



Kong API Gateway

Document Create Date : 7 June 2024

Version : 1.0

Status : Initial Version

Document Author : Tripop Raksa

Document Reviewer : Kittimasak Wangsri

## คำนำ

เอกสารฉบับนี้มีวัตถุประสงค์เพื่อให้ความรู้และข้อมูลเกี่ยวกับการใช้งาน Kong API Gateway ซึ่งเป็นเครื่องมือสำคัญในการบริหารจัดการ API และเพิ่มประสิทธิภาพในการเชื่อมต่อบริการต่างๆ Kong API Gateway เป็นแพลตฟอร์มที่มีความยืดหยุ่นสูง และสามารถปรับแต่งได้ตามความต้องการขององค์กร เพื่อให้มั่นใจในการทำงานร่วมกันระหว่างบริการต่างๆ ได้อย่างมีประสิทธิภาพ

ในเอกสารฉบับนี้จะครอบคลุมถึงการติดตั้งและการตั้งค่าเบื้องต้นของ Kong API Gateway การสร้างและบริหารจัดการ API การกำหนดนโยบายการรักษาความปลอดภัย และการควบคุมการเข้าถึง รวมถึงการควบคุมปริมาณการใช้งาน API และการตรวจสอบและบันทึกข้อมูลเพื่อดูแลและตรวจสอบเหตุการณ์ที่สำคัญต่างๆ ในระบบ

เราเชื่อว่าเอกสารฉบับนี้จะเป็นประโยชน์สำหรับนักพัฒนาระบบ ผู้ดูแลระบบเครือข่าย และผู้ที่สนใจในเทคโนโลยีการจัดการ API และหวังว่าจะสามารถช่วยให้ท่านมีความรู้ที่ได้รับไปประยุกต์ใช้ในการทำงานได้อย่างมีประสิทธิภาพ

นายไตรภพ รักษา

ผู้จัดทำรายงาน

วันที่ 07 เดือน มิถุนายน พ.ศ. 2567

## สารบัญ

Kong API Gateway .....	5
API Gateway คืออะไร? .....	5
Kong API Gateway คือ ? .....	5
ส่วนประกอบของ Kong API Gateway.....	5
Kong จะประกอบไปด้วย.....	6
ทำไมถึงต้องใช้ Kong API Gateway .....	8
ตัวอย่าง Use Case .....	8
Demo Kong API Gateway .....	10
ติดตั้ง Kong .....	10
DEMO - 1 .....	15
กำหนด Services.....	15
สร้าง Route .....	17
กำหนด Consumers.....	24
API Keys.....	25
JWT .....	26
ทดสอบการใช้งาน .....	29
แบบ Public .....	29
แบบ Authentication .....	30
Plugin Rate Limiting .....	33
Plugin File Log ใน Logging.....	35
Plugins Basic Auth .....	38
DEMO – 2 .....	42
Code .....	42
กำหนด Service.....	49
สร้าง Route .....	51

เพิ่ม Plugin.....	54
Prometheus + Grafana .....	57
Prometheus .....	60
Grafana .....	64

## Kong API Gateway

### API Gateway คืออะไร?

API Gateway เปรียบเสมือนประตูหน้าบ้านที่เอาไว้เปิดรับ request ต่างๆ จาก Frontend ก่อนที่จะทำการส่งต่อมายัง Backend เพื่อเรียกใช้งาน service ต่างๆ

- ช่วยกรอง Traffic หรือ Request ที่เข้ามามากจากภายนอก อีกทั้งยังช่วยจำกัด Traffic Limit ได้อีกด้วย
- เป็น API Management ภายใต้ตัวที่ช่วยจัดการกับ API ต่างๆ ที่จะถูกส่งต่อไปข้างในอีกทีหนึ่ง
- เปิดเผยตัว Endpoint ของ API โดยจะเปิดเผยเพียงแค่ Endpoint เดียวเท่านั้น (Single Entry Point)
- ช่วยในเรื่องความปลอดภัย (Authentication, Logging และอื่นๆ)

### Kong API Gateway คือ ?

Kong API Gateway คือ Open-source Software โดยเป็นเครื่องมือที่ทำหน้าที่เป็น API Gateway ที่เป็นตัวกลางในการจัดการ Microservices ต่างๆ มาไว้ที่เดียว เพื่อควบคุมและจัดการให้ง่ายขึ้น ไม่ว่าจะเป็นเรื่องของความปลอดภัย สิทธิ์การเข้าถึง การ Monitoring การกำหนด Rate Limiting เป็นต้น

Kong API Gateway จะมี 2 รุ่นดังนี้

- Kong EE (Enterprise Edition) รุ่นเสียเงิน (มีให้ทดลองใช้งาน) มีหน้า Admin (GUI) ให้ใช้งาน
- Kong CE (Community Edition) รุ่นฟรี ไม่มีหน้า Admin (GUI) ให้ใช้งาน

### ส่วนประกอบของ Kong API Gateway

1. Kong Gateway เป็น core หลักที่ทำหน้าที่เป็น API Gateway ซึ่งจะคอย routing service (ส่งต่อ request ต่อไปยัง upstream service) และนอกจากรส่ง request ต่อไปยังปลายทางแล้ว plugins ต่างๆ ก็จะถูกเรียกใช้งานตอนที่มี request วิ่งเข้ามายัง proxy นี้ การใช้งาน Gateway จะเข้าผ่านทาง port
  - 8000 เป็นการเข้าถึงผ่านทาง HTTP Protocol
  - 8443 เป็นการเข้าถึงผ่านทาง HTTPS Protocol
2. Kong Database การใช้งาน Kong นั้นจะต้องระบุวิธีจัดการเก็บ configuration ซึ่ง Kong นั้น Support อยู่ 2 รูปแบบด้วยกันคือ
  - แบบมี Database หรือ Imperative รูปแบบนี้จะจัดเก็บ configuration ลงใน database ซึ่ง Kong จะ Support แค่ PostgreSQL เท่านั้น
  - แบบ DBless หรือ Declarative รูปแบบนี้จะใช้ config file (นามสกุล .yml) แทนการใช้ database ซึ่งการเปลี่ยนแปลงหรือ update เราจะใช้ decK เป็นตัวช่วย

Database (Imperative)	DBless (Declarative)
1. เหมาะกับการใช้คนเข้าไปจัดการ	1. เหมาะกับการทำ Automation
2. ง่าย เหมาะสำหรับผู้เริ่มต้น	2. ต้องคอย sync ให้ kong.yml ตรงกับ config ใน database
3. การแก้ไขทำได้ยากกว่าเมื่อเวลาผ่านไป	3. อ่านไฟล์ kong.yml จะรู้ว่ามีการ routes หรือติดตั้ง plugins ลงมาที่ไหนบ้าง

3. Kong Management ส่วนนี้เป็นเครื่องมือที่ใช้จัดการ Kong ซึ่งมีอยู่หลายทางเลือกด้วยกัน

- Kong API เมื่อเราติดตั้ง Kong เราจะได้เฉพาะ Proxy Service ที่ทำหน้าที่เป็น Gateway เท่านั้น เราจะไม่มีหน้า UI มาช่วยในการจัดการ เราต้องคุยกับ API โดยตรง ซึ่งจะเปิดให้บริการอยู่ที่ port
  - 8001 สำหรับเข้าถึง API ด้วย HTTP Protocol
  - 8444 สำหรับเข้าถึง API ด้วย HTTPS Protocol
- Kong Manager ถ้าคุณใช้งาน Kong Enterprise คุณจะได้รับ Kong Manager มาเป็น UI เราจะจัดการ Kong ได้ง่ายขึ้น
- Konga ในการนี้ใช้ Community Edition ต้องใช้ Konga ที่เป็น Open-Source มาเป็น UI
- Deck จะเป็น Command Line ที่ช่วยจัดการ Kong ในแบบ Declarative

## Kong จะประกอบไปด้วย

การเริ่มใช้งาน Kong ต้องเข้าใจ 5 ส่วนหลักของ Kong ก่อน

1. Service : เป็นการระบุที่อยู่ของ upstream services (service ที่ให้บริการจริงๆ)
2. Route : เป็นการระบุเส้นทางเพื่อเข้าถึง upstream services ที่เราได้ระบุไว้ในขั้นตอนก่อนหน้า ซึ่งการกำหนด route สามารถใช้ได้ทั้ง
  - Host สามารถใช้ชื่อ host ในกรณีที่ต้องการตัวนำทางได้ เช่น api.example.com
  - Path นอกจากชื่อ host แล้วยังสามารถใช้ path เป็นตัวนำทางได้ เช่น api.example.com/users
  - Method ในบางกรณีจะใช้ method ในกรณีที่ต้องการตัวนำทาง เช่น CQRS patterns เราจะแยก service ออกเป็น command service สำหรับการ Insert, Update, Delete และ Query service สำหรับการอ่านหรือการ query (Read-only) ซึ่งตอน implement เราจะใช้การแยก route ด้วย method โดยใช้ GET method วิ่งเข้าไปที่ Query service อื่นๆ (POST, PUT, PATCH, DELETE) จะวิ่งเข้า command service
  - Headers สามารถกำหนด route ด้วยชื่อนุ่ลที่อยู่ใน headers ได้ เช่น x-api-key: 2 จะเป็นการ route ไปยัง service key 2 แต่ถ้าไม่มี header อาจจะวิ่งไปยัง key 1

3. Plugin : เป็นอีกหนึ่งสิ่งที่ทำให้ Kong ได้รับความนิยมมากๆ เพราะ plugins ของ Kong นั้นมีเยอะมากๆ ซึ่งจะแบ่งออกเป็น 7 กลุ่มหลักๆดังนี้
  1. Authentication : กลุ่มนี้ทำหน้าที่ตรวจสอบสิทธิ์การเข้าใช้งานระบบ ซึ่ง support ทั้ง JWT, OAuth2 และ LDAP
  2. Security : กลุ่มนี้จะดูแลเรื่องความปลอดภัยของระบบ เช่น Bot detection, IP restriction หรือ การ Enable CORS
  3. Traffic Control : กลุ่มนี้จะระบุวิธีส่ง Request ต่อไปยัง upstream service เช่น ทำ Canary Release, Rate Limiting หรือ Request Size Limiting
  4. Serverless : เป็นกลุ่มที่ทำหน้าที่เป็นตัวเชื่อมต่อไปยัง serverless บน Cloud Service Provider เช่น AWS Lambda, Azure Function หรือ Apache OpenWhisk
  5. Transformation : เป็น plugins ที่ทำหน้าที่แปลง request และ response ให้อยู่ในรูปแบบที่ต้องการ เช่น CorrelationID สำหรับ Microservices หรือ gRPC Gateway สำหรับแปลง Request ในรูปแบบของ REST ไปเป็น gRPC
  6. Analytics and Monitoring : กลุ่มนี้ถือว่ามีความสำคัญกับการจัดการ API มากใน Operation Phase เพราะเราต้องนำ Log และ Metric ส่งไปยัง monitoring server ซึ่ง Kong support ทั้ง DataDog, Prometheus และ Open Telemetry
  7. Logging : กลุ่มนี้จะทำการส่ง Log ขึ้นไปยัง Logstash server ซึ่ง Kong support ทั้ง Filelog, HTTP Log และ Kafka Log รวมทั้งสามารถเชื่อมต่อกับ cloud service ยอดนิยมอย่าง Loggly ได้อีกด้วย
4. Consumers จะเก็บสิ่งที่สามารถ reference ไปยัง user ได้ซึ่งมีอยู่ 2 ทางเลือก คือ
  - Username เก็บ username ลงในระบบโดยตรง
  - CustomID เลือกเก็บ UUID ที่ซึ่งไปยัง user แทน และสามารถใช้ Service ID แทน user ที่เป็น Machine ได้อีกด้วย
5. Upstream and Targets ในการสร้าง route และ service ใน Gateway จะเป็นการระบุ service ปลายทางแค่ตัวเดียว แต่ถ้าต้องการจะระบุ upstream services มากกว่า 1 ตัว (Load Balancer) เราจะต้องสร้างเป็น upstream และ targets โดยที่เราต้องกำหนด
  - Targets เป็นการระบุที่อยู่ของ upstreaming service แต่ละตัว โดยที่ต้องกำหนด URL และ weight (ยิ่งกำหนด weight มาก Load Balancer ก็จะส่ง request เข้ามาที่ service ตัวนี้มาก)
  - Upstream จะเป็นการรวม targets ให้อยู่ภายใต้ service เดียวกัน และเราจะระบุ algorithm ในการกระจาย load เข้าไปยัง target แต่ละตัว

## ทำไมถึงต้องใช้ Kong API Gateway

1. การจัดการ API ที่มีประสิทธิภาพ (Efficient API Management)
  - Routing
  - Load Balancing
2. Security
  - Authentication and Authorization (OAuth2, JWT, Basic Auth and ACL)
  - Rate Limiting and Throttling
3. Monitoring and Logging
  - มี plugin สำหรับตรวจสอบและบันทึกข้อมูล เช่น Prometheus, ELK Stack ทำให้สามารถติดตามประสิทธิภาพ และการทำงานของ API ได้อย่างละเอียด
4. Multi-Protocol Support
  - รองรับการทำงานกับหลายโปรโตคอล เช่น HTTP/HTTPS, gRPC, GraphQL

## ตัวอย่าง Use Case

### Use case บริษัท E – Commerce

บริษัท A เป็นบริษัทอีคอมเมิร์ซที่มีแพลตฟอร์มออนไลน์สำหรับการขายสินค้า และบริการต่างๆ บริษัทต้องการเพิ่มประสิทธิภาพ และความปลอดภัยในการจัดการ API หลายตัวที่ให้บริการลูกค้า รวมถึงการบริหารจัดการการเข้าถึง และการบันทึกข้อมูลการใช้งาน

### การใช้งาน Kong API Gateway

1. การจัดการ API หลายตัว (API Management)
  - Description : บริษัท A มี API หลายตัวที่ใช้ในการให้บริการ เช่น Product API, Order API, Payment API เป็นต้น
  - Implementation : ใช้ Kong API Gateway เพื่อจัดการ API เหล่านี้ในที่เดียว รวมถึงการกำหนด routing สำหรับ API แต่ละตัว ทำให้การบริหารจัดการง่ายขึ้น และมีประสิทธิภาพมากขึ้น
2. การตรวจสอบ และการควบคุมการเข้าถึง (Authentication and Access Control)
  - Description : บริษัทต้องการให้การเข้าถึง API ปลอดภัย และให้เฉพาะผู้ใช้ที่มีสิทธิ์เท่านั้นที่สามารถเข้าถึงได้
  - Implementation : ใช้ Plugin JWT (JSON Web Token) สำหรับการยืนยันตัวตน และการควบคุมการเข้าถึง ผู้ใช้ที่ทำการล็อกอินจะได้รับ JWT ซึ่งจะถูกตรวจสอบทุกครั้งที่มีการเรียก API

### 3. การป้องกันการโจมตีแบบ DDoS (DDoS Protection)

- Description : บริษัทต้องการป้องกันการโจมตีแบบ DDoS ที่อาจทำให้บริการหยุดชะงัก
- Implementation : ใช้ Plugin Rate Limiting เพื่อจำกัดอัตราการเรียก API จากผู้ใช้แต่ละคน และป้องกันการเรียก API ที่มากเกินไปในช่วงเวลาสั้นๆ

### 4. การบันทึก และการตรวจสอบการใช้งาน API (Logging and Monitoring)

- Description : บริษัทต้องการบันทึก และตรวจสอบการใช้งาน API เพื่อการวิเคราะห์ และปรับปรุงประสิทธิภาพ
- Implementation : ใช้ Plugin Logging เพื่อบันทึกข้อมูลการเรียก API ทั้งหมดไปยังระบบบันทึกข้อมูล เช่น ELK Stack หรือ Splunk เพื่อการวิเคราะห์ และการตรวจสอบย้อนหลัง

### 5. การจัดการการกระจายโหลด (Load Balancing)

- Description : บริษัทต้องการกระจายโหลดการใช้งานไปยังเซิร์ฟเวอร์หลาย台เพื่อป้องกันการทำงานหนักของเซิร์ฟเวอร์เดียว
- Implementation : ใช้ Kong Upstream และ Targets เพื่อกำหนด และจัดการกลุ่มเซิร์ฟเวอร์ที่ให้บริการ API โดย Kong จะกระจายโหลดการใช้งานไปยังเซิร์ฟเวอร์ต่างๆ อย่างสมดุล

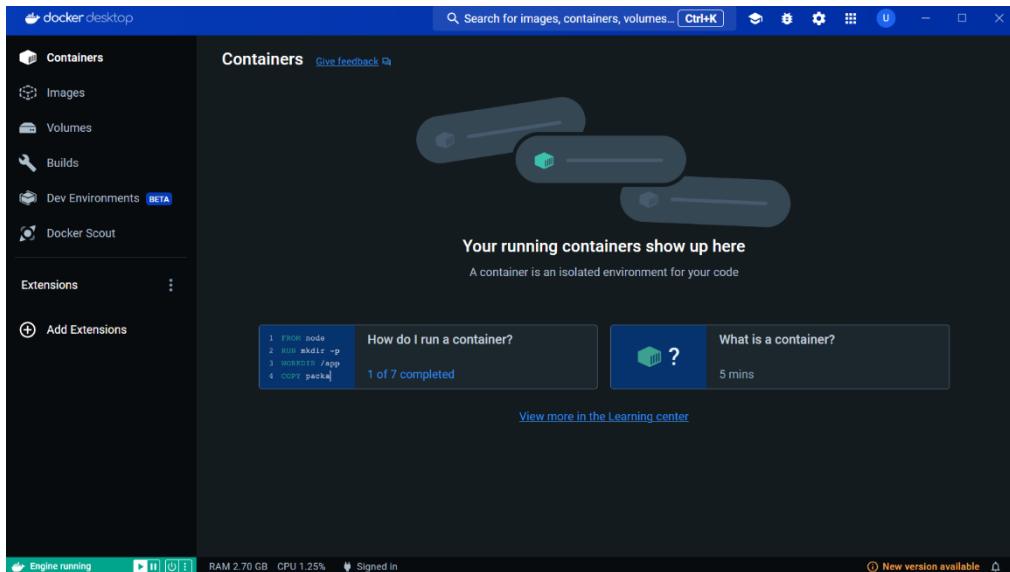
### 6. การป้องกัน และเข้ารหัสข้อมูล (Data Security)

- Description : บริษัทต้องการให้ข้อมูลที่ถูกส่งผ่าน API มีความปลอดภัย และถูกเข้ารหัส
- Implementation : ใช้ Certificates SSL/TLS และการตั้งค่า SNI เพื่อเข้ารหัสการเชื่อมต่อระหว่าง client และ Kong เพื่อป้องกันการดักฟังข้อมูล

