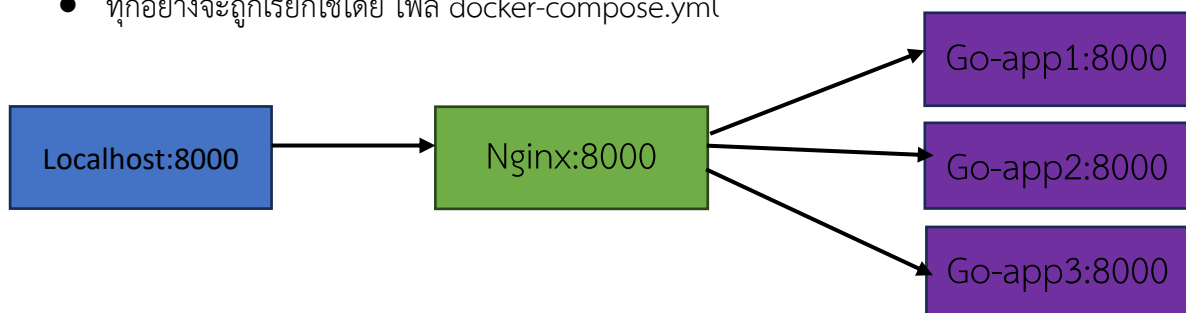




CRDU api Docker Run On AWS

Use go-lang postgre and nginx.

- เขียนเชื่อมต่อ DB postgresql สามารถรับ Create Read Update and Delete.
- ใช้ nginx เป็น Proxy ในการรับข้อมูล
- ลง PG-admin4 ในการจัดการข้อมูล
- ทุกอย่างจะถูกเรียกใช้โดย ไฟล์ docker-compose.yml



หลังจากเขียนโค้ดเสร็จแล้วก็มาลอง run โดยใช้คำสั่ง `$ docker compose up`

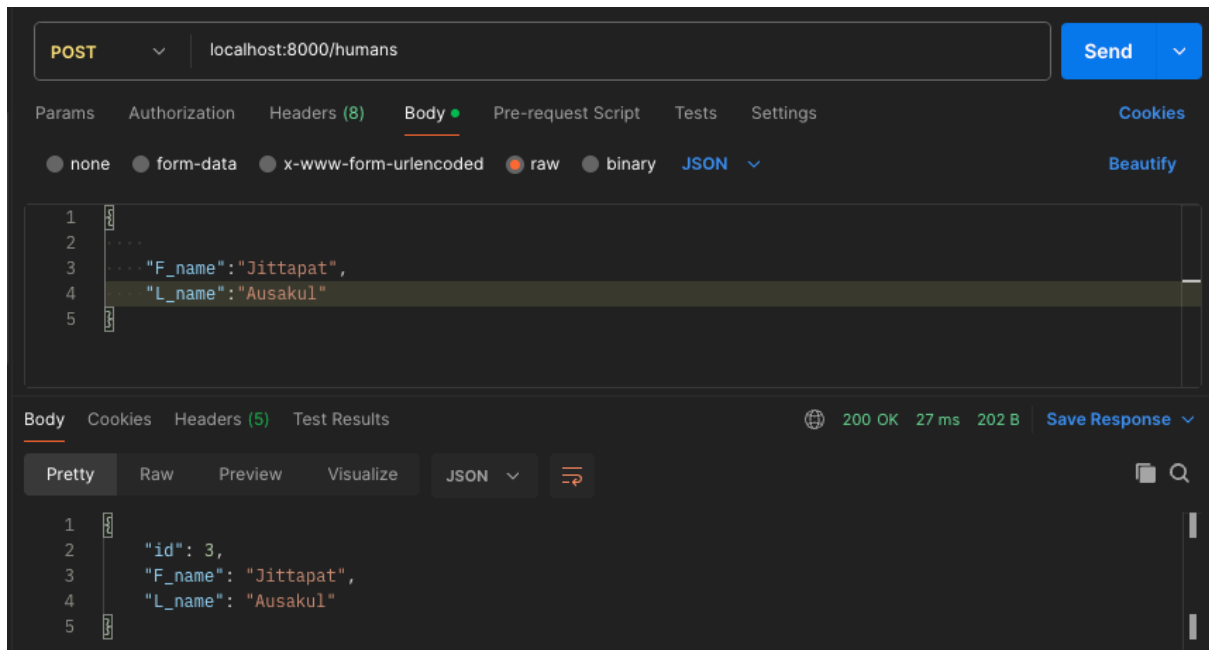
<input type="checkbox"/>		go-crud-dor	Running (4/4)		1 minute ago	0.13%				
<input type="checkbox"/>		go_db d5e2708cf	Running	postgres:12	5432:5432	1 minute ago	0.07%			
<input type="checkbox"/>		pg-admin 83ce0564b	Running	dpag/pgadmin4	5050:5050	1 minute ago	0.04%			
<input type="checkbox"/>		go-app-1 9f4ddaaf5c	Running	francescoxx/go-app		1 minute ago	0.02%			
<input type="checkbox"/>		nginx 4016cefc6b	Running	nginx:latest	8000:8000	1 minute ago	0%			

