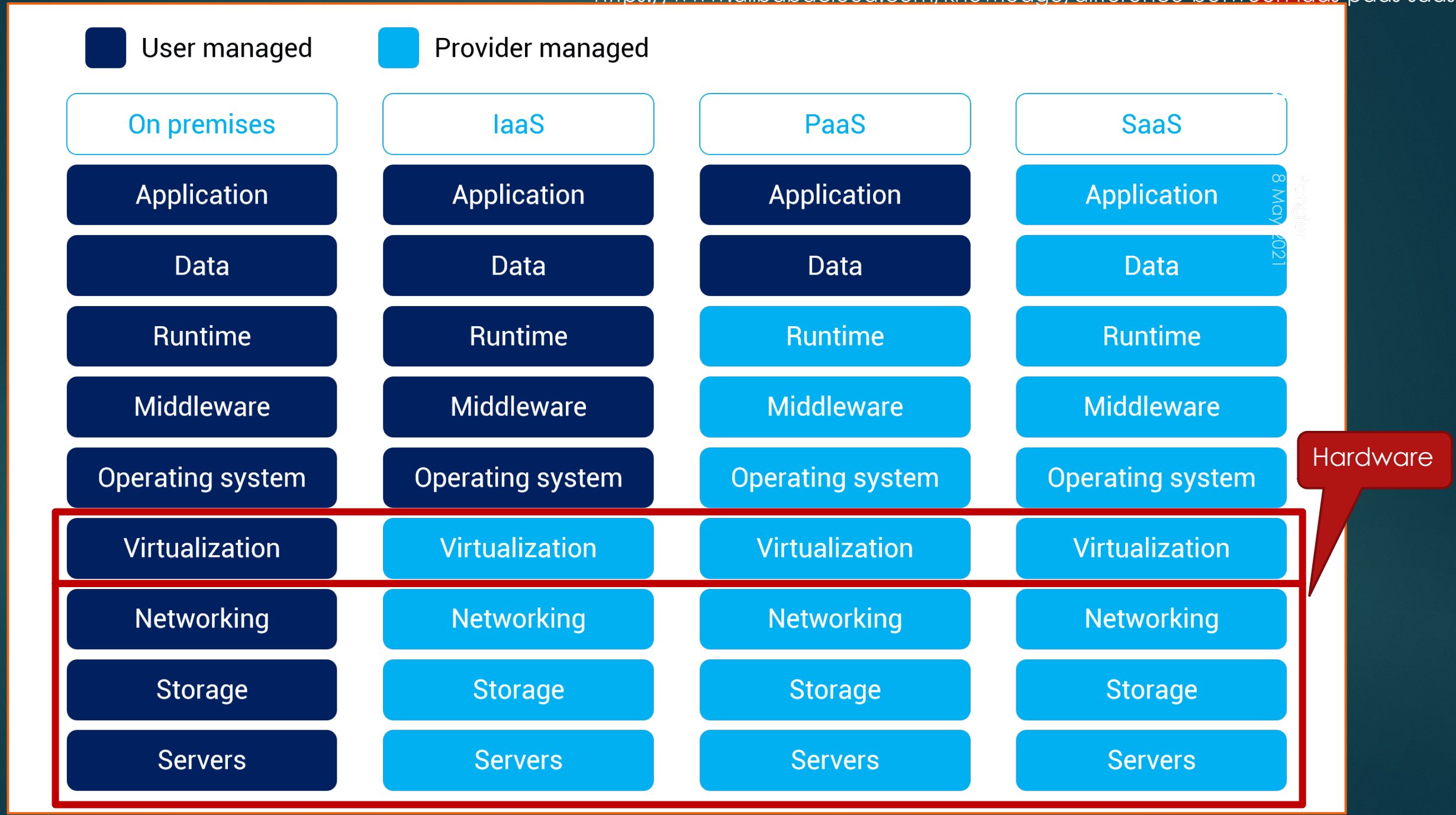
Hierarchy Example III



Cloud

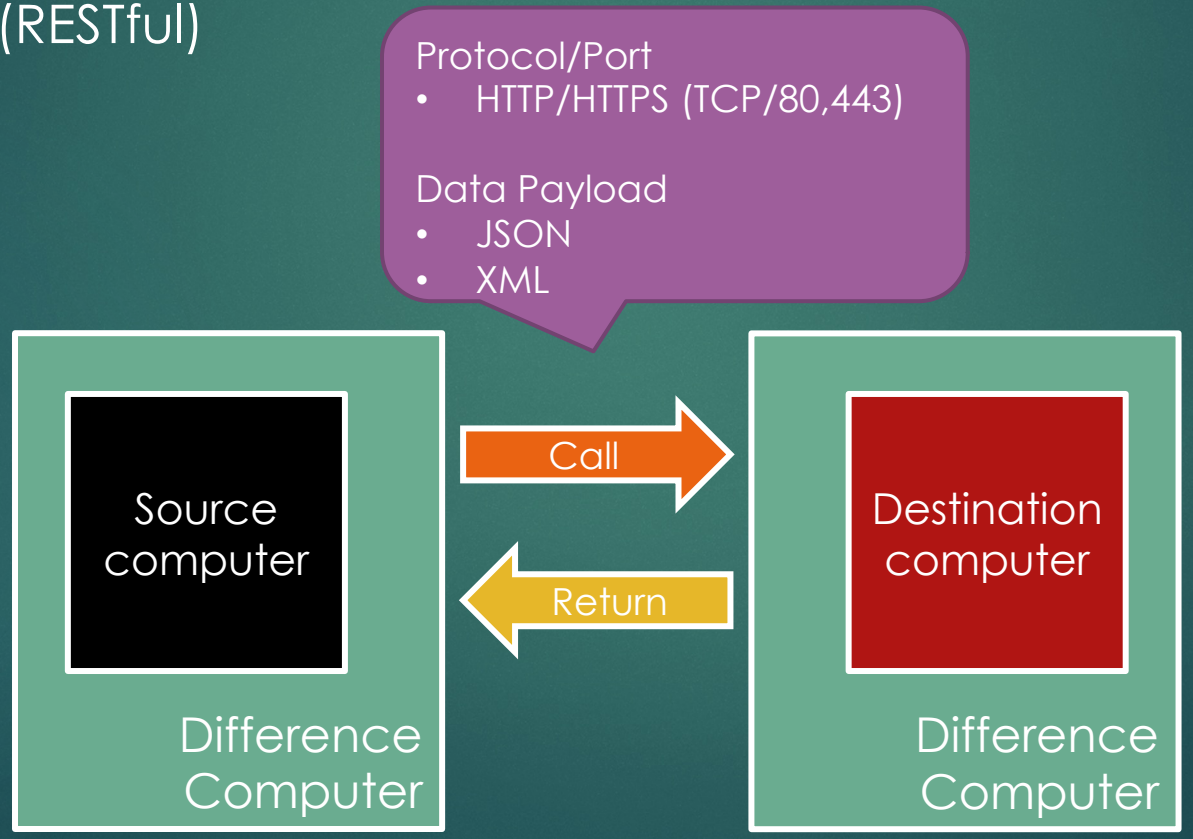
- IaaS (Infrastructure As A Service)
  - Rent a server
- PaaS (Platform As A Service)
  - Deploy a python web API
- SaaS (Software As A Service)
  - Using a mail account
  - Using a Google drive

https://www.alibabacloud.com/knowledge/difference-between-iaas-paas-saas



Web API

- Exchange data between two or more objects by using HTTP protocol (RESTful)



Understanding HTTP Protocol

- HTTP = Hyper Text Transfer Protocol
- HTTPS = Secure using SSL/TLS

- HTTP Request Method
  - GET
  - POST
  - PUT
  - DELETE
  - PATCH

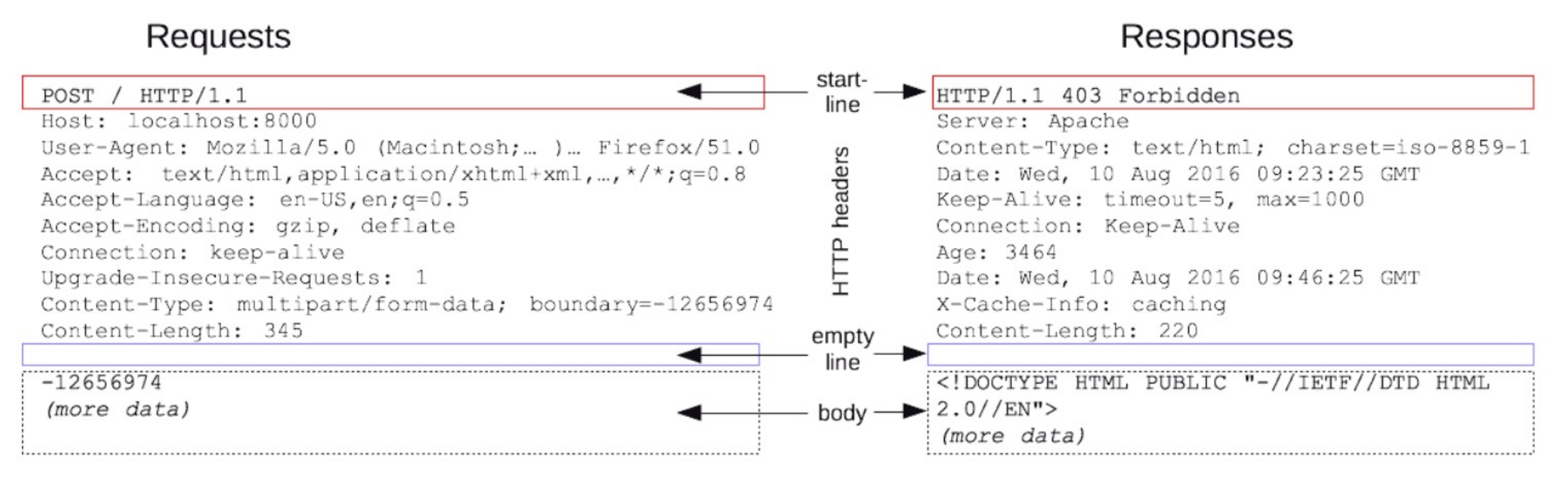
- HTTP = Hyper Text Transfer Protocol
- HTTPS = Secure using SSL/TLS