## Demo Kong API Gateway

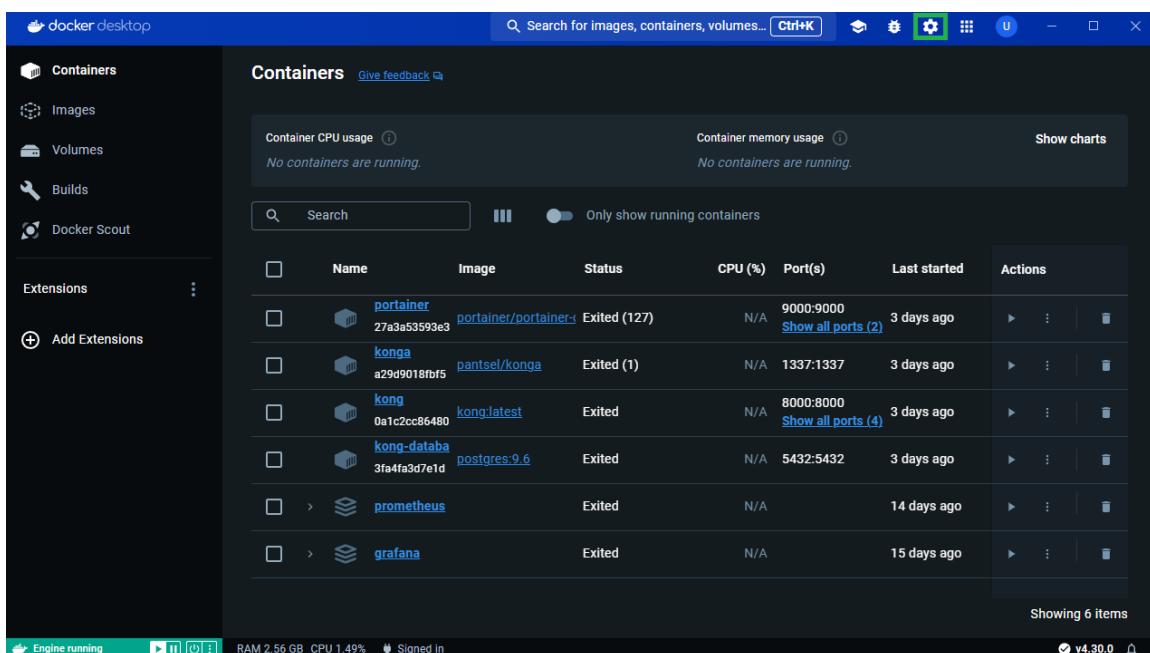
### ติดตั้ง Kong

ขั้นตอนแรกทำการติดตั้ง Docker โดยไปที่เว็บไซต์ <https://www.docker.com/get-started/> แล้วทำการดาวน์โหลดและติดตั้งตัว Docker หลังจากติดตั้งเสร็จแล้วจะเห็นไอคอน Docker Desktop ขึ้นมาอยู่ตรงหน้า Desktop ของเรา และลองเปิดตัว Docker ขึ้นมาแล้วทำการ login ดูว่าสามารถเข้าใช้งานได้ด้วย

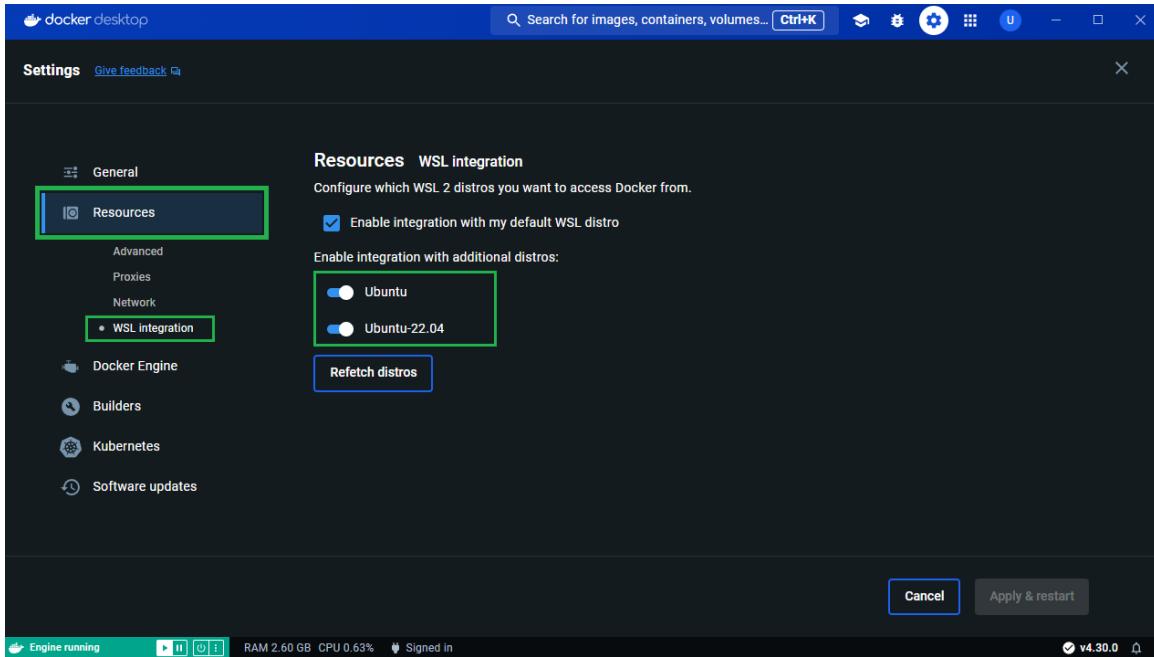


กรณีใช้ Ubuntu ในการรันคำสั่งของ Docker ต่างๆ

ไปที่ปุ่ม Setting บนขวาบาน



ต่อมาไปที่ Resources → WSL integration และทำการเปิด Ubuntu ที่เราใช้



หลังจากที่เรามี Docker แล้วจะมาทำการติดตั้ง Kong กัน

โดยทางผมจะใช้ตัว Git Bash ใน Terminal ของ Virtual Studio Code ในการรันคำสั่ง (หรือจะใช้ Terminal ของ Ubuntu ก็ได้)

### 1. สร้าง Docker Network

```
$ docker network create kong-net
```

```
TRIPOP@DESKTOP-59RR9SH MINGW64 ~
● $ docker network create kong-net
71ddff33679e92b1db12e40b8755b1610b21572dc99208067eba564509d5b235
```

### 2. สร้าง Database และเชื่อมต่อเข้ากับ Docker Network

```
$ docker volume create pg_data
```

```
TRIPOP@DESKTOP-59RR9SH MINGW64 ~
● $ docker volume create pg_data
pg_data
```

```
docker run -d --name kong-database \
--network=kong-net \
--restart unless-stopped \
-v pg_data:/var/lib/postgresql/data \
-p 5432:5432 \
-e "POSTGRES_USER=kong" \
-e "POSTGRES_DB=kong" \
-e "POSTGRES_PASSWORD=kong" \
postgres:9.6
```

```
TRIPOP@DESKTOP-59RR9SH MINGW64 ~
$ docker run -d --name kong-database \
--network=kong-net \
--restart unless-stopped \
-v pg_data:/var/lib/postgresql/data \
-p 5432:5432 \
-e "POSTGRES_USER=kong" \
-e "POSTGRES_DB=kong" \
-e "POSTGRES_PASSWORD=kong" \
postgres:9.6
3fa4fa3d7e1da8ab4ef3b2807fc25aa7dd3a833749f30487ce7005d4f93a75c7
```

### 3. สั่ง Migration ฐานข้อมูล

```
docker run --rm \
--network=kong-net \
-e "KONG_DATABASE=postgres" \
-e "KONG_PG_HOST=kong-database" \
-e "KONG_PG_USER=kong" \
-e "KONG_PG_PASSWORD=kong" \
-e "KONG_CASSANDRA_CONTACT_POINTS=kong-database" \
kong:latest kong migrations bootstrap
```

### 4. สั่งให้ Kong เชื่อมโยงกับ Database พร้อมกับ Start Kong

```
docker run -d --name kong \
--pull=always \
--restart unless-stopped \
--network=kong-net \
```

```
-e "KONG_DATABASE=postgres" \
-e "KONG_PG_HOST=kong-database" \
-e "KONG_PG_USER=kong" \
-e "KONG_PG_PASSWORD=kong" \
-e "KONG_PROXY_ACCESS_LOG=/dev/stdout" \
-e "KONG_ADMIN_ACCESS_LOG=/dev/stdout" \
-e "KONG_PROXY_ERROR_LOG=/dev/stderr" \
-e "KONG_ADMIN_ERROR_LOG=/dev/stderr" \
-e "KONG_ADMIN_LISTEN=0.0.0.0:8001, 0.0.0.0:8444 ssl" \
-p 8000:8000 \
-p 8443:8443 \
-p 8001:8001 \
-p 8444:8444 \
kong:latest
```

เราสามารถใช้ Kong ผ่าน <http://localhost:8001>

คำอธิบายการใช้งาน Port ต่างๆ ของ Kong

- 8000 สำหรับการเข้าถึง API Gateway ผ่าน HTTP Request
- 8443 สำหรับการเข้าถึง API Gateway ผ่าน HTTPS Request
- 8001 สำหรับการเข้าถึง Kong ผ่าน HTTP Request
- 8444 สำหรับการเข้าถึง Kong ผ่าน HTTPS Request

แต่ Kong จะไม่มี GUI ให้เข้าใช้งาน ดังนั้นเราจะไปใช้งาน Konga เพื่อการใช้งาน Kong ที่ง่ายขึ้น

วิธีติดตั้ง Konga Web UI เพื่อช่วยใช้งาน Kong

1. สร้างฐานข้อมูล และ Migration ฐานข้อมูลสำหรับ Konga

```
docker run --rm \
--network=kong-net \
pantsel/konga -c prepare -a postgres -u postgresql://kong:kong@kong-
database:5432/konga
```

## 2. Start Konga

```
docker run -d -p 1337:1337 \
--restart unless-stopped \
--network kong-net \
-e "TOKEN_SECRET=2E55D9A3EFB8542DC145C19EA3D5E" \
-e "DB_ADAPTER=postgres" \
-e "DB_URI=postgresql://kong:kong@kong-database:5432/konga" \
-e "NODE_ENV=production" \
--name konga \
pantsel/konga
```

หลังจากรันคำสั่งเสร็จก็สามารถเข้าไปใช้งานได้ผ่าน <http://localhost:1337>

เมื่อเข้าไปแล้วให้ทำการ Register และทำการ Login หลังจากนั้นทำการกำหนดชื่อ API Gateway และ URL ที่ใช้จัดการ Kong

Name \*

MyAPI

Kong Admin URL \*

<http://kong:8001>

เมื่อตั้งค่า Kong เสร็จก็จะแสดงหน้าตา Dashboard ต่างๆให้เราใช้งาน

The screenshot shows the Kong Admin UI dashboard. On the left is a dark sidebar with navigation links: DASHBOARD, API GATEWAY (INFO, SERVICES, ROUTES, CONSUMERS, PLUGINS, UPSTREAMS, CERTIFICATES), APPLICATION (USERS, SNAPSHOTS), and SETTINGS. The main area has several cards: 1) CONNECTIONS: ACTIVE 10, READING 0, WRITING 10, WAITING 0, ACCEPTED 29, HANDLED 29, Total Requests: 27. 2) NODE INFO: HostName 0a1c2cc86480, Tag Line Welcome to kong, Version 3.6.1, LUA Version LuaJIT 2.1.0-20231117, Admin listen ["0.0.0.0:8001","0.0.0.0:8444 ssl"]. 3) TIMERS: Pending (empty), Running (bar chart from 0 to 250). 4) DATASTORE INFO: DBMS postgres, Host kong-database, Database kong, User kong, Port 5432. 5) PLUGINS: A list of available plugins including response-rate-limiting, syslog, loggly, datadog, ldsp-auth, statsd, bot-detection, aws-lambda, request-termination, prometheus, proxy-cache, session, acme, grpc-gateway, grpc-web, pre-function, post-function, azure-functions, zipkin, opentelemetry, ai-proxy, ai-prompt-decorator, ai-prompt-template. At the bottom, it says KONGA 0.14.9, GitHub, Issues, Support the project, and Connected to MyAPI.

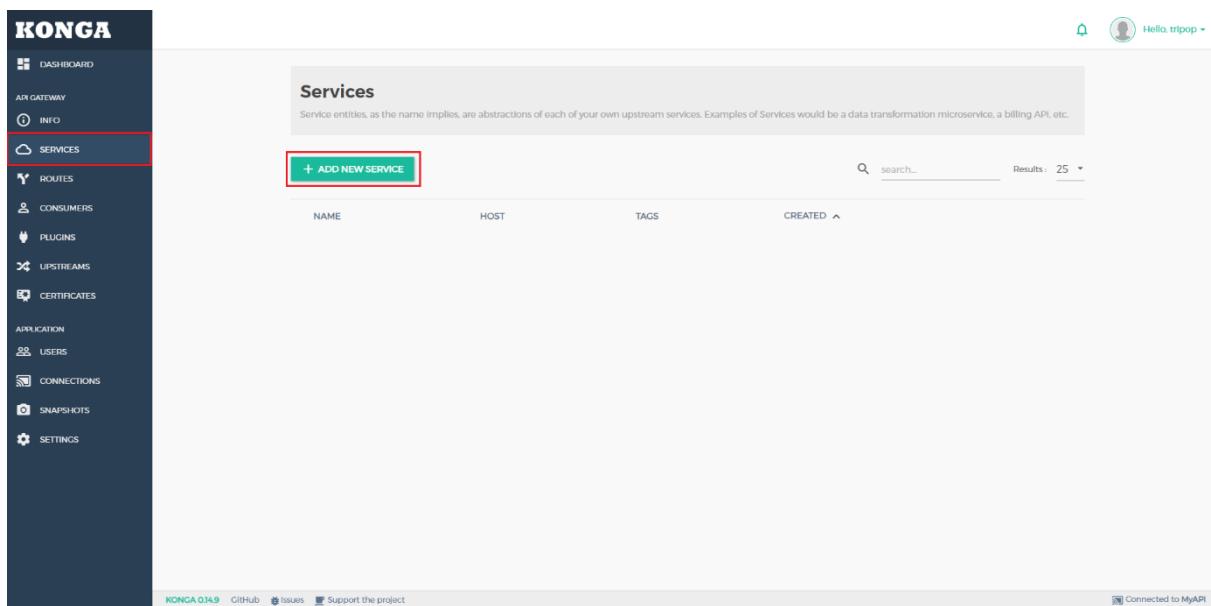
## DEMO - 1

ในตัวอย่างนี้จะใช้ข้อมูล JSON Placeholder API ของ <https://jsonplaceholder.typicode.com/> ซึ่งเป็น Free fake data โดยจะใช้ Konga สร้าง service ออกมานา 2 แบบดังนี้

1. Authenticated endpoints เข้าถึง API ได้โดยต้องแนบ API Keys หรือ JWT มากับ HTTP Headers
  - /todos (Headers with API Keys)
  - /users (Headers with JWT)
2. Public endpoints เข้าถึง API ได้โดยไม่ต้องแนบ API Keys หรือ JWT มากับ HTTP Headers
  - /photos
  - /posts

## กำหนด Services

เมื่อเข้ามาหน้า URL ของ Konga และให้ไปที่หน้า Services --> ADD NEW SERVICE



The screenshot shows the Konga API Gateway interface. On the left, there's a dark sidebar with various menu options: DASHBOARD, API GATEWAY, INFO, SERVICES (which is highlighted with a red box), ROUTES, CONSUMERS, PLUGINS, UPSTREAMS, CERTIFICATES, APPLICATION, USERS, CONNECTIONS, SNAPSHOTs, and SETTINGS. The main content area is titled 'Services' and contains a sub-instruction: 'Service entities, as the name implies, are abstractions of each of your own upstream services. Examples of Services would be a data transformation microservice, a billing API, etc.' Below this, there's a search bar and a results dropdown set to '25'. At the top right, there's a user icon and the text 'Hello, tripop'. In the center, there's a table with four columns: NAME, HOST, TAGS, and CREATED (with an upward arrow). At the top of the table, there's a red box around the '+ ADD NEW SERVICE' button. The bottom of the screen shows some footer links: KONGA 0.14.9, GitHub, Issues, Support the project, and a note 'Connected to MyAPI'.

จะแสดงหน้า CREATE SERVICE ออกมากิ่งๆ ข้อมูลดังนี้

Name : JSON-Placeholder  
Description : API Resource of json placeholder  
Protocol : https  
Host : jsonplaceholder.typicode.com  
Port : 443  
Path : /  
Retries : 5  
Connect timeout : 60000  
Write timeout : 60000  
Read timeout : 60000

**CREATE SERVICE** X

---

<b>Name</b> <small>(optional)</small>	JSON-Placeholder
The service name.	
<b>Description</b> <small>(optional)</small>	API Resource of json placeholder
An optional service description.	
<b>Tags</b> <small>(optional)</small>	Optionally add tags to the service
<b>Url</b> <small>(shorthand-attribute)</small>	Shorthand attribute to set <code>protocol</code> , <code>host</code> , <code>port</code> and <code>path</code> at once. This attribute is write-only (the Admin API never "returns" the url).
<b>Protocol</b> <small>(semi-optional)</small>	https
The protocol used to communicate with the upstream. It can be one of <code>http</code> or <code>https</code> .	
<b>Host</b> <small>(semi-optional)</small>	jsonplaceholder.typicode.com
The host of the upstream server.	
<b>Port</b> <small>(semi-optional)</small>	443
The upstream server port. Defaults to <code>80</code> .	
<b>Path</b> <small>(optional)</small>	/
The path to be used in requests to the upstream server. Empty by default.	
<b>Retries</b> <small>(optional)</small>	5
The number of retries to execute upon failure to proxy. The default is <code>5</code> .	
<b>Connect timeout</b> <small>(optional)</small>	60000
The timeout in milliseconds for establishing a connection to your upstream server. Defaults to <code>60000</code> .	
<b>Write timeout</b> <small>(optional)</small>	60000
The timeout in milliseconds between two successive write operations for transmitting a request to the upstream server. Defaults to <code>60000</code> .	
<b>Read timeout</b> <small>(optional)</small>	60000
The timeout in milliseconds between two successive read operations for transmitting a request to the upstream server. Defaults to <code>60000</code> .	
<b>Client certificate</b> <small>(optional)</small>	Certificate ( <code>id</code> ) to be used as client certificate while TLS handshaking to the upstream server.

✓ SUBMIT SERVICE

เมื่อคุณ SUBMIT SERVICE แล้วก็จะได้ Service ที่เราสร้างอອกมา

The screenshot shows a list of services. There is one item listed:

NAME	HOST	TAGS	CREATED
JSON-Placeholder API Resource of json placeholder	Jsonplaceholder.typicode.com		May 23, 2024

Buttons at the top include '+ ADD NEW SERVICE' and a search bar.