และ Run แบบ scale app ด้วยคำสั่ง `$ docker compose up --scale go-app=3 -d`

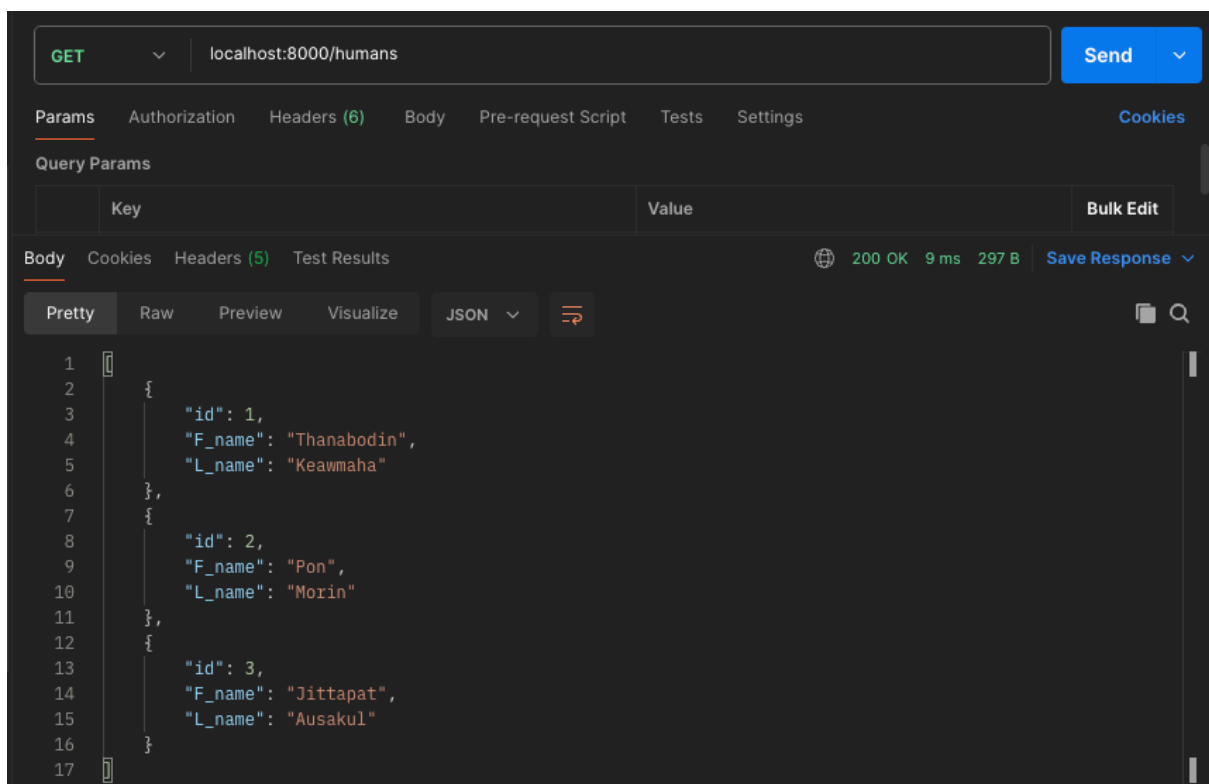
<input type="checkbox"/>	Name	Image	Status	Port(s)	Last started	CPU (%)	Actions
<input type="checkbox"/>	go-crud-dock		Running (6/6)		32 seconds ago	0.24%	
<input type="checkbox"/>	pg-admin d885b068dd8	dpage/pgadmin4	Running	5050:5050	33 seconds ago	0.15%	
<input type="checkbox"/>	go_db 18381aad822	postgres:12	Running	5432:5432	33 seconds ago	0.09%	
<input type="checkbox"/>	go-app-1 1f32de69e93i	francescoxx/go-app:1	Running		33 seconds ago	0%	
<input type="checkbox"/>	go-app-2 c99f0f9dac9f	francescoxx/go-app:1	Running		32 seconds ago	0%	
<input type="checkbox"/>	go-app-3 a295705d92b	francescoxx/go-app:1	Running		32 seconds ago	0%	
<input type="checkbox"/>	nginx 47fd5afe37ee	nginx:latest	Running	8000:8000	32 seconds ago	0%	

ใช้โปรแกรม Postman ในการทดสอบ api

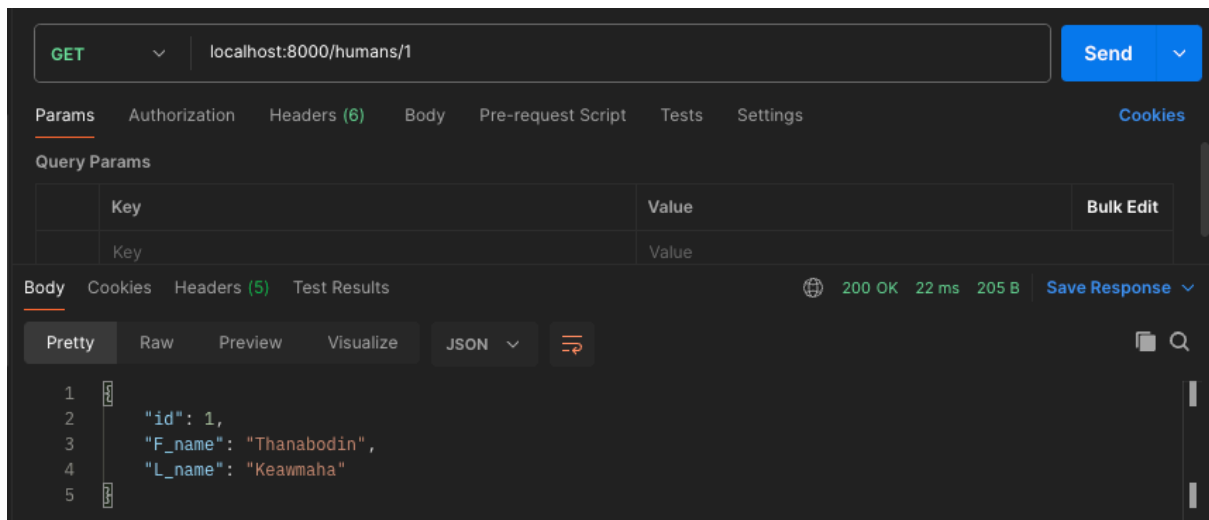
Create ข้อมูล POST : localhost:8000/humans



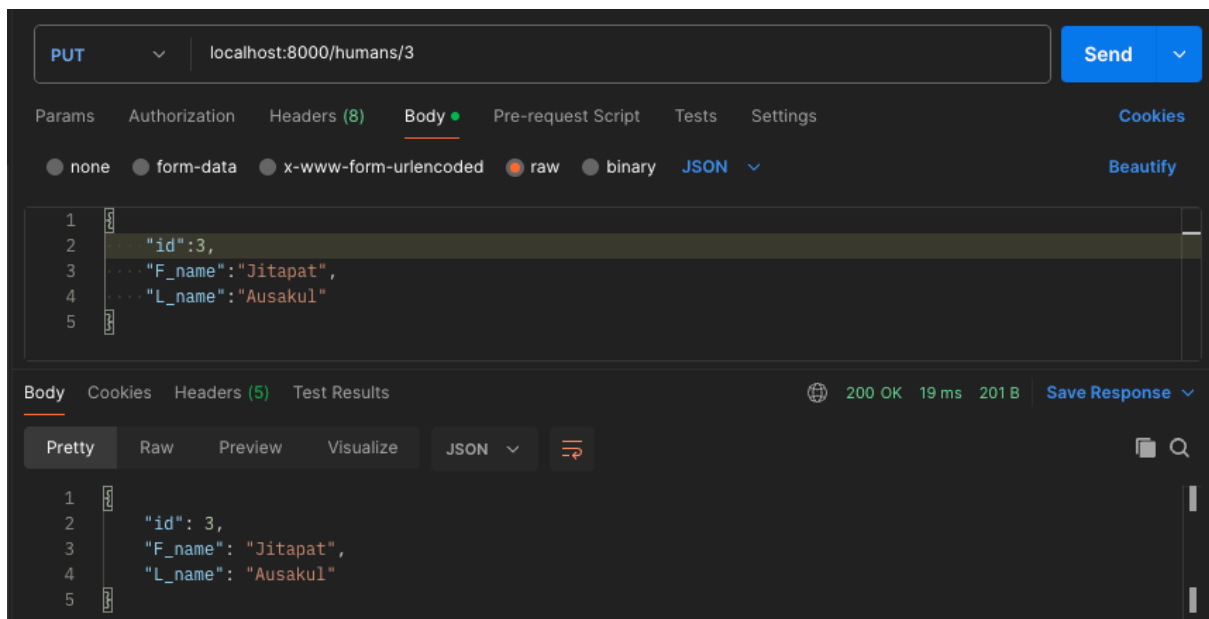
Read ข้อมูลทั้งหมด GET : localhost:8000/humans