- HTTP Response Code
  - 1XX
  - 2XX
  - 3XX
  - 4XX
  - 5XX

https://developer.mozilla.org/en-US/docs/Web/HTTP/Status

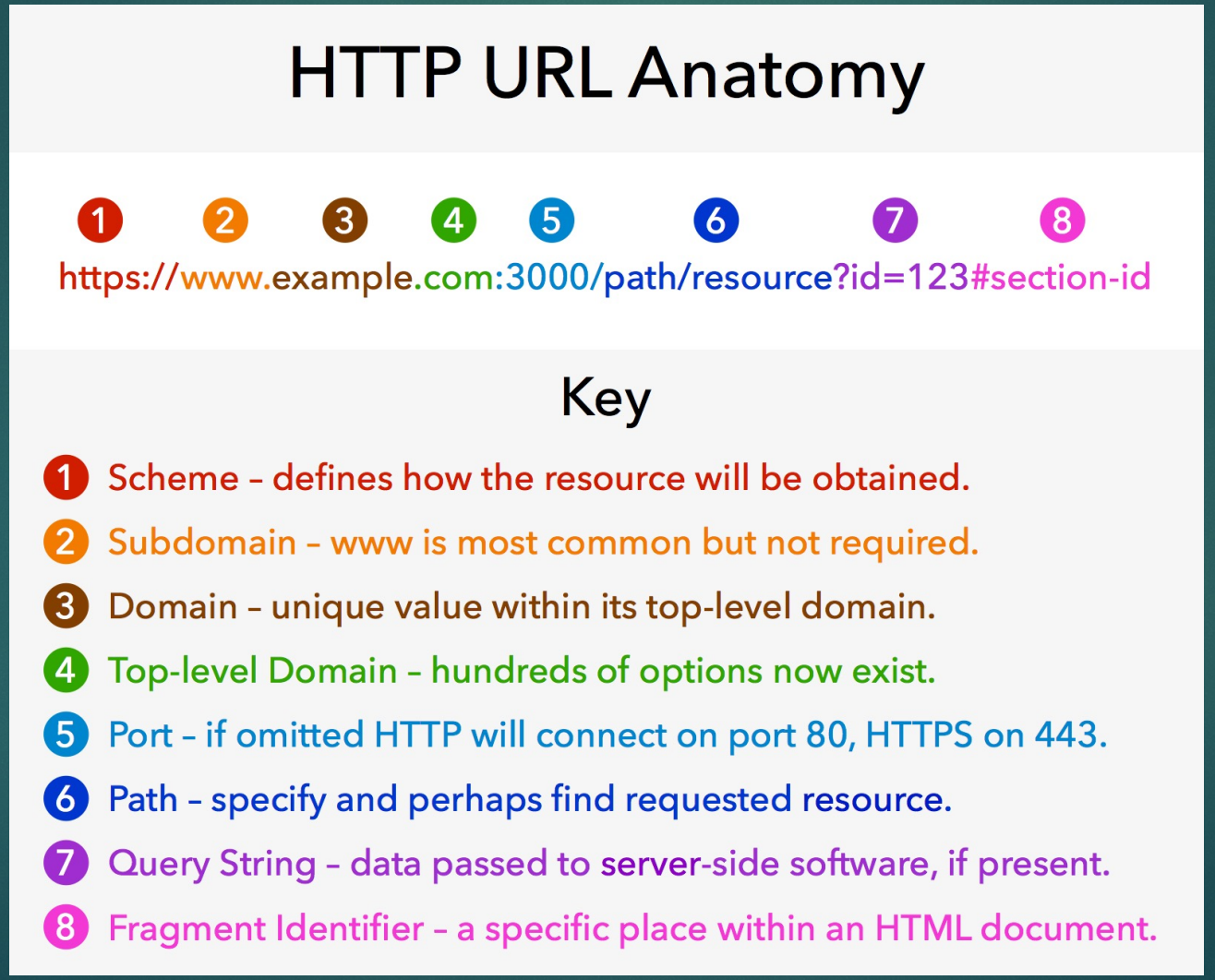
Understanding HTTP Protocol

- Payload (HTTP Message)
  - HTTP Header
  - HTTP Body



https://developer.mozilla.org/en-US/docs/Web/HTTP/Messages

Understanding HTTP Protocol



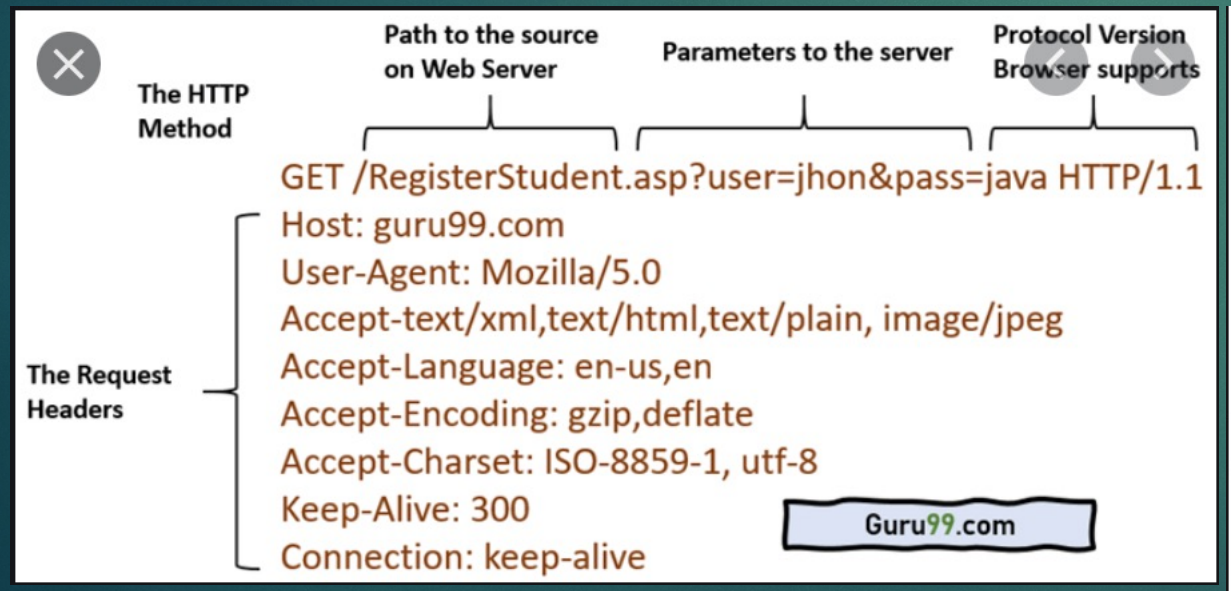
https://cascadingmedia.com/insites/2015/02/url-anatomy-seo-tips.html