## สร้าง Route

กดเข้าไปที่ตัว Service ที่เราสร้างแล้วไปที่หน้า Route --> ADD ROUTE

The screenshot shows the 'Routes' section of the JSON-Placeholder service. It includes a search bar and a table header:

Name / ID	Hosts	Paths	Protocols	Methods	Regex priority	Created
-----------	-------	-------	-----------	---------	----------------	---------

No data found.

Buttons at the top right include '+ ADD ROUTE'.

User Route (/users)

กำหนดค่าดังนี้

Name : Users

Path : /users (หลังระบุเสร็จแล้วให้กด Enter ที่ช่อง Path)

Path handling : v1

Https redirect status code : 426

Regex priority : 0

Strip Path : NO

Protocols : http, https

**ADD ROUTE TO JSON-PLACEHOLDER**

\* For hosts, paths, methods and protocols, snis, sources, headers and destinations press enter to apply every value you type

Name (optional)	Users
Tags (optional)	Optional tags for the route.
Hosts (semi-optional)	A list of domain names that match this Route. For example: example.com. At least one of hosts, paths, or methods must be set.
Paths (semi-optional)	/users X
Headers (semi-optional)	One or more lists of values indexed by header name that will cause this Route to match if present in the request. The <code>Host</code> header cannot be used with this attribute. hosts should be specified using the <code>hosts</code> attribute. Field values format example: x-some-header:foo,bar
Path handling	v1
Https redirect status code (optional)	426
Regex priority (optional)	0
Methods (semi-optional)	A list of HTTP methods that match this Route. At least one of hosts, paths, or methods must be set.
Strip Path (optional)	NO
Preserve Host (optional)	NO
Protocols (semi-optional)	http X https X

เมื่อกด SUBMIT ROUTE แล้วก็จะได้ Route ที่เราสร้างขึ้นมา

**Service JSON-Placeholder**

services / show

**Routes**

+ ADD ROUTE

Name / ID	Hosts	Paths	Protocols	Methods	Regex priority	Created
Users	-	/users	http, https	-	0	May 23, 2024

ต่อมาทำการเพิ่ม Plugin JWT ให้กับ /users กดเข้าไปที่ Route : Users ไปที่หน้า Plugin --> ADD PLUGIN

Route Users

routes / route

Route Details

Plugins

+ ADD PLUGIN

Eligible consumers beta

Assigned plugins

search plugins...

Name	Consumer	Created
no data found...		

เลือก Authentication : JWT (ADD PLUGIN)

กำหนดค่าดังนี้

key claim name : iss

secret is base64 : NO

run on preflight : YES

maximum expiration : 0

ADD JWT

Verify requests containing HS256 or RS256 signed JSON Web Tokens (as specified in RFC 7519). Each of your Consumers will have JWT credentials (public and secret keys) which must be used to sign their JWTs. A token can then be passed through the Authorization header or in the request's URI and Kong will either proxy the request to your upstream services if the token's signature is verified, or discard the request if not. Kong can also perform verifications on some of the registered claims of RFC 7519 (exp and nbf).

consumer

The CONSUMER ID that this plugin configuration will target. This value can only be used if authentication has been enabled so that the system can identify the user making the request. If left blank, the plugin will be applied to all consumers.

uri param names

Tip: Press **Enter** to accept a value.  
A list of querystring parameters that Kong will inspect to retrieve JWTs.

cookie names

Tip: Press **Enter** to accept a value.

key claim name

iss

The name of the claim in which the key identifying the secret must be passed.

secret is base64

NO  
If true, the plugin assumes the credential's secret to be base64 encoded. You will need to create a base64 encoded secret for your consumer, and sign your JWT with the original secret.

claims to verify

Tip: Press **Enter** to accept a value.  
A list of registered claims (according to RFC 7519) that Kong can verify as well. Accepted values: exp, nbf.

anonymous

run on preflight

YES

maximum expiration

0

header names

Tip: Press **Enter** to accept a value.

+ ADD PLUGIN

## กลับไปที่หน้า Service ของเรา และสร้าง Todos Route (/todos)

กำหนดค่าดังนี้

Name : Todos

Path : /todos (หลังระบุแล้วให้กด Enter ที่ช่อง Path)

Path handling : v1

Https redirect status code : 426

Regex priority : 0

Strip Path : NO

Protocols : http, https

**ADD ROUTE TO JSON-PLACEHOLDER**

\* For hosts, paths, methods and protocols, sinks, sources, headers and destinations press enter to apply every value you type

**Name**  (optional)  
The name of the Route.

**Tags**  (optional)  
Optionally add tags to the route

**Hosts**  (semi-optional)  
A list of domain names that match this Route. For example: example.com. At least one of hosts, paths, or methods must be set.

**Paths**  (semi-optional)  
A list of paths that match this Route. For example: /my-path. At least one of hosts, paths, or methods must be set.

**Headers**  (semi-optional)  
One or more lists of values indexed by header name that will cause this Route to match if present in the request. The Host header cannot be used with this attribute: hosts should be specified using the hosts attribute.  
Field values format example: x-some-header:foo,bar

**Path handling**  (optional)  
Controls how the Service path, Route path and requested path are combined when sending a request to the upstream. See above for a detailed description of each behavior. Accepted values are: "v0", "v1". Defaults to "v1".

**Https redirect status code**  (optional)  
The status code Kong responds with when all properties of a Route match except the protocol, i.e. if the protocol of the request is **HTTP** instead of **HTTPS**, **Location** header is injected by Kong if the field is set to 301, 302, 307 or 308. Defaults to **426**.

**Regex priority**  (optional)  
A number used to choose which route resolves a given request when several routes match it using regexes simultaneously. When two routes match the path and have the same regex\_priority, the older one (lowest created\_at) is used. Note that the priority for non-regex routes is different (longer non-regex routes are matched before shorter ones). Defaults to **0**.

**Methods**  (semi-optional)  
A list of HTTP methods that match this Route. At least one of hosts, paths, or methods must be set.

**Strip Path**  (optional)  
When matching a Route via one of the paths, strip the matching prefix from the upstream request URL.

**Preserve Host**  (optional)  
When matching a Route via one of the hosts domain names, use the request Host header in the upstream request headers. By default set to false, and the upstream Host header will be that of the Service's host

**Protocols**  (semi-optional)

เมื่อกด SUBMIT ROUTE ก็จะได้ Route : Todos เพิ่มเข้ามา

The screenshot shows the 'Service JSON-Placeholder' interface. On the left, there's a sidebar with 'Service Details' (blue), 'Routes' (selected, green), 'Plugins' (grey), and 'Eligible consumers' (beta). The main area is titled 'Routes' with a search bar 'search routes...'. A table lists two routes:

Name / ID	Hosts	Paths	Protocols	Methods	Regex priority	Created	Actions
Todos	-	/todos	http, https	-	0	May 23, 2024	<span>EDIT</span> <span>DELETE</span>
Users	-	/users	http, https	-	0	May 23, 2024	<span>EDIT</span> <span>DELETE</span>

เพิ่ม Plugin Key Auth ให้กับ /todos กดเข้าไปที่ Route : Todos ไปที่หน้า Plugin --> ADD PLUGIN

เลือก Authentication : Key Auth (ADD PLUGIN)

กำหนดค่าดังนี้
key names : x-api-key
hide credentials : NO
key in header : YES
key in query : NO
key in header : NO
run on preflight : YES

**ADD KEY AUTH**

Add Key Authentication (also referred to as an API key) to your APIs. Consumers then add their key either in a querystring parameter or a header to authenticate their requests.

**consumer**

The CONSUMER ID that this plugin configuration will target. This value can only be used if authentication has been enabled so that the system can identify the user making the request. If left blank, the plugin will be applied to all consumers.

**key names** **x-api-key**

Tip: Press **Enter** to accept a value.  
Describes an array of comma separated parameter names where the plugin will look for a key. The client must send the authentication key in one of those key names, and the plugin will try to read the credential from a header or the querystring parameter with the same name.

**hide credentials** **NO**  
An optional boolean value telling the plugin to hide the credential to the upstream API server. It will be removed by Kong before proxying the request.

**anonymous**

**key in header** **YES**

**key in query** **NO**

**key in body** **NO**

**run on preflight** **YES**

**✓ ADD PLUGIN**

ต่อมาเราจะสร้าง Route แบบ Public กัน

กลับไปที่หน้า Service ที่เราสร้าง และไปที่ Routes --> ADD ROUTE

Photos Route (/photos)

กำหนดค่าดังนี้

Name : Photos

Path : /photos

path handling : v1

Https redirect status code : 426

Regex priority : 0

Strip Path : NO

Protocols : http, https

## Posts Route (/posts)

กำหนดค่าดังนี้
Name : Posts
Path : /posts
path handling : v1
Https redirect status code : 426
Regex priority : 0
Strip Path : NO
Protocols : http, https

เมื่อสร้างเสร็จก็จะมี Routes ทั้งหมดดังนี้

The screenshot shows the 'Service JSON-Placeholder' interface. On the left, there's a sidebar with 'Service Details' (info icon), 'Routes' (selected, indicated by a green bar), 'Plugins' (gear icon), and 'Eligible consumers' (two people icon, beta). The main area is titled 'Routes' with a search bar 'search routes...'. A '+ ADD ROUTE' button is at the top right. Below is a table listing four routes:

	Name / ID	Hosts	Paths	Protocols	Methods	Regex priority	Created	Actions
1	Posts	-	/posts	http, https	-	0	May 23, 2024	<span>EDIT</span> <span>DELETE</span>
2	Photos	-	/photos	http, https	-	0	May 23, 2024	<span>EDIT</span> <span>DELETE</span>
3	Todos	-	/todos	http, https	-	0	May 23, 2024	<span>EDIT</span> <span>DELETE</span>
4	Users	-	/users	http, https	-	0	May 23, 2024	<span>EDIT</span> <span>DELETE</span>

## กำหนด Consumers

ในส่วนนี้จะเป็นการเพิ่ม Consumer User สำหรับ Authentication ของ /users และ /todos

ไปที่ CONSUMERS --> CREATE CONSUMER

The screenshot shows the Konga UI for managing API consumers. The left sidebar has a 'CONSUMERS' option selected, indicated by a red box. The main area displays a table of consumers with columns: USERNAME, CUSTOM\_ID, TAGS, and CREATED. One consumer, 'tripop', is listed with a creation timestamp of 'May 08 2024 @10:58'. A 'DELETE' button is located to the right of the consumer row.

ที่หน้า CREATE CONSUMER

กำหนด username : test (กำหนดชื่อได้ตามที่เราต้องการ)

### CREATE CONSUMER

username  
(semi-optional)  The username of the consumer. You must send either this field or `custom_id` with the request.

custom\_id  
(semi-optional) Field for storing an existing ID for the consumer, useful for mapping Kong with users in your existing database. You must send either this field or `username` with the request.

Tags  
(optional) Optionally add tags to the consumer

**✓ SUBMIT CONSUMER**

เมื่อคุณ SUBMIT ก็จะได้ CONSUMER ที่เราสร้างขึ้นมา

The screenshot shows the 'Consumers' section of the Kong API Manager. At the top, there is a green button labeled '+ CREATE CONSUMER'. Below it is a search bar with the placeholder 'search...' and a results counter 'Results: 25'. The main area displays a table with columns: USERNAME, CUSTOM\_ID, TAGS, and CREATED. Two rows are present:

USERNAME	CUSTOM_ID	TAGS	CREATED
test	-		May 23 2024 @01:45
trip0	-		May 08 2024 @10:08

Each row has a red 'DELETE' button on the right.

## API Keys

จากนั้นเข้าไปที่ Consumer ที่เราสร้าง Credentials --> API KEYS --> CREATE API KEY

The screenshot shows the 'CONSUMER: test' page. At the top, there is a navigation bar with 'consumers / edit consumer'. Below it are several tabs: Details (1), Groups, Credentials (highlighted with a red box), Accessible Routes, and Plugins. The 'Credentials' tab is active. On the left, there is a sidebar with 'BASIC' (2) and 'API KEYS' (highlighted with a red box). The main content area shows a heading 'Api Keys' with the subtext 'You have not created any keys for this consumer yet'. On the right, there is a green button '+ CREATE API KEY' (3).

เราสามารถกด create แล้วทำการ submit ได้เลย ทางตัว API Key จะสุ่ม Key ออกมายังเรา หรือจะทำการกำหนด API Key เองก็ได้เหมือนกัน

The screenshot shows the 'CONSUMER: test' page again. The 'Credentials' tab is active. The 'API KEYS' tab is selected (highlighted with a red box). The main content area shows a heading 'Api Keys' with a table listing one key:

#	key	created
1.	6AKoS2INTPgst7IUCZgkPXB1rtXr2XWO	May 23, 2024

On the right, there is a red 'DELETE' button.

## JWT

ไปที่ เมนู Consumer --> Consumer (ที่เราได้สร้างเอาไว้ : test) --> Credentials --> JWT --> CREATE JWT

The screenshot shows the 'CONSUMER: test' page with the 'edit consumer' link at the top left. Below it are tabs: 'Details' (with an info icon), 'Groups' (with a people icon), 'Credentials' (with a shield icon, highlighted by a red box labeled '1'), 'Accessible Routes' (with a cloud icon), and 'Plugins' (with a plug icon). Under the 'Credentials' tab, there are sections for 'BASIC', 'API KEYS', 'HMAC', and 'OAUTH2'. The 'OAUTH2' section is highlighted with a red box labeled '2'. In the 'JWT' section, there is a button labeled '+ CREATE JWT' highlighted with a red box labeled '3'.

### CREATE JWT

Create JWT for **test**

**key**  
(optional)

A unique string identifying the credential. If left out, it will be auto-generated. However, usage of this key is **mandatory** while crafting your token, as specified in the next section.

**algorithm**  
(optional)

HS256

The algorithm used to verify the token's signature. Can be **HS256** or **RS256**.

**rsa\_public\_key**  
(optional)

If **algorithm** is **RS256**, the public key (in PEM format) to use to verify the token's signature.

**secret**  
(optional)

If **algorithm** is **HS256**, the secret used to sign JWTs for this credential. If left out, will be auto-generated. If **algorithm** is **RS256**, this is the private key (in PEM format) to use to verify the token's signature.

**✓ SUBMIT**

ในส่วนของ CREATE JWT ก็เหมือนของ API KEY จะกรอกข้อมูลในส่วน key หรือ secret เองก็ได้ กรณีถ้าไม่กรอกเองก็กด SUBMIT สร้างตัว Kong จะสร้าง key และ secret ออกมาให้ ตัวอย่างดังนี้

The screenshot shows the Kong API UI under the 'consumers' section. The consumer is named 'test'. The 'Credentials' tab is selected, and the 'JWT' sub-tab is active. A JSON object representing the JWT is displayed:

```
{
  "algorithm": "HS256",
  "consumer": {
    "id": "7960669b-d929-43ce-9645-46e8efa61ce3"
  },
  "id": "35454645-31d0-4a84-a914-c64ae5b9ce47",
  "created_at": 1716404141,
  "key": "SFB4wQZQYQ1dwoW2TuM1YGaWV8KWYFwa",
  "tags": null,
  "secret": "ijsjBHFU0l0HZRVf9ur0kS8pu7Bv7b9I",
  "rsa_public_key": null
}
```

Below the JSON, there are 'Created' and 'DELETE' buttons. A red box highlights the 'secret' field.

#### ทดสอบ Generate JWT (Manual)

ตัวอย่างในครั้งนี้จะทดลองใช้ JWT Builder แบบ Manual ของ <http://jwtbuilder.jamiekurtz.com/> โดย

##### กำหนดค่าดังนี้

Issuer : SFB4wQZQYQ1dwoW2TuM1YGaWV8KWYFwa (key)

Issued At : วัน เวลา ที่ออก Token

Expiration : วัน เวลา ที่ Token หมดอายุ

Audience : ชื่อผู้ที่รับรอง Token

Subject : id หรือ username ของผู้ใช้ Token

Additional Claims (สามารถเพิ่มได้ตามที่เราต้องการ)

First Name : Tripop

Surname : Raksa

Signed JSON Web Token

key : ijsjBHFU0l0HZRVf9ur0kS8pu7Bv7b9I (secret)

ผลลัพธ์เมื่อกด Generate

eyJ0eXAiOiJKV1QiLCJhbGciOiJIUzI1NiJ9eyJpc3MiOiJTRkl0d1FaUVlRMWR3b1cyVHVNMVlHYVdWOEtXWUZ3YSIsImhlhdCI6MTcxNjQwNDYyMSwiZXhwIjoxNzQ3OTQwNjIxLCJhdWQiOiJ3d3cuZXhhbXBsZS5jb20iLCJzdWIiOiJ0ZXN0liwiRmlyc3QgTmFtZSI6IlRyaXBvcCIsIlN1cm5hbWUiOiJSYWtzYSJ9. q-w\_ZHDS7Cdy0rnACGvVBEMtssqg3nZDQYGsKYd-U3M

**Standard JWT Claims**

<b>Issuer</b>	SFB4wQZQYQ1dwoW2TuM1YGaWV8KWFwa	Identifier (or, name) of the server or system issuing the token. Typically a DNS name, but doesn't have to be.
<b>Issued At</b>	2024-05-22T19:03:41.983Z	Date/time when the token was issued. (defaults to now) <b>now</b>
<b>Expiration</b>	2025-05-22T19:03:41.983Z	Date/time at which point the token is no longer valid. (defaults to one year from now) <b>now</b> <b>in 20 minutes</b> <b>in 1 year</b>
<b>Audience</b>	www.example.com	Intended recipient of this token; can be any string, as long as the other end uses the same string when validating the token. Typically a DNS name.
<b>Subject</b>	test	Identifier (or, name) of the user this token represents.

**Additional Claims**

Claim Type	Value
First Name	Tripop
Surname	Raksa

Use this section to define 0 or more custom claims for your token. The claim type can be anything, and so can the value.  
If recipient of the token is a .NET Framework application, you might want to follow the Microsoft [ClaimType names](#). You can also use the .NET-oriented claim buttons below.

**Buttons:** clear all | add one | add email claim | add name claim (.NET) | add role claim (.NET) | add email claim (.NET)

**Generated Claim Set (plain text)**

```
{
  "iss": "SFB4wQZQYQ1dwoW2TuM1YGaWV8KWFwa",
  "iat": 1716404621,
  "exp": 1747940621,
  "aud": "www.example.com",
  "sub": "test",
  "First Name": "Tripop",
  "Surname": "Raksa"
}
```

This section displays the claims that will be signed and base64-encoded into a complete JSON Web Token.