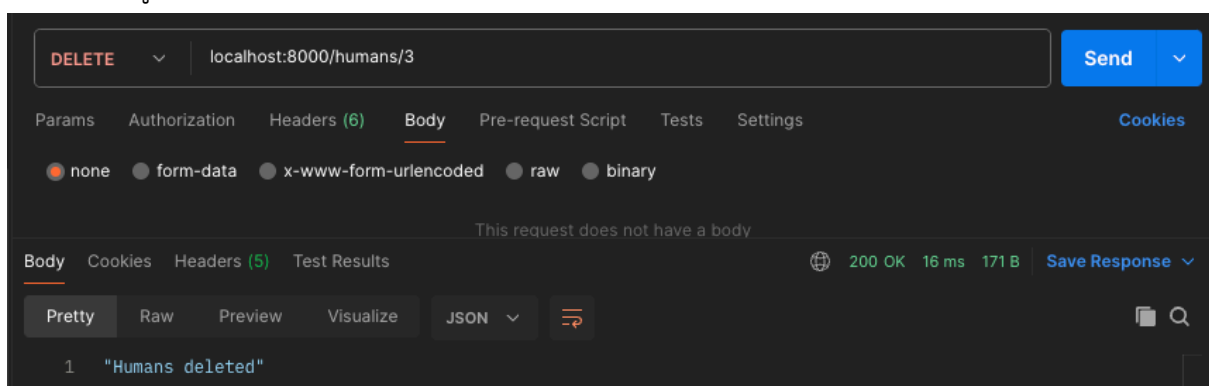
Read ข้อมูลตามไอดี GET : localhost:8000/humans/{id}



