

API

Application Programming Interface

<https://github.com/9meo/api>

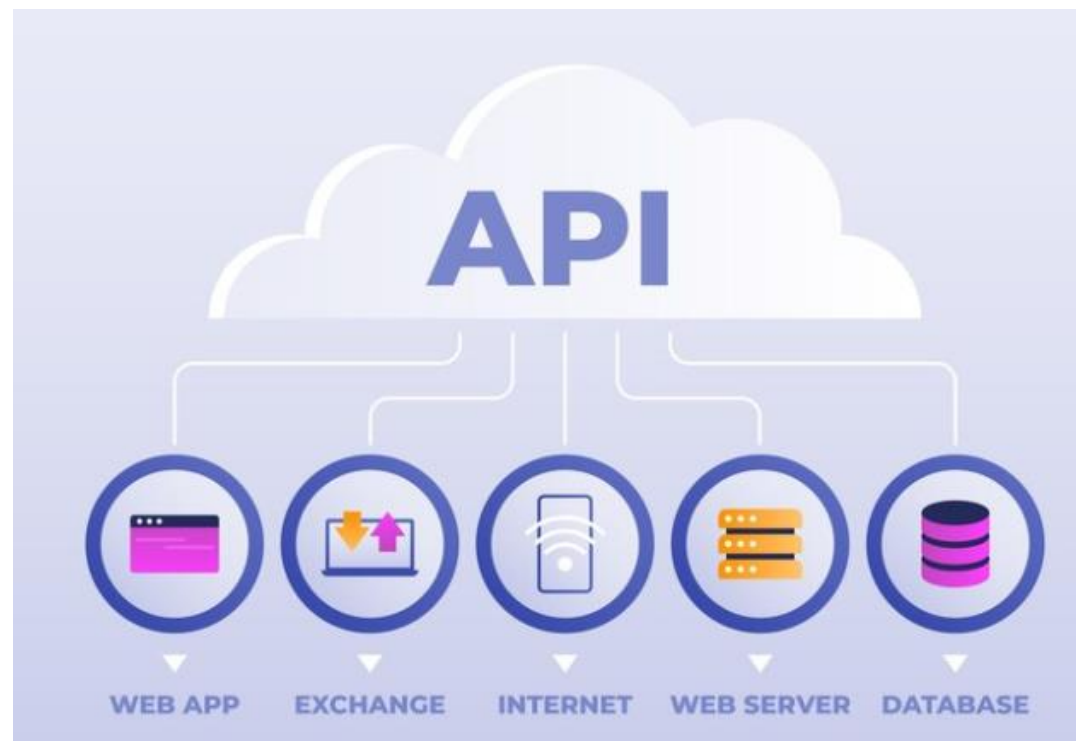


API

- ☐ ทำความรู้จักกับ API
- ☐ ความสำคัญของ API ใน
โลกของการพัฒนาซอฟต์แวร์
- ☐ วัตถุประสงค์ของคลาสนี้

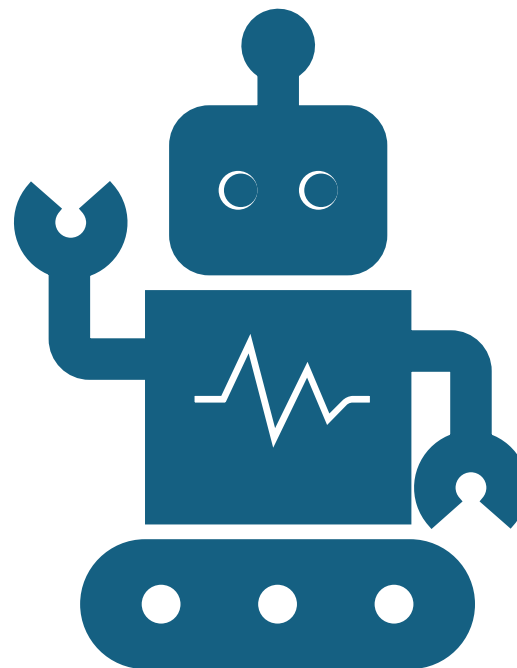
API คืออะไร?

- API (Application Programming Interface) คือชุดของกฎเกณฑ์และโปรโตคอลที่ช่วยให้ซอฟต์แวร์ต่างๆ สามารถสื่อสารกันได้ หรือเป็นช่องทางในการเข้าถึงฟังก์ชันหรือข้อมูลของแอปพลิเคชันหนึ่งๆ โดยแอปพลิเคชันอื่นๆ
- ตัวอย่างง่ายๆ: เมื่อคุณใช้แอปบนโทรศัพท์เพื่อดูอากาศ แอปนั้นใช้ **API** เพื่อรับข้อมูลอากาศจากบริการภูมิอากาศ



ประโยชน์ของ API:

- **อัตโนมัติ:** อำนวยความสะดวกในการทำงานอัตโนมัติ ช่วยลดงานที่ซ้ำซากจำเจ
- **การรวมระบบ:** ช่วยให้สามารถรวมฟังก์ชันการทำงานของซอฟต์แวร์ต่างๆ เข้าด้วยกันได้ง่าย
- **การปรับขยาย:** ช่วยให้แอปพลิเคชันสามารถปรับขยายได้ง่ายขึ้น ตอบสนองต่อความต้องการของผู้ใช้และธุรกิจ



ข้อจำกัดของ **API**:

- ข้อจำกัดด้านการเข้าถึง: **API** บางตัวอาจมีข้อจำกัดในการใช้งานซึ่งผู้พัฒนาต้องปฏิบัติตาม
- ความปลอดภัย: การใช้งาน **API** ต้องมีการจัดการความปลอดภัยอย่างระมัดระวังเพื่อป้องกันข้อมูลรั่วไหล





ประเภทของ API

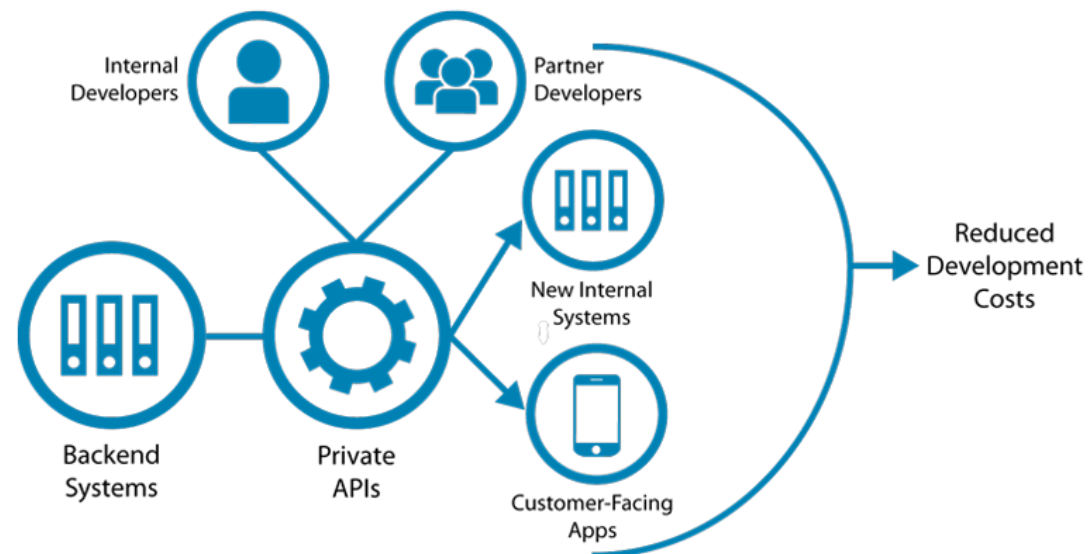
1. Public API (หรือ Open API)

API ที่เปิดให้นักพัฒนาทั่วไปสามารถเข้าถึงและใช้งานได้ มักจะมีเอกสารคู่มือที่ชัดเจนและอยู่ภายใต้ข้อตกลงการใช้งาน (Terms of Service)

ประเภทของ API

2. Private API (Internal API)

API ที่ถูกใช้งานภายในองค์กร ไม่เปิดเผยให้ภายนอก ช่วยในการปรับปรุงการทำงานร่วมกันและประสิทธิภาพของซอฟต์แวร์ภายใน



ประเภทของ API

3. Partner API

API ที่ถูกใช้งานระหว่างองค์กรที่มีความร่วมมือกัน มีการควบคุมการเข้าถึงมากกว่า **Public API** แต่ยังเปิดกว้างกว่า **Private API**



RESTful API

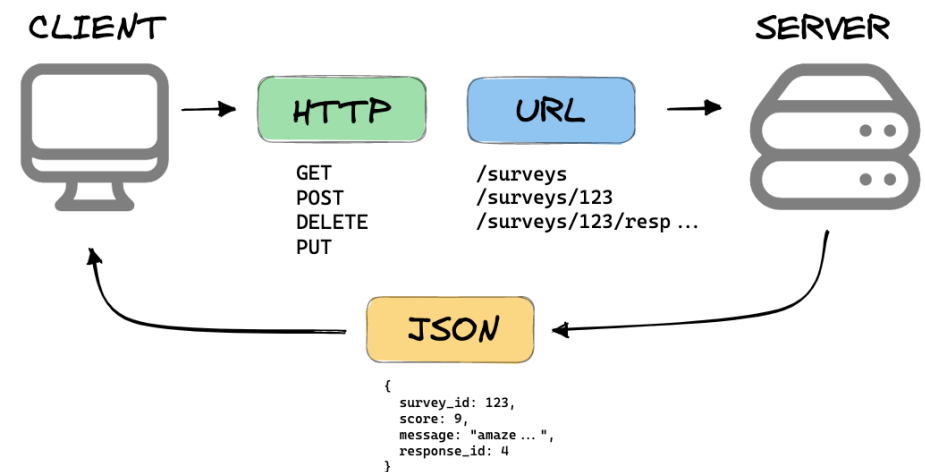
REST (Representational State Transfer)

เป็นสถาปัตยกรรมที่ใช้ในการสร้างการสื่อสารระหว่างระบบ cloud และ server

หลักการของ RESTful API:

- **Uniform Interface:** การทำงานของ API ต้องมีการออกแบบที่เป็นมาตรฐานเดียวกัน
- **Stateless:** ไม่เก็บสถานะของผู้ใช้บน server แต่จะคำขอต้องมีข้อมูลเพียงพอในการดำเนินการต่อ
- **Cacheable:** คำขอตอบสนองสามารถเก็บในแคชได้ เพื่อลดการโหลดบน server

WHAT IS A REST API?



RESTful API Method:

GET: to retrieve data from the server.