**Signed JSON Web Token**

Key: ijsjBFHU0i0HZRVf9ur0kS8pu7Bv7b9i **32** HS256 **Create Signed JWT**

```
eyJ0eXAiOiJKV1QiLCJhbGciOiJIUzI1NiJ9.eyJpc3Mi0iJTRkI0d1FaUVlRMW3b1cyVHWNVlHVVdloEtXwUZ3YSis1mlhdCIGMTcxNjQmNDYyIiSwiZXhwIjoxNzQ3OTQwNjIxLCJhdQ0i0i33d3cuZXhhXBsZ55jb20iLCjzdwIi0102ZXw0I1wiRmlyc3QgTmftZSI61lRyaBvcCISi1h1cm5hbWUi0i15YVtzYSJ9.q-w_ZHD57Cdy0rnACGvVBEMtssqg3nDQYGsKyd-U3M
```

**Buttons:** Copy JWT to Clipboard

## Result

```
{
  "iss": "SFB4wQZQYQ1dwoW2TuM1YGaWV8KWFwa",
  "iat": 1716404621,
  "exp": 1747940621,
  "aud": "www.example.com",
  "sub": "test",
  "First Name": "Tripop",
  "Surname": "Raksa"
}
```

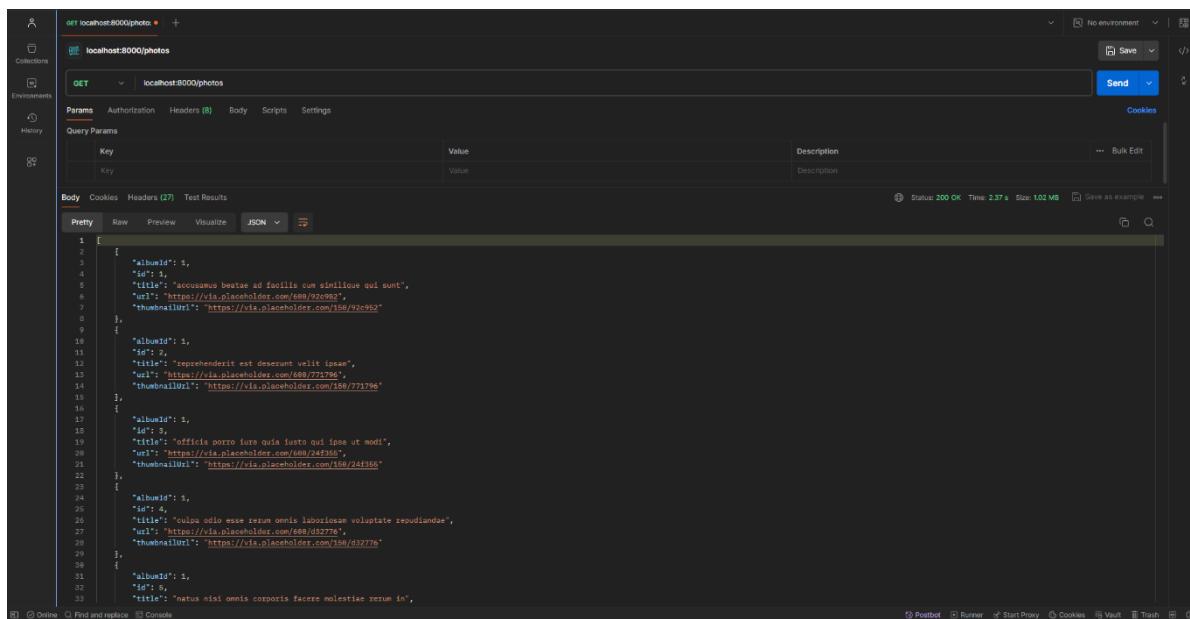
## ทดสอบการใช้งาน

การใช้งานจะแบ่งเป็น 2 ประเภทดังต่อไปนี้

แบบ Public

ทดสอบเรียก API แบบ GET สำหรับ /photos และ /posts โดยใช้ Postman

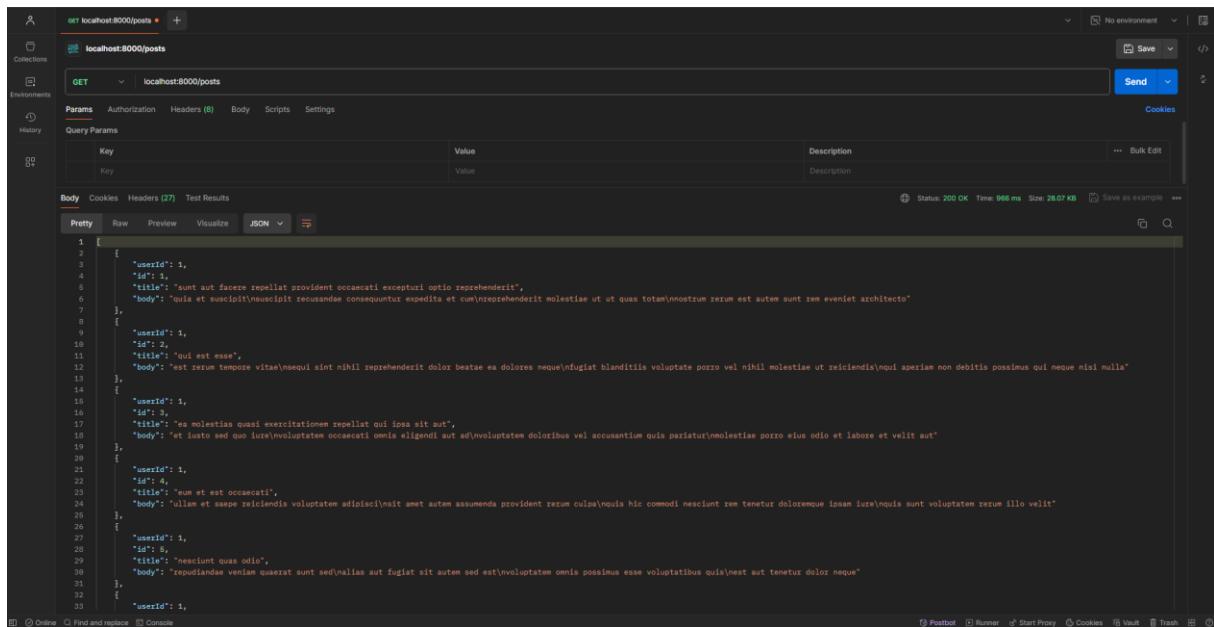
/photos



The screenshot shows the Postman interface with a GET request to `localhost:8000/photos`. The response is a JSON array containing three photo objects:

```
[{"id": 1, "title": "accusamus beatae ad facilis cum similique qui sunt", "url": "https://via.placeholder.com/640x320c02", "thumbnaillurl": "https://via.placeholder.com/150x92c092"}, {"id": 2, "title": "reprehenderit est deserunt velit ipsam", "url": "https://via.placeholder.com/680x771796", "thumbnaillurl": "https://via.placeholder.com/150x771796"}, {"id": 3, "title": "officia porro iure nesciunt isto qui idem ut modi", "url": "https://via.placeholder.com/680x24336", "thumbnaillurl": "https://via.placeholder.com/150x24336"}]
```

/posts



The screenshot shows the Postman interface with a GET request to `localhost:8000/posts`. The response is a JSON array containing four post objects:

```
[{"id": 1, "userId": 1, "title": "sunt aut facere repellat provident occaecati excepturi optio reprehenderit", "body": "quia et suscipit\\nuscipit recusandae consequuntur expedita et cum\\nreprehenderit molestiae ut ut quas totam\\nnostrum rerum est autem sunt rem eveniet architecto"}, {"id": 2, "userId": 1, "title": "qui est esse", "body": "est rerum tempore vitae\\nsequi sint nihil reprehenderit dolor beatas ea dolores neque\\nfugiat blanditiis voluptate porro vel nihil molestiae ut reiciendis\\nqui aperiam non debitis possimus qui neque nisi nulla"}, {"id": 3, "userId": 1, "title": "ea molestias quasi exercitationem repellat qui ipsa sit aut", "body": "et iusto sed quo iure\\nvolutatem occaecati omnis eligendi aut\\ndivulpatem doloribus vel accusantium quis pariatur\\nmolestias porro eius odio et labore et velit aut"}, {"id": 4, "userId": 1, "title": "eum et est occaecati", "body": "ullam et sapiente voluptatum adipisci\\nsit amet autem assumenda provident rerum culpa\\nquis hic commodi nesciunt rem tenetur doloremque ipsam iure\\nquis sunt voluptatem rerum illo velit"}, {"id": 5, "userId": 1, "title": "nesciunt quas odio", "body": "repudiandae veniam querat sunt sed\\nalias aut fugiat sit autem sed est\\nvolutatem omnis possimus esse voluptatibus quis\\nest aut tenetur dolor neque"}]
```

## แบบ Authentication

ทดสอบเรียก API แบบ GET สำหรับ /todos และ /users (โดยไม่ส่ง Key) โดยใช้ Postman

จะพบปัญหาว่าไม่สามารถเข้าถึง API ได้ดังนี้

/users

The screenshot shows the Postman interface with a request to `localhost:8000/users`. The response status is 401 Unauthorized, with a message "unauthorized".

```
1 {
2     "message": "unauthorized"
3 }
```

/todos

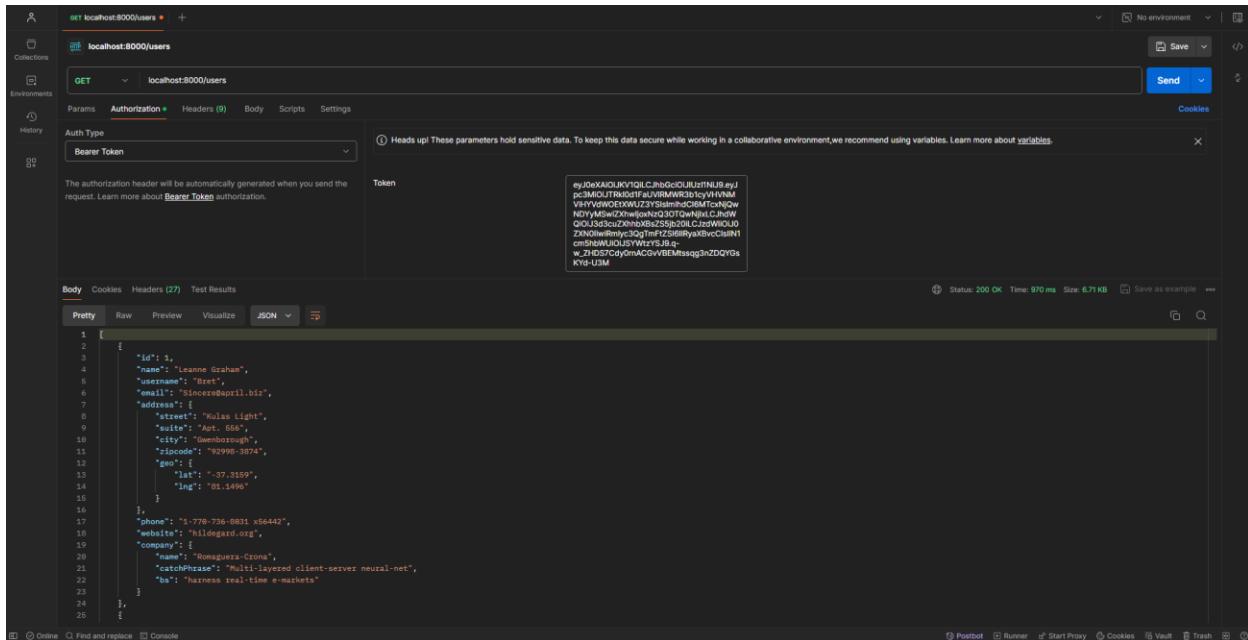
The screenshot shows the Postman interface with a request to `localhost:8000/todos`. The response status is 401 Unauthorized, with a message "No API key found in request" and a request ID.

```
1 {
2     "message": "No API key found in request",
3     "request_id": "36429869cb4fb738b7ecab6f399d8962"
4 }
```

ทดสอบส่ง key ไปด้วย

/users

นำตัว Token ที่เรา generate ได้มาใส่ในส่วนของ Bearer Token



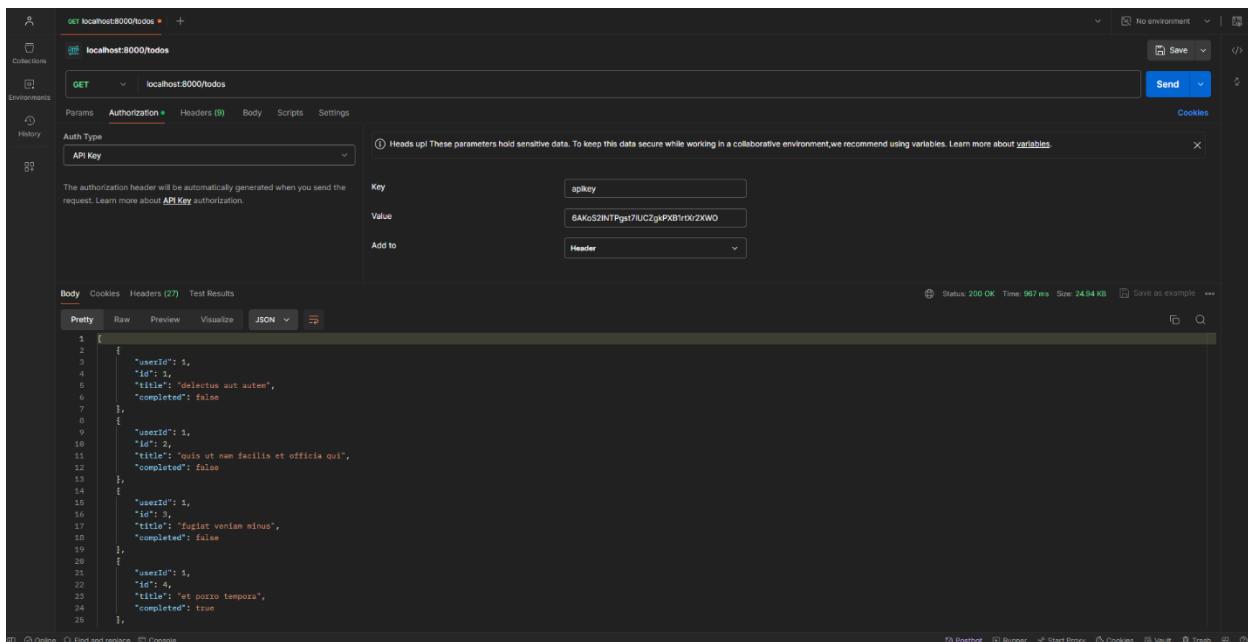
The screenshot shows a Postman request to `localhost:8000/users` using the `GET` method. The `Authorization` field is set to `Bearer Token`. The response status is `200 OK` with a time of `970 ms` and a size of `6.71 KB`. The response body is a JSON object representing a user:

```
1 [
2   {
3     "id": 1,
4     "name": "Leanne Graham",
5     "username": "bret",
6     "email": "Sincere@april.biz",
7     "address": {
8       "street": "Kulas light",
9       "suite": "Apt. 556",
10      "city": "Gwenborough",
11      "zipcode": "92998-3874",
12      "geo": {
13        "lat": "-37.3569",
14        "lng": "81.1496"
15      }
16    },
17    "phone": "1-770-736-8031 x56442",
18    "website": "hildegard.org",
19    "company": {
20      "name": "Rönsger-Crona",
21      "catchPhrase": "Multi-layered client-server neural-net",
22      "bs": "harness real-time e-markets"
23    }
24  ],
25 ]
```

/todos

นำ key ที่เราสร้างใน API KEYS มาใส่ในส่วนของ API Key ของ Postman

#6AKoS2lNTPgst7lUCZgkPXB1rtXr2XWO



The screenshot shows a Postman request to `localhost:8000/todos` using the `GET` method. The `Authorization` field is set to `API Key`, with the value `apikey` and `Header` selected. The response status is `200 OK` with a time of `967 ms` and a size of `24.94 KB`. The response body is a JSON array of todos:

```
1 [
2   {
3     "userId": 1,
4     "id": 1,
5     "title": "delectus aut autem",
6     "completed": false
7   },
8   {
9     "userId": 1,
10    "id": 2,
11    "title": "quis ut nam facilis et officia qui",
12    "completed": false
13  },
14  {
15    "userId": 1,
16    "id": 3,
17    "title": "eiusmod tempor inimicus",
18    "completed": false
19  },
20  {
21    "userId": 1,
22    "id": 4,
23    "title": "et porro tempora",
24    "completed": true
25 },
```

ถ้าหากว่าจำ Key name ไม่ได้ว่าตั้งชื่อไว้แบบไหนสามารถเข้าไปดูได้ที่

Services —> Routes —> Todos (Route ที่เราใช้ตัว Plugin Key Auth) —> Plugin —> key-auth

The screenshot shows the 'Route Todos' configuration page. In the 'Assigned plugins' section, there is one entry: 'key-auth'. The 'key-auth' plugin is listed with the following details:

Name	Consumer	Created
key-auth	All consumers	May 23, 2024

Below the table are icons for search, edit, and delete.

### EDIT KEY AUTH

X

Add Key Authentication (also referred to as an API key) to your APIs. Consumers then add their key either in a querystring parameter or a header to authenticate their requests.

ENABLED

#### consumer

The CONSUMER ID that this plugin configuration will target. This value can only be used if authentication has been enabled so that the system can identify the user making the request. If left blank, the plugin will be applied to all consumers.

key names

apikey X

Tip: Press `Enter` to accept a value.

Describes an array of comma separated parameter names where the plugin will look for a key. The client must send the authentication key in one of those key names, and the plugin will try to read the credential from a header or the querystring parameter with the same name.

hide credentials

NO

An optional boolean value telling the plugin to hide the credential to the upstream API server. It will be removed by Kong before proxying the request.

#### anonymous

key in header

YES

key in query

YES

key in body

NO

run on preflight

YES

✓ SUBMIT CHANGES

## Plugin Rate Limiting

Rate Limiting คือ การจำกัดจำนวนครั้งในการเข้าถึง API ในช่วงเวลาที่เรากำหนด เช่น เราสามารถเข้าใช้งาน API ได้สูงสุด 10 requests ใน 1 นาที หรือ 3 request ใน 1 วินาที เป็นต้น โดยหน่วยของเวลาสามารถกำหนดเป็น second, minute, hour, day, month, year

โดยในตัวอย่างนี้เราจะกำหนด แค่ 10 Minute

เข้าไปที่ Route —> Plugin —> ADD PLUGIN —> Traffic Control —> Rate Limiting (ADD PLUGIN)

**EDIT RATE LIMITING** X

Rate limit how many HTTP requests a developer can make in a given period of seconds, minutes, hours, days, months or years. If the API has no authentication layer, the Client IP address will be used, otherwise the Consumer will be used if an authentication plugin has been configured.

**consumer** **ENABLED**

**second**  
The amount of HTTP requests the developer can make per second. At least one limit must exist.

**minute** **10**  
The amount of HTTP requests the developer can make per minute. At least one limit must exist.

**hour**  
The amount of HTTP requests the developer can make per hour. At least one limit must exist.

**day**  
The amount of HTTP requests the developer can make per day. At least one limit must exist.

**month**  
The amount of HTTP requests the developer can make per month. At least one limit must exist.

**year**  
The amount of HTTP requests the developer can make per year. At least one limit must exist.

**limit by** **consumer** ▼

The entity that will be used when aggregating the limits: consumer, credential, ip. If the consumer or the credential cannot be determined, the system will always fallback to ip.