Update ข้อมูล PUT : localhost:8000/humans/{id}

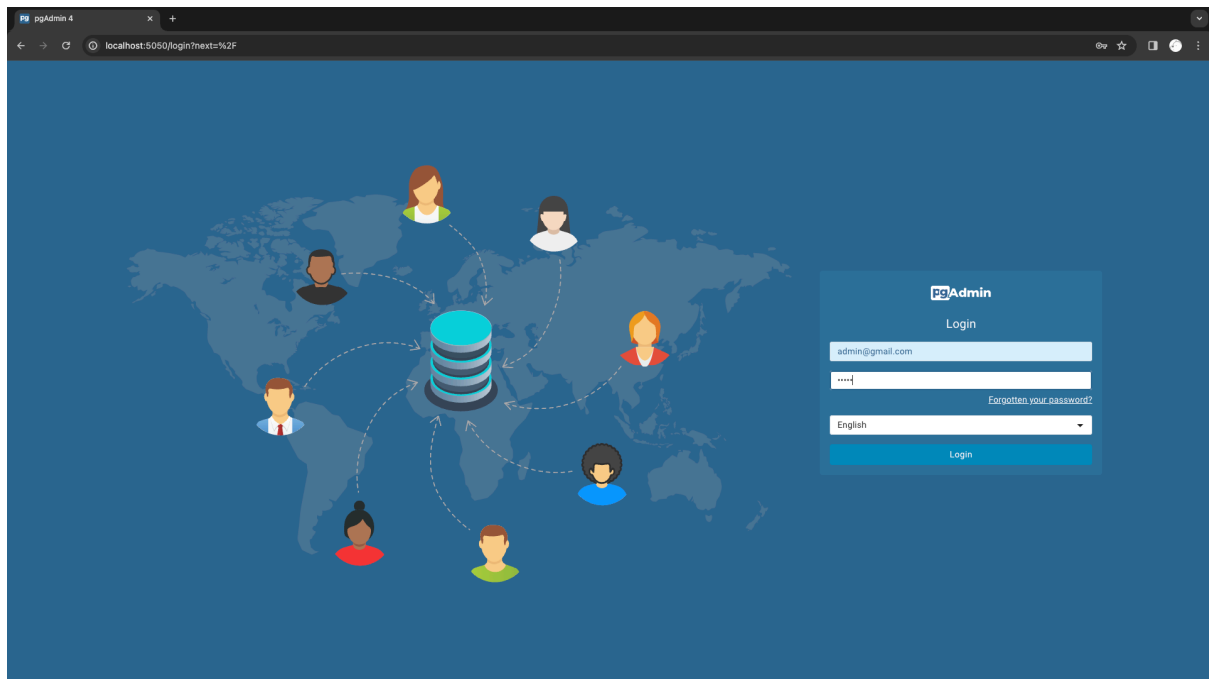


Delete ข้อมูล DELETE : localhost:8000/humans/{id}

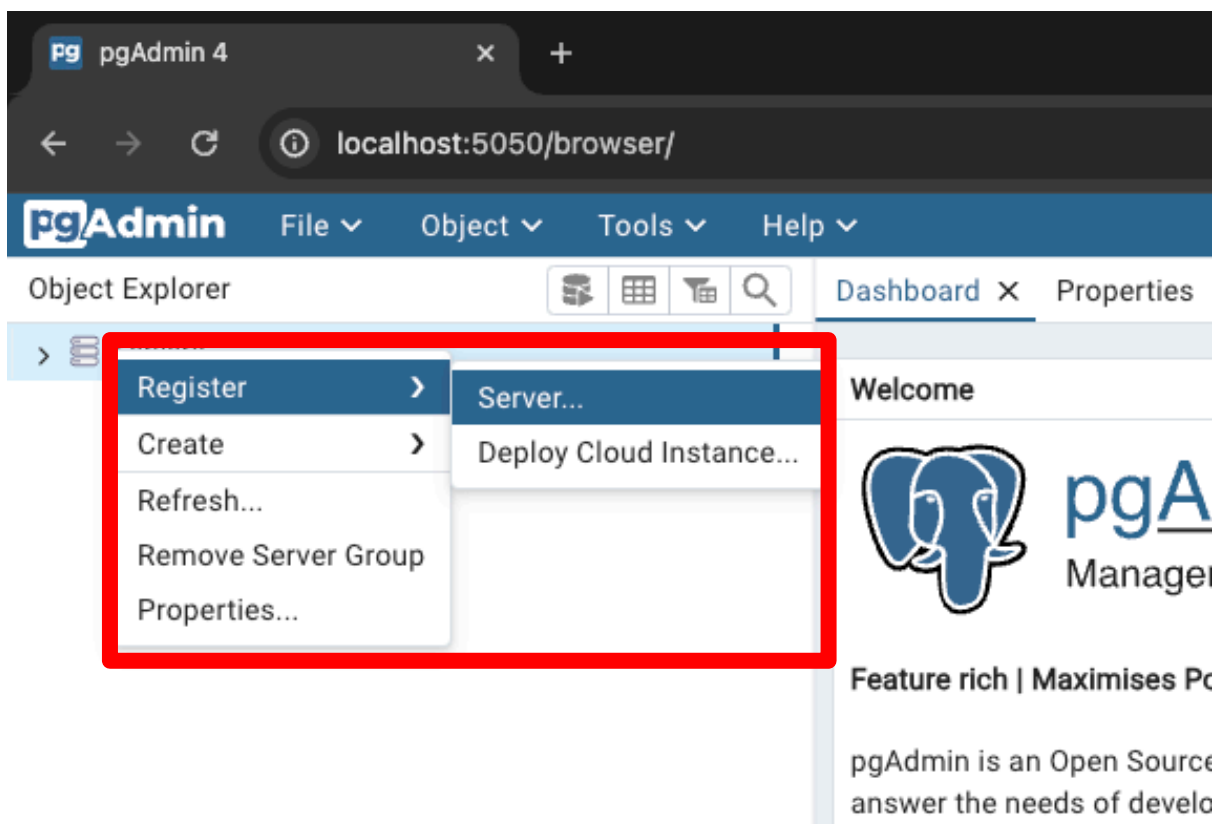


PG-admin

login



Connect to Database



กรอกข้อมูลในการเข้าสู่ Database

Register - Server

General

Connection

Parameters

SSH Tunnel

Advanced

Host name/address

go_db

Port

5432

Maintenance database

postgres

Username

postgres

Kerberos authentication?

☐

Password

.....

Save password?

☐

Role

Service

Close

Reset

Save

pgAdmin 4

localhost:5050/browser/

pgAdmin

File Object Tools Help

Object Explorer

Dashboard Properties SQL Statistics Dependencies Dependents Processes public.humans/postgres/postgres@postgres

postgres (1)

- Databases (1)
 - postgres
 - Cast
 - Catalogs
 - Event Triggers
 - Extensions
 - Foreign Data Wrappers
 - Languages
 - Publications
 - Schemas (1)
 - public
 - Aggregates
 - Collations
 - Domains
 - FTS Configurations
 - FTS Dictionaries
 - FTS Parsers
 - FTS Templates
 - Foreign Tables
 - Functions
 - Materialized Views
 - Operators
 - Procedures
 - Sequences
 - Tables (1)
 - humans
 - Columns
 - Constraints
 - Indexes
 - RLS Policies
 - Rules
 - Triggers
 - Trigger Functions
 - Types
 - Views
 - Subscriptions

Query

Query History

Scratch Pad

1 SELECT * FROM public.humans

2 ORDER BY id ASC

Data Output Messages Notifications

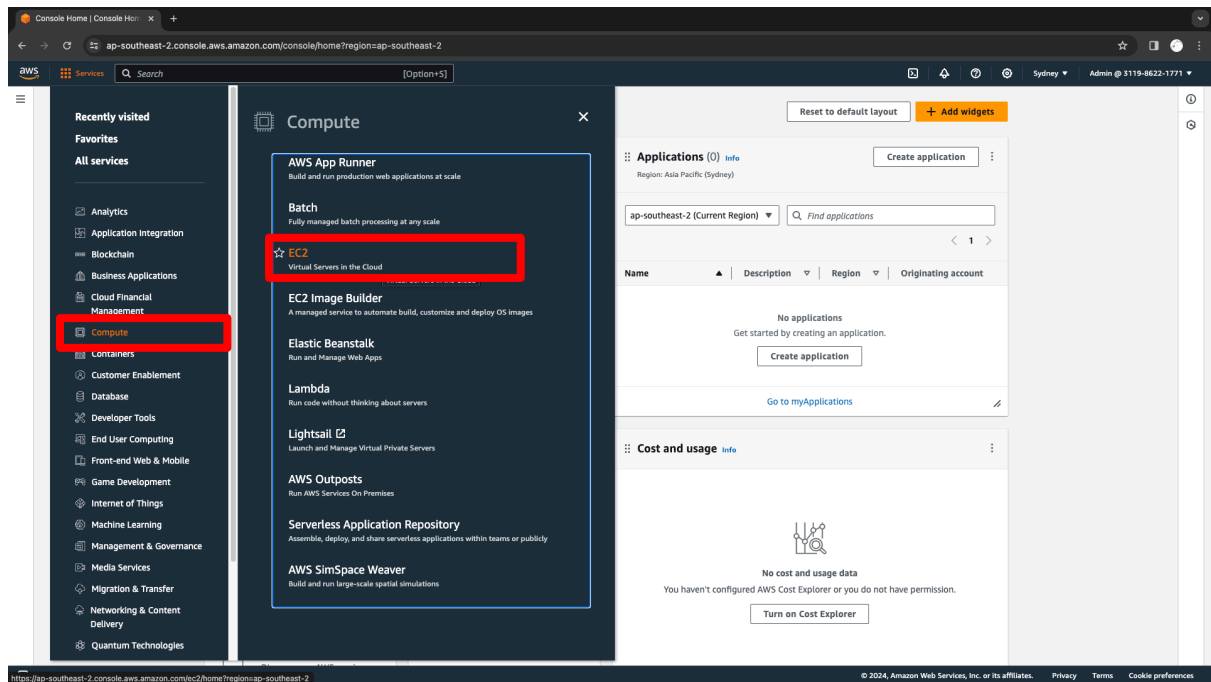
	id	lname	lname
	[PK] integer	text	text
1	1	Thanabodin	Keawmaha
2	2	Pon	Morin