Understanding HTTP Protocol

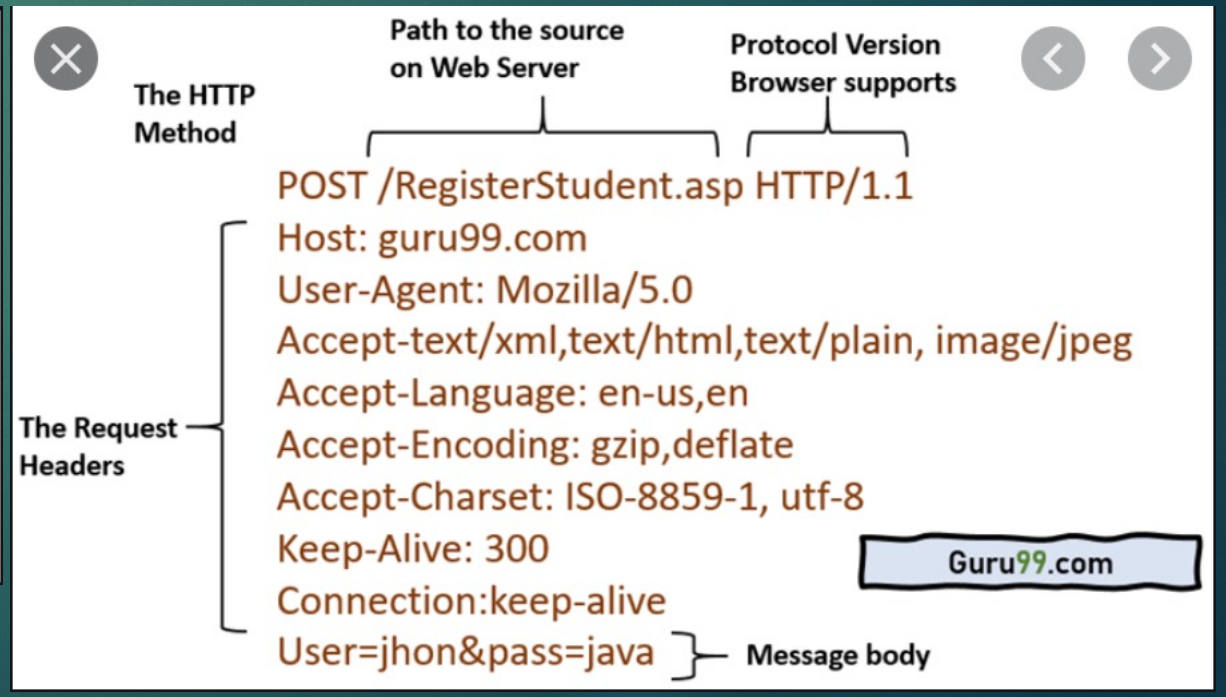
Pongier
8 May 2021

- GET vs POST method

http://guru99.com/RegisterStudent.asp?user=jhon&pass=java



http://guru99.com/RegisterStudent.asp



First calling Web API

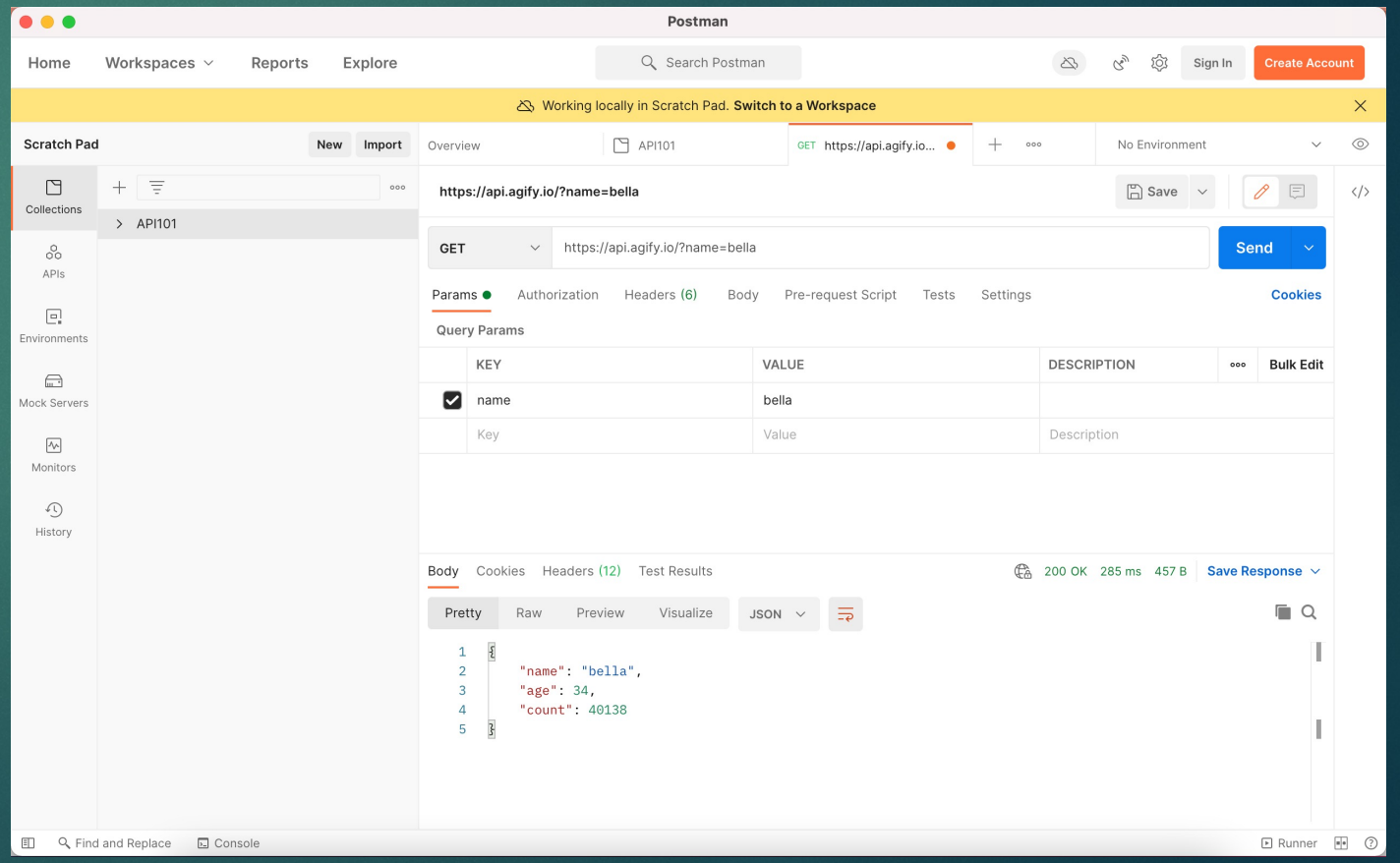
- Most helpful tool
- Postman
  - https://www.postman.com

- API
  - https://api.agify.io/?name=bella
- Keyword
  - URL
  - HTTP Method
  - Parameter/Payload (Both Request and Response)
  - Is need subscriber (e.g. token, key)

Postman

Pongier
8 May 2021

- Create a "Collection" and rename to "API101"
- Creare a "Request" with following setting
  - Method: GET
  - URL: https://api.agify.io/?name=bella
- Send the request
- Observed the responsing
- Try the other one API
  - https://api.exchangerate-api.com/v4/latest/THB



More Advanced API

- https://openweathermap.org/api/hourly-forecast
- https://jsonplaceholder.typicode.com/guide/

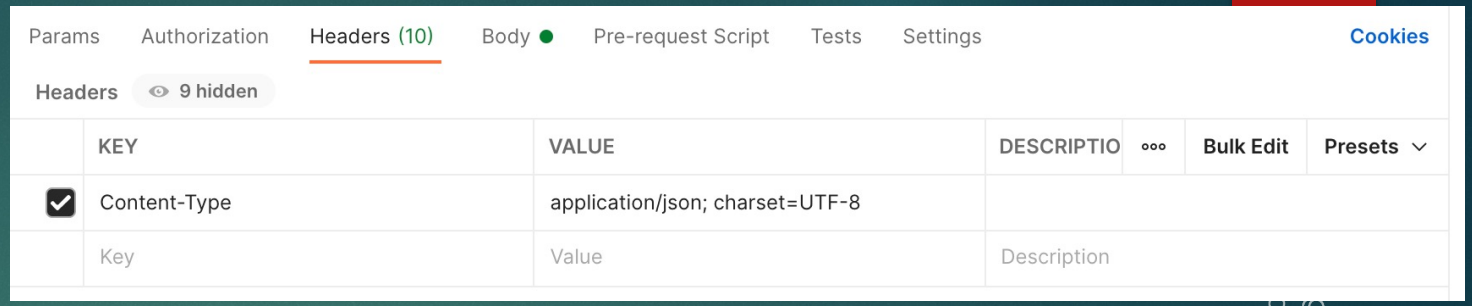
https://jsonplaceholder.typicode.com/posts

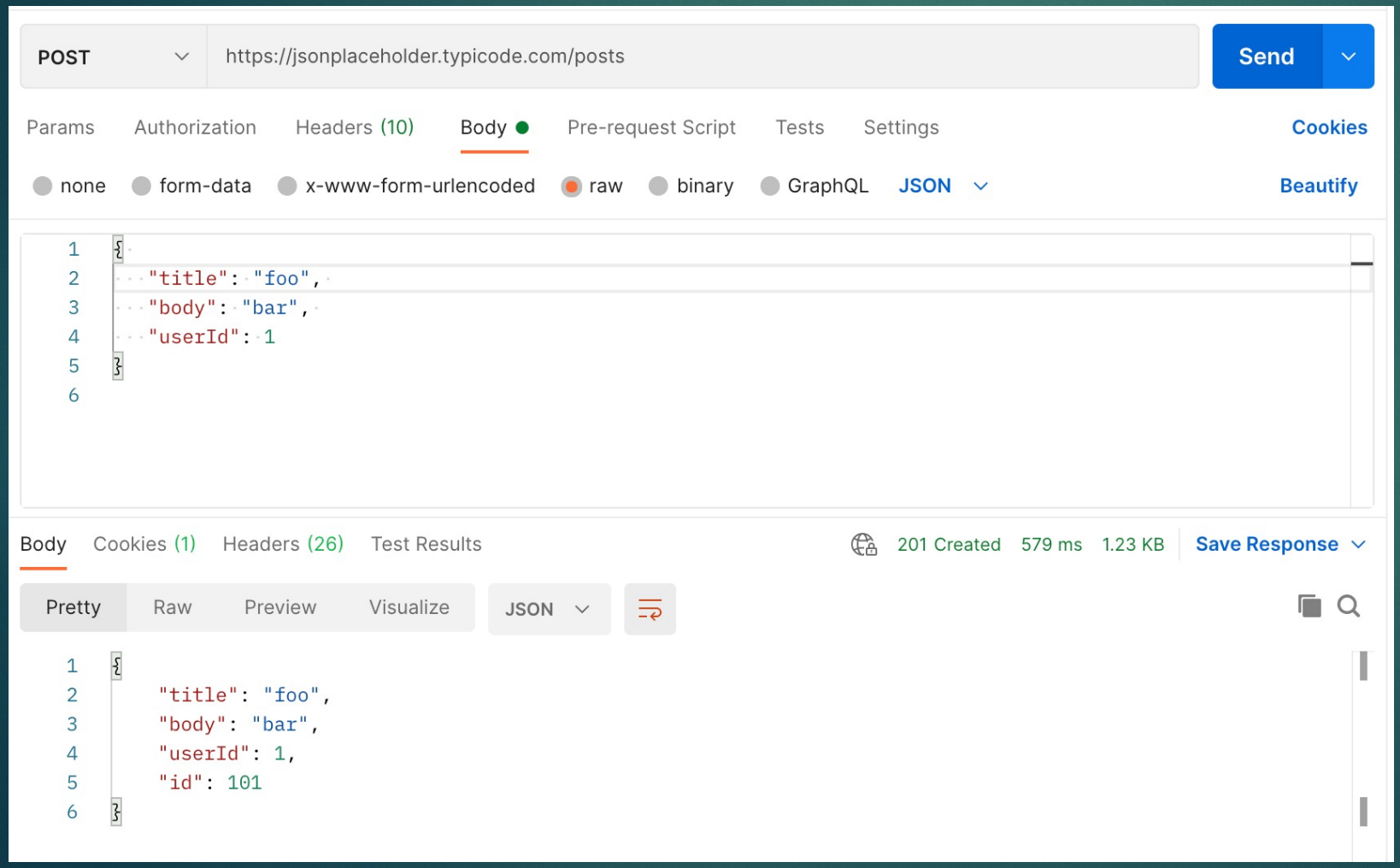
Creating a resource

```
fetch('https://jsonplaceholder.typicode.com/posts', {
  method: 'POST',
  body: JSON.stringify({
    title: 'foo',
    body: 'bar',
    userId: 1,
  }),
  headers: {
    'Content-type': 'application/json; charset=UTF-8',
  },
})
  .then((response) => response.json())
  .then((json) => console.log(json));
```


```
{
  "title": "foo",
  "body": "bar",
  "userId": 1
}
```