## กด SUBMIT

KONGA

DASHBOARD

API GATEWAY

INFO

SERVICES

ROUTES

CONSUMERS

PLUGINS

UPSTREAMS

CERTIFICATES

APPLICATION

USERS

CONNECTIONS

SNAPSHOTS

SETTINGS

Route Photos

routes / route

Route Details

Plugins

Eligible consumers beta

Assigned plugins

+ ADD PLUGIN

search plugins...

Name	Consumer	Created
rate-limiting	All consumers	May 23, 2024

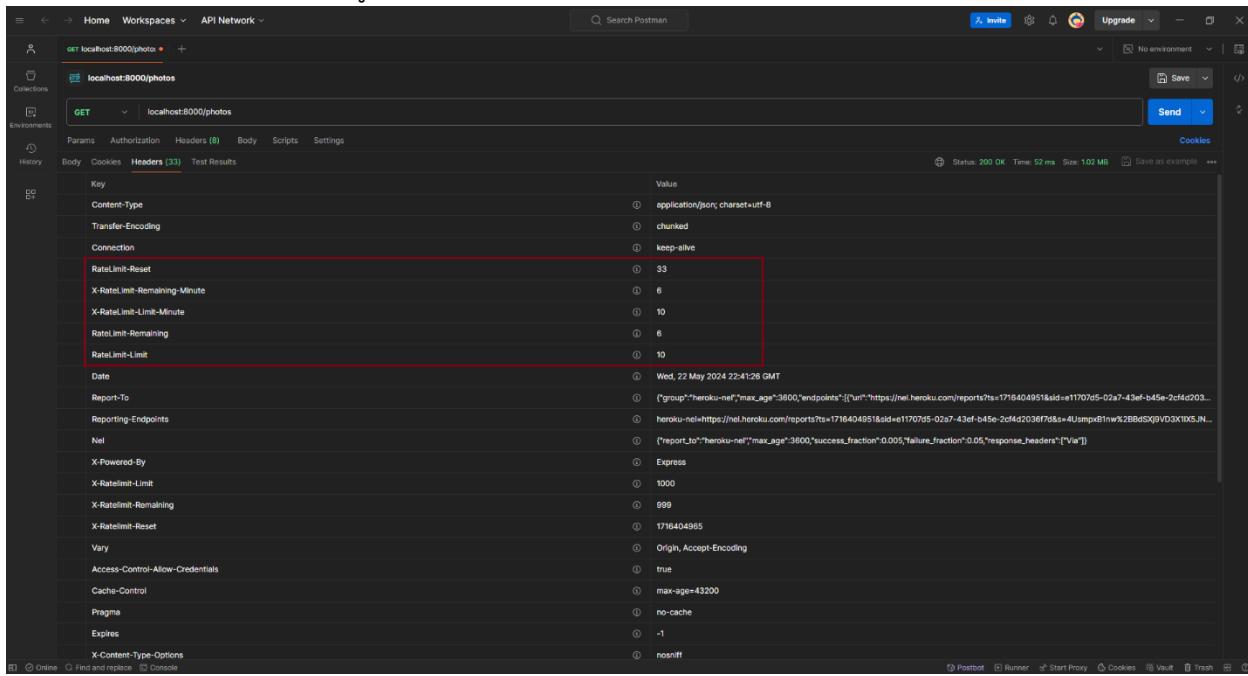
Connected to Demo

จากนั้นทดสอบเข้าใช้งาน API ผ่าน Route ที่เราสร้างไว้ในขั้นตอนก่อนหน้านี้ และทำการ Reload page เกิน 10 ครั้ง จะได้ผลลัพธ์เป็น Error รหัส 429 ดังรูป



## ANSWER

หรือว่าจะลองใช้กับ postman ดูว่า Rate Limit เราเหลือจำนวนเท่าไหร่



The screenshot shows the Postman application interface. In the top navigation bar, there are tabs for Home, Workspaces, and API Network. Below the navigation, there's a search bar labeled "Search Postman". On the left side, there are sections for Collections, Environments, and History. The main workspace shows a GET request to "localhost:8000/photos". The "Headers" tab is currently selected, showing a list of headers with their values. One specific header, "X-RateLimit-Reset", is highlighted with a red box, and its value is displayed as a list of numbers: 33, 6, 10, 6, 10. Other visible headers include Content-Type, Transfer-Encoding, Connection, Date, Report-To, Reporting-Endpoints, Nel, X-Powered-By, X-RateLimit-Limit, X-RateLimit-Remaining, X-RateLimit-Reset, Vary, Access-Control-Allow-Credentials, Cache-Control, Pragma, Expires, and X-Content-Type-Options. At the bottom of the interface, there are buttons for "Send", "Save", and "Cancel". Status information at the bottom right indicates a "Status: 200 OK", "Time: 52 ms", and "Size: 1.02 MB".

นอกจากการใช้งาน plugins โดยสร้างเป็น global plugins ตามตัวอย่างด้านบนแล้วเรายังสามารถเลือกสร้าง plugins ตาม scope ที่กำหนด ซึ่ง scope ของ plugins มีดังนี้

- Global plugins : plugins จะมีผลกับทุกๆ request ที่เข้ามาใน API Gateway
- Route plugins : plugins จะมีผลเมื่อวิ่งผ่านเฉพาะ route ที่ add plugins เข้ามาเท่านั้น
- Service plugins : plugins จะมีผลเฉพาะ service ที่ได้ add plugins เข้ามาเท่านั้น
- Consumer plugins : plugins จะมีผลเฉพาะ consumer หรือ user บางคนที่ได้ add plugins เข้ามาในระบบเท่านั้น

## Plugin File Log ใน Logging

ความสำคัญ และคุณสมบัติของ Plugin File Log

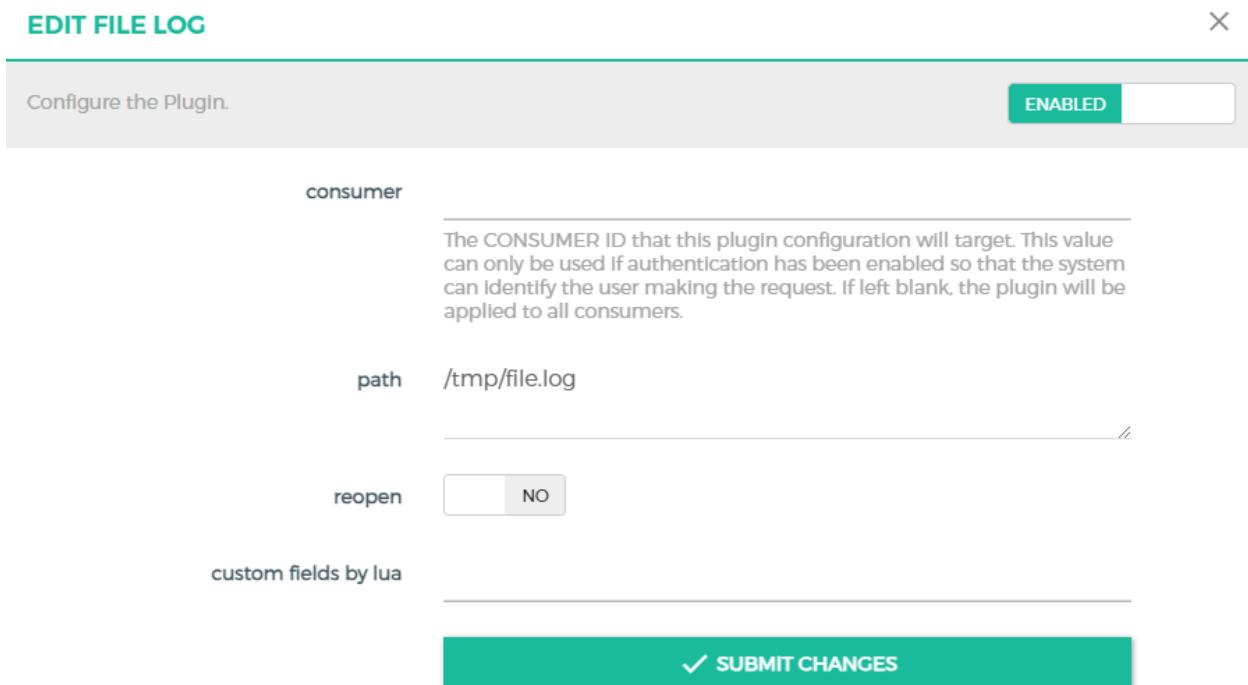
1. การบันทึกข้อมูลการเข้าใช้งาน
  - Plugin จะบันทึกข้อมูลการใช้งาน API ที่ได้รับการประมวลผลโดย Kong ลงไฟล์ที่กำหนดไว้
  - ข้อมูลที่บันทึกประกอบด้วยรายละเอียดของ request และ responses
2. ความยืดหยุ่นในการตั้งค่า
  - สามารถกำหนดรูปแบบของข้อมูลที่ต้องการบันทึกลงในไฟล์ได้
  - รองรับการบันทึกข้อมูลในรูปแบบ JSON

### 3. การวิเคราะห์ และติดตาม

- ช่วยในการตรวจสอบ และวิเคราะห์การเข้าใช้งาน API
- สามารถใช้ข้อมูลที่บันทึกในการตรวจสอบปัญหา และปรับปรุงประสิทธิภาพของระบบ

ในตัวอย่างนี้จะยกตัวอย่างเป็น File Log ในตัว File Log จะเป็นการเก็บไฟล์อยู่ใน docker ของ Kong เราใส่แค่ path ให้ plugin รู้ว่าต้องบันทึกลงในไหน

ไปที่ Route ที่เราต้องการ —> Plugins —> ADD PLUGIN —> LOGGING —> File Log



เมื่อเรากด SUBMIT แล้วให้ไปที่ Postman เพื่อลอง GET ข้อมูลแล้วจากนั้นเปิด Ubuntu

run คำสั่งตามนี้

```
$ docker ps -a
```

```
student@DESKTOP-59RR95H:~$ docker ps -a
CONTAINER ID        IMAGE               COMMAND                  CREATED             STATUS              PORTS                 NAMES
43af28f835a0        pantsel/konga      "/app/start.sh"         53 minutes ago     Up 53 minutes      0.0.0.0:1337->1337/tcp, 0.0.0.0:8000-8001/tcp, 0.0.0.0:8443-8444->8443-8444/tcp   konga
7c7c1a701892        kong:latest        "/docker-entrypoint..."  55 minutes ago     Up 55 minutes (healthy)   0.0.0.0:8000-8001->8000-8001/tcp, 0.0.0.0:8443-8444->8443-8444/tcp   kong
31db9746515b        postgres:9.6       "docker-entrypoint.s..."  55 minutes ago     Up 55 minutes      0.0.0.0:5432->5432/tcp   kong-database
27a353593e3        portainer/portainer-ce:latest  "/portainer"          2 weeks ago       Exited (127) 2 days ago   0.0.0.0:9000->9000/tcp, 8000/tcp, 0.0.0.0:9443->9443/tcp   portainer
fdc39bc8324        prom/prometheus      "/bin/prometheus --c..."  2 weeks ago       Exited (0) 34 minutes ago   0.0.0.0:9000->9000/tcp, 8000/tcp, 0.0.0.0:9443->9443/tcp   prometheus
983a8f28e87e        grafana/grafana      "/run.sh"              2 weeks ago       Exited (0) 33 minutes ago   0.0.0.0:9000->9000/tcp, 8000/tcp, 0.0.0.0:9443->9443/tcp   grafana
student@DESKTOP-59RR95H:~$ |
```

```
§ docker exec -it 7c7c(ส่วนนี้ดูตาม Container id ของเรา) sh
```

```
student@DESKTOP-59RR9SH:~$ docker exec -it 7c7c sh  
$ |
```

ใช้คำสั่ง `ls` เพื่อดูว่ามีไฟล์เดอร์ หรือไฟล์อะไรบ้าง

```
$ ls
```

ใช้คำสั่ง cd ไป tmp ที่เราเก็บ file log ไว้

```
$ cd tmp
```

ใช้คำสั่ง ls จะเห็นว่ามีไฟล์ file.log อยู่

\$ ls

```
student@DESKTOP-59R9SH:~$ docker exec -it 7c7c sh
$ ls
bin  boot  dev  docker-entrypoint.sh  etc  home  lib  lib32  lib64  libx32  media  mnt  opt  proc  root  run  sbin  srv  sys  tmp  usr  var
$ cd tmp
$ ls
file.log
```

ใช้คำสั่ง cat เพื่ออ่าน file.log

```
$ cat file.log
```

สังเกตดูจาก user-agent มันจะหาว่าต้นทางเป็นใคร และยังบอกข้อมูลต่างๆที่เราสร้าง service ไว้ เช่น host, port เป็นต้น

หมายเหตุ ในส่วนค่าที่ได้ของ file.log สามารถนำไปใส่ใน <https://jsonpathfinder.com/> เพื่อให้ถูกรายจัดขึ้นได้ โดยนำเนื้อหาในไฟล์ไปใส่แล้วกด Beautify

## Plugins Basic Auth

### ความสำคัญของ Plugin Basic Auth

1. การยืนยันตัวตนพื้นฐาน:
  - ใช้สำหรับยืนยันตัวตนของผู้ใช้ที่ต้องการเข้าถึง API
  - เหมาะสำหรับการใช้งานที่ไม่ต้องการระบบยืนยันตัวตนที่ซับซ้อน
2. การรักษาความปลอดภัย:
  - ช่วยป้องกันการเข้าถึง API โดยไม่ได้รับอนุญาต
  - เพิ่มระดับความปลอดภัยให้กับ API ของคุณ
3. การตั้งค่าและใช้งานง่าย:
  - ง่ายต่อการตั้งค่าและใช้งาน
  - สามารถสนับสนุนรวมกับระบบการจัดการผู้ใช้และรหัสผ่านได้ง่าย

ในตัวอย่างจะใช้ Route /posts

ไปที่ Route ที่เราต้องการ —> Plugins —> ADD PLUGIN —> Authentication —> Basic Auth

เราจะ ADD PLUGIN โดยที่ไม่ต้องตั้งค่าอะไรใด้เลย

**ADD BASIC AUTH** ×

Add Basic Authentication to your APIs, with username and password protection. The plugin will check for valid credentials in the [Proxy-Authorization](#) and [Authorization](#) header (in this order).

**consumer**

The CONSUMER ID that this plugin configuration will target. This value can only be used if authentication has been enabled so that the system can identify the user making the request. If left blank, the plugin will be applied to all consumers.

**anonymous**

**hide credentials**  **NO**

An optional boolean value telling the plugin to hide the credential to the upstream API server. It will be removed by Kong before proxying the request.

**realm** **service**

**✓ ADD PLUGIN**

เมื่อเราสร้าง Plugin และลองเราลงไปที่ <http://localhost:8000/posts>



จะเห็นว่ามีการให้ Sign in เพื่อเข้าสู่ <http://localhost:8000/posts> นี้เมื่อเรากด Cancel ก็จะ Error ไม่สามารถเข้าดูข้อมูลของ /posts ได้



เราจะไปสร้าง Username และ Password เพื่อมาทำการ Sign in กัน

ไปที่ Consumer เข้าไปที่ Consumer ของเรา —> Credentials —> Basic —> CREATE CREDENTIALS

ให้ทำการสร้าง username และ password ที่ต้องการ (ในตัวอย่างจะตั้ง username: admin pass: admin)

## BASIC AUTH

Manage Basic Auth credentials for **tripop**

<b>username</b> <i>(required)</i>	admin
The username to use in the Basic Authentication	
<b>password</b> <i>(optional)</i>	admin
The password to use in the Basic Authentication	
<b>SUBMIT</b>	

เมื่อเราสร้างเสร็จแล้วตัว password จะถูกเข้ารหัสไว้เพื่อความปลอดภัย

### CONSUMER: tripop

consumers / edit consumer

Details Groups Credentials Accessible Routes Plugins

**BASIC**

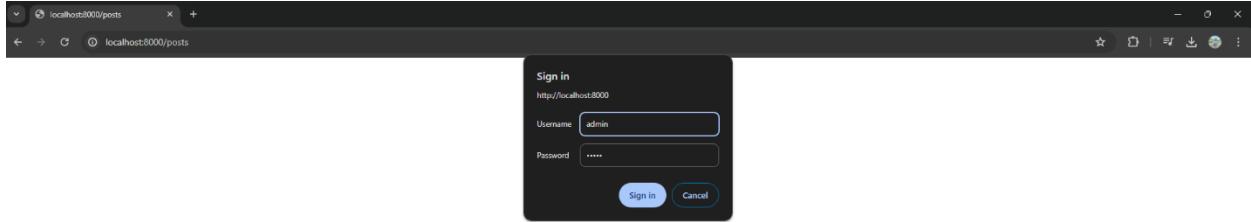
API KEYS HMAC OAUTH2 JWT

**Basic Auth**

#	username	password	created	
1.	admin	78fb4eddcf56a26fdf3ae86ba741045ff3a6fff88	May 23, 2024	<b>DELETE</b>

+ CREATE CREDENTIALS

จากนั้นลองไปที่ <http://localhost:8000/posts> อีกรังแล้วลองใส่ Username และ Password ที่เราได้สร้างไว้



เมื่อ Sign in แล้วก็จะเข้าถึง <http://localhost:8000/posts> ตัวนี้ได้

```
pretty print □
[{"userId": 1,
 "id": 1,
 "title": "sunt aut facere repellat provident occaecati excepturi optio reprehenderit",
 "body": "quia et suscipit\\nscipit\\nrecusandae consequuntur expedita et cum\\nreprehenderit molestiae ut quas totam\\nnostrum rerum est autem sunt rem eveniet architecto"
},
 {"userId": 1,
 "id": 2,
 "title": "qui est esse",
 "body": "est rerum tempore vitae\\nsequi sint nihil reprehenderit dolor beatae ea dolores neque\\nfugiat blanditiis voluptate porro vel nihil molestiae ut reiciendis\\nqui aperiam non debitis possimus qui neque nisi nulla"
},
 {"userId": 1,
 "id": 3,
 "title": "ea molestias quasi exercitationem repellat qui ipsa sit aut",
 "body": "et iusto sed quo iure\\nvolutpatem occaecati omnis eligendi aut ad\\nvolutpatem doloribus vel accusantium quis parlatur\\nmolestiae porro eius odio et labore et velit aut"
},
 {"userId": 1,
 "id": 4,
 "title": "eum et est occaecati",
 "body": "ullam et saepe reiciendis voluptatem adipisci\\nsit amet autem assumenda provident rerum culpa\\nquis hic commodi nesciunt rem tenetur doloremque ipsam iure\\nquis sunt voluptatem rerum illo velit"
},
 {"userId": 1,
 "id": 5,
 "title": "nesciunt quas edipis",
 "body": "repudiandae veniam querat sunt sed\\nalias aut fugiat sit autem sed est\\nvolutpatem omnis possimus esse voluptatibus quis\\nest aut tenetur dolor neque"
},
 {"userId": 1,
 "id": 6,
 "title": "dolorem eum magni eos aperiam quia",
 "body": "ut aspernatur corporis harum nihil quis provident sequi\\nmollitia nobis aliquid molestiae\\nperspiciatis et ea nemo ab reprehenderit accusantium quas\\nvolutpatem dolores velit et doloremque molestiae"
},
 {"userId": 1,
 "id": 7,
 "title": "magnam facilis autem",
 "body": "dolore placeat quibusdam ea quo vitae\\nmagni quis enim qui quis quo nemo aut saepe\\nquidem repellat excepturi ut quia\\nsumt ut sequi eos ea sed quas"
},
 {"userId": 1,
 "id": 8,
 "title": "dolorem dolore est ipsam",
 "body": "dignissimos aperiam dolorem qui eum\\nfacilis quibusdam animi sint suscipit qui sint possimus cum\\nquerat magni maiores excepturi\\nipsam ut commodi dolor voluptatum modi aut vitae"
},
 {"userId": 1,
 "id": 9,
 "title": "nesciunt iure omnis dolorem tempora et accusantium",
 "body": "consectetur animi nesciunt iure dolore\\nveniam autem ut quam aut nobis\\net est aut quod aut provident voluptas autem voluptas"
},
 {"userId": 1,
 "id": 10,
 "title": "optio molestias id quia eum",
 "body": "qua et expedita modi cum officia vel magni\\ndoloribus qui repudiandae\\nvero nisi sit\\nquo veniam quod sed accusamus veritatis error"
}]
```