Total rows: 2 of 2 Query complete 00:00:00.113

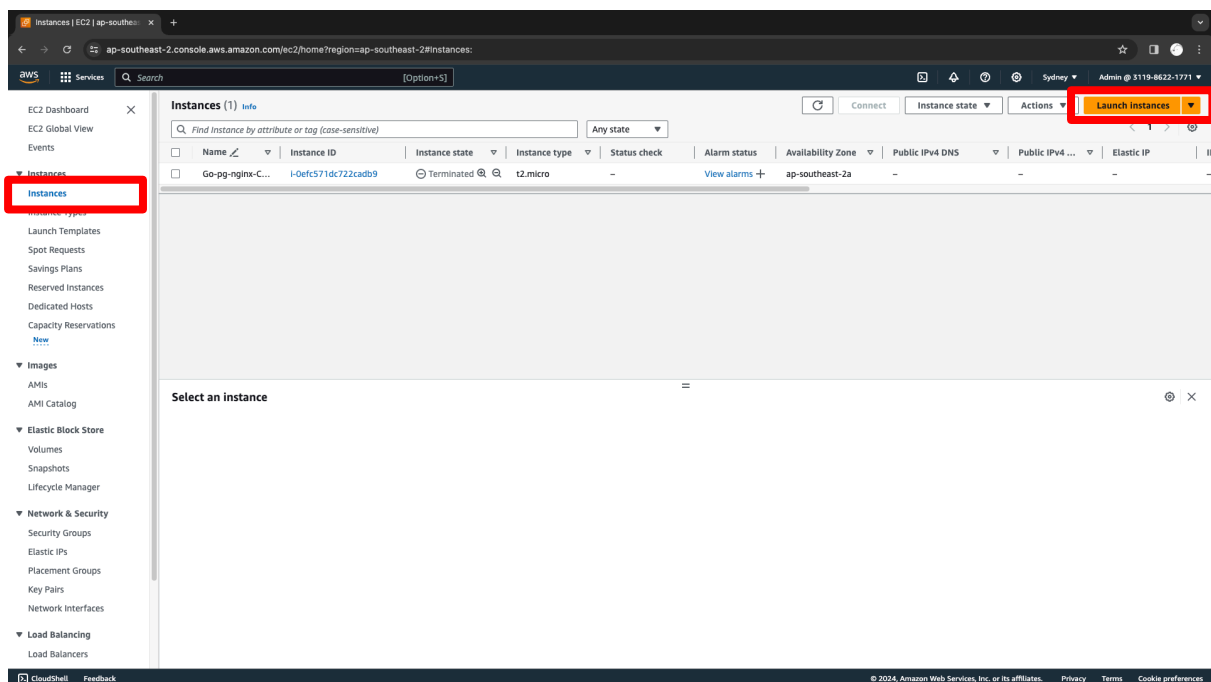
Successfully run. Total

ใช้ AWS ในการเป็น server run docker

Login AWS แล้วหา EC2



จากนั้น เลือก instances แล้วเลือก Launch instances



ตั้งชื่อ Instance

Name and tags [Info](#)

Name

CRDU api Docker

Add additional tags

เลือก OS ในการสร้าง virtual machines เลือก ubuntu version 22.04 Free tier eligible

Quick Start

Amazon Linux

aws

macOS

Mac

Ubuntu

ubuntu

Windows

Microsoft

Red Hat

Red Hat

SUSE Linux

SUS

Browse more AMIs

Including AMIs from AWS, Marketplace and the Community

Amazon Machine Image (AMI)

Ubuntu Server 22.04 LTS (HVM), SSD Volume Type

Free tier eligible

ami-04f5097681773b989 (64-bit (x86)) / ami-0b71cd1a5da0c93ec (64-bit (Arm))

Virtualization: hvm ENA enabled: true Root device type: ebs

Description

Canonical, Ubuntu, 22.04 LTS, amd64 jammy image build on 2023-12-07

Architecture

AMI ID

64-bit (x86)

ami-04f5097681773b989

Verified provider

เลือก Instance type เป็น t2.micro เพราะเป็น Free tier eligible

▼ Instance type [Info](#) | [Get advice](#)

Instance type

t2.micro

Free tier eligible

Family: t2 1 vCPU 1 GiB Memory Current generation: true

On-Demand SUSE base pricing: 0.0146 USD per Hour

On-Demand Linux base pricing: 0.0146 USD per Hour

On-Demand Windows base pricing: 0.0192 USD per Hour

On-Demand RHEL base pricing: 0.0746 USD per Hour

☒ All generations

[Compare instance types](#)

[Additional costs apply for AMIs with pre-installed software](#)


ตั้งชื่อไฟล์ Key เพื่อจะได้ SSH เข้าไปใช้งาน

▼ Key pair (login) [Info](#)

You can use a key pair to securely connect to your instance. Ensure that you have access to the selected key pair before you launch the instance.

Key pair name - *required*

▼

 [Create new key pair](#)

ตั้งค่า Network

▼ Network settings [Info](#) Edit

Network [Info](#)
vpc-0cc3d96c9207d80e7

Subnet [Info](#)
No preference (Default subnet in any availability zone)

Auto-assign public IP [Info](#)
Enable

Firewall (security groups) [Info](#)
A security group is a set of firewall rules that control the traffic for your instance. Add rules to allow specific traffic to reach your instance.

☒ Create security group ☐ Select existing security group

We'll create a new security group called 'launch-wizard-2' with the following rules:

☒ Allow SSH traffic from
Helps you connect to your instance

Anywhere
0.0.0.0/0 ▼

☒ Allow HTTPS traffic from the internet
To set up an endpoint, for example when creating a web server

☒ Allow HTTP traffic from the internet
To set up an endpoint, for example when creating a web server

กดสร้าง Instance

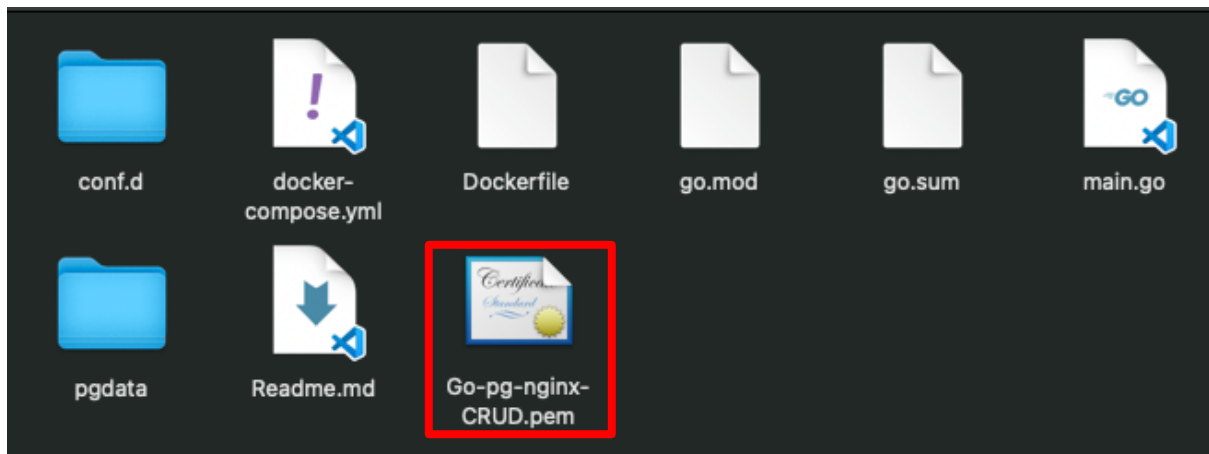
Cancel **Launch instance**
[Review commands](#)