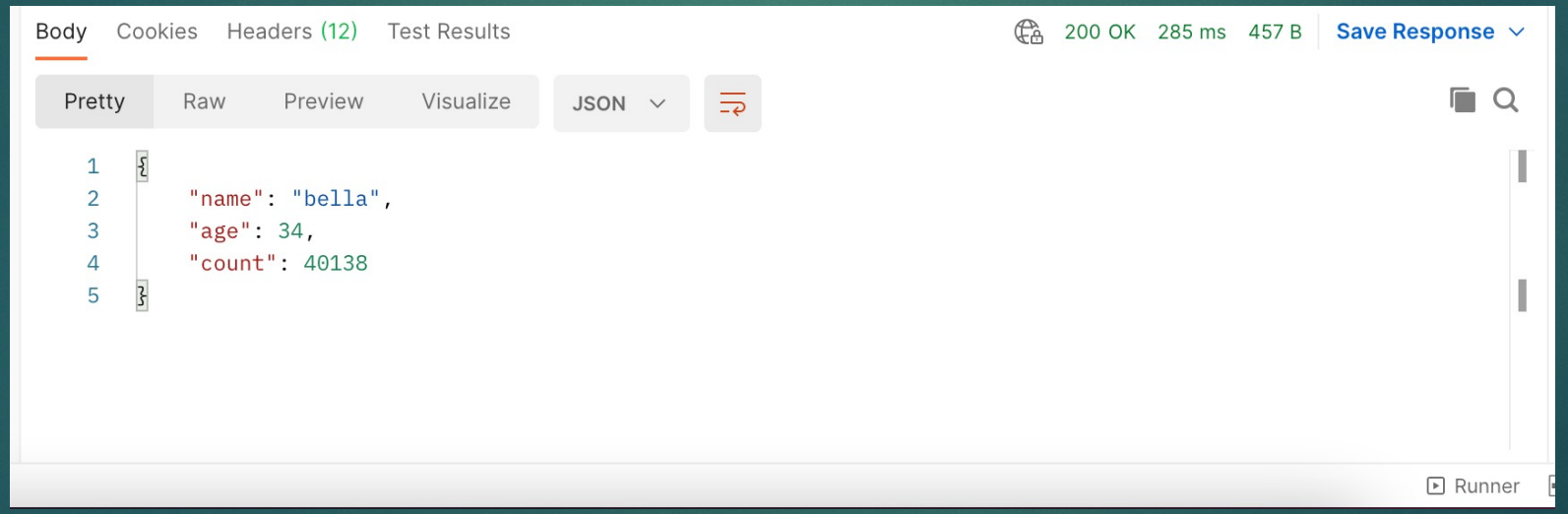
Advanced API





Observed the Responding

- The responding is JSON format



Restful & JSON

- RESTful
  - https://iamgique.medium.com/restful-api-กับ-rest-api-ต่างกันนะรู้ยัง-2c70c42990e3
- RESTful web service
  - Carrier = HTTP Protocol
  - Data Payload = JSON (or XML or HTML or other format)
- Other word
  - SOAP, REST, …

Restful & JSON

- HTTP Request Method for RESTful service
  - https://www.restapitutorial.com/lessons/httpmethods.html

HTTP Verb	CRUD	Entire Collection (e.g. /customers)	Specific Item (e.g. /customers/{id})
POST	Create	201 (Created), 'Location' header with link to /customers/{id} containing new ID.	404 (Not Found), 409 (Conflict) if resource already exists..
GET	Read	200 (OK), list of customers. Use pagination, sorting and filtering to navigate big lists.	200 (OK), single customer. 404 (Not Found), if ID not found or invalid.
PUT	Update/Replace	405 (Method Not Allowed), unless you want to update/replace every resource in the entire collection.	200 (OK) or 204 (No Content). 404 (Not Found), if ID not found or invalid.
PATCH	Update/Modify	405 (Method Not Allowed), unless you want to modify the collection itself.	200 (OK) or 204 (No Content). 404 (Not Found), if ID not found or invalid.
DELETE	Delete	405 (Method Not Allowed), unless you want to delete the whole collection—not often desirable.	200 (OK). 404 (Not Found), if ID not found or invalid.

Restful & JSON

- JSON (Java Script Object Notation)
- Key : Value

- “key”:”value”
- [] is array
- {}

https://www.oxygenxml.com/xml_editor/json_editor.html

API with Python

- Create your RESTful API with Python
- Useful framework
  - Django
  - **Flask**
  - FastAPI
- Did you remember
  - Src/Dst IP Addr
  - Src/Dst Protocol,Port
  - Data Payload

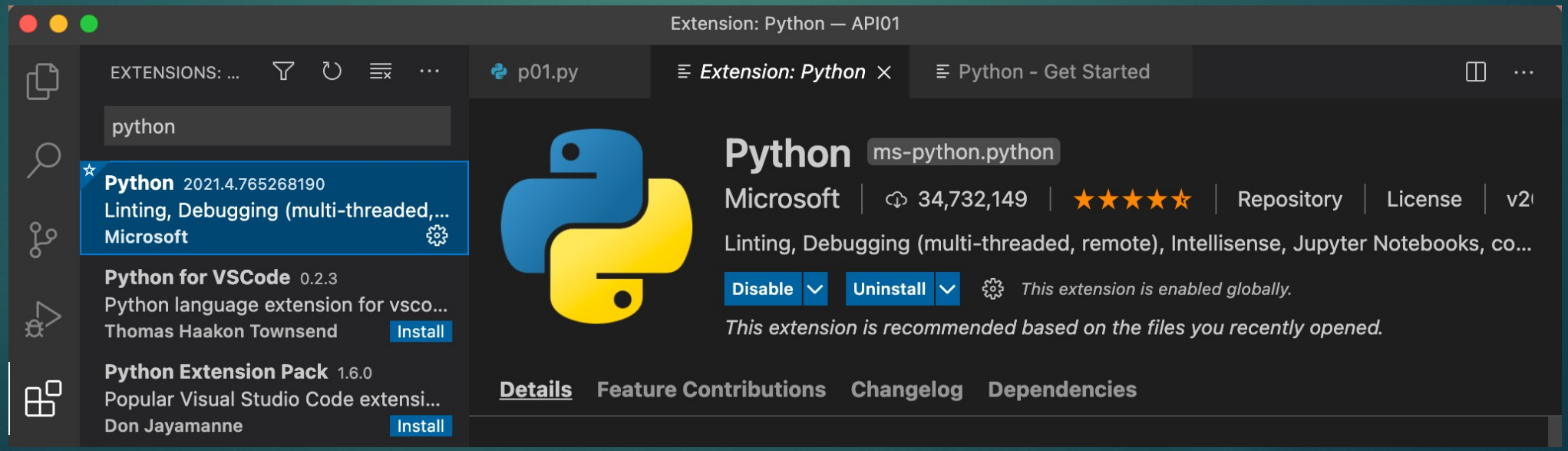
Flask Overview

- https://palletsprojects.com/p/flask/
- https://flask.palletsprojects.com/en/1.1.x/quickstart/

```python
# save this as app.py
from flask import Flask, escape, request

app = Flask(__name__)

@app.route('/')
def hello():
    name = request.args.get("name", "World")
    return f'Hello, {escape(name)}!'
```