GET

/pet/{petId} Find pet by ID

PUT

/pet Update an existing pet

DELETE

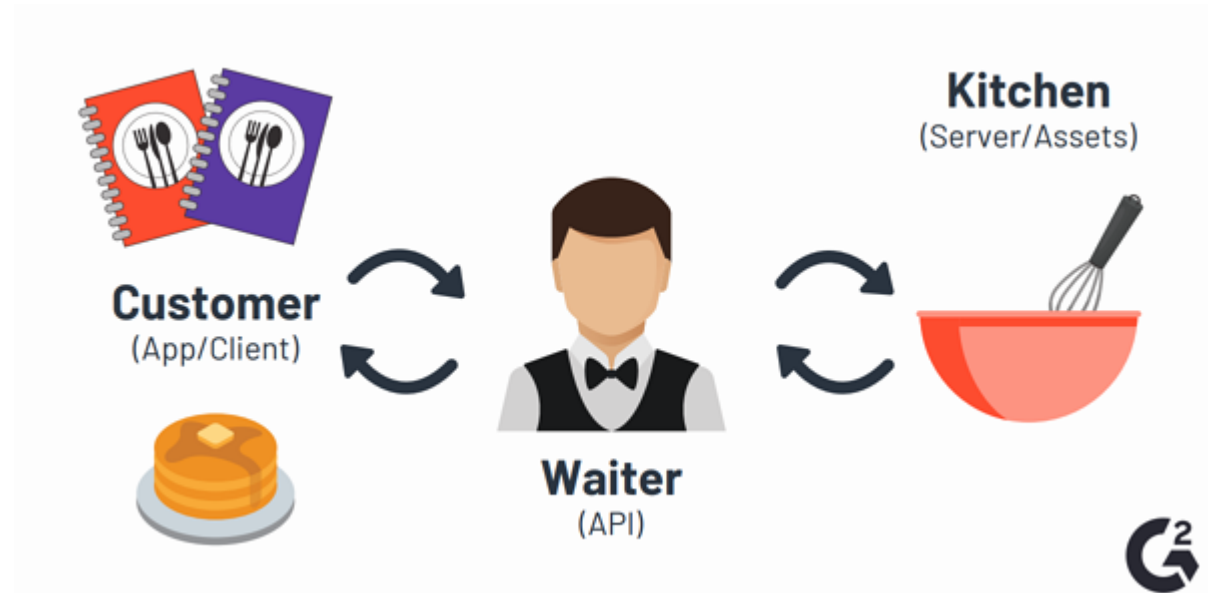
/pet/{petId} Deletes a pet

POST

/pet/{petId}/uploadImage uploads an image

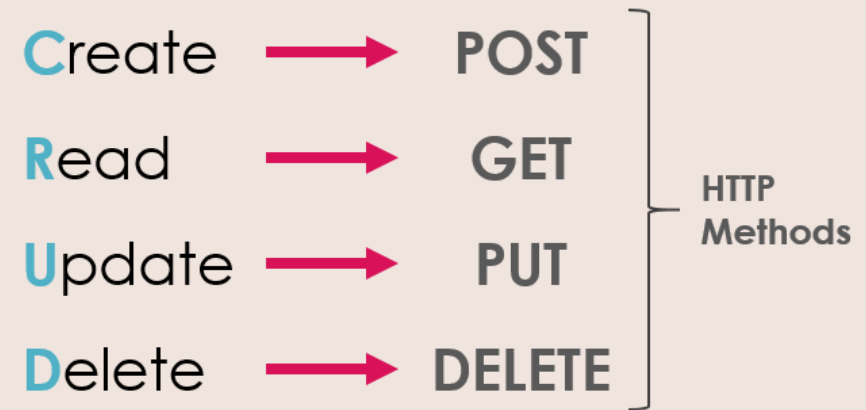
RESTful API Method:

POST: sends data to the server and creates a new resource.



RESTful API Method:

PUT: update an existing resource.



RESTful API Method:

DELETE : delete a resource specified by its URI.

```
// DELETE task with id = 1
fetch('https://jsonplaceholder.typicode.com/todos/1', {
  method: 'DELETE'
})
// empty response: {}
```

JSON ในการสื่อสารข้อมูล

- **JSON (JavaScript Object Notation)** เป็นรูปแบบการแลกเปลี่ยนข้อมูลที่เบาและอ่านง่าย ใช้สำหรับแลกเปลี่ยนข้อมูลระหว่างเบราว์เซอร์และเซิร์ฟเวอร์
- **คุณสมบัติ:** โครงสร้างที่ง่ายดาย, สามารถใช้กับภาษาโปรแกรมมิ่งหลายภาษา, และสนับสนุนโดยระบบ **API** ส่วนใหญ่

```
{  
  "user": {  
    "id": "12345",  
    "name": "John Doe",  
    "email": "john.doe@example.com"  
  }  
}
```

ข้อดีของการใช้ JSON:

- **ประสิทธิภาพ:** ข้อมูลมีขนาดเล็ก, ประมวลผลได้เร็ว
- **ความเข้ากันได้:** รองรับโดยเว็บเบราว์เซอร์และเครื่องมือพัฒนาซอฟต์แวร์ส่วนใหญ่
- **ความยืดหยุ่น:** สามารถปรับแต่งโครงสร้างข้อมูลได้ตามความต้องการ

Xml

```
<?xml version="1.0" encoding="UTF-8" ?>
<root>
  <apiVersion>v1</apiVersion>
  <kind>Pod</kind>
  <metadata>
    <name>hello-pod</name>
    <labels>
      <app>hello</app>
    </labels>
  </metadata>
  <spec>
    <containers>
      <name>hello-container</name>
      <image>tmkube/hello</image>
      <ports>
        <containerPort>8000</containerPort>
      </ports>
    </containers>
  </spec>
</root>
```

Json

```
{
  "apiVersion": "v1",
  "kind": "Pod",
  "metadata": {
    "name": "hello-pod",
    "labels": {
      "app": "hello"
    }
  },
  "spec": {
    "containers": [
      {
        "name": "hello-container",
        "image": "tmkube/hello",
        "ports": [
          {
            "containerPort": 8000
          }
        ]
      }
    ]
  }
}
```

Yaml

```
apiVersion: v1
kind: Pod
metadata:
  name: hello-pod
  labels:
    app: hello
spec:
  containers:
    - name: hello-container
      image: tmkube/hello
      ports:
        - containerPort: 8000
```

การใช้ API ในโปรเจกต์จริง

Social Media API

- API ที่ช่วยให้ผู้พัฒนาสามารถเชื่อมต่อและโต้ตอบกับแพลตฟอร์มโซเชียลมีเดีย
- ตัวอย่างการใช้งาน: การดึงคอนเทนต์จากโซเชียลมีเดียไปยังเว็บไซต์, การเข้าถึงและจัดการคอมเมนต์, การโพสต์เนื้อหาโดยอัตโนมัติผ่าน API
- ตัวอย่าง API: Facebook Graph API, Twitter API

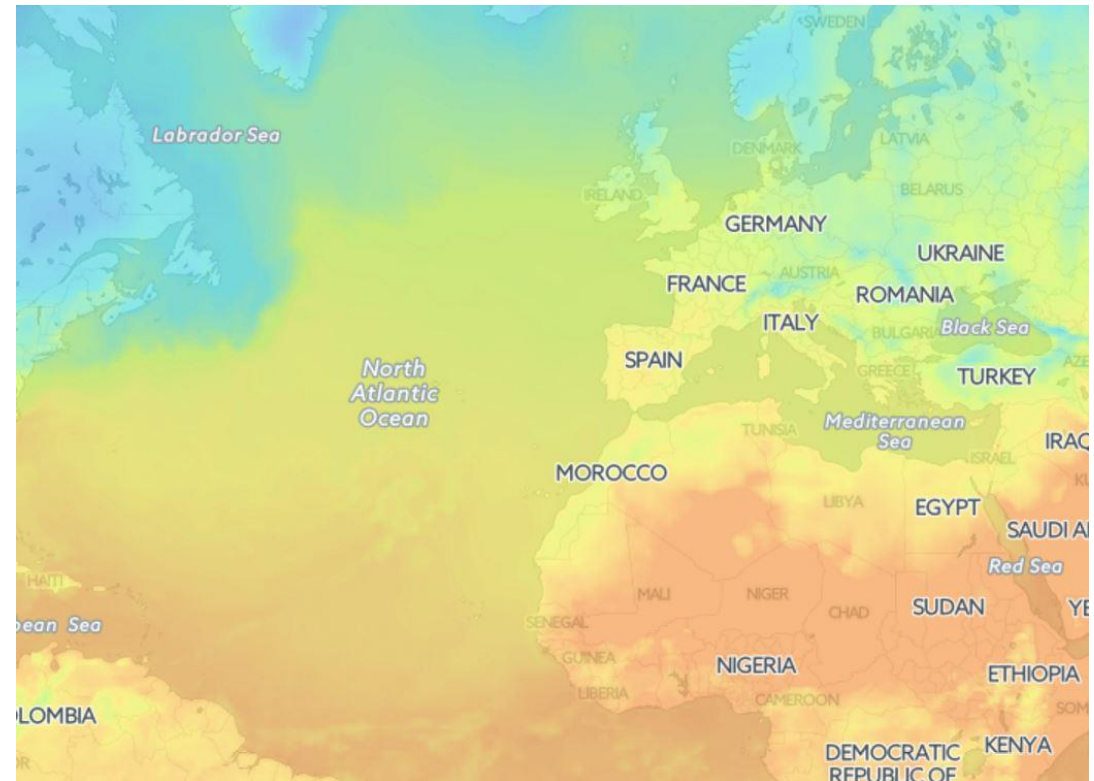
Graph API



การใช้ API ในโปรเจกต์จริง

Weather API

- API ที่ให้ข้อมูลสภาพอากาศทั่วโลกจากแหล่งข้อมูลต่างๆ
- ตัวอย่างการใช้งาน: แอปพลิเคชันที่แสดงข้อมูลอากาศสด, การรวมข้อมูลอากาศเข้ากับแพลตฟอร์มธุรกิจรวมหรือการท่องเที่ยว
- ตัวอย่าง **API**: OpenWeatherMap, AccuWeather API



การใช้ API ในโปรเจกต์จริง

Payment Gateway API

- API ที่ช่วยให้แอปพลิเคชันสามารถรับชำระเงินออนไลน์ผ่านวิธีการต่างๆ
- ตัวอย่างการใช้งาน: การรวมระบบชำระเงินเข้ากับเว็บไซต์ e-commerce, การจัดการการชำระเงินและการคืนเงิน
- ตัวอย่าง API: PayPal, Stripe API