## DEMO – 2

ทำระบบ Monitoring Microservice ด้วย Kong, Prometheus และ Grafana

โดยใน Demo นี้เราจะมีโครงสร้างไฟล์ดังนี้

Kong\_dock

|--> docker-compose.yml

|--> kong.Dockerfile

|--> Prometheus.yml

### Code

โค้ดส่วนของไฟล์ docker-compose.yml

```
version: "3.7"
services:
  kong-database:
    image: postgres
    container_name: kong-postgres
    restart: on-failure
    volumes:
      - kong_data:/var/lib/postgresql/data
    environment:
      POSTGRES_USER: kong
      POSTGRES_PASSWORD: ${KONG_PG_PASSWORD:-kong}
      POSTGRES_DB: kong
      POSTGRES_HOST_AUTH_METHOD: trust
    ports:
      - "5432:5432"
  healthcheck:
    test: ["CMD", "pg_isready", "-U", "kong"]
    interval: 30s
    timeout: 30s
```

```
retries: 3
networks:
- default

kong-migration:
image: kong:latest
container_name: kong-migration
command: >
/bin/sh -c "
apt-get update && apt-get install -y postgresql-client &&
chown -R _apt /var/lib/apt/lists/partial &&
while ! pg_isready -h kong-database -U kong; do
sleep 1;
done;
kong migrations bootstrap
"
restart: on-failure
environment:
KONG_PG_HOST: kong-database
depends_on:
kong-database:
condition: service_healthy
networks:
- default

kong:
build:
context: .
dockerfile: kong.Dockerfile
environment:
KONG_DATABASE: postgres
KONG_PG_HOST: kong-database
KONG_PG_DATABASE: kong
KONG_PG_USER: kong
KONG_PG_PASSWORD: ${KONG_PG_PASSWORD:-kong}
```

```
KONG_PROXY_LISTEN: 0.0.0.0:8000, 0.0.0.0:8443 ssl
KONG_PROXY_LISTEN_SSL: 0.0.0.0:8443
KONG_ADMIN_LISTEN: 0.0.0.0:8001
KONG_STATUS_LISTEN: 0.0.0.0:8001
depends_on:
  - kong-database
  - kong-migration
healthcheck:
  test: ["CMD", "kong", "health"]
  interval: 10s
  timeout: 10s
  retries: 10
ports:
  - 8000:8000
  - 8001:8001
  - 8443:8443
  - 8444:8444
networks:
  - default

konga:
  container_name: konga
  image: pantsel/konga:latest
  ports:
    - 1337:1337
  networks:
    - default

prometheus:
  image: prom/prometheus:latest
  container_name: prometheus
  ports:
    - "9090:9090"
  volumes:
    - ./prometheus.yml:/etc/prometheus/prometheus.yml
```

```
- prometheus_data:/prometheus
command:
- "--config.file=/etc/prometheus/prometheus.yml"
networks:
- default

node_exporter:
image: prom/node-exporter:latest
container_name: node_exporter
expose:
- "9100:9100"
networks:
- default

grafana:
image: grafana/grafana
ports:
- "3000:3000"
external_links:
- prometheus
networks:
- webproxy
- default

networks:
webproxy:
external:
name: webproxy
default:
external:
name: kong_network

volumes:
kong_data: {}
prometheus_data: {}
```

หมายเหตุ โดยในโค้ดส่วนของ command ใน kong-migration นี้จะใช้เป็นคำสั่งของ bash, ubuntu ไม่สามารถรันใน powershell ได้ โดยของ powershell จะเป็นดังนี้

```
kong-migration:  
image: kong:latest  
container_name: kong-migration  
command: >  
powershell -Command "  
choco install postgresql --version 13.0.0 --yes --no-progress;  
while (!{Test-NetConnection -ComputerName kong-database -Port  
5432}.TcpTestSucceeded) { Start-Sleep -Seconds 1 };  
kong migrations bootstrap  
"  
restart: on-failure  
environment:  
  KONG_PG_HOST: kong-database  
depends_on:  
  kong-database:  
    condition: service_healthy  
networks:  
  - default
```

โค้ดส่วนของไฟล์ kong.Dockerfile

```
FROM kong:latest  
CMD ["sh", "-c", "kong migrations bootstrap && ./docker-entrypoint.sh kong docker-start"]  
USER kong
```

โค้ดส่วนของไฟล์ prometheus.yml

```
global:  
  scrape_interval: 15s  
  
scrape_configs:
```

```
- job_name: "prometheus"
  static_configs:
    - targets: ["localhost:9090"]

- job_name: "node_exporter"
  static_configs:
    - targets: ["node_exporter:9100"]

- job_name: "kong"
  scrape_interval: 5s
  static_configs:
    - targets: ["kong:8001"]
  metrics_path: /metrics
```

หรือจะใช้ Git เพื่อโคลนโปรเจคนี้มา

```
$ git clone https://github.com/POPPROJECT/Kong_Dock.git
```

หลังจากสร้างไฟล์ และใส่โค้ดทั้งหมดแล้วให้รันคำสั่งดังนี้

Step 1 สร้าง docker network

```
$ docker network create kong_network
$ docker network create webproxy
```

Step 2 รันคำสั่ง docker compose เพื่อ start โปรเจค

```
$ docker-compose up -d
```

```
TRIPOP@DESKTOP-59RR9SH MINGW64 /e/project_intern/Kong_dock
```

- \$ docker network create webproxy  
59878cd6259e3e62105e2c6d0ddef9064e139059e1d4ab357d4fb5a41d0a035b

```
TRIPOP@DESKTOP-59RR9SH MINGW64 /e/project_intern/Kong_dock
```

- \$ docker network create kong\_network  
45a158f9c37a2384753c1fd0070d3e2e54b6fbfae97016b8c024960990483705

[+] Running 9/9			
✓ Volume "kong_dock_kong_data"	Created	0.0s	
✓ Volume "kong_dock_prometheus_data"	Created	0.0s	
✓ Container kong_dock-grafana-1	Started	2.7s	
✓ Container kong-postgres	Healthy	33.2s	
✓ Container prometheus	Started	2.7s	
✓ Container konga	Started	2.7s	
✓ Container node_exporter	Started	2.5s	
✓ Container kong-migration	Started	32.7s	
✓ Container kong_dock-kong-1	Started	33.0s	

ถ้าดูจากใน docker จะเห็นว่า container Kong-migration นั้นหยุดทำงาน เพราะว่า Kong-migration หยุดทำงานโดยแสดงรหัส 0 นั้นแสดงว่าการทำงานเสร็จสมบูรณ์ โปรแกรมจะจบการทำงานและหยุดทำงาน

kong_dock		Running (6/7)	47.47%	42 seconds ago	⋮	⋮	⋮
└─ konga	f1df56fe1d24	pantsel/konga:latest	Running	21.55% 1337:1337	1 minute ago	⋮	⋮
└─ grafana-1	8486f28fd70	grafana/grafana	Running	0.04% 3000:3000	1 minute ago	⋮	⋮
└─ prometheus	2228f92e11dd	prom/prometheus:latest	Running	0% 9090:9090	1 minute ago	⋮	⋮
└─ node_exporter	ac5fd440ee60	prom/node-exporter:latest	Running	0%	1 minute ago	⋮	⋮
└─ kong-postgres	cadac4cde2eb	postgres	Running	8.57% 5432:5432	1 minute ago	⋮	⋮
└─ kong-migration	6c0af7b6869b	kong:latest	Exited	0%	42 seconds ago	▶	⋮
└─ kong-1	a82c73d01de9	kong_dock-kong	Running	17.31% 8000:8000	42 seconds ago	⋮	⋮

kong-migration exited with code 0

ต่อมาให้เราไปที่ <http://localhost:1337> ทำการสมัครสมาชิก และทำการล็อกอินเพื่อเข้าหน้า dashboard ของตัว Konga

ให้กรอกข้อมูลในส่วนของ Name (จะตั้งชื่ออะไรก็ได้) และ Kong Admin URL (<http://kong:8001>)

KONGA

DASHBOARD

APPLICATION

USERS

CONNECTIONS

SNAPSHOTS

SETTINGS

Welcome!

First of all, let's setup a connection to Kong Admin.

Select a connection type.

DEFAULT KEY AUTH JWT AUTH BASIC AUTH

Kong will connect directly to Kong's admin API. This method is mainly suitable for demo scenarios or internal access (ex: localhost). Kong's admin API **should not** be publicly exposed.

Name\* MyAPI

Kong Admin URL\* http://kong:8001

✓ CREATE CONNECTION

KONGA 0.1.6 GitHub Issues Support the project Connected to N/A

เมื่อกรอกข้อมูลเสร็จกด CREATE CONNECTION ก็จะเข้ามาที่หน้า Dashboard

The screenshot shows the KONGA API Gateway Dashboard. On the left, there's a sidebar with various menu items: DASHBOARD, API GATEWAY (INFO, SERVICES, ROUTES, CONSUMERS, PLUGINS, UPSTREAMS, CERTIFICATES), APPLICATION (USERS, CONNECTIONS, SNAPSHOT, SETTINGS). The main area is titled 'CONNECTIONS' and displays the following metrics: ACTIVE (11), READING (0), WRITING (10), WAITING (1), ACCEPTED (24), and HANDLED (24). Below these metrics are three cards: 'NODE INFO' (HostName: a82c73d01de9, Tag Line: Welcome to kong, Version: 3.7.0, LUA Version: LuaJIT 2.1.0-20231117, Admin listen: ["0.0.0.0:8001"]), 'TIMERS' (a chart showing Pending and Running tasks over time), and 'DATASTORE INFO' (DBMS: postgres, Host: kong-database, Database: kong, User: kong, Port: 5432). At the bottom of the dashboard, there are links for GitHub, Issues, and Support the project, along with a note 'Connected to MyAPI'.

## กำหนด Service

ในตัวอย่างนี้จะใช้ข้อมูล JSON Placeholder API ของ <https://jsonplaceholder.typicode.com/> ซึ่งเป็น Free fake data โดยเราจะใช้ Konga สร้าง Service ออกมานะ

ต่อมาเราไปสร้าง SERVICES โดยไปที่ SERVICES --> ADD NEW SERVICE

The screenshot shows the 'Services' page in KONGA. The sidebar on the left has the 'SERVICES' item highlighted with a red box. In the center, there's a button labeled '+ ADD NEW SERVICE' also highlighted with a red box. A search bar and a results dropdown are visible at the top right. Below the button is a table with columns: NAME, HOST, TAGS, and CREATED. The table currently shows 25 results. At the bottom of the page, there are links for GitHub, Issues, and Support the project, along with a note 'Connected to MyAPI'.

จะเข้ามาที่หน้า CREATE SERVICE โดยกรอกข้อมูลดังนี้

Name : JSON-Placeholder  
Description : API Resource of json placeholder (จะใส่หรือไม่ใส่ก็ได้)  
Protocol : https  
Host : jsonplaceholder.typicode.com  
Port : 443  
Path : /  
Retries : 5  
Connect timeout : 60000  
Write timeout : 60000  
Read timeout : 60000

**CREATE SERVICE**

Name <i>(optional)</i>	JSON-Placeholder The service name.
Description <i>(optional)</i>	API Resource of json placeholder An optional service description.
Tags <i>(optional)</i>	Optionally add tags to the service
Url <i>(shorthand-attribute)</i>	Shorthand attribute to set <code>protocol</code> , <code>host</code> , <code>port</code> and <code>path</code> at once. This attribute is write-only (the Admin API never "returns" the url).
Protocol <i>(semi-optional)</i>	https The protocol used to communicate with the upstream. It can be one of <code>http</code> or <code>https</code> .
Host <i>(semi-optional)</i>	jsonplaceholder.typicode.com The host of the upstream server.
Port <i>(semi-optional)</i>	443 The upstream server port. Defaults to <code>80</code> .
Path <i>(optional)</i>	/ The path to be used in requests to the upstream server. Empty by default.
Retries <i>(optional)</i>	5 The number of retries to execute upon failure to proxy. The default is <code>5</code> .
Connect timeout <i>(optional)</i>	60000 The timeout in milliseconds for establishing a connection to your upstream server. Defaults to <code>60000</code> .
Write timeout <i>(optional)</i>	60000 The timeout in milliseconds between two successive write operations for transmitting a request to the upstream server. Defaults to <code>60000</code> .
Read timeout <i>(optional)</i>	60000 The timeout in milliseconds between two successive read operations for transmitting a request to the upstream server. Defaults to <code>60000</code> .
Client certificate <i>(optional)</i>	Certificate ( <code>id</code> ) to be used as client certificate while TLS handshaking to the upstream server.
<b>✓ SUBMIT SERVICE</b>	

เมื่อคุณ SUBMIT SERVICE แล้วก็จะได้ Service ที่เราสร้างออกมานะ

The screenshot shows the KONGA API Gateway dashboard. On the left, there's a sidebar with various navigation options: DASHBOARD, API GATEWAY, INFO, SERVICES (which is selected and highlighted in blue), ROUTES, CONSUMERS, PLUGINS, UPSTREAMS, CERTIFICATES, APPLICATION, USERS, CONNECTIONS, SNAPSHOT, and SETTINGS. The main content area is titled "Services". It contains a sub-header: "Service entities, as the name implies, are abstractions of each of your own upstream services. Examples of Services would be a data transformation microservice, a billing API, etc.". Below this is a search bar with a magnifying glass icon and the placeholder "search...", and a dropdown menu showing "Results: 25". A green button labeled "+ ADD NEW SERVICE" is at the top left of the list. The list itself has columns: NAME, HOST, TAGS, and CREATED. There is one entry: "JSON-Placeholder" with the host "jsonplaceholder.typicode.com", created on "Jun 7, 2024", and a red "DELETE" button. At the bottom of the page, there are links for "KONGA 0.14.9", "GitHub", "Issues", and "Support the project", along with a note "Connected to MyAPI".

## สร้าง Route

กดเข้าไปที่ตัว Service ที่เราสร้าง Routes --> ADD ROUTE

The screenshot shows the KONGA API Gateway dashboard. The sidebar is identical to the previous screenshot. The main content area is titled "Service JSON-Placeholder" and shows the URL "services / show". Under "Service Details", there are three tabs: "Routes" (selected and highlighted in red), "Plugins", and "Eligible consumers beta". A red box highlights the "Routes" tab. To the right, there's a section titled "Routes" with a red box highlighting the "+ ADD ROUTE" button. Below this is a table with columns: Name / ID, Hosts, Paths, Protocols, Methods, Regex priority, and Created. The table currently displays "no data found...". At the bottom of the page, there are links for "KONGA 0.14.9", "GitHub", "Issues", and "Support the project", along with a note "Connected to MyAPI".

โดยในตัวอย่างนี้เราจะใช้เป็น Users Route (/users)

กำหนดค่าดังนี้

Name : Users

Path : /users (หลังระบุเสร็จแล้วให้กด Enter ที่ช่อง Path)

Path handling : v1

Https redirect status code : 426

Regex priority : 0

Strip Path : NO

Protocols : http, https

**ADD ROUTE TO JSON-PLACEHOLDER** X

\* For hosts, paths, methods and protocols, snis, sources, headers and destinations press enter to apply every value you type

**Name**  (optional)  
The name of the Route.

**Tags**  (optional)  
Optionally add tags to the route

**Hosts**  (semi-optional)  
A list of domain names that match this Route. For example: example.com. At least one of hosts, paths, or methods must be set.

**Paths**  (semi-optional)  
A list of paths that match this Route. For example: /my-path. At least one of hosts, paths, or methods must be set.

**Headers**  (semi-optional)  
One or more lists of values indexed by header name that will cause this Route to match if present in the request. The Host header cannot be used with this attribute: hosts should be specified using the hosts attribute.  
Field values format example: x-some-header:foo,bar

**Path handling**   
Controls how the Service path, Route path and requested path are combined when sending a request to the upstream. See above for a detailed description of each behavior. Accepted values are: "v0", "v1". Defaults to "v1".

**Https redirect status code**  (optional)  
The status code Kong responds with when all properties of a Route match except the protocol, i.e. if the protocol of the request is `HTTP` instead of `HTTPS`, `Location` header is injected by Kong if the field is set to 301, 302, 307 or 308. Defaults to 426.

**Regex priority**  (optional)  
A number used to choose which route resolves a given request when several routes match it using regexes simultaneously. When two routes match the path and have the same regex\_priority, the older one (lowest created\_at) is used. Note that the priority for non-regex routes is different (longer non-regex routes are matched before shorter ones). Defaults to 0.

**Methods**  (semi-optional)  
A list of HTTP methods that match this Route. At least one of hosts, paths, or methods must be set.

**Strip Path**  (optional)  
When matching a Route via one of the paths, strip the matching prefix from the upstream request URL.

**Preserve Host**  (optional)  
When matching a Route via one of the hosts domain names, use the request Host header in the upstream request headers. By default set to false, and the upstream Host header will be that of the Service's host.