รอน Instance state จะเป็น Running และสังเกตตรง Public IPV4 address

Instance summary for i-00e29e76903805db3 (CRDU api Docker) [Info](#)
Updated less than a minute ago

Instance ID i-00e29e76903805db3 (CRDU api Docker)	Public IPv4 address 54.252.139.219 open address
IPv6 address -	Instance state Running

เราจะได้ key มาเก็บไว้อย่างดี



เปิด terminal ขึ้นมาละเข้าไปยังโพลเดอร์ที่มี key อยู่ และใช้คำสั่ง

\$ ssh -i <ชื่อไฟล์ Key> ubuntu@< Public IPV4 address > และ พิมพ์ yes

```
thanabodinkeawmaha@Mac-Thanabodin go-crud-docker % ssh -i Go-pg-nginx-CRUD.pem ubuntu@54.252.139.219
The authenticity of host '54.252.139.219 (54.252.139.219)' can't be established.
ED25519 key fingerprint is SHA256:URhGLYjDWlob/hdSUEogcS1ATDj8A/xd0+3ibBKE2j4.
This key is not known by any other names.
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
```

ก็เข้ามาใน ubuntu ใน virtual machines ของ AWS

```
System information as of Fri Feb  2 19:36:25 UTC 2024

System load:  0.0                Processes:            95
Usage of /:   20.6% of 7.57GB    Users logged in:     0
Memory usage: 21%               IPv4 address for eth0: 172.31.1.239
Swap usage:   0%

ubuntu@ip-172-31-1-239:~$
```

ลง Docker

Add Docker's official GPG key:

\$ sudo apt-get update

\$ sudo apt-get install ca-certificates curl

\$ sudo install -m 0755 -d /etc/apt/keyrings

\$ sudo curl -fsSL https://download.docker.com/linux/ubuntu/gpg -o
/etc/apt/keyrings/docker.asc

\$ sudo chmod a+r /etc/apt/keyrings/docker.asc

```
ubuntu@ip-172-31-1-239:~$ sudo apt-get update
sudo apt-get install ca-certificates curl
sudo install -m 0755 -d /etc/apt/keyrings
sudo curl -fsSL https://download.docker.com/linux/ubuntu/gpg -o /etc/apt/keyrings/docker.asc
sudo chmod a+r /etc/apt/keyrings/docker.asc
```

Add the repository to Apt sources:

\$ echo \

"deb [arch=\$(dpkg --print-architecture) signed-by=/etc/apt/keyrings/docker.asc]

https://download.docker.com/linux/ubuntu \

\$(. /etc/os-release && echo "\$VERSION_CODENAME") stable" | \

sudo tee /etc/apt/sources.list.d/docker.list > /dev/null

\$ sudo apt-get update

```
ubuntu@ip-172-31-1-239:~$ echo \
"deb [arch=$(dpkg --print-architecture) signed-by=/etc/apt/keyrings/docker.asc] https://download.docker.com/linux/ubuntu \
$(. /etc/os-release && echo "$VERSION_CODENAME") stable" | \
sudo tee /etc/apt/sources.list.d/docker.list > /dev/null
sudo apt-get update
```

Install the Docker packages.

\$ sudo apt-get install docker-ce docker-ce-cli containerd.io docker-buildx-plugin docker-
compose-plugin

```
ubuntu@ip-172-31-1-239:~$ sudo apt-get install docker-ce docker-ce-cli containerd.io docker-buildx-plugin docker-compose-plugin
```

หลังจากนั้นก็เอาโค้ดที่เราเขียนขึ้น GitHub จากนั้นก็ clone ลงมายัง ubuntu ของ AWS

```
ubuntu@ip-172-31-1-239:~$ git clone https://github.com/Thanabodin19/go-crud-Docker.git
Cloning into 'go-crud-Docker'...
remote: Enumerating objects: 36, done.
remote: Counting objects: 100% (36/36), done.
remote: Compressing objects: 100% (26/26), done.
remote: Total 36 (delta 16), reused 28 (delta 8), pack-reused 0
Receiving objects: 100% (36/36), 5.54 KiB | 944.00 KiB/s, done.
Resolving deltas: 100% (16/16), done.
ubuntu@ip-172-31-1-239:~$ ls
go-crud-Docker
```