ความปลอดภัยในการใช้งาน API

ความสำคัญของความปลอดภัย API:

- **ความเสี่ยง:** API ที่ไม่ปลอดภัยอาจนำไปสู่การรั่วไหลของข้อมูลส่วนบุคคล, การโจมตีทางไซเบอร์, และความเสียหายต่อชื่อเสียง
- **ความจำเป็น:** การรักษาความปลอดภัย API คือส่วนสำคัญในการปกป้องแอปพลิเคชันและข้อมูลผู้ใช้



แนวทางการรักษาความปลอดภัย:

- **Authentication** (การตรวจสอบสิทธิ์): ใช้มาตรฐาน OAuth, API keys หรือ JWT (JSON Web Tokens) เพื่อยืนยันตัวตนผู้ใช้
- **Authorization** (การอนุญาต): จัดการสิทธิ์การเข้าถึงข้อมูลโดยใช้ RBAC (Role-Based Access Control) หรือ ABAC (Attribute-Based Access Control)
- **Encryption** (การเข้ารหัสข้อมูล): ใช้ HTTPS ในการเข้ารหัสข้อมูลที่ส่งผ่านเครือข่าย
- **Rate Limiting**: จำกัดจำนวนคำขอที่ส่งถึง API เพื่อป้องกันการโจมตีแบบ DDoS
- **Monitoring and Logging**: ติดตามและบันทึกการใช้งาน API เพื่อตรวจจับและตอบสนองต่อกิจกรรมที่ผิดปกติ



ตัวอย่างเครื่องมือความปลอดภัย API:

- **API Gateways:** เช่น Amazon API Gateway, Kong, Apigee
- **Web Application Firewalls (WAF):** ปกป้อง API จากการโจมตีที่เป็นที่รู้จัก





การเรียนรู้ API GET/POST

Python

```
import requests

response = requests.get("https://api.example.com/data")
data = response.json()
print(data)
```

```
import requests

data = {'key': 'value'}
response = requests.post('https://api.example.com/data', json=data)
print(response.text)
```

JavaScript

```
fetch('https://api.example.com/data')  
  .then(response => response.json())  
  .then(data => console.log(data))  
  .catch(error => console.error('Error:', error));
```

```
fetch('https://api.example.com/data', {  
  method: 'POST',  
  headers: {  
    'Content-Type': 'application/json',  
  },  
  body: JSON.stringify({key: 'value'})  
})  
  .then(response => response.json())  
  .then(data => console.log(data))  
  .catch(error => console.error('Error:', error));
```


Java

```
URL url = new URL("https://api.example.com/data");
URLConnection con = (URLConnection) url.openConnection();
con.setRequestMethod("GET");
BufferedReader in = new BufferedReader(new InputStreamReader(con.getInputStream()));
String inputLine;
StringBuilder response = new StringBuilder();
while ((inputLine = in.readLine()) != null) {
    response.append(inputLine);
}
in.close();
System.out.println(response.toString());
```

Java

```
URL url = new URL("https://api.example.com/data");
URLConnection con = (URLConnection) url.openConnection();
con.setRequestMethod("POST");
con.setDoOutput(true);
String jsonString = "{\"key\": \"value\"}";
try(OutputStream os = con.getOutputStream()) {
    byte[] input = jsonString.getBytes("utf-8");
    os.write(input, 0, input.length);
}
try(BufferedReader br = new BufferedReader(
    new InputStreamReader(con.getInputStream(), "utf-8"))) {
    StringBuilder response = new StringBuilder();
    String responseLine = null;
    while ((responseLine = br.readLine()) != null) {
        response.append(responseLine.trim());
    }
    System.out.println(response.toString());
}
```

C#

```
using System.Net.Http;
using System.Threading.Tasks;

public class Program
{
    public static async Task Main(string[] args)
    {
        using (HttpClient client = new HttpClient())
        {
            HttpResponseMessage response = await client.GetAsync("https://api.example.com");
            string data = await response.Content.ReadAsStringAsync();
            Console.WriteLine(data);
        }
    }
}
```

C#

```
using System.Net.Http;
using System.Text;
using System.Threading.Tasks;

public class Program
{
    public static async Task Main(string[] args)
    {
        using (HttpClient client = new HttpClient())
        {
            var content = new StringContent("{\"key\": \"value\"}", Encoding.UTF8, "application/json");
            HttpResponseMessage response = await client.PostAsync("https://api.example.com/data", content);
            string result = await response.Content.ReadAsStringAsync();
            Console.WriteLine(result);
        }
    }
}
```

PHP

```
<?php
$url = 'https://api.example.com/data';

// สร้าง session cURL
$ch = curl_init();
curl_setopt($ch, CURLOPT_URL, $url);
curl_setopt($ch, CURLOPT_RETURNTRANSFER, true);

// ส่ง GET request
$response = curl_exec($ch);
if (!$response) {
    die('Error: ' . curl_error($ch) . ' - Code: ' . curl_errno($ch));
}

curl_close($ch);
echo 'Response: ' . $response;
?>
```

PHP

```
<?php
$url = 'https://api.example.com/data';
$data = array('key' => 'value');

// ใช้ cURL เพื่อส่ง POST request
$ch = curl_init($url);
curl_setopt($ch, CURLOPT_RETURNTRANSFER, true);
curl_setopt($ch, CURLOPT_POSTFIELDS, http_build_query($data));
curl_setopt($ch, CURLOPT_POST, true);

// รับ response
$response = curl_exec($ch);
if (!$response) {
    die('Error: "' . curl_error($ch) . '" - Code: ' . curl_errno($ch));
}

curl_close($ch);
echo 'Response: ' . $response;
?>
```





AI FOR THAI

APIs & SERVICES

Language



BASIC NLP
ประมวลผลภาษา



TAG SUGGESTION
แนะนำป้ายกำกับ



MACHINE TRANSLATION
แปลภาษา



SENTIMENT ANALYSIS
วิเคราะห์ความคิดเห็น

Vision



CHARACTER RECOGNITION
แปลงภาพอักษรเป็นข้อความ



OBJECT RECOGNITION
รู้จำวัตถุ



FACE ANALYTICS
วิเคราะห์ใบหน้า



PERSON & ACTIVITY
ANALYTICS
วิเคราะห์บุคคล

Conversation



SPEECH TO TEXT
แปลงเสียงพูดเป็นข้อความ



TEXT TO SPEECH
แปลงข้อความเป็นเสียงพูด



CHATBOT
สร้างแชทบอต

Registration

First Name*

Your first name

Last Name*

Your last name

Email*

your-email@mail.com

Username*

(Allow only A-Z, a-z, 0-9, '_')

Organization Name*

Organization

Organization Type*

กรุณาเลือกอาชีพของท่าน

- ☐ accept the **User Agreement**
- ☐ accept the **PDPA & Privacy Policy**



I'm not a robot



reCAPTCHA
Privacy - Terms

RESET

REGISTER

TLex+

LexTo+

Longan

Word Segmentation

POS Tagging

Named Entity
Recognition

Grapheme to Phoneme

Soundex

Word Approximation

Word Similarity

Text Cleansing

Text Summarization

Text Parser

ทีเล็กซ์พลัส บริการตัดคำด้วยเทคนิคการเรียนรู้ของเครื่อง (Machine learning) โดยใช้อัลกอริทึม Conditional Random Fields

จำนวนอักขระ = 17/1000

ทดสอบตัดคำภาษาไทย



วิเคราะห์



AI FOR THAI

[Home](#) [About](#) [Services](#) [Demo](#) [Use cases](#) [Corpus](#) [Developer](#) [News & Events](#) [Contact us](#) [FAQs](#)

Host	https://api.aiforthai.in.th/lextoplus
Method	GET/POST
Header	Apikey : VLc8IopU62H0kiSfodQCOPnVRHFagOzf Content-Type : application/x-www-form-urlencoded
Parameter	text : ข้อความที่ต้องการตัดคำ norm : การสกรูปตัวอักษรที่พิมพ์ซ้ำ (normalize) 0 = ไม่ทำการ normalize (default) 1 = ทำ normalize

CURL

GET

```
curl --location --request GET 'https://api.aiforthai.in.th/lextoplus?norm=1&text=พนักงานโรงแรม%20XYZ%20ให้บริการต้อนรับดีมาก' \  
--header 'Apikey: VLc8IopU62H0kiSfodQCOPnVRHFagOzf'
```

POST

```
curl --location --request POST 'https://api.aiforthai.in.th/lextoplus' \  
--header 'Apikey: VLc8IopU62H0kiSfodQCOPnVRHFagOzf' \  
--data 'norm=1&text=พนักงานโรงแรม XYZ ให้บริการต้อนรับดีมาก'
```

GET

```
1.  <?php
2.
3.  $curl = curl_init();
4.
5.  curl_setopt_array($curl, array(
6.      CURLOPT_URL => "https://api.aiforthai.in.th/lextoplus?text=".urlencode("พนักงานโรงแรม XYZ ให้บริการต้อนรับดีมากกก")."&norm=1",
7.      CURLOPT_RETURNTRANSFER => true,
8.      CURLOPT_ENCODING => "",
9.      CURLOPT_MAXREDIRS => 10,
10.     CURLOPT_TIMEOUT => 30,
11.     CURLOPT_HTTP_VERSION => CURL_HTTP_VERSION_1_1,
12.     CURLOPT_CUSTOMREQUEST => "GET",
13.     CURLOPT_HTTPHEADER => array(
14.         "Apikey: VLc8IopU62H0kiSfodQCOPnVRHFagOzf"
15.     )
16. ));
17.
18. $response = curl_exec($curl);
19. $err = curl_error($curl);
20.
21. curl_close($curl);
22.
23. if ($err) {
24.     echo "cURL Error #:" . $err;
25. } else {
26.     echo $response;
27. }
28. ?>
29.
```

POST

```
1. <?php
2.
3. $curl = curl_init();
4.
5. curl_setopt_array($curl, array(
6.     CURLOPT_URL => "https://api.aiforthai.in.th/lextoplus",
7.     CURLOPT_RETURNTRANSFER => true,
8.     CURLOPT_ENCODING => "",
9.     CURLOPT_MAXREDIRS => 10,
10.    CURLOPT_TIMEOUT => 30,
11.    CURLOPT_HTTP_VERSION => CURL_HTTP_VERSION_1_1,
12.    CURLOPT_CUSTOMREQUEST => "POST",
13.    CURLOPT_POSTFIELDS => "text=พนักงานโรงแรม XYZ ให้บริการต้อนรับดีมากกก&norm=1",
14.    CURLOPT_HTTPHEADER => array(
15.        "Content-Type: application/x-www-form-urlencoded",
16.        "Apikey: VLc8IopU62H0kiSfodQCOPnVRHFagOzf"
17.    )
18. ));
19.
20. $response = curl_exec($curl);
21. $err = curl_error($curl);
22.
23. curl_close($curl);
24.
25. if ($err) {
26.     echo "cURL Error #:" . $err;
27. } else {
28.     echo $response;
29. }
30. ?>
```