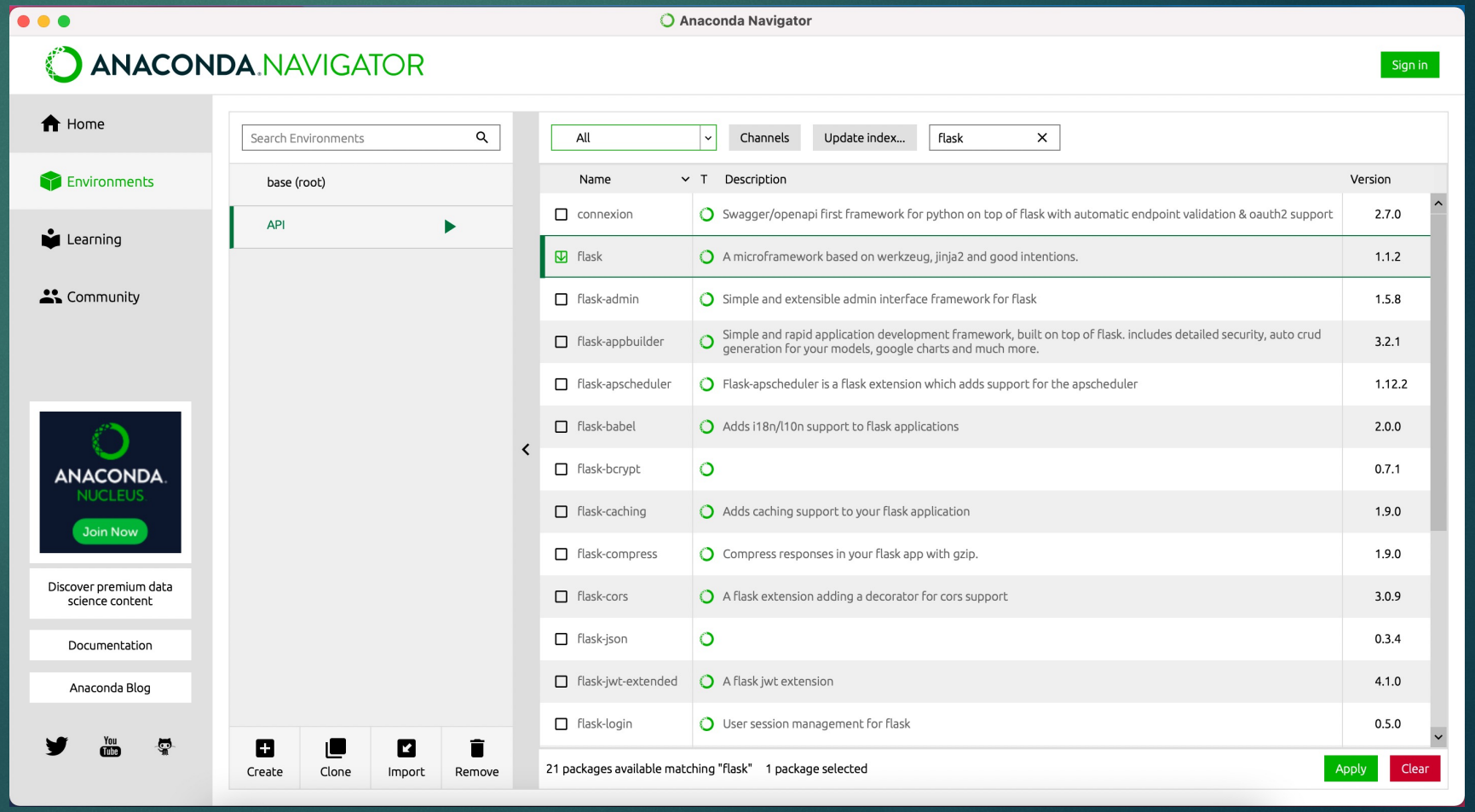
Prepared tools

- Python & Env (example using Anaconda)
  - https://www.anaconda.com
- VS Code & Python Extension
  - https://code.visualstudio.com



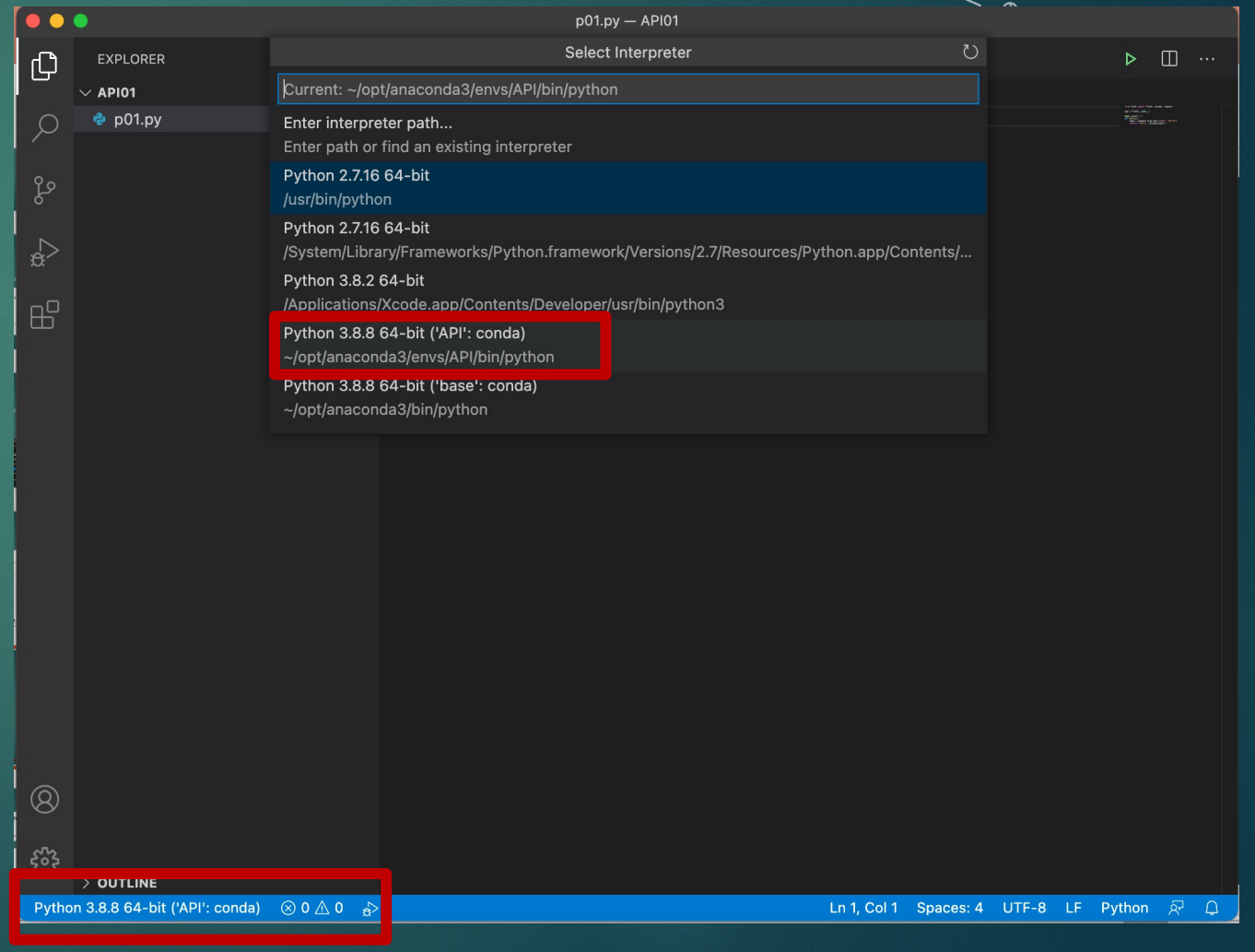
Prepared tools

- pip install flask



VS Code

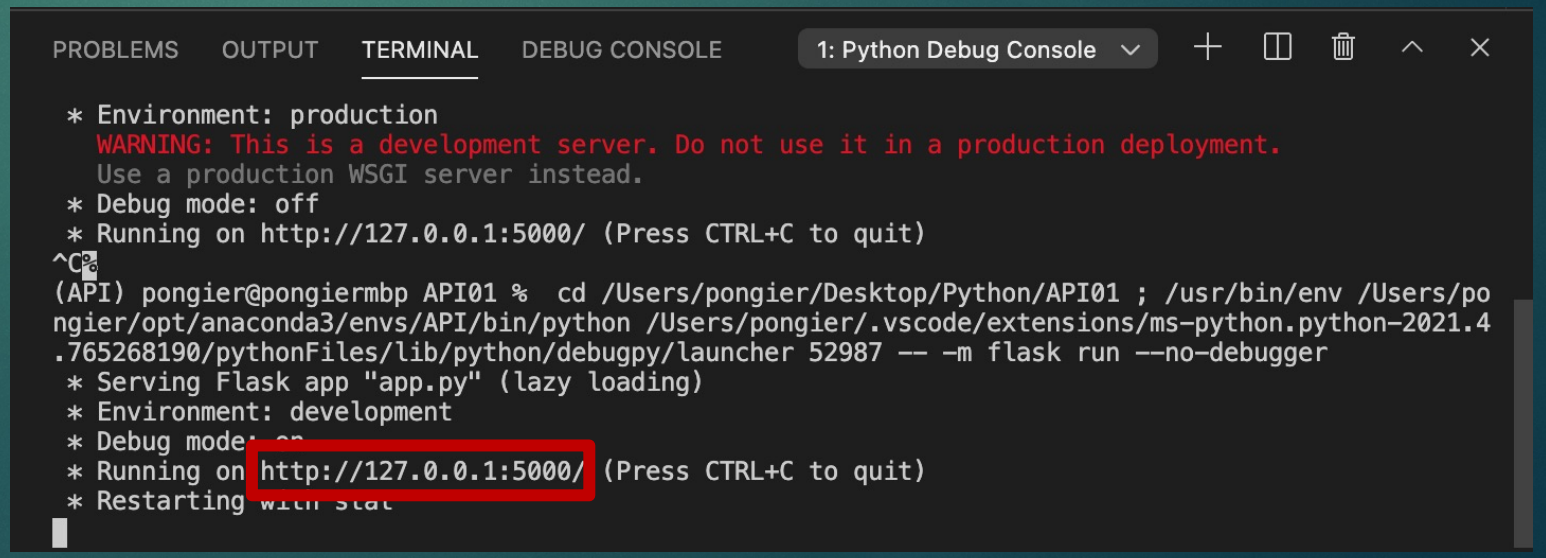
- Create new file named “app.py”
- Select Python (Anaconda API)