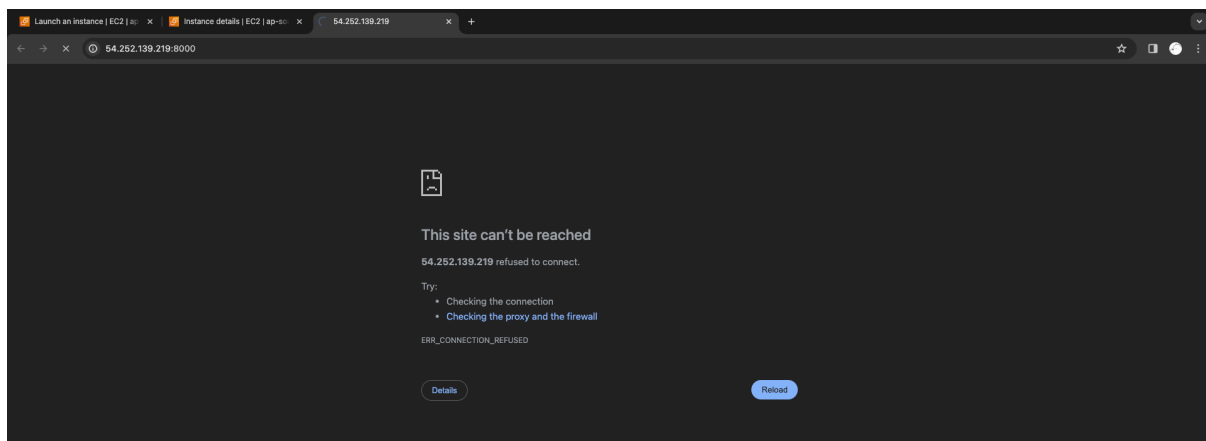
เข้าไปยังโฟลเดอร์ทำการ build Dockerfile ด้วยคำสั่ง `$ sudo docker compose build`

```
ubuntu@ip-172-31-1-239:~/go-crud-Docker$ sudo docker compose build
[+] Building 17.9s (10/10) FINISHED
=> [go-app internal] load build definition from Dockerfile
=> => transferring dockerfile: 338B
=> [go-app internal] load metadata for docker.io/library/golang:1.17-alpine3.13
=> [go-app internal] load .dockerignore
=> => transferring context: 2B
=> [go-app 1/5] FROM docker.io/library/golang:1.17-alpine3.13@sha256:33c95bc78fa434bae90c45d04e5a02612492a0b1176369fb09f0d82eb1e55a43b
=> => resolve docker.io/library/golang:1.17-alpine3.13@sha256:33c95bc78fa434bae90c45d04e5a02612492a0b1176369fb09f0d82eb1e55a43b
=> => sha256:33c95bc78fa434bae90c45d04e5a02612492a0b1176369fb09f0d82eb1e55a43b 1.65kB / 1.65kB
=> => sha256:5cae70ebb5ca78778f682a994e4df09bdc8c382c0f647058665cb6186dbe3351 1.36kB / 1.36kB
=> => sha256:a9baab48e7f24a25ea1591bc125de93e87ecb13c3e5ab25ca2cc0792da6f70fd 5.37kB / 5.37kB
=> => sha256:5758d4e389a3f662e94a85fb76143dbe338b64f8d2a65f45536a9663b05305ad 2.82MB / 2.82MB
=> => sha256:04b7a40ca5d5102ba31433c4330802d08e749d55f483003b5ed1f75e257aa9ec 281.80kB / 281.80kB
=> => sha256:452a8c64b8e1aafdc8bccac5c7c4cef87cea0b1176369fb09f0d82eb1e55a43b 154B / 154B
=> => extracting sha256:5758d4e389a3f662e94a85fb76143dbe338b64f8d2a65f45536a9663b05305ad
=> => sha256:5d2dd47ef1ba23bf9cfbb38067b00bb9322a284ef7dd0a0f4fc25dab6324a9fe 110.03MB / 110.03MB
=> => sha256:4c343abe76065bf3c2d8082f9815b6e3707e9565c1f7f92cfeb81f74bed02db4f 156B / 156B
=> => extracting sha256:04b7a40ca5d5102ba31433c4330802d08e749d55f483003b5ed1f75e257aa9ec
=> => extracting sha256:452a8c64b8e1aafdc8bccac5c7c4cef87cea0b1176369fb09f0d82eb1e55a43b
=> => extracting sha256:5d2dd47ef1ba23bf9cfbb38067b00bb9322a284ef7dd0a0f4fc25dab6324a9fe
=> => extracting sha256:4c343abe76065bf3c2d8082f9815b6e3707e9565c1f7f92cfeb81f74bed02db4f
=> [go-app internal] load build context
=> => transferring context: 47.97kB
=> [go-app 2/5] WORKDIR /app
=> [go-app 3/5] COPY . .
=> [go-app 4/5] RUN go get -d -v ./...
=> [go-app 5/5] RUN go build -o api .
=> [go-app] exporting to image
=> => exporting layers
=> => writing image sha256:4b04eed95177ca53c822fcf78e0deaad280a4963c67bd7ac4cae06be1af4df14
=> => naming to docker.io/francescoxx/go-app:1.0.1
```

จากนั้นก็ run docker compose เพื่อสร้าง service ทั้งหมด

```
ubuntu@ip-172-31-1-239:~/go-crud-Docker$ sudo docker compose up --scale go-app=3 -d
.: Network go-crud-docker_default Created
.: Volume "go-crud-docker_pgdata" Created
✓ Container pg-admin Started
✓ Container go_db Started
✓ Container go-crud-docker-go-app-3 Started
✓ Container go-crud-docker-go-app-1 Started
✓ Container go-crud-docker-go-app-2 Started
✓ Container nginx Started
```

จากนั้นใช้ Public IPV4 address ตามด้วย Port ที่จะเข้าถึง จะเห็นว่ายังเข้าไม่ได้



ต้องเข้าไปเพิ่ม Port ที่ AWS Instance เลือก Security และเลือก Security Groups

The screenshot shows the AWS Management Console interface with the 'Security' tab selected. The 'Security groups' section is highlighted with a red box, showing a single group: 'sg-0ff25bce24b9e3b15 (launch-wizard-2)'. Below it, the 'Inbound rules' section is visible, showing a table of rules.

Name	Security group rule ID	Port range	Protocol
-	sgr-09c6badae77542588	22	TCP
-	sgr-0f404f570c0b793c0	443	TCP
-	sgr-0f80ff248beed283e	80	TCP

เลือก Edit inbound rules

The screenshot shows the 'Edit inbound rules' button highlighted with a red box. Other buttons like 'Manage tags' and 'Refresh' are also visible.

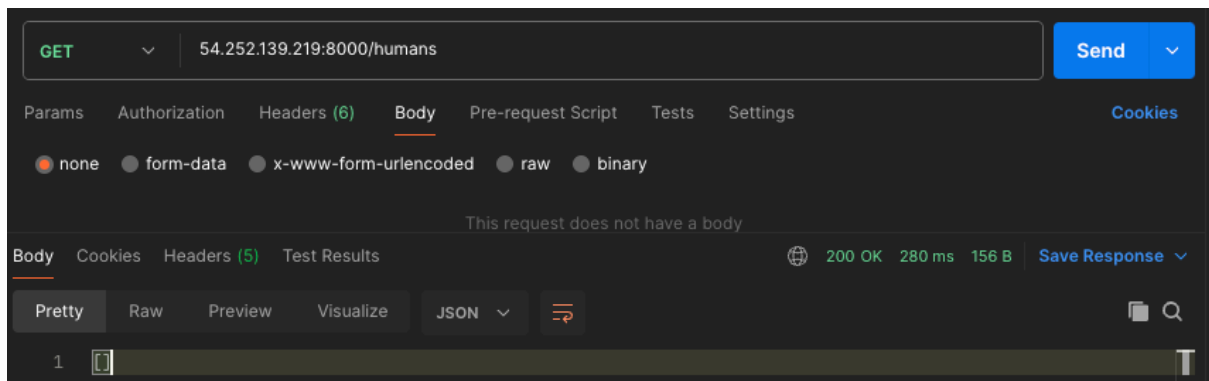
เลือก Add rule

The screenshot shows the 'Add rule' button highlighted with a red box. The rule name 'sgr-0f80ff248beed283e' and the protocol 'HTTP' are visible.