GET

```
1. import requests
2.
3. url = 'https://api.aiforthai.in.th/lextoplus'
4.
5. headers = {'Apikey': "VLc8IopU62H0kiSfodQCOPnVRHFagOzf"}
6.
7. text = 'พนักงานโรงแรม XYZ ให้บริการต้อนรับดีมากกก'
8.
9. params = {'text':text, 'norm':'1'}
10.
11. response = requests.get(url, params=params, headers=headers)
12.
13. print(response.json())
```

POST

```
1. import requests
2.
3. url = 'https://api.aiforthai.in.th/lextoplus'
4.
5. headers = {'Apikey': "VLc8IopU62H0kiSfodQCOPnVRHFagOzf"}
6.
7. text = 'พนักงานโรงแรม XYZ ให้บริการต้อนรับดีมากกก'
8.
9. params = {'text':text, 'norm':'1'}
10.
11. response = requests.post(url, data=params, headers=headers)
12.
13. print(response.json())
```



FastAPI

สร้าง **API** ด้วยตัวเอง

Download Visual Studio Code

Download Visual Studio Code

Free and built on open source. Integrated Git, debugging and extensions.



↓ Windows

Windows 10, 11

User Installer	x64	Arm64
System Installer	x64	Arm64
.zip	x64	Arm64
CLI	x64	Arm64



↓ .deb

Debian, Ubuntu

↓ .rpm

Red Hat, Fedora, SUSE

.deb	x64	Arm32	Arm64
.rpm	x64	Arm32	Arm64
.tar.gz	x64	Arm32	Arm64
Snap	Snap Store		
CLI	x64	Arm32	Arm64