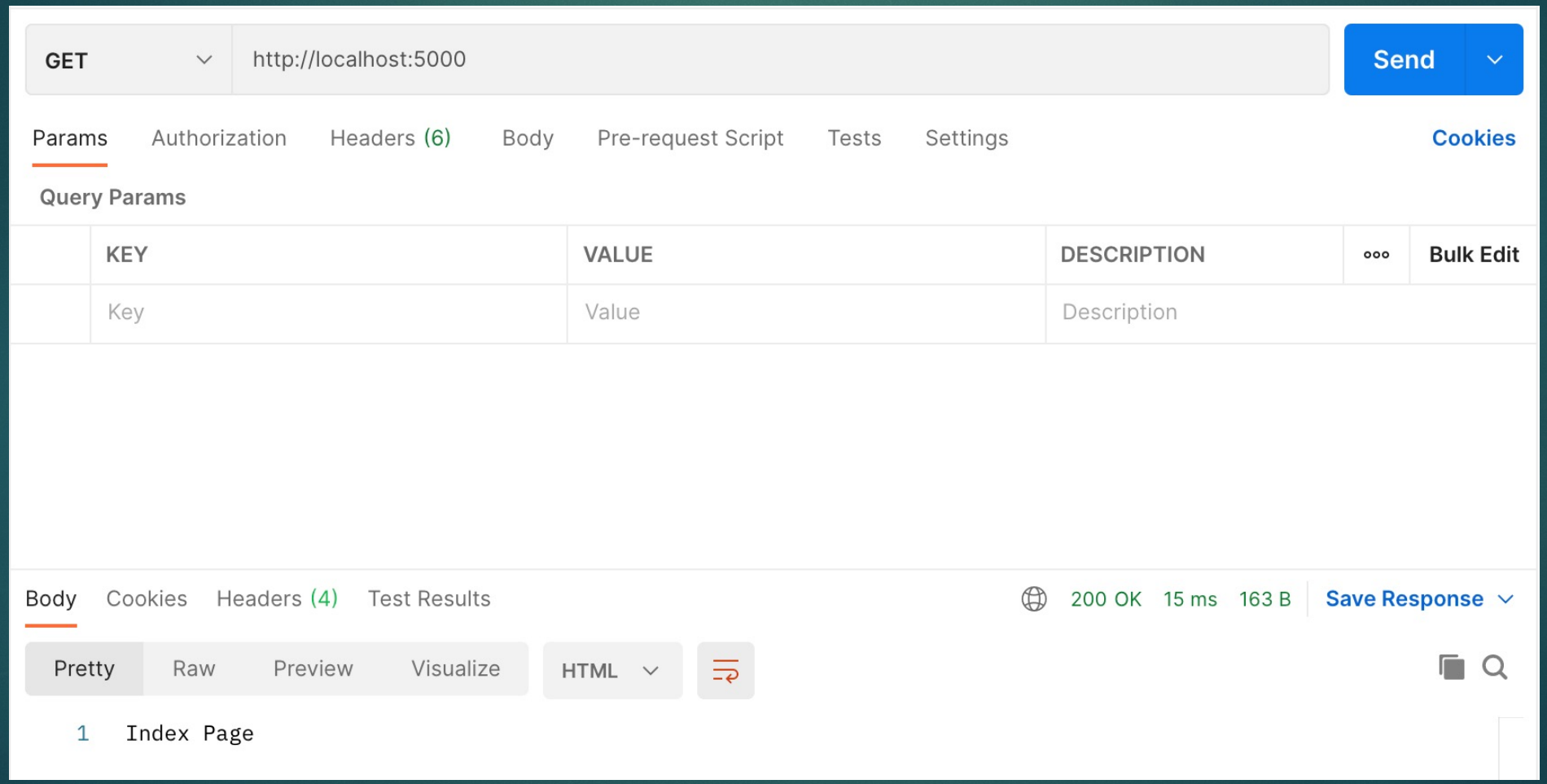
First API with Python Flask


```python
from flask import Flask, escape, request

app = Flask(__name__)

@app.route('/')
def index():
    return 'Index Page'
```

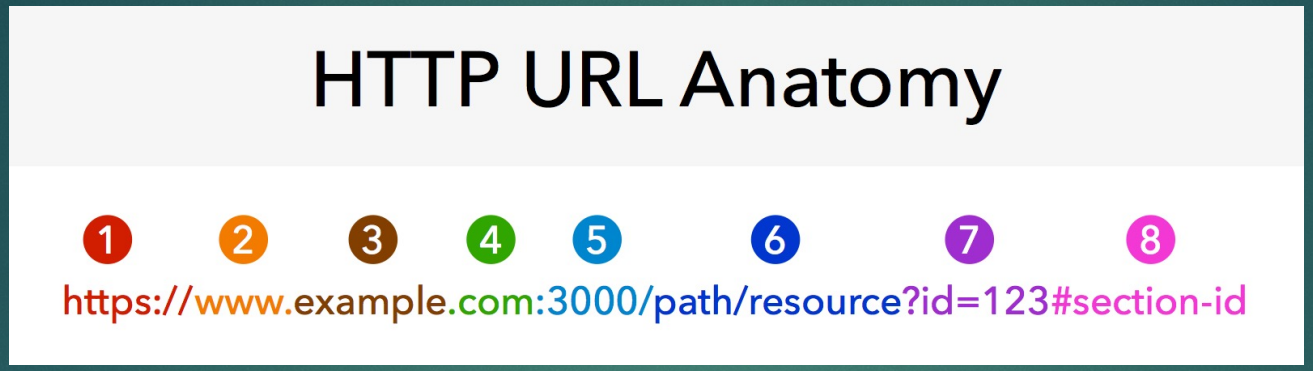
- Run -> Start Debuging -> Flask



Test with Postman

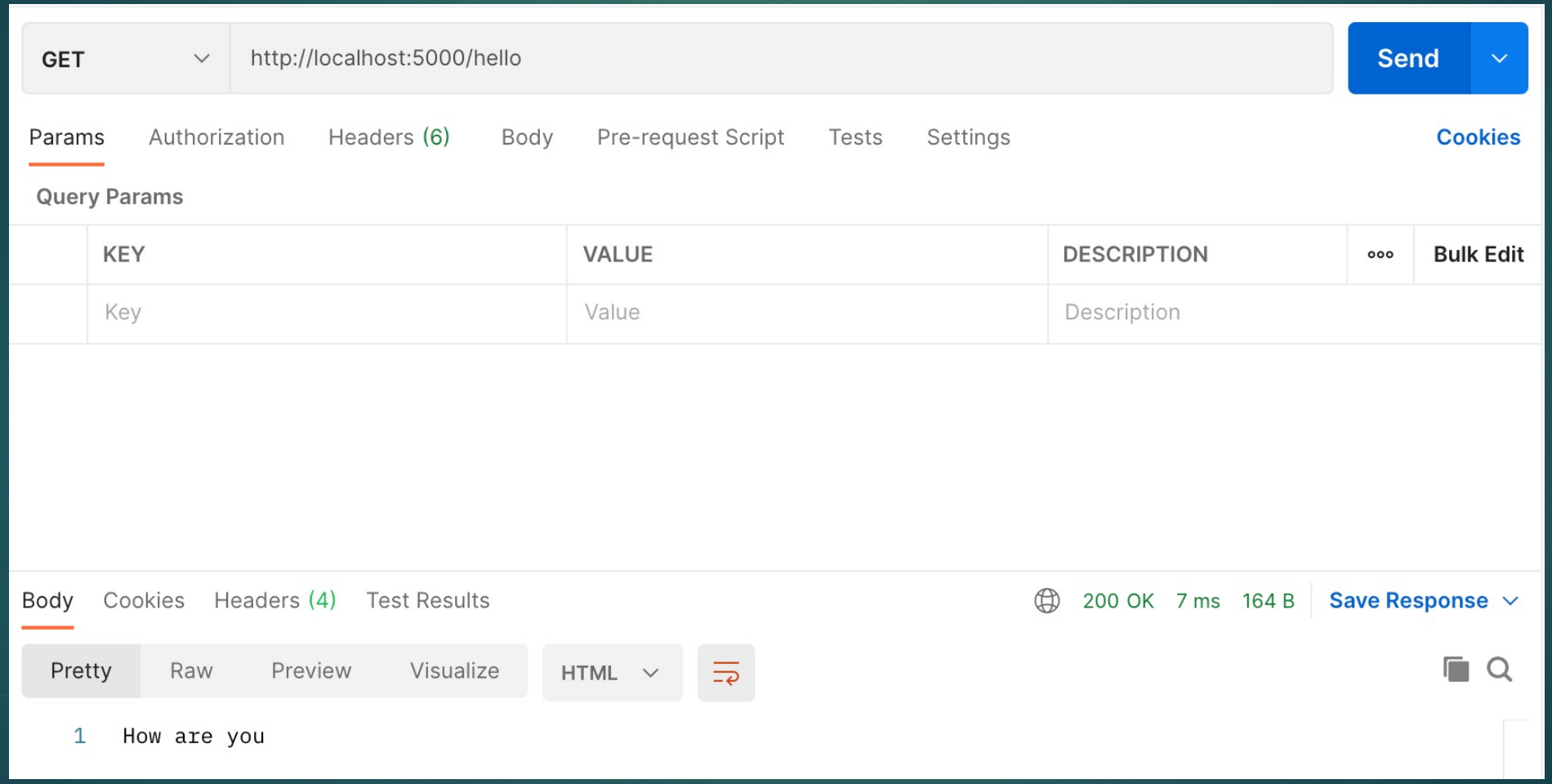


Flask Route




```python
from flask import Flask, escape, request

app = Flask(__name__)

@app.route('/')
def index():
    return 'Index Page'

@app.route('/hello')
def hello():
    return 'How are you'
```