**Protocols**   (semi-optional)

เมื่อคุณ SUBMIT ROUTE แล้วเราจะได้ Route ที่เราสร้างออกมานะ

The screenshot shows the KONGA API Gateway interface. On the left, there's a sidebar with various navigation options: DASHBOARD, API GATEWAY, INFO, SERVICES, ROUTES, CONSUMERS, PLUGINS, UPSTREAMS, CERTIFICATES, APPLICATION, USERS, CONNECTIONS, SNAPSHTOS, and SETTINGS. The 'ROUTES' option is currently selected. The main content area is titled 'Service JSON-Placeholder' and shows the 'Routes' tab selected. It displays a table with one row of data:

Name / ID	Hosts	Paths	Protocols	Methods	Regex priority	Created	Actions
Users	/users	http, https	-	0	0	Jun 7, 2024	<span>EDIT</span> <span>DELETE</span>

At the bottom of the main content area, there are links for 'KONGA 0.14.9', 'GitHub', 'Issues', and 'Support the project'. A status bar at the bottom right indicates 'Connected to MyAPI'.

เราจะสามารถเข้าดู Route ที่เราสร้างได้ที่ <http://localhost:8000/path> (path ก็คือ path ที่เรากำหนดไว้) โดยที่ตัวอย่างสร้างไว้จะเป็น path : users --> <http://localhost:8000/users>

The screenshot shows a browser window displaying the JSON response for the '/users' endpoint. The URL in the address bar is 'localhost:8000/users'. The response is a list of user objects:

```
[{"id": 1, "name": "Leanne Graham", "username": "Bret", "email": "Sincere@april.biz", "address": {"street": "Kulas Light", "suite": "Apt. 556", "city": "Gwenborough", "zipcode": "92998-3874", "geo": {"lat": "-37.3159", "lng": "81.4968"}}, "phone": "1-770-736-8031 x56422", "website": "hildegard.org"}, {"company": {"name": "Romonera-Crona", "catchPhrase": "Multi-layered client-server neural-net", "bs": "harness real-time e-markets"}}, {"id": 2, "name": "Ervin Howell", "username": "Antonette", "email": "Shanna@alissa.tv", "address": {"street": "Victor Plains", "suite": "Suite 879", "city": "Wisokyburgh", "zipcode": "96566-7711", "geo": {"lat": "-43.9599", "lng": "-34.4618"}}, "phone": "010-692-6593 x09125", "website": "anastasia.net"}, {"company": {"name": "Deckow-Crist", "catchPhrase": "Proactive didactic contingency", "bs": "synergize scalable supply-chains"}}, {"id": 3, "name": "Clementine Bauch", "username": "Samantha", "email": "Natalie@kian.com"}]
```

## เพิ่ม Plugin

เราจะสามารถเลือกได้ว่าจะเพิ่ม plugins ให้กับตัว service (ทุก route ที่อยู่ใน service จะใช้ plugin ที่เพิ่มมา) หรือ route ที่เราสร้างได้

โดยตัวอย่างนี้จะใช้แค่ Plugins Rate Limiting เพิ่มให้กับ Route : Users

เข้าไปที่ Routes --> Users (route ที่เราสร้าง) --> Plugins --> ADD PLUGIN

The screenshot shows the KONGA API Gateway interface. On the left, there's a sidebar with various navigation options like Dashboard, API GATEWAY, INFO, SERVICES, ROUTES, CONSUMERS, PLUGINS, UPSTREAMS, CERTIFICATES, APPLICATION, USERS, CONNECTIONS, SNAPSHOT, and SETTINGS. The main area is titled 'Route Users' and shows a list of assigned plugins. At the top right, there's a user profile and a 'Hello, tripop' message. Below the title, it says 'routes / route'. Under 'Route Details', the 'Plugins' tab is selected (marked with a red box and number 1). In the main content area, there's a section titled 'Assigned plugins' with a search bar and a table header for 'Name', 'Consumer', and 'Created'. A red box highlights the '+ ADD PLUGIN' button at the top right of this section (marked with a red box and number 2). At the bottom, there are links for 'KONGA 0.14.9', 'GitHub', 'Issues', 'Support the project', and a note 'Connected to MyAPI'.

ในหน้า ADD PLUGIN ไปที่ Traffic Control --> Rate Limiting กด ADD PLUGIN

The screenshot shows a modal window titled 'ADD PLUGIN' for the 'TRAFFIC CONTROL' section. On the left, there's a sidebar with 'Authentication', 'Security', 'Traffic Control' (which is selected and highlighted in green), 'Serverless', 'Analytics & Monitoring', 'Transformations', 'Logging', and 'Other'. The main area has five sections, each with an 'ADD PLUGIN' button: 'Rate Limiting' (illustrated with a graph icon), 'Response Ratelimiting' (illustrated with a double-headed arrow icon), 'Request Size Limiting' (illustrated with a funnel icon), 'Request Termination' (illustrated with a stop sign icon), and 'Proxy Cache' (illustrated with a network icon). Each section also has a brief description below its icon.

เราจะสามารถกำหนดได้ว่าจะสามารถให้เข้า path : /users ได้กี่ครั้งโดยรายละเอียดตามนี้

Second : กี่ครั้ง/นาที

Minute : กี่ครั้ง/นาที

Hour : กี่ครั้ง/ชั่วโมง

Day : กี่ครั้ง/วัน

Month : กี่ครั้ง/เดือน

Year : กี่ครั้ง/ปี

โดยครั้งนี้จะกำหนดเป็น 10 ครั้ง/นาที (ใน 1 นาทีจะเข้าได้ 10 ครั้ง)

**EDIT RATE LIMITING**

Rate limit how many HTTP requests a developer can make in a given period of seconds, minutes, hours, days, months or years. If the API has no authentication layer, the Client IP address will be used, otherwise the Consumer will be used if an authentication plugin has been configured.

**consumer**

The CONSUMER ID that this plugin configuration will target. This value can only be used if authentication has been enabled so that the system can identify the user making the request. If left blank, the plugin will be applied to all consumers.

**second**

10

The amount of HTTP requests the developer can make per second. At least one limit must exist.

**minute**

10

The amount of HTTP requests the developer can make per minute. At least one limit must exist.

**hour**

10

The amount of HTTP requests the developer can make per hour. At least one limit must exist.

**day**

10

The amount of HTTP requests the developer can make per day. At least one limit must exist.

**month**

10

The amount of HTTP requests the developer can make per month. At least one limit must exist.

**year**

10

The amount of HTTP requests the developer can make per year. At least one limit must exist.

**limit by**

consumer

The entity that will be used when aggregating the limits: consumer, credential, ip. If the consumer or the credential cannot be determined, the system will always fallback to ip.

## กด SUBMIT

The screenshot shows the Konga API Gateway interface. On the left, there's a dark sidebar with various navigation options like Dashboard, API Gateway, Info, Services, Routes (which is selected), Consumers, Plugins, Upstreams, Certificates, Application, Users, Connections, Snapshots, and Settings. The main content area has a header 'Route Users' with a breadcrumb 'routes / route'. Below this, there are two tabs: 'Route Details' (selected) and 'Plugins'. Under 'Route Details', there's a note 'Eligible consumers beta'. Under 'Plugins', there's a section titled 'Assigned plugins' with a search bar 'search plugins...'. A table lists one plugin: 'rate-limiting' by 'All consumers' created on 'Jun 7, 2024'. There's a red 'DELETE' button next to it. At the bottom of the main content area, there are links for 'KONGA 0.14.9', 'GitHub', 'Issues', 'Support the project', and a note 'Connected to MyAPI'.

ถ้าหากเราเข้า <http://localhost:8000/users> เกิน 10 ครั้งใน 1 นาทีก็จะแจ้ง ERROR อกกมา

The screenshot shows a browser window with the URL 'localhost:8000/users'. The page title is 'Error'. The content of the page is: 'API rate limit exceeded.' followed by 'request\_id: d149c609ae558fd0a00936a6ffcf152'. The browser's address bar also shows 'localhost:8000/users'.

หรือจะใช้เป็น Postman เพื่อดูว่าเรายังสามารถดู Rate Limiting ได้ว่าเหลือเท่าไหร'

The screenshot shows the Postman interface with a successful API call to `localhost:8000/users`. The Headers tab is active, displaying the following key-value pairs:

Key	Value
X-RateLimit-Remaining-Minute	8
RateLimit-Limit	10
RateLimit-Remaining	8
RateLimit-Reset	37
X-RateLimit-Limit-Minute	10

## Prometheus + Grafana

เราจะใช้ Prometheus จัดเก็บข้อมูลต่างๆ ของ Cloud Server และ Service ในแบบ Time Series และใช้ Grafana สำหรับดึงข้อมูลมาแสดงบน Real Time Dashboard

ก่อนอื่นเราจะไป ADD Plugin ใน Konga เพื่อ Monitor Consumer

ไปที่เมนู CONSUMERS --> CREATE CONSUMER

The screenshot shows the Konga UI with the Consumers page open. A new consumer is being created, with the username set to "tripop". The page includes a search bar and a results dropdown.

โดยเราสามารถกรอก username ได้ตามต้องการ

### CREATE CONSUMER

username  
(semi-optional)

DevOps

The username of the consumer. You must send either this field or `custom_id` with the request.

custom\_id  
(semi-optional)

Field for storing an existing ID for the consumer, useful for mapping Kong with users in your existing database. You must send either this field or `username` with the request.

Tags  
(optional)

Optional tags for the consumer.

✓ SUBMIT CONSUMER

กด SUBMIT CONSUMER

The screenshot shows the Konga dashboard with the Consumers section selected. A new consumer named "DevOps" has been created and is listed in the table. The table columns are USERNAME, CUSTOM\_ID, TAGS, and CREATED. The "DevOps" entry has a timestamp of Jun 07 2024 @14:28 and a red "DELETE" button.

เมื่อเข้ามาที่ Consumer ที่เราสร้างก็จะได้รายละเอียดหน้าตาดังนี้

The screenshot shows the Konga dashboard with the Consumers section selected. The consumer "DevOps" is being viewed. The consumer details page includes tabs for Details, Groups, Credentials, Accessible Routes, and Plugins. The "Details" tab is active, showing the "username" field with "DevOps". Below it is the "custom\_id" field, which is currently empty. There is also a "Tags" field with an optional note. A "SUBMIT CHANGES" button is at the bottom.

โดยรายละเอียดในส่วนของ CONSUMER ก็จะมีตามนี้

Details : แสดงรายละเอียดของผู้ใช้

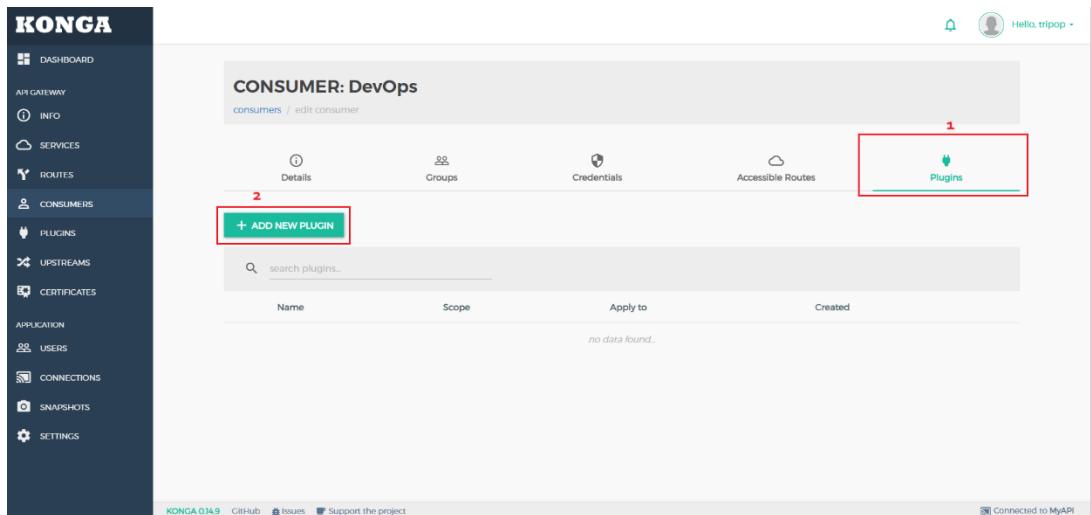
Groups : อนุญาตให้กำหนดผู้ใช้ในกลุ่มต่างๆ

Credentials : ใช้สำหรับจัดการข้อมูลประจำตัวของผู้ใช้

Accessible Routes : แสดง route ที่ผู้ใช้สามารถเข้าถึงได้

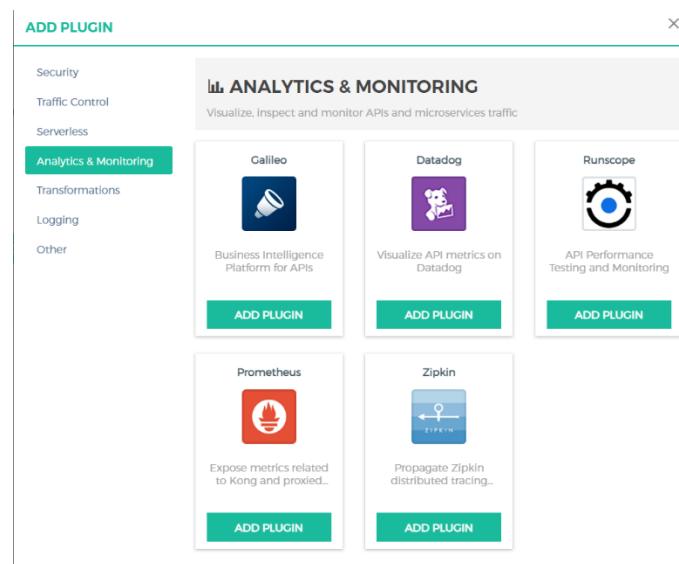
Plugins : จัดการ plugin ที่เกี่ยวข้องกับผู้ใช้นี้

เมื่อเราเข้ามาที่ Consumer ที่เราสร้างก็ไปที่ Plugins —> ADD NEW PLUGIN



The screenshot shows the Konga API Gateway interface. On the left, there's a sidebar with various navigation options like Dashboard, API Gateway, Info, Services, Routes, Consumers, Plugins, Upstreams, Certificates, Application, Users, Connections, Snapshots, and Settings. The main area is titled 'CONSUMER: DevOps' under 'consumers / edit consumer'. It has tabs for Details, Groups, Credentials, Accessible Routes, and Plugins (which is highlighted with a red box labeled '1'). Below these tabs is a search bar and a table with columns Name, Scope, Apply to, and Created. A red box labeled '2' highlights the '+ ADD NEW PLUGIN' button. At the bottom, there's a footer with 'KONGA 0.14.9', 'GitHub', 'Issues', 'Support the project', and a note 'Connected to MyAPI'.

เลือก Analytics & Monitoring —> Prometheus (ADD PLUGIN) และกด ADD PLUGIN อีกครั้งโดยที่ไม่ config plugin



The screenshot shows the 'ADD PLUGIN' dialog box. On the left, there's a sidebar with categories: Security, Traffic Control, Serverless, Analytics & Monitoring (selected and highlighted in green), Transformations, Logging, and Other. The main area is titled 'ANALYTICS & MONITORING' and contains five plugin options: Galileo, DataDog, Runscope, Prometheus, and Zipkin. Each option has a small icon, a brief description, and a green 'ADD PLUGIN' button below it.

ນີ້ອກດີ ADD PLUGIN ເສື່ອງກີ່ຈະໄດ້ plugin Prometheus ມາແລ້ວ

KONGA

CONSUMER: DevOps

+ ADD NEW PLUGIN

Name	Scope	Apply to	Created
prometheus	global	All Entrypoints	Jun 7, 2024

DELETE

## Prometheus

ເຂົ້າ Prometheus ຜ່ານ <http://localhost:9090>

Prometheus

Alerts Graph Status Help

Use local time  Enable query history  Enable autocomplete  Enable highlighting  Enable linter

Expression (press Shift+Enter for newlines)

Graph

Evaluation time

No data queried yet

Add Panel

ໄປທີ target ເພື່ອດູວວ່າ Endpoint ຂອງເຮົາມີ error ອະໄຣ້ໂກ

The screenshot shows the Prometheus web interface with the 'Status' dropdown menu open, highlighting the 'Targets' option. The main panel displays a table with no data queried yet. A status bar at the bottom indicates the URL is localhost:9090/targets.

ໜ້າ target

The screenshot shows the 'Targets' page in the Prometheus interface. It lists three healthy targets:

- kong (1/1 up)**:  
Endpoint: http://kong:8001/metrics, State: UP, Labels: instance="kong:8001", job="kong", Last Scrape: 7.453s ago, Scrape Duration: 7.191ms.
- node\_exporter (1/1 up)**:  
Endpoint: http://node\_exporter:9100/metrics, State: UP, Labels: instance="node\_exporter:9100", job="node\_exporter", Last Scrape: 5.748s ago, Scrape Duration: 48.750ms.
- prometheus (1/1 up)**:  
Endpoint: http://localhost:9090/metrics, State: UP, Labels: instance="localhost:9090", job="prometheus", Last Scrape: 193.000ms ago, Scrape Duration: 3.775ms.