↓ Mac

macOS 10.15+

.zip	Intel chip	Apple silicon	Universal
CLI	Intel chip	Apple silicon	

<https://code.visualstudio.com/download>

1. Install “wsl” extension

ค้นหา wsl

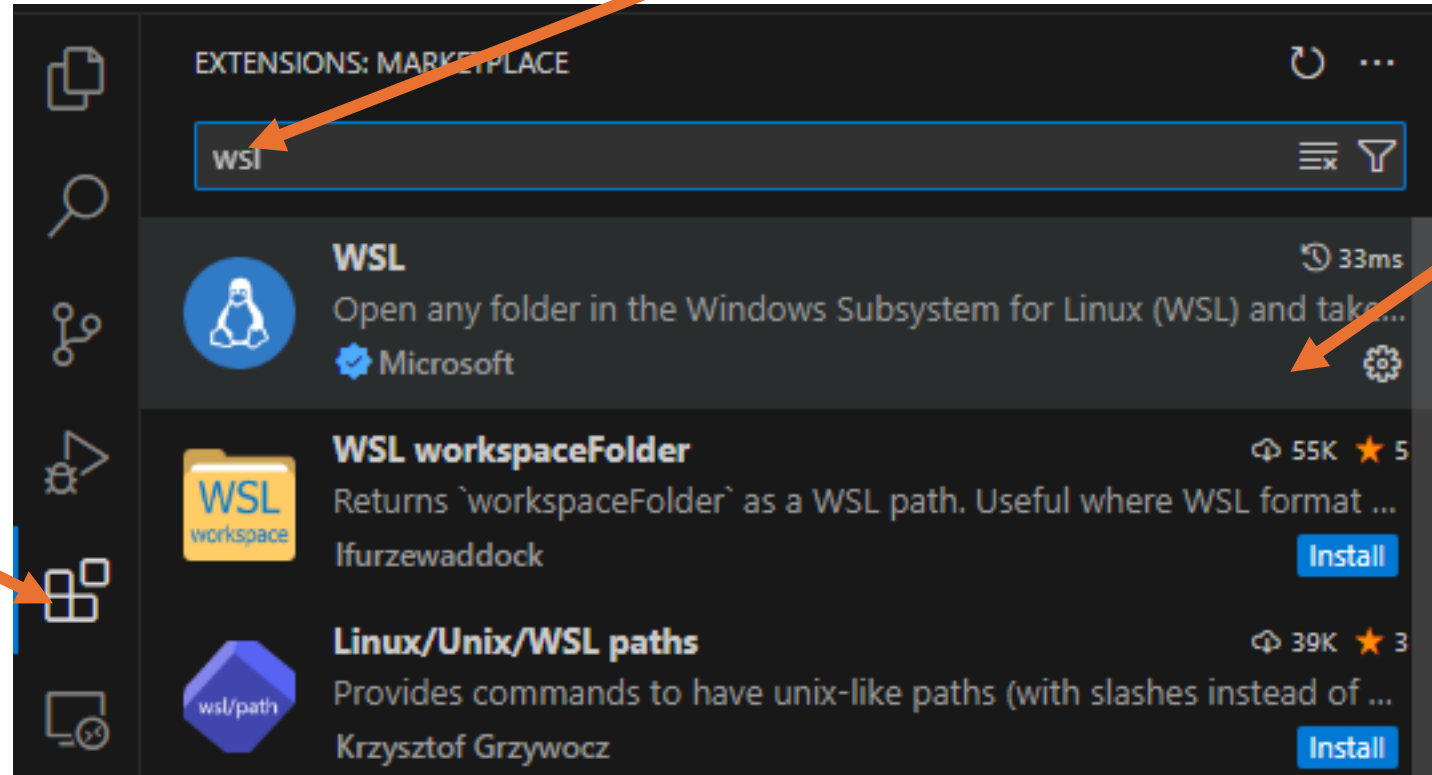
2

กด install

3

เลือก tab extension

1



2. Connect to “wsl”

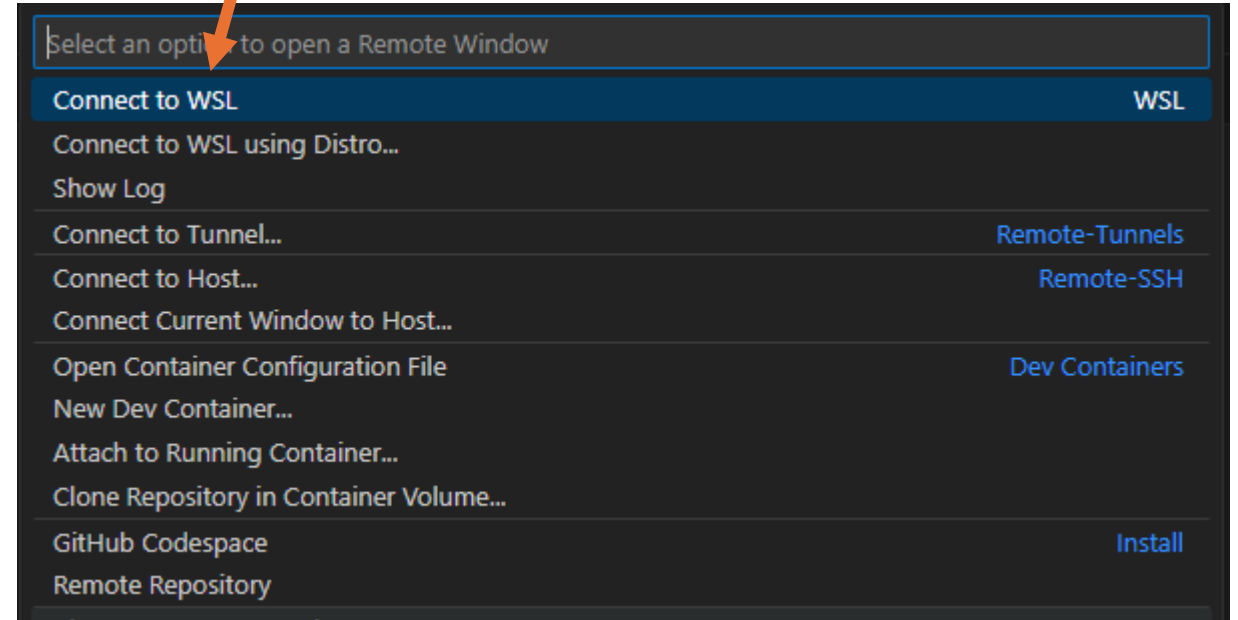
Click มุมซ้ายล่าง

1



เลือก Connect to WSL

2

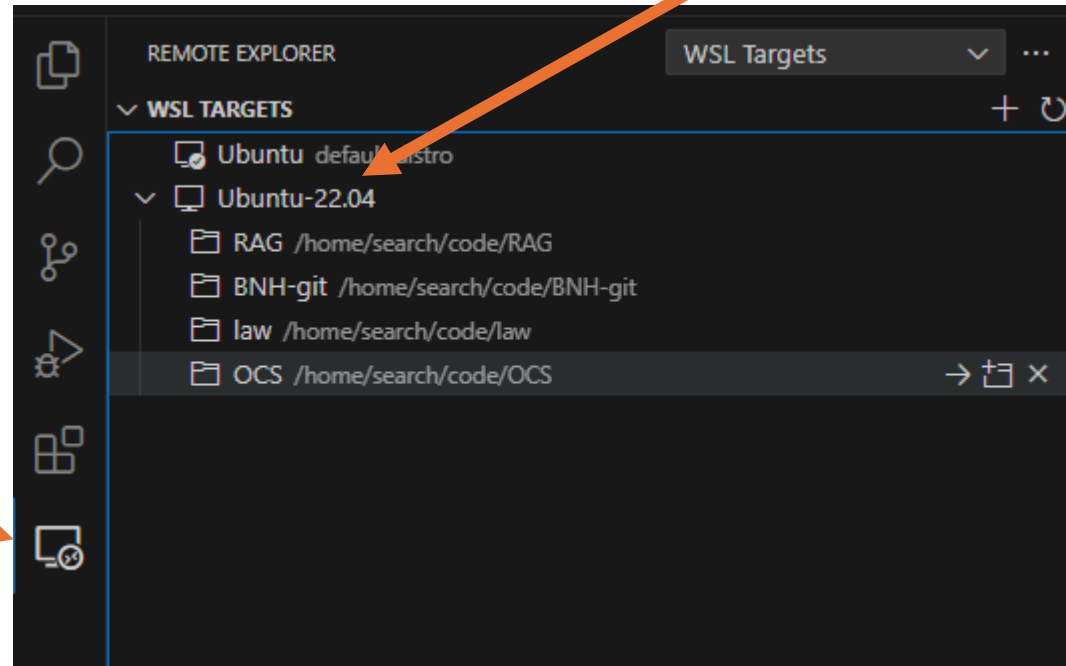


3. Select workspace

เลือก tab Remote Explorer

เลือก Ubuntu

2



connect

3



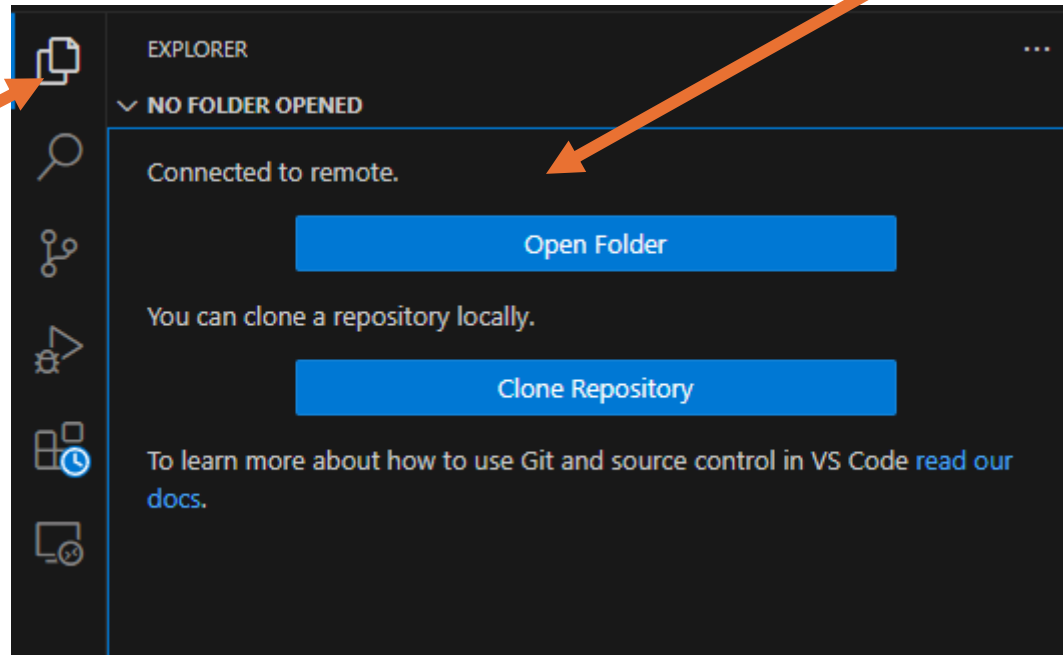
4. Open Folder

Open Folder

2

1

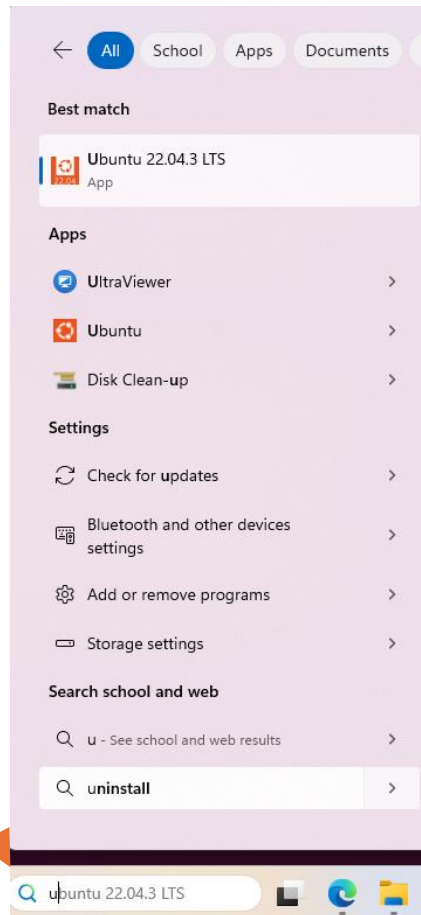
เลือก tab
Explorer



5. Prepare Folder

เปิด Ubuntu terminal

2



1

ค้นหา
Ubuntu
ที่ได้ติดตั้งไว้

```
search ~  
} mkdir api  
search ~  
} cd api  
search ~/api  
} pwd  
/home/search/api  
search ~/api  
}
```