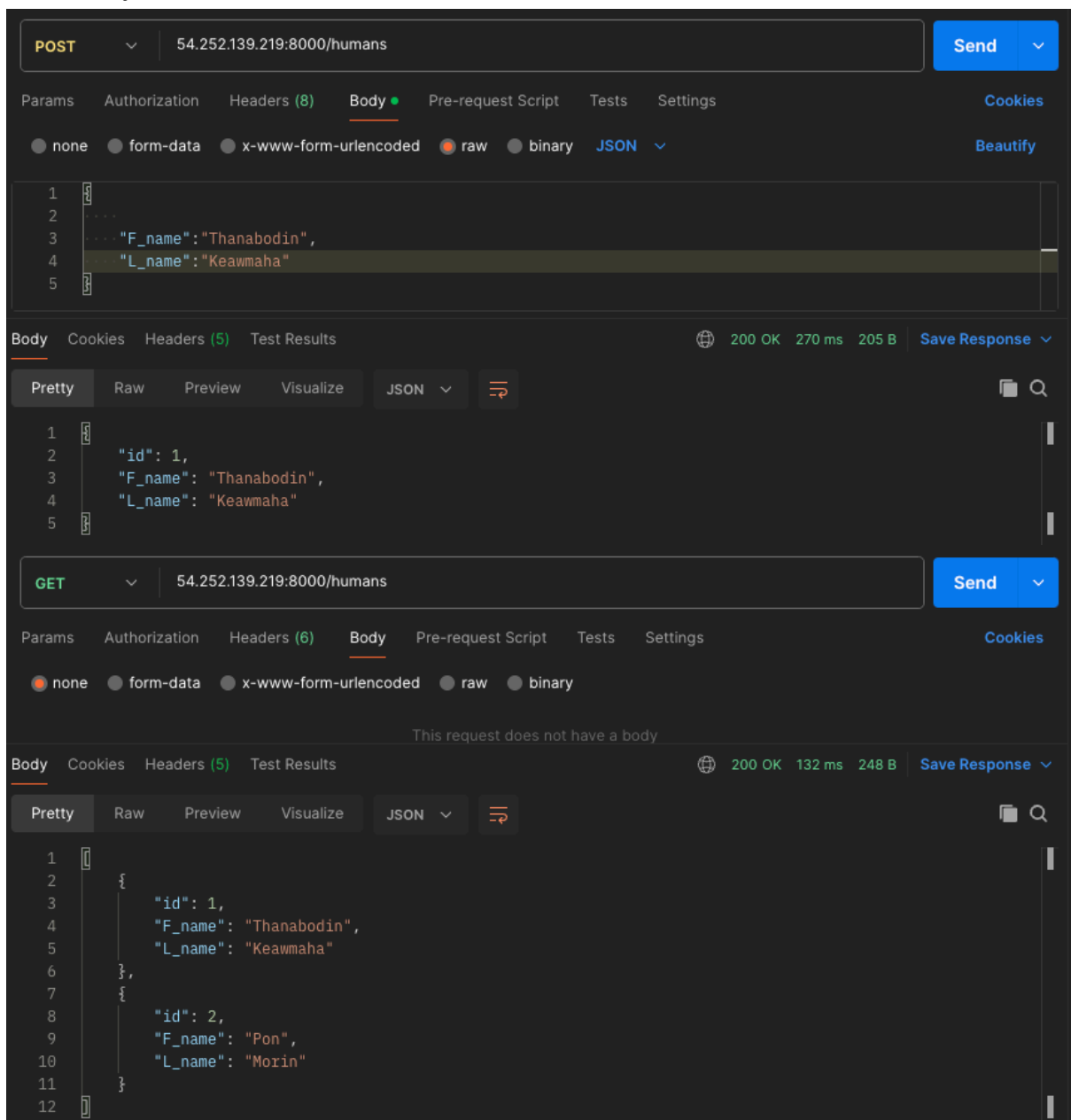
เลือก Custom TCP Port 8000

The screenshot shows the 'Add rule' configuration form. The 'Custom TCP' option is selected, and the port '8000' is entered. The 'Anywhere...' option is selected for the source. The '0.0.0.0/0' option is selected for the destination.

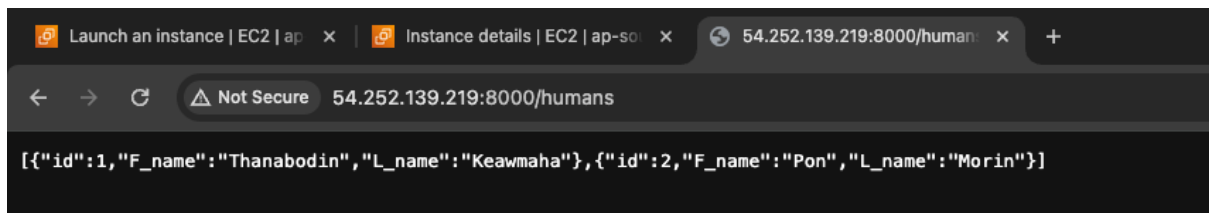
ลองทดสอบ api จะเห็นได้ว่าสามารถใช้งานได้



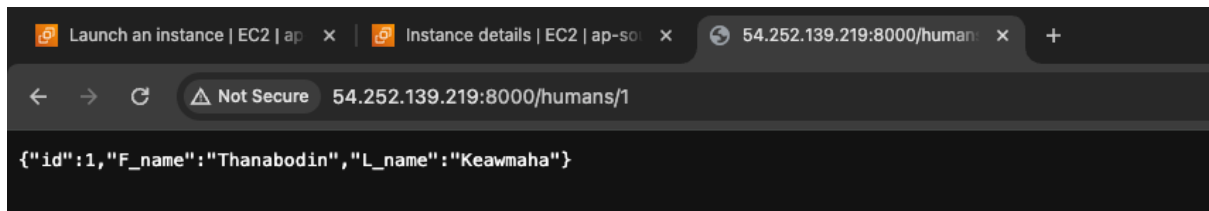
ลองเพิ่มข้อมูล



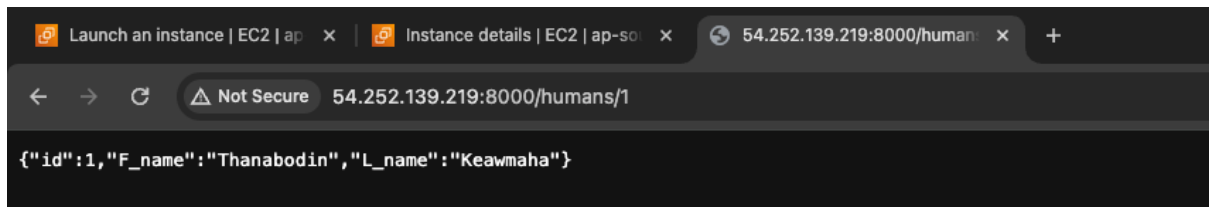
ลองใช้ Public IPV4 address ตามด้วย Port 8000/humans



ลองใช้ Public IPV4 address ตามด้วย Port 8000/humans/1



ลองใช้ Public IPV4 address ตามด้วย Port 8000/humans/2



ลองใช้ Public IPV4 address ตามด้วย Port 5050