Test with Postman

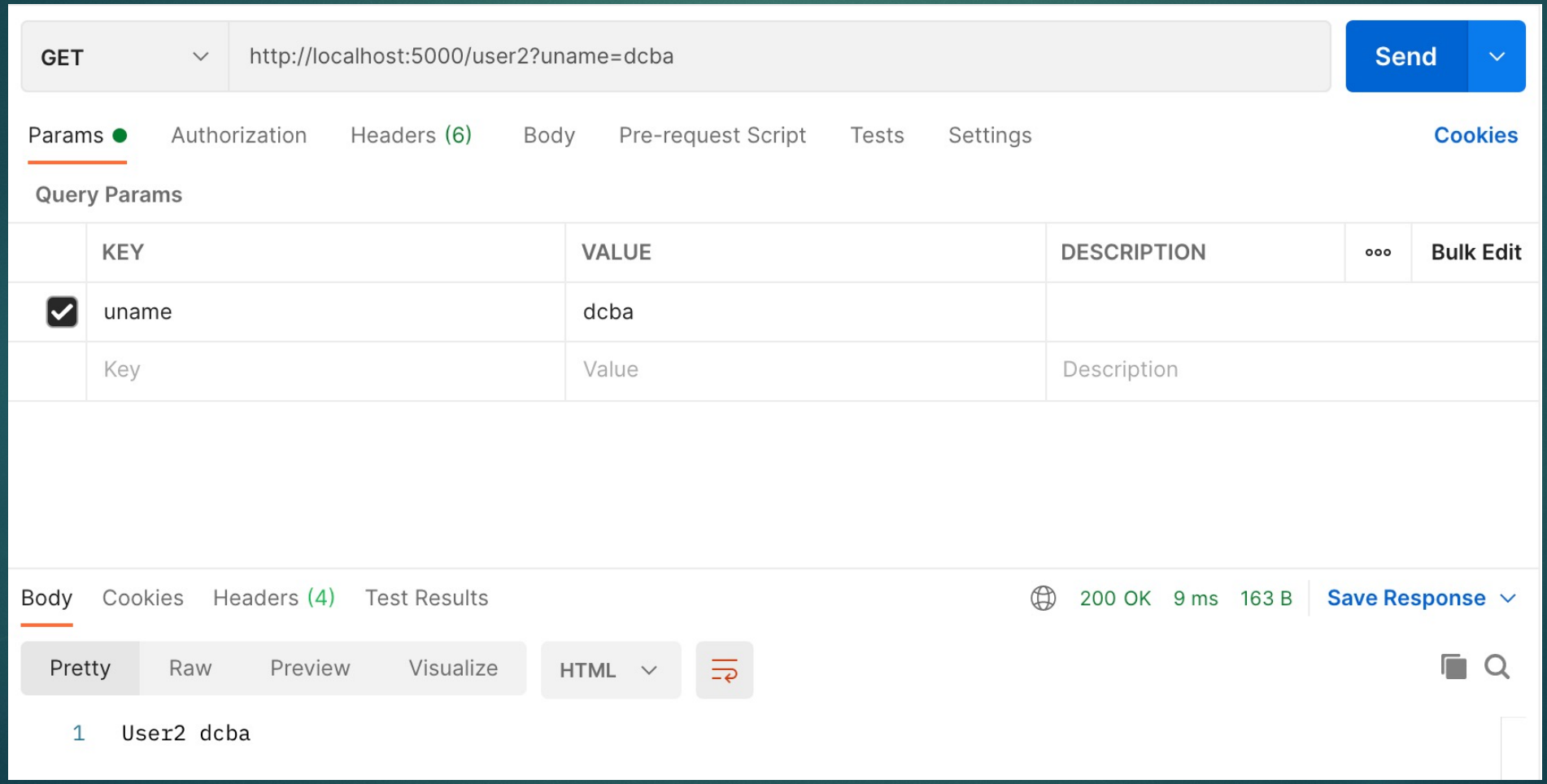


Flask with parameters

Security purpose


```
from markupsafe import escape

@app.route('/user/<username>')
def show_user_profile(username):
    # http://YOUR_SERVER/user/abcd
    return 'User %s' % escape(username)

@app.route('/post/<int:post_id>')
def show_post(post_id):
    # http://YOUR_SERVER/post/9
    return 'Post %d' % post_id

@app.route('/user2')
def login():
    # http://YOUR_SERVER/user2?uname=dcba
    var1 = request.args.get('uname')
    return 'User2 %s' % escape(var1)
```

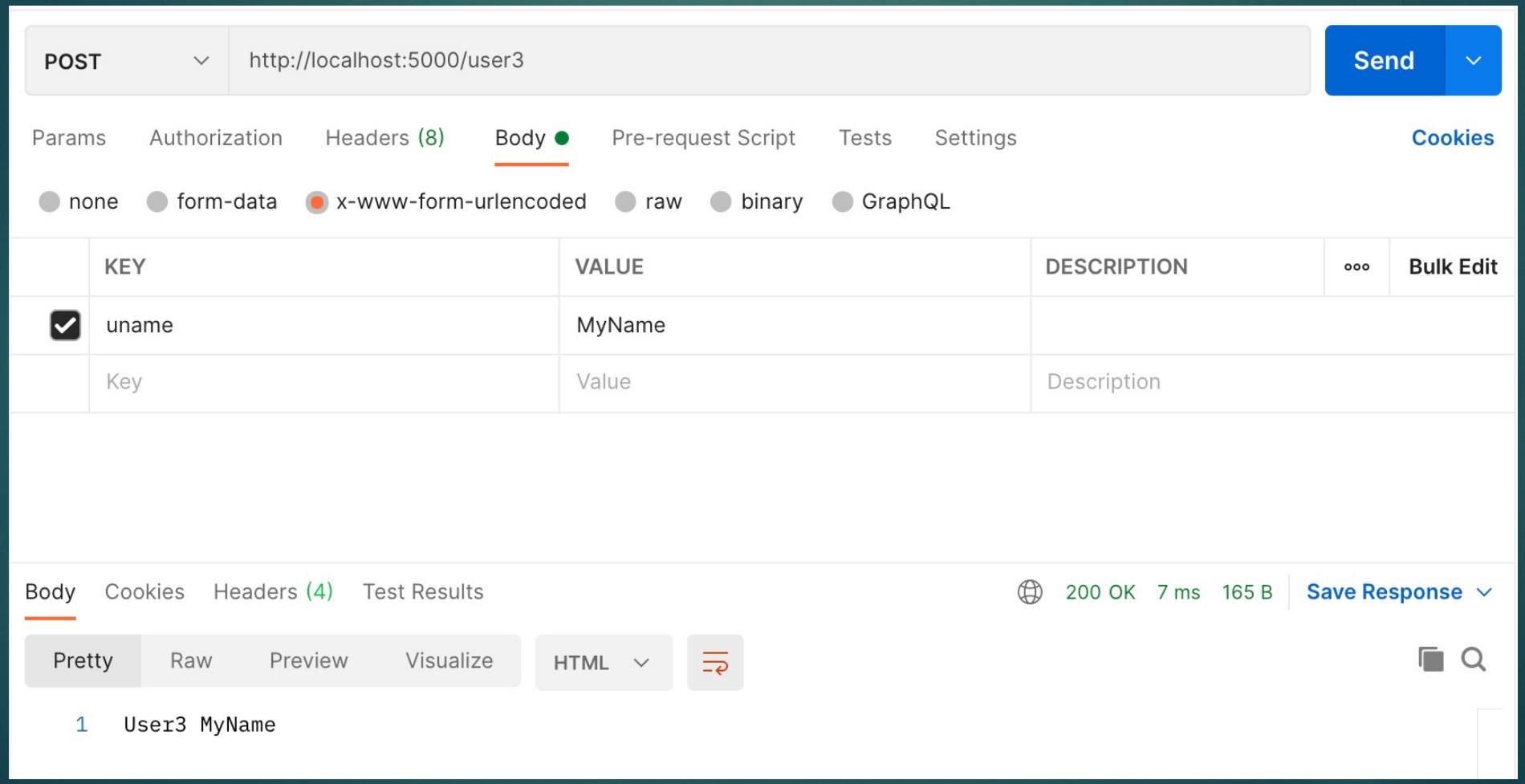
Test with Postman



Flask HTTP Method


```
@app.route('/user3', methods=['POST'])
def login3():
    var1 = request.form['uname']
    return 'User3 %s' % escape(var1)
```