เตรียม workspace

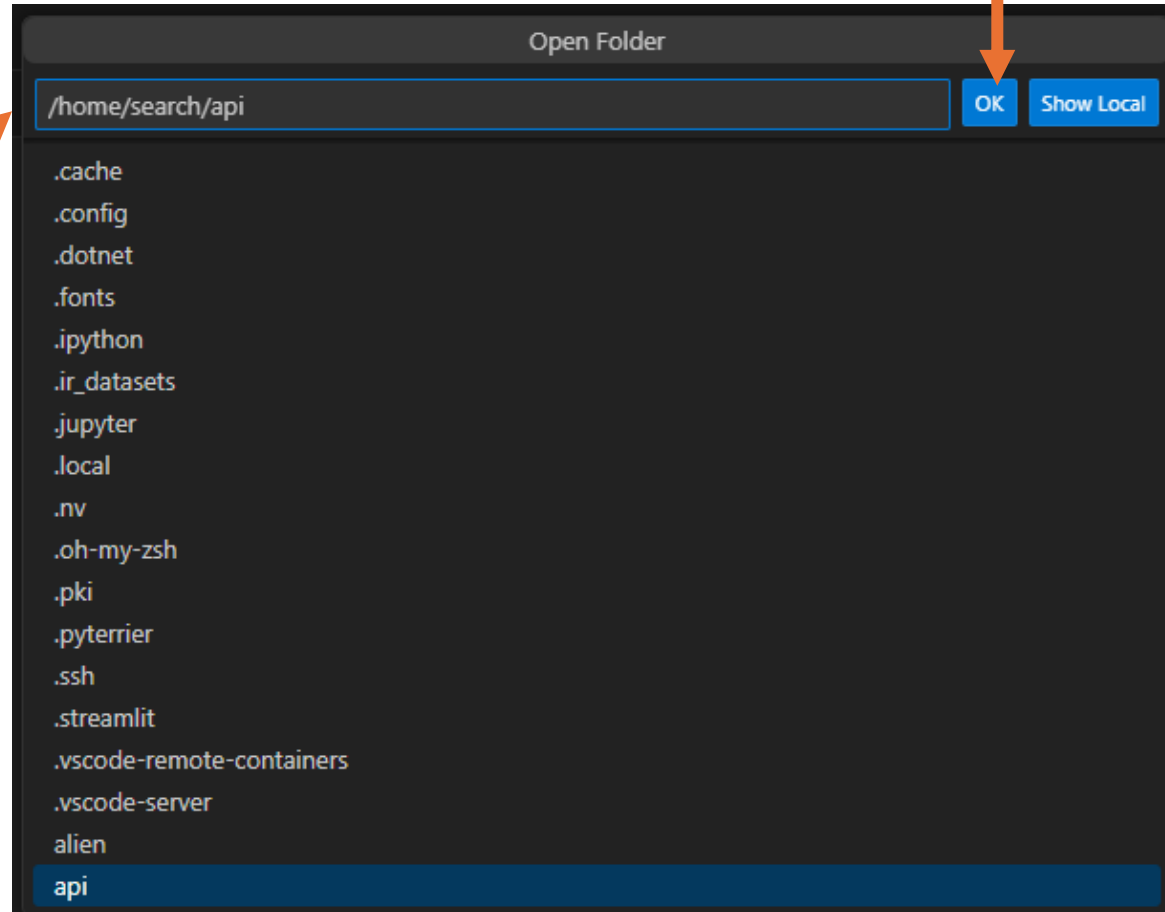
3

6. Select Folder

เลือก Folder ที่ต้องการ

กด OK

2

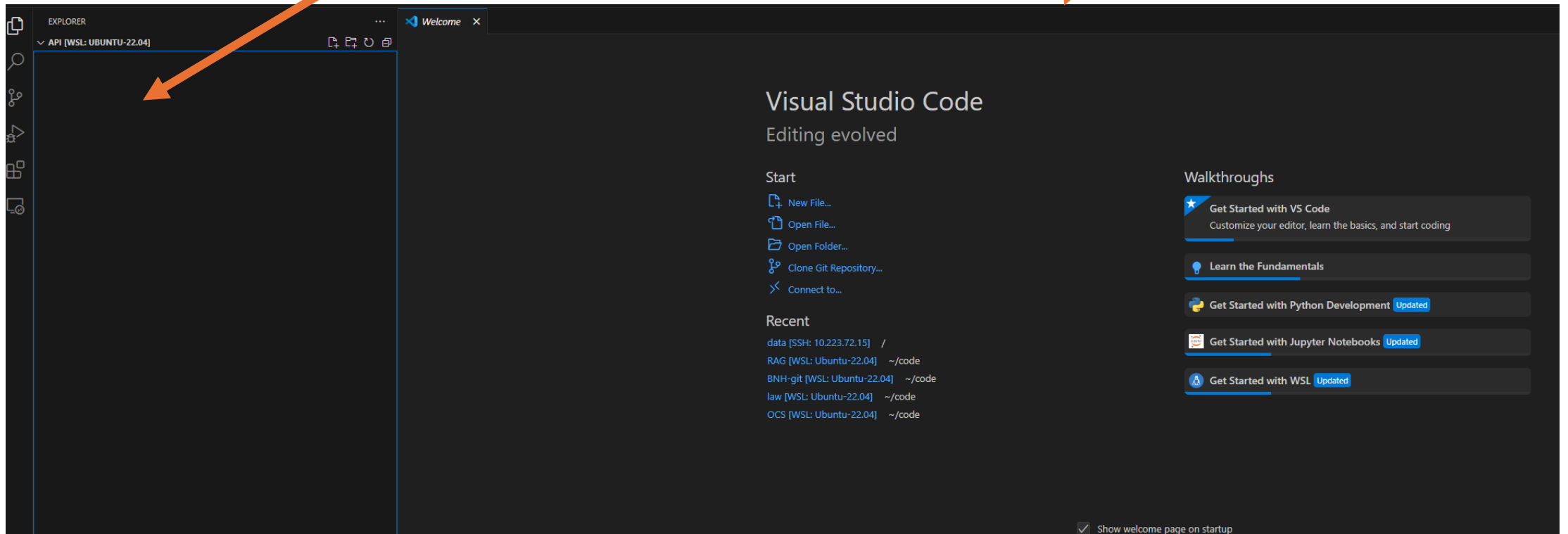


7. Workspace

1 File Explorer

Coding Area

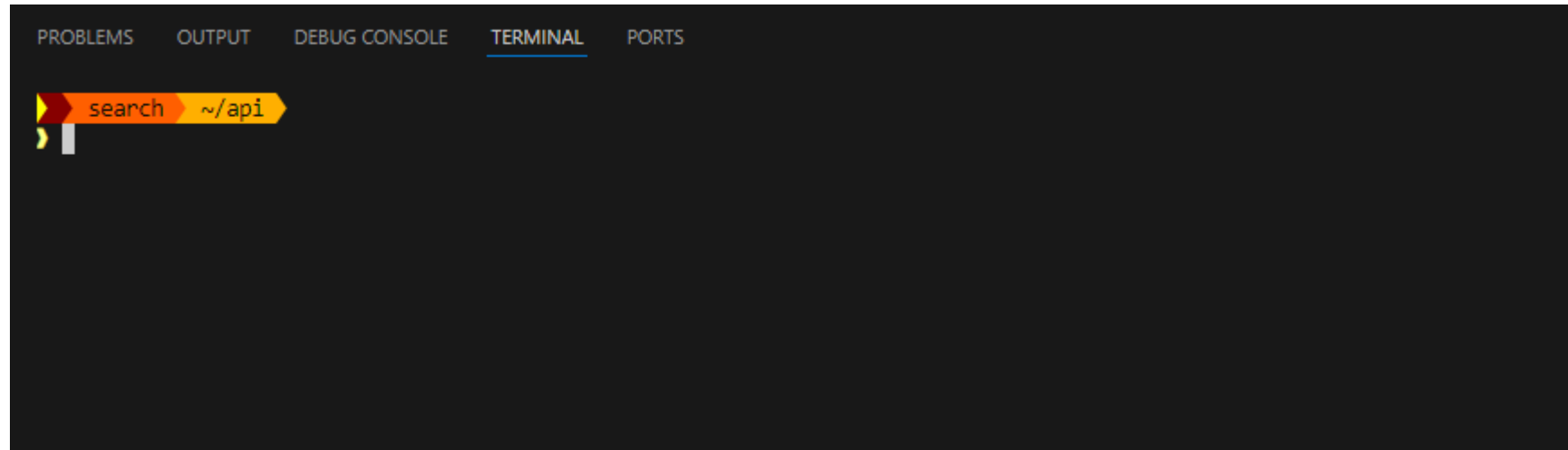
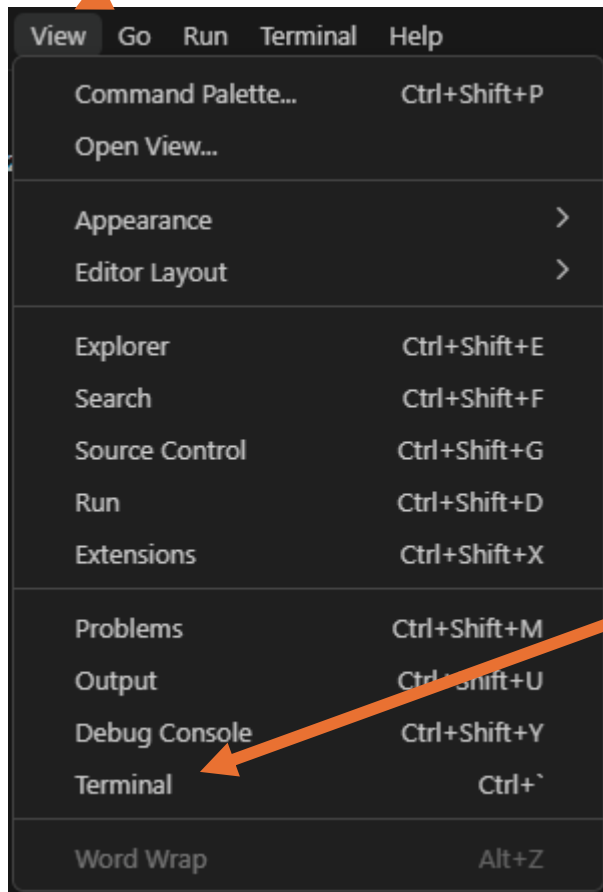
2



8. Open Terminal

1

menu → view



Terminal

2

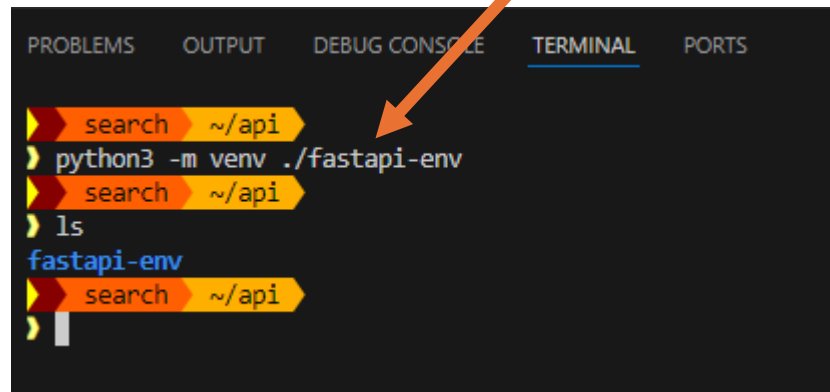
8. sudo apt install python3.10-venv

```
➤ sudo apt install python3.10-venv
[sudo] password for search:
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
The following additional packages will be installed:
  python3-pip-whl python3-setuptools-whl
The following NEW packages will be installed:
  python3-pip-whl python3-setuptools-whl python3.10-venv
0 upgraded, 3 newly installed, 0 to remove and 5 not upgraded.
Need to get 2473 kB of archives.
After this operation, 2884 kB of additional disk space will be used.
Do you want to continue? [Y/n] y
Get:1 http://archive.ubuntu.com/ubuntu jammy-updates/universe amd64 python3-pip-whl all 22.0.2+dfsg-1ubuntu0.4 [1680 kB]
Get:2 http://archive.ubuntu.com/ubuntu jammy-updates/universe amd64 python3-setuptools-whl all 59.6.0-1.2ubuntu0.22.04.1 [788 kB]
Get:3 http://archive.ubuntu.com/ubuntu jammy-updates/universe amd64 python3.10-venv amd64 3.10.12-1~22.04.3 [5716 B]
Fetched 2473 kB in 1s (2002 kB/s)
Selecting previously unselected package python3-pip-whl.
(Reading database ... 61161 files and directories currently installed.)
Preparing to unpack .../python3-pip-whl_22.0.2+dfsg-1ubuntu0.4_all.deb ...
Unpacking python3-pip-whl (22.0.2+dfsg-1ubuntu0.4) ...
Selecting previously unselected package python3-setuptools-whl.
Preparing to unpack .../python3-setuptools-whl_59.6.0-1.2ubuntu0.22.04.1_all.deb ...
Unpacking python3-setuptools-whl (59.6.0-1.2ubuntu0.22.04.1) ...
Selecting previously unselected package python3.10-venv.
Preparing to unpack .../python3.10-venv_3.10.12-1~22.04.3_amd64.deb ...
Unpacking python3.10-venv (3.10.12-1~22.04.3) ...
```

9. สร้าง virtual environment

1

`python3 -m venv ./fastapi-env`



The screenshot shows a terminal window with a dark background. At the top, there are tabs for 'PROBLEMS', 'OUTPUT', 'DEBUG CONSOLE', 'TERMINAL' (which is selected), and 'PORTS'. The terminal history shows the following commands and outputs:

- `search ~/api`
- `python3 -m venv ./fastapi-env`
- `search ~/api`
- `ls`
- `fastapi-env`
- `search ~/api`

An orange arrow points from the blue circle containing the number '1' to the `python3 -m venv ./fastapi-env` command in the terminal.

10. Activate environment

1 source fastapi-env/bin/activate

```
> search ~/api  
> source fastapi-env/bin/activate  
> search ~/api  
(fastapi-env) >
```

11. ติดตั้ง library ที่เกี่ยวข้อง คือ FastAPI และ Uvicorn

```
(fastapi-env) ➤ pip install fastapi  
130 ➤➤➤ search ➤ ~/api  
(fastapi-env) ➤ pip install uvicorn
```

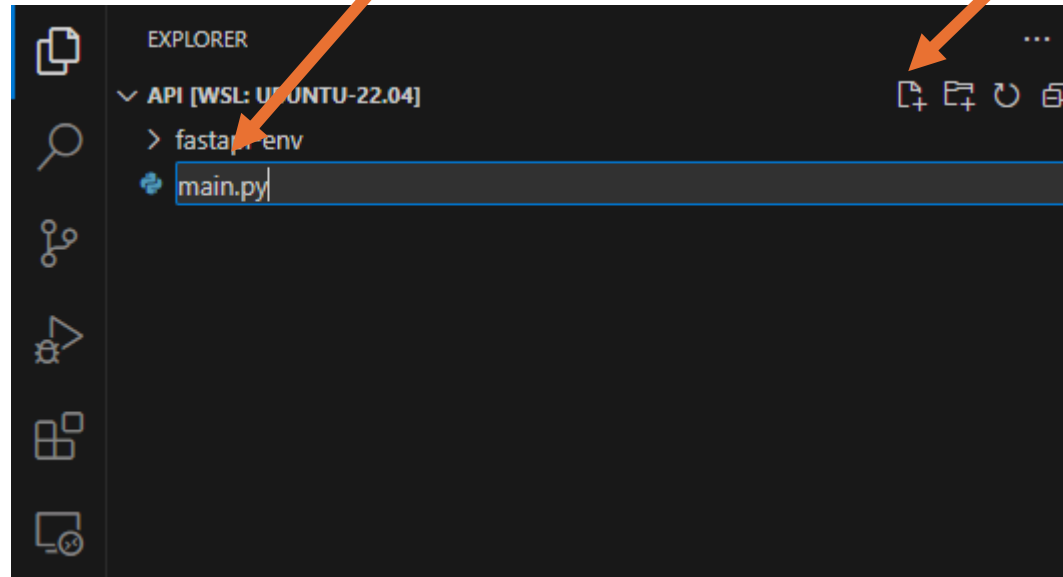
First API

ใส่ชื่อไฟล์ main.py

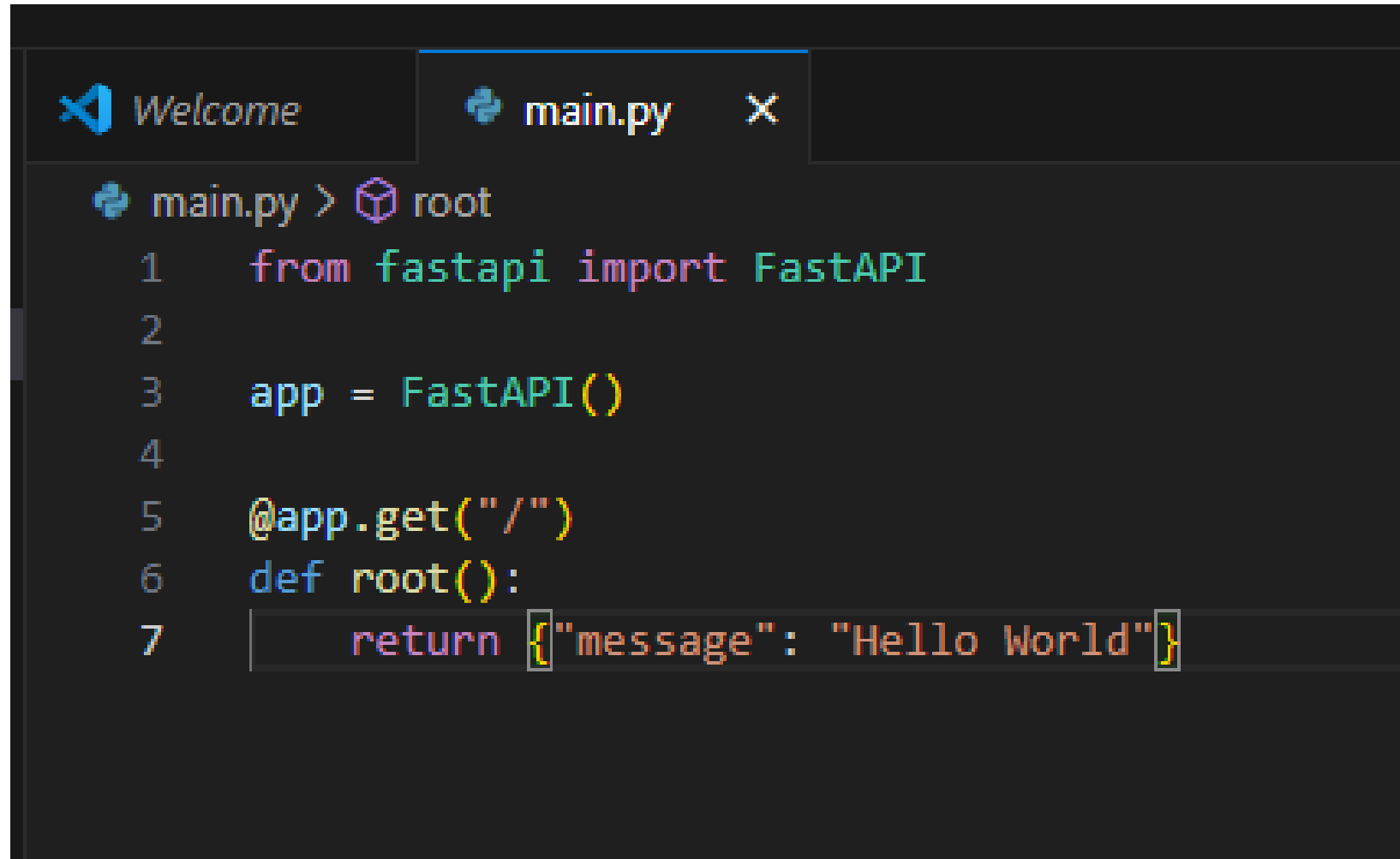
2

1

New file



First API



The image shows a code editor window with two tabs: 'Welcome' and 'main.py'. The 'main.py' tab is active. The code in the editor is as follows:

```
main.py > root
1  from fastapi import FastAPI
2
3  app = FastAPI()
4
5  @app.get("/")
6  def root():
7      return {"message": "Hello World"}
```

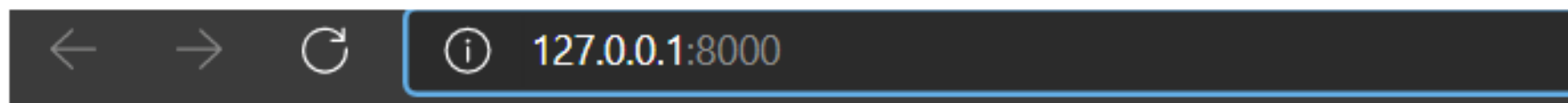
uvicorn main:app --reload

```
(fastapi-env) > uvicorn main:app --reload  
  
INFO:      Will watch for changes in these directories: ['/home/search/api']  
INFO:      Uvicorn running on http://127.0.0.1:8000 (Press CTRL+C to quit)  
INFO:      Started reloader process [214116] using WatchFiles  
INFO:      Started server process [214118]  
INFO:      Waiting for application startup.  
INFO:      Application startup complete.  
INFO:      127.0.0.1:41408 - "GET / HTTP/1.1" 200 OK  
INFO:      127.0.0.1:41408 - "GET /favicon.ico HTTP/1.1" 404 Not Found
```

```
█
```


Open Browser

ทำการเปิด <http://127.0.0.1:8000> บน web browser ดู จะเห็น response ที่ส่งกลับมาจาก route แรกที่เราสร้างขึ้น



```
{"message": "Hello World"}
```

http://127.0.0.1:8000/docs

ที่นี่เรามาลองเปิด API document โดยเข้าไปที่ <http://127.0.0.1:8000/docs> ที่เป็นของ Swagger UI ซึ่งจะแสดง API route ที่เราสร้างขึ้นมา

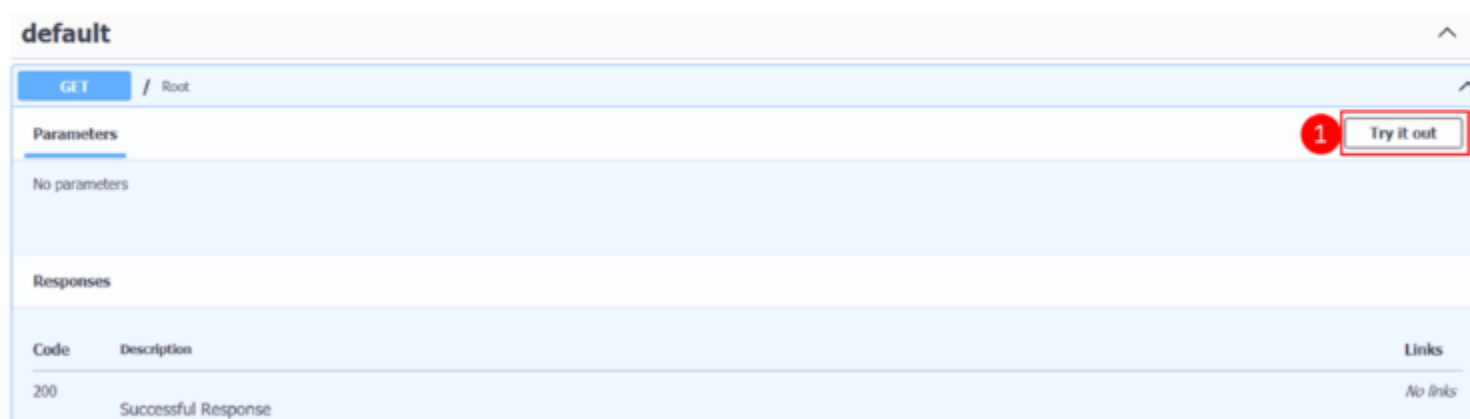
FastAPI 0.1.0 OAS3
/openapi.json

default

GET / Root

Swagger UI

ในหน้านี้เราสามารถยิงทดสอบ API จริงดูได้เลย ทำให้ไม่ต้องเสียเวลาสร้าง request ขึ้นมาเอง การทดสอบให้เลือก route ที่ต้องการทดสอบ จากนั้นกดปุ่ม Try it out และปุ่ม Execute เพื่อทำการส่ง request ไป จะเห็น response กลับมาเป็นผลลัพธ์ข้อความว่า Hello World ของเรา



The screenshot shows an API documentation interface for a 'default' endpoint. The endpoint is a GET request to the 'Root' path. The 'Parameters' section indicates 'No parameters'. The 'Responses' section shows a 200 status code with the description 'Successful Response' and 'No links'. A red circle with the number 1 highlights the 'Try it out' button.

Code	Description	Links
200	Successful Response	No links

หลังกดเลือก route ให้กดปุ่ม Try it out

default

GET / Root

Parameters Cancel

No parameters

2 Execute Clear

Responses

curl

```
curl -X 'GET' \
  'http://127.0.0.1:8000/' \
  -H 'accept: application/json'
```

Request URL

```
http://127.0.0.1:8000/
```

Server response

Code	Details
200	3 ผลลัพธ์ Response body <pre>{ "message": "hello world" }</pre> Download

กดปุ่ม Execute เพื่อส่ง request แล้วจะมี response กลับมาตามการทำงานของ route

ลองสร้าง route ที่มีการรับ parameter

โดยเราลองมาเริ่มที่ method GET ที่มีการรับ path parameter สำหรับตัวอย่างนี้ เราจะกำหนด route เป็น “/user/{user_id}” มีตัวแปรที่รับมาคือ user_id ซึ่งเราสามารถกำหนดประเภทของตัวแปรได้ ในที่นี้เราจะกำหนดให้เป็น integer โดยภายในฟังก์ชันเราจะไม่มีการทำอะไร แต่จะส่งค่า user_id ที่ได้รับมาคืนกลับไปเพื่อแสดงผลเลย route นี้สามารถเขียนได้ตามโค้ดข้างล่างนี้

```
@app.get("/user/{user_id}")
async def get_user_by_id(user_id: int):
    return {"user_id": user_id}
```

เมื่ออัปเดตโค้ดเรียบร้อยแล้ว **server** จะทำการ **restart** ให้อัตโนมัติ เมื่อเราทำการ **refresh** หน้า **Swagger** ก็จะได้เห็น **route** ใหม่ที่เราสร้างขึ้น และมีส่วนตัวแปร **user_id** ไว้ให้เราทดลองกรอกค่าตัวแปรได้เลย จะเห็นว่ามีกรอบตรวจสอบค่าที่เราใส่เข้าไป เช่น ในตัวอย่างข้างล่าง หากเรากรอก “one” เข้าไปและกดปุ่ม **Execute** จะมีการแจ้งเตือนว่าใส่ค่าไม่ตรงประเภทของตัวแปรที่กำหนด