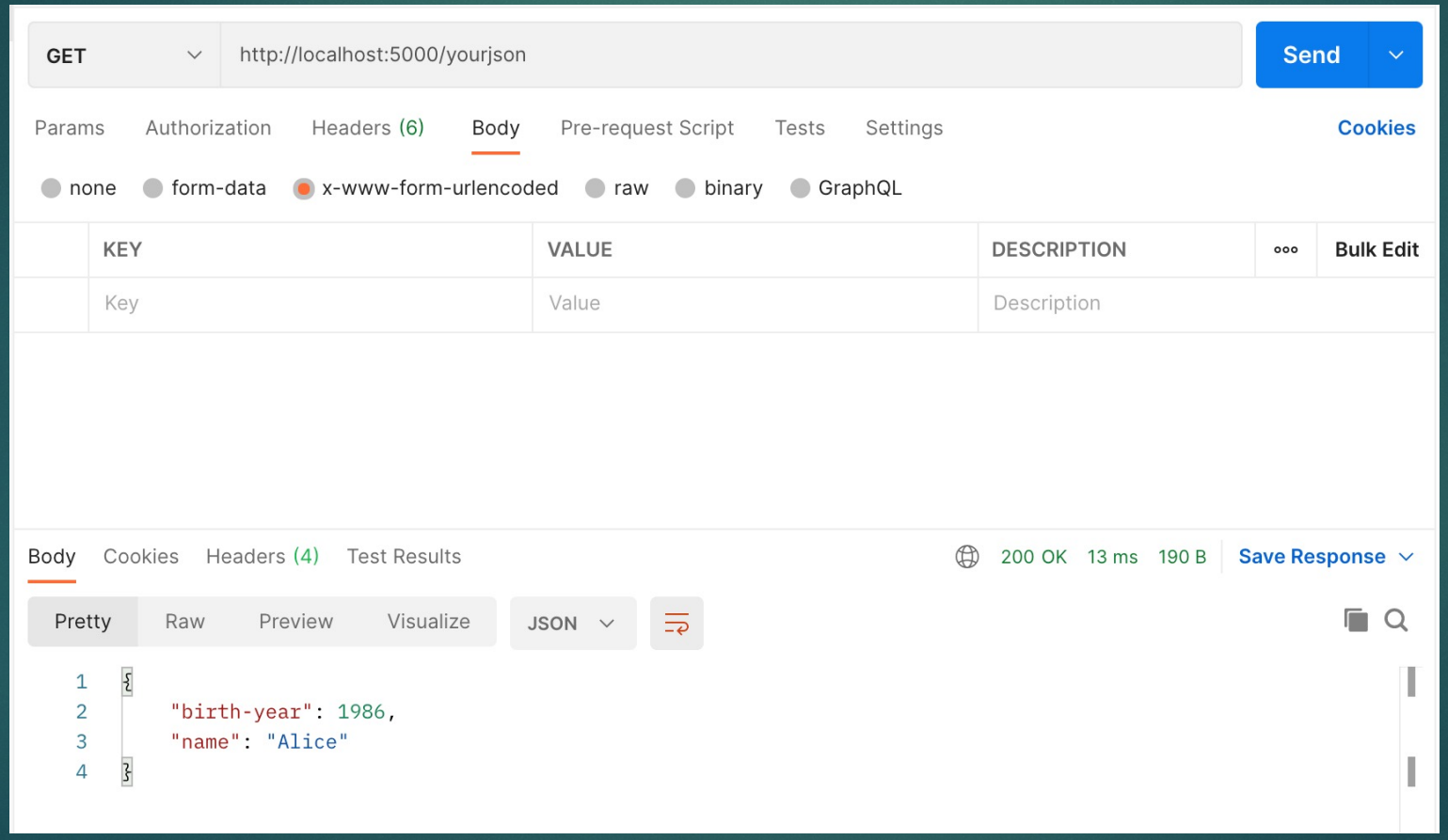
Test with Postman



Flask with JSON output

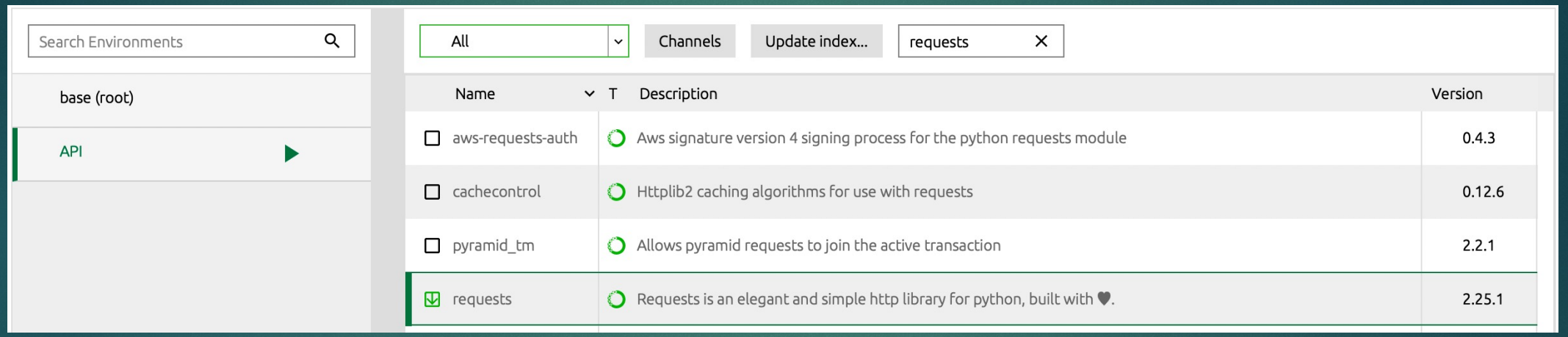

```
from flask import Flask, escape, request, jsonify

@app.route('/yourjson')
def ojson():
    person = {"name": "Alice", "birth-year": 1986}
    return jsonify(person)
```

Test with Postman



Flask call other API

- pip install requests

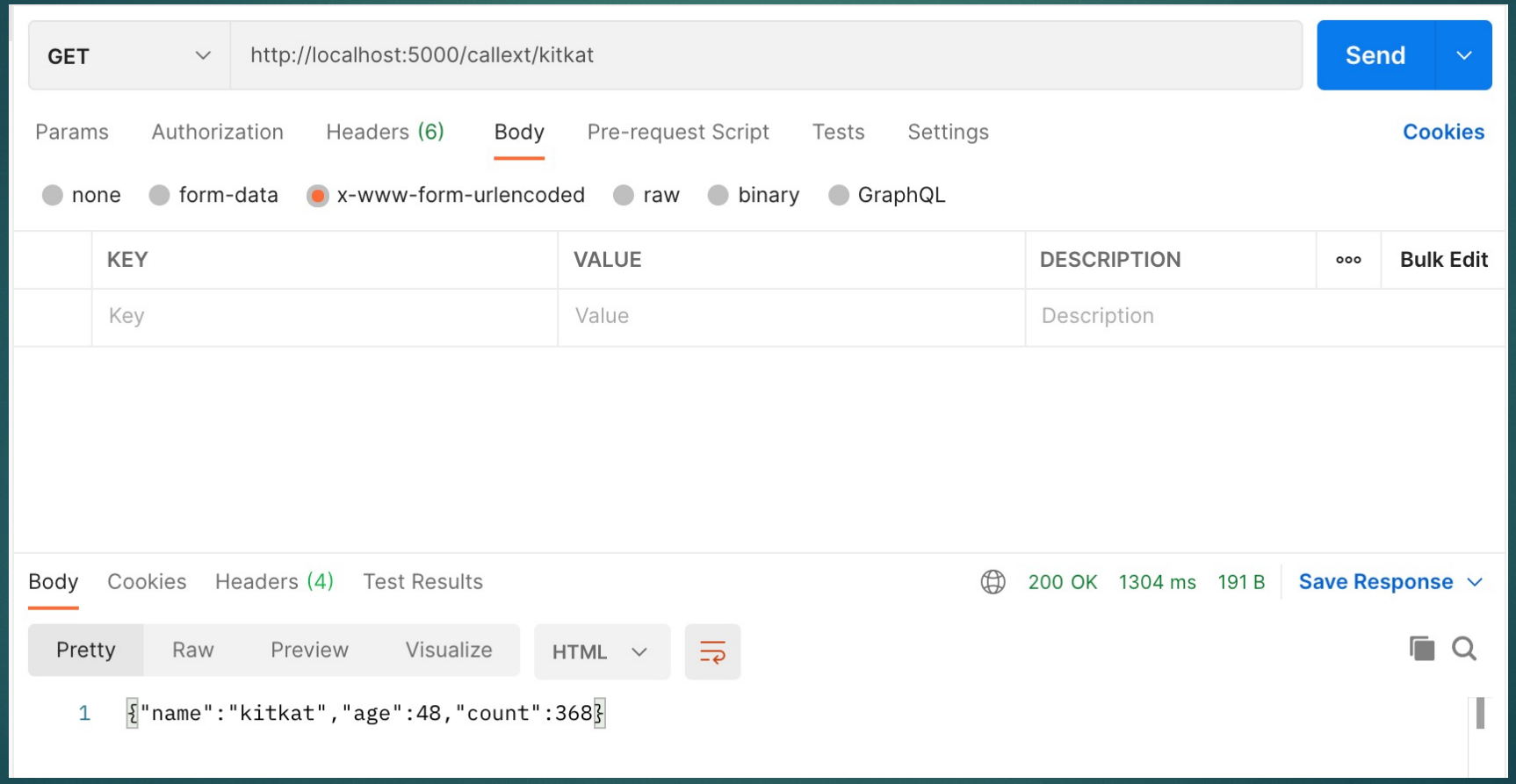


Flask call other API


```
import requests

@app.route('/callext/<username>')
def callext(username):
# call to https://api.agify.io/?name=bella
r = requests.get('https://api.agify.io/?name=' + escape(username))
return r.text
```

- More for call other API with POST method
  - https://www.w3schools.com/python/ref_requests_post.asp

Test with Postman



Flask API with Prediction model

Pongier
8 May 2021

```
def predictmodel():
    TRAIN_SET_LIMIT = 1000
    TRAIN_SET_COUNT = 100

    TRAIN_INPUT = list()
    TRAIN_OUTPUT = list()
    for i in range(TRAIN_SET_COUNT):
        a = randint(0, TRAIN_SET_LIMIT)
        b = randint(0, TRAIN_SET_LIMIT)
        c = randint(0, TRAIN_SET_LIMIT)
        op = a + (2*b) + (3*c)
        TRAIN_INPUT.append([a, b, c])
        TRAIN_OUTPUT.append(op)

    predictor = LinearRegression(n_jobs=-1)
    predictor.fit(X=TRAIN_INPUT, y=TRAIN_OUTPUT)

    X_TEST = [[10, 20, 30]]
    outcome = predictor.predict(X=X_TEST)
    coefficients = predictor.coef_

    print('Outcome : {}\nCoefficients : {}'.format(outcome, coefficients))

    return 'Outcome : {}\nCoefficients : {}'.format(outcome, coefficients)
```

```
@app.route('/predict')
def callpred():
    return predictmodel()
```

Q&A

- A&Q

Python API for Machine Learning

- API 101 (08 May 2021)
  - Basic API with Python3 and Flask
- API 201 (Tentative 15 May 2021)
  - Advance API input
    - Image
  - Train model and save model
  - Using saved model and conduct API
  - Trip & Trick for API (Speed and Problem)
  - Cloud Platform example