GET

/user/{user_id} Get User By Id

Parameters

Name	Description
user_id * required integer (path)	<div>one</div>

Execute

แจ้งเตือนเมื่อใส่ค่าตัวแปรผิดประเภท

หากกรอกค่าถูกต้อง และกดปุ่ม **execute** ผลลัพธ์ที่เป็น **user_id** ก็จะออกมาตามโค้ดที่เราเขียนไปข้างต้น ดังรูปข้างล่าง

The screenshot shows a REST client interface with the following sections:

- Parameters:** A table with columns 'Name' and 'Description'. It contains one parameter: **user_id** (integer, required, path) with a value of **1**. There are 'Execute' and 'Clear' buttons.
- Responses:**
 - Curl:** `curl -X 'GET' \ 'http://127.0.0.1:8000/user/1' \ -H 'accept: application/json'`
 - Request URL:** `http://127.0.0.1:8000/user/1`
 - Server response:**

Code	Details
200	<p>Response body: <code>{ "user_id": 1 }</code></p> <p>Response headers: <code>content-length: 13 content-type: application/json date: Mon, 13 Sep 2021 10:15:33 GMT server: uvicorn</code></p>

เมื่อใส่ค่าถูกประเภท จะได้ response ออกมา

ทีนี้ลองเพิ่ม **query parameter** กันดูบ้าง โดยสามารถใส่ตัวแปรเพิ่มเข้าไปในฟังก์ชัน **get_user_by_id** ต่อกับ **user_id** ได้เลย ในตัวอย่างนี้ เราจะทำการเพิ่มตัวแปรชื่อ **type** ประเภทตัวแปรเป็น **string** และกำหนดให้สามารถเลือกได้ว่าจะใส่ค่ามาหรือไม่ใส่ค่ามาก็ได้ โดยการเพิ่ม **Optional** เข้าไป หากอยากกำหนดค่า **default** เมื่อไม่ได้มีการใส่ค่ามา ก็สามารถใช้ **= "default"** ได้เลย ในตัวอย่างจะกำหนดค่า **default** เป็น **"normal"** โค้ดที่แก้ไขเพิ่มเติมจะเป็นดังนี้

```
from typing import Optional

@app.get("/user/{user_id}")
async def get_user_by_id(user_id: int, type: Optional[str] = "normal"):
    return {"user_id": user_id, "type": type}
```


กลับมา refresh หน้า swagger ของเรา ลองทดสอบไม่ใส่ค่าตัวแปร **type** เมื่อกดปุ่ม **execute** จะได้ค่า **default** ที่ตั้งไว้คือ “normal” คือนกลับมาแทน ดังตัวอย่างข้างล่าง

The screenshot shows the Swagger UI for a GET endpoint `/user/{user_id}` with the description "Get User By Id". The "Parameters" section lists two parameters: `user_id` (integer, path, required) with a value of `1`, and `type` (string, query) with a value of `type`. Below the parameters are "Execute" and "Clear" buttons. The "Responses" section shows the curl command, request URL, and server response. The response body is a JSON object: `{ "user_id": 1, "type": "normal" }`. The response headers include `content-length: 29`, `content-type: application/json`, `date: Mon, 13 Sep 2021 10:47:32 GMT`, and `server: uvicorn`.

Name	Description
<code>user_id</code> * required integer (path)	<input type="text" value="1"/>
<code>type</code> string (query)	<input type="text" value="type"/>

Execute **Clear**

Responses

Curl

```
curl -X 'GET' \
  'http://127.0.0.1:8000/user/1' \
  -H 'accept: application/json'
```

Request URL

```
http://127.0.0.1:8000/user/1
```

Server response

Code	Details
200	<p>Response body</p> <pre>{ "user_id": 1, "type": "normal" }</pre> <p>Response headers</p> <pre>content-length: 29 content-type: application/json date: Mon, 13 Sep 2021 10:47:32 GMT server: uvicorn</pre>

ผลลัพธ์เมื่อกำหนด field type เป็น optional และ default value เป็น “normal”

GET

/user/{user_id} Get User By Id

Parameters

Cancel

Name	Description
user_id * required	
integer	1
(path)	
type	
string	special
(query)	

ExecuteClear

Responses

Curl

```
curl -X 'GET' \
'http://127.0.0.1:8000/user/1?type=special' \
-H 'accept: application/json'
```

Request URL

```
http://127.0.0.1:8000/user/1?type=special
```

Server response

Code	Details
200	<div>Response body<pre>{ "user_id": 1, "type": "special" }</pre><div>Download</div></div> <div>Response headers<pre>content-length: 30 content-type: application/json date: Fri, 24 Sep 2021 02:19:13 GMT server: uvicorn</pre></div>

ผลลัพธ์เมื่อทดลองกำหนดค่า field type เป็น special
จะเห็นว่าใน request url มีการส่งค่า query parameter ของ type เพิ่มขึ้นมา

กำหนด request body ด้วย pydantic

ขั้นตอนต่อไป เราจะมาลองทำ **method POST** เพื่อใช้สำหรับสร้าง **item** ขึ้นมา โดยรายละเอียดของ **item** ที่ผู้ใช้ต้องการสร้างขึ้นนี้ เราจะกำหนดให้ส่งมาใน **request body** ในรูปแบบของ **JSON** ซึ่งเราจะใช้ **library pydantic** มาช่วยในการจัดการและช่วยตรวจสอบความถูกต้องของโครงสร้างของข้อมูลที่ถูกส่งเข้ามา

เริ่มต้นจากการ import BaseModel class ของ pydantic ซึ่งสามารถเขียน import ได้ตามนี้

```
from pydantic import BaseModel
```

จากนั้นเราจะเขียน class Item ที่สืบทอด BaseModel มา โดยเราจะกำหนด field และ type ของข้อมูลที่เราต้องการในนี้

```
class Item(BaseModel):  
    id: int  
    name: str  
    item_type: Optional[str] = "a"  
    total: int
```

จากนั้นสร้าง route ชื่อ “/item” และกำหนดประเภทของตัวแปร โดยใช้ class Item ที่เราเพิ่งสร้างขึ้นมา

```
@app.post("/item")
def create_item(item: Item):
    return item
```

เมื่อเปิด swagger คุณก็จะเห็น route /item ที่เป็น post method โดยมี request body เป็นรูปแบบเดียวกันกับ class Item

The image shows a Swagger UI interface for a REST API endpoint. At the top, there's a green bar with the text "POST /item Create Item". Below this, there's a section for "Parameters" with a "Try it out" button. The "Parameters" section indicates "No parameters". Below that, there's a "Request body" section with a red "required" label and a dropdown menu set to "application/json". At the bottom, there's a section for "Example Value" and "Schema". The "Example Value" section shows a JSON object: {"id": 0, "name": "string", "item_type": "a", "total": 0}.

POST /item Create Item

Parameters Try it out

No parameters

Request body required application/json

Example Value | Schema

```
{
  "id": 0,
  "name": "string",
  "item_type": "a",
  "total": 0
}
```

POST /item กับ request body ตามโครงสร้างของ Item

หากเราทดลองใส่ **request body** โดยมี **fields** ไม่ครบตามที่ต้องการ ระบบก็จะส่งข้อความ **error** กลับ เช่น ตัวอย่างในรูปข้างล่างไม่ได้ทำการส่ง **field total** ไปใน **request body** ด้วย ระบบก็ตอบกลับมามี **field total** ที่หายไป

POST /item Create Item

Parameters Cancel Reset

No parameters

Request body **required** application/json

```
{  
  "id": 0,  
  "name": "string",  
  "item_type": "a"  
}
```

Execute Clear

ไม่ได้กำหนดค่า total ใน request body

Responses

Curl

```
curl -X 'POST' \
  'http://127.0.0.1:8000/item' \
  -H 'accept: application/json' \
  -H 'Content-Type: application/json' \
  -d '{
    "id": 0,
    "name": "string",
    "item_type": "a"
  }'
```

Request URL

<http://127.0.0.1:8000/item>

Server response

Code

Details

422

Error: Unprocessable Entity

Response body

```
{
  "detail": [
    {
      "loc": [
        "body",
        "item",
        "total"
      ],
      "msg": "field required",
      "type": "value_error.missing"
    }
  ]
}
```



Download

Response headers

```
content-length: 96
content-type: application/json
date: Mon, 13 Sep 2021 06:11:00 GMT
server: uvicorn
```

response error กลับมาว่า field total ไม่มีค่า

FastAPI and DATABASE

SQLAlchemy

```
pip install databases, sqlalchemy, aiosqlite
```

Connect Database

```
from typing import List

import databases
import sqlalchemy
from fastapi import FastAPI
from pydantic import BaseModel

# SQLAlchemy specific code, as with any other app
DATABASE_URL = "sqlite:///./test.db"
# DATABASE_URL = "postgresql://user:password@postgresserver/db"

database = databases.Database(DATABASE_URL)
```

Create a metadata object.

Create a table notes using the metadata object.

```
metadata = sqlalchemy.MetaData()
```

```
notes = sqlalchemy.Table(  
    "notes",  
    metadata,  
    sqlalchemy.Column("id", sqlalchemy.Integer, primary_key=True),  
    sqlalchemy.Column("text", sqlalchemy.String),  
    sqlalchemy.Column("completed", sqlalchemy.Boolean),  
)
```

Create an engine.

Create all the tables from the metadata object.

```
engine = sqlalchemy.create_engine(  
    DATABASE_URL, connect_args={"check_same_thread": False}  
)  
metadata.create_all(engine)
```

Create models

Create Pydantic models for:

- Notes to be created (NoteIn).
- Notes to be returned (Note).

```
class NoteIn(BaseModel):  
    text: str  
    completed: bool
```

```
class Note(BaseModel):  
    id: int  
    text: str  
    completed: bool
```

Connect and disconnect

```
main_app_lifespan = app.router.lifespan_context

@asynccontextmanager
async def lifespan_wrapper(app):
    print("sub startup")
    await database.connect()
    async with main_app_lifespan(app) as maybe_state:
        yield maybe_state
    print("sub shutdown")
    await database.disconnect()

app.router.lifespan_context = lifespan_wrapper
```

Read notes

```
@app.get("/notes/", response_model=List[Note])
async def read_notes():
    query = notes.select()
    return await database.fetch_all(query)
```

Notice the `response_model=List[Note]`

It uses `typing.List`.

That documents (and validates, serializes, filters) the output data, as a list of Notes.

Create notes

```
@app.post("/notes/", response_model=Note)
async def create_note(note: NoteIn):
    query = notes.insert().values(text=note.text, completed=note.completed)
    last_record_id = await database.execute(query)
    return {**note.model_dump(), "id": last_record_id}
```