ถ้าหากว่า target ของ node\_exporter หรือ kong เกิด error ให้ลองแก้ด้านในไฟล์ Prometheus.yml

```
global:
  scrape_interval: 15s

scrape_configs:
  - job_name: "prometheus"
    static_configs:
      - targets: ["localhost:9090"]

  - job_name: "node_exporter"
    static_configs:
      - targets: ["node_exporter:9100"]

  - job_name: "kong"
    static_configs:
      - targets: ["kong:8001"]
```

เปลี่ยนมาใช้ inet (localhost ของเรา) ให้เป็นดังนี้

```
global:
  scrape_interval: 15s

scrape_configs:
  - job_name: "prometheus"
    static_configs:
      - targets: ["inet [REDACTED]:9090"]

  - job_name: "node_exporter"
    static_configs:
      - targets: ["inet [REDACTED].4.9100"]

  - job_name: "kong"
    static_configs:
      - targets: ["kong:8001"]
```

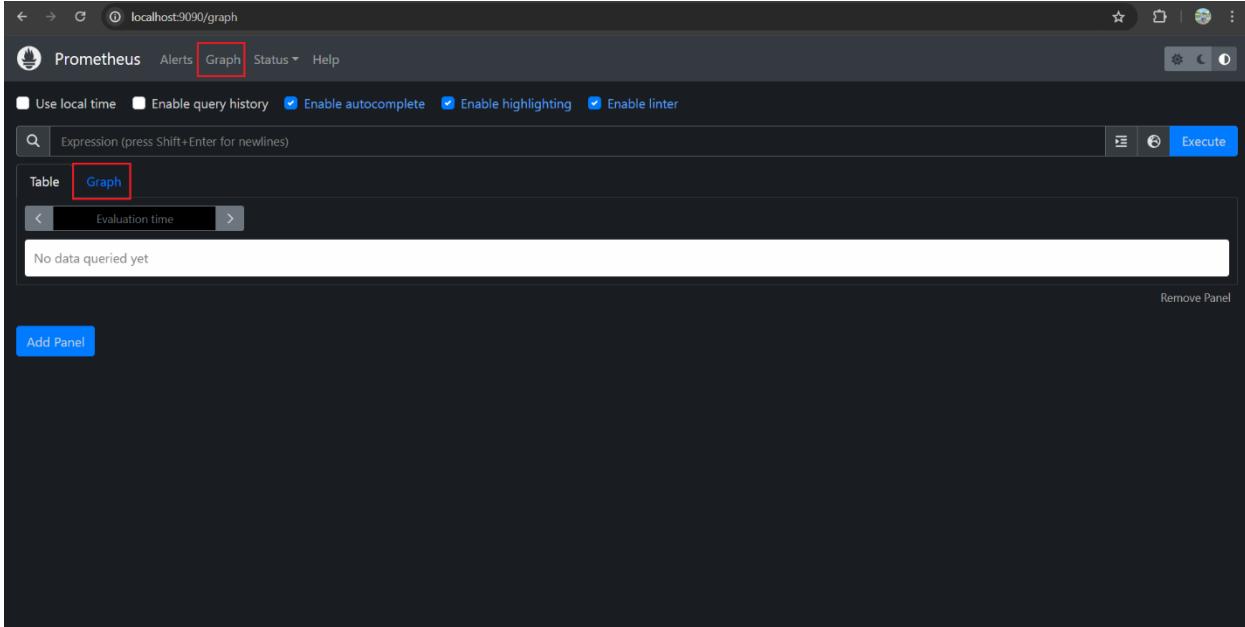
โดยวิธีดูก็คือ เปิด Ubuntu พิมพ์คำสั่ง

```
$ ifconfig
```

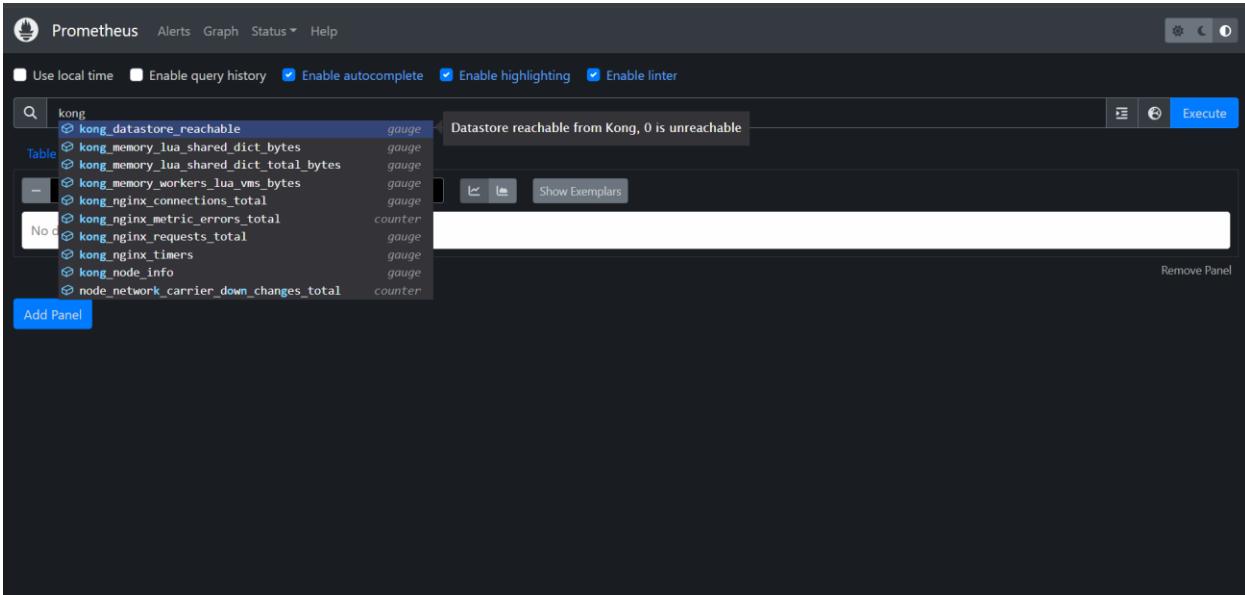
```
student@DESKTOP-59RR9SH:~$ ifconfig
eth0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
  inet [REDACTED]  netmask 255.255.240.0  broadcast [REDACTED]
    ether 00:15:5d:0c:ac:55  txqueuelen 1000  (Ethernet)
      RX packets 17419  bytes 23840299 (23.8 MB)
      RX errors 0  dropped 0  overruns 0  frame 0
      TX packets 4624  bytes 442357 (442.3 KB)
      TX errors 0  dropped 0  overruns 0  carrier 0  collisions 0

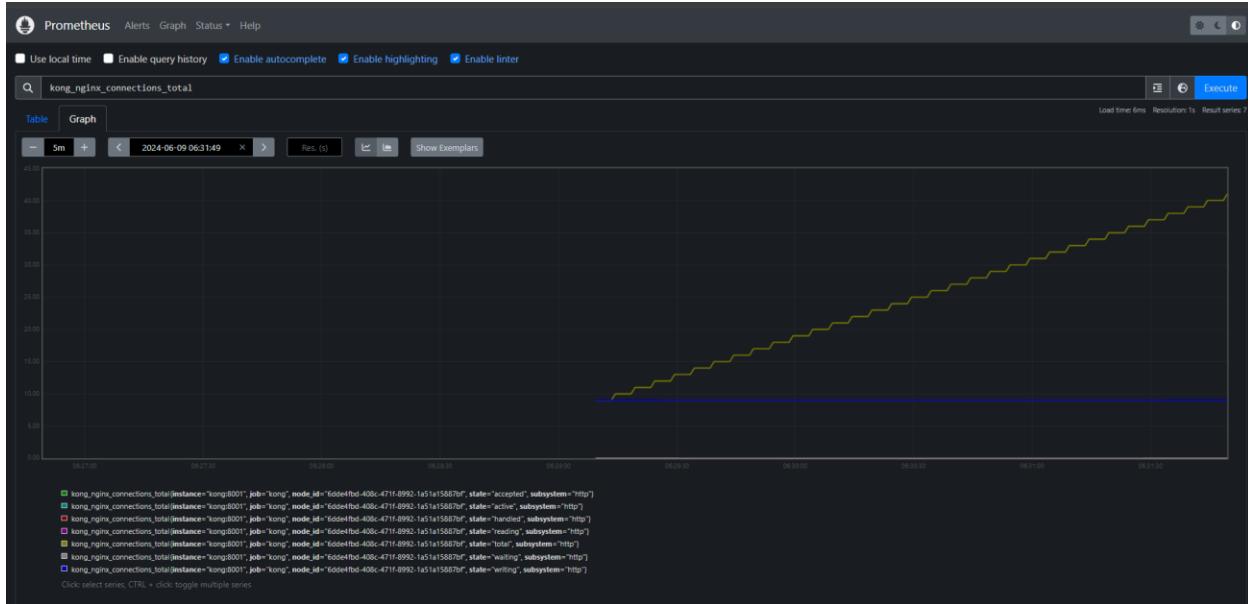
lo: flags=73<UP,LOOPBACK,RUNNING>  mtu 65536
  inet 127.0.0.1  netmask 255.0.0.0
    loop  txqueuelen 1000  (Local Loopback)
      RX packets 4802  bytes 4511567 (4.5 MB)
      RX errors 0  dropped 0  overruns 0  frame 0
      TX packets 4802  bytes 4511567 (4.5 MB)
      TX errors 0  dropped 0  overruns 0  carrier 0  collisions 0
```

## ໄປໄສ Graph



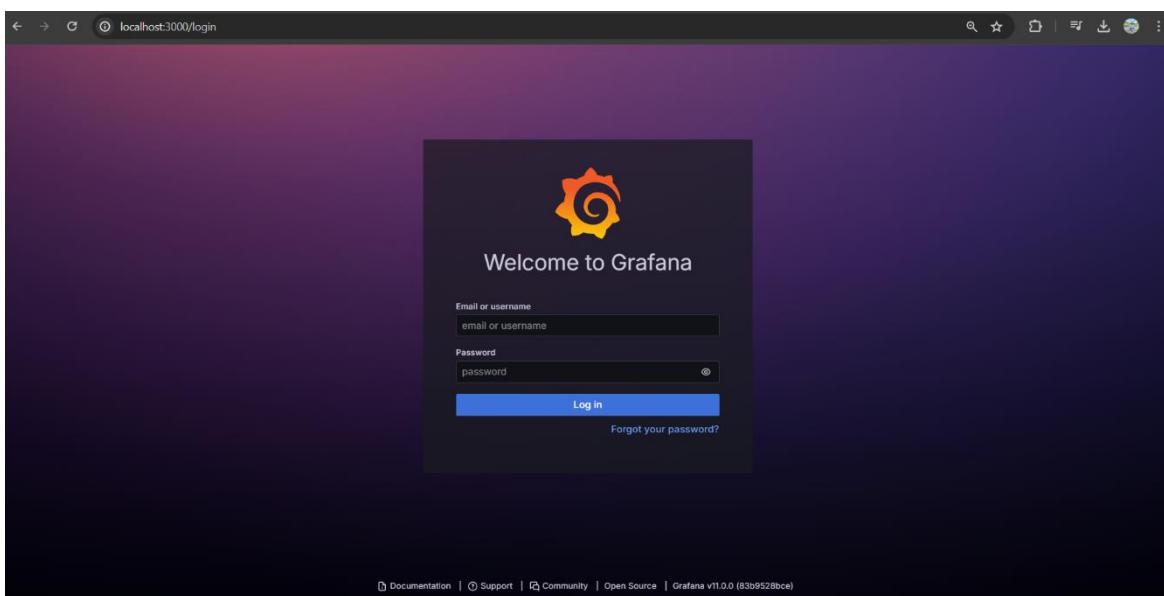
ໂດຍໃນຂ່ອງ Expression ເຮັດວຽກ search ເພື່ອຄູ່ຄ່າຕ່າງໆຂອງ Kong ໄດ້



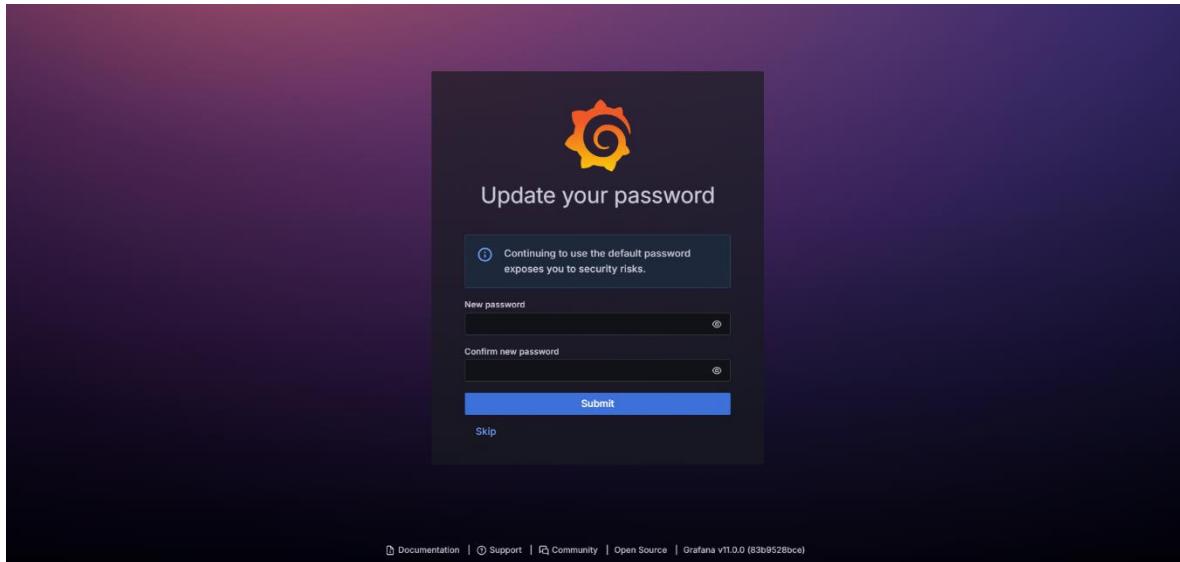


## Grafana

Login เข้า Grafana ได้ที่ <http://localhost:3000> โดย username : admin และ password : admin



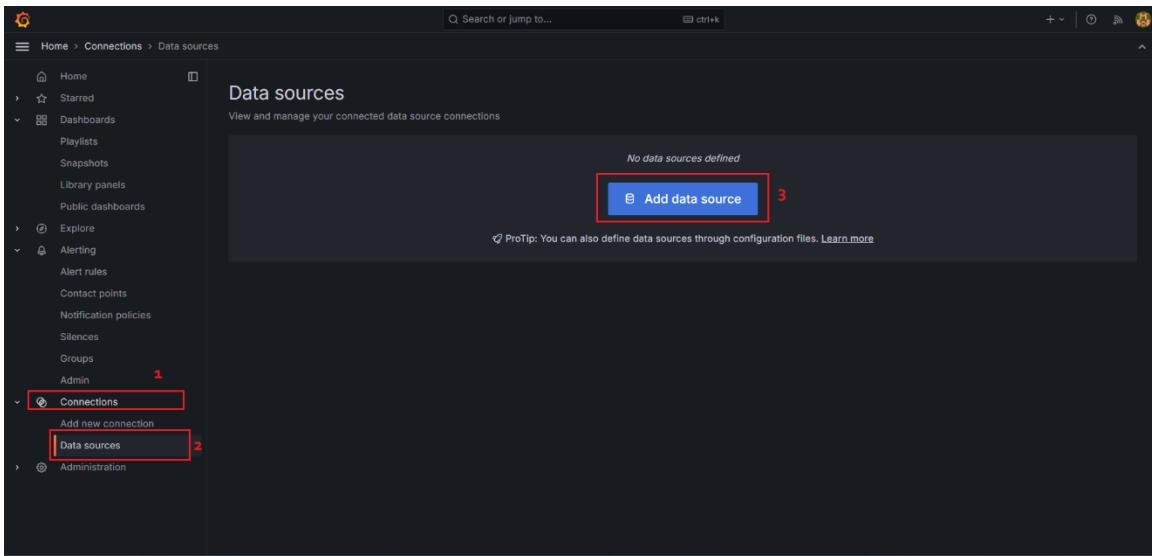
เราสามารถเปลี่ยน password ใหม่ หรือจะ skip ไปได้



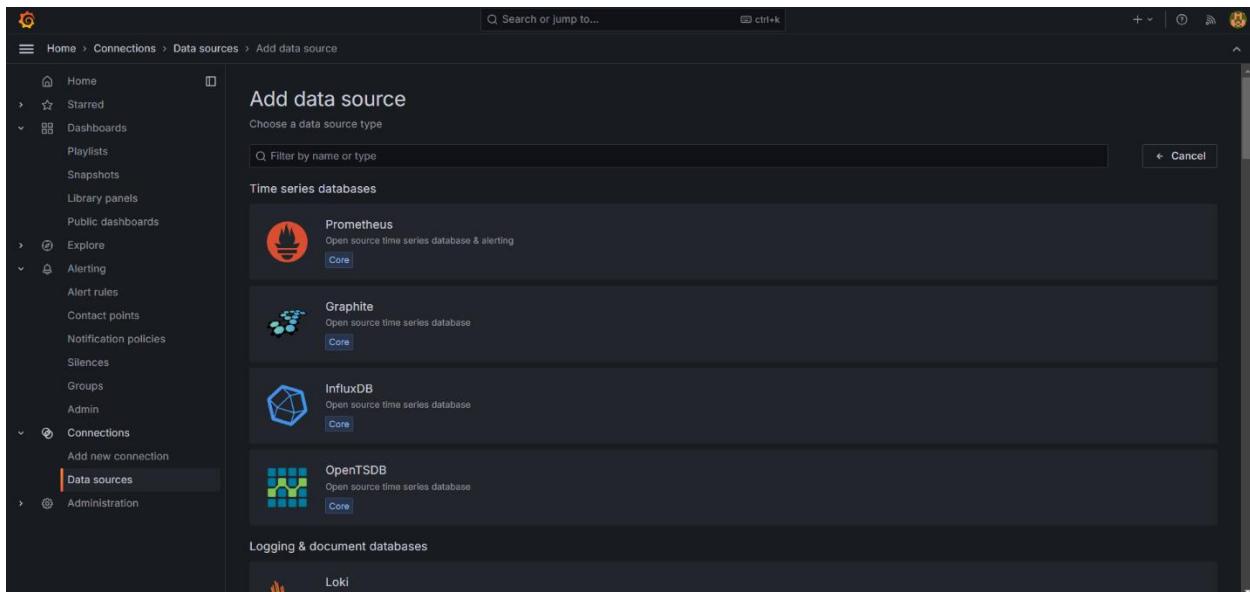
เมื่อ login เข้ามาแล้วก็จะเข้าหน้าที่หน้าหลักของ Grafana

A screenshot of the Grafana home dashboard. On the left is a sidebar with navigation links: Home, Starred, Dashboards, Playlists, Snapshots, Library panels, Public dashboards, Explore, Alerting, Alert rules, Contact points, Notification policies, Silences, Groups, Admin, Connections, Data sources, and Administration. The main area has a header "Welcome to Grafana" and "Need help? Documentation Tutorials Community Public Slack". It features several cards: "Basic" (Tutorial, DATA SOURCE AND DASHBOARDS, Grafana fundamentals), "DATA SOURCES" (Add your first data source, Learn how in the docs), "DASHBOARDS" (Create your first dashboard, Learn how in the docs), "Dashboards" (Starred dashboards, Recently viewed dashboards), and "Latest from the blog" (Snowflake data visualization: all the latest features to monitor metrics, enhance security, and more). The dashboard has a dark theme with light-colored cards.

ໄປທ໌ Connections --> Data sources --> Add data source



ເລືອກ Prometheus



ตรงส่วน Connection ใส่ URL : <http://prometheus:9090>

The screenshot shows the 'Settings' tab for a 'prometheus' data source in Grafana. At the top, there's a note to 'Configure your Prometheus data source below' and an option to skip configuration for a managed service. Below this, the 'Name' is set to 'prometheus' and the 'Default' toggle is on. A note says 'Before you can use the Prometheus data source, you must configure it below or in the config file. For detailed instructions, [view the documentation](#)'. Under the 'Connection' section, the 'Prometheus server URL' is set to 'http://prometheus:9090'. The 'Authentication' section is collapsed.

ในส่วนของ Other เลือก HTTP method เป็น GET และกด save & test

The screenshot shows the 'Settings' tab for the same 'prometheus' data source. In the 'Other' section, the 'HTTP method' is set to 'GET'. A success message at the bottom states 'Successfully queried the Prometheus API.' and suggests building a dashboard or querying the Explore view. At the bottom, there are 'Delete' and 'Save & test' buttons.

เราจะได้ Data sources ของเราที่ตั้งจาก Prometheus มา

The screenshot shows the Grafana interface with the sidebar navigation open. The 'Connections' section is selected, and 'Data sources' is highlighted. A single data source entry for 'prometheus' is listed, connected to 'Prometheus | http://prometheus:9090' with the 'default' role. There are buttons for 'Build a dashboard' and 'Explore'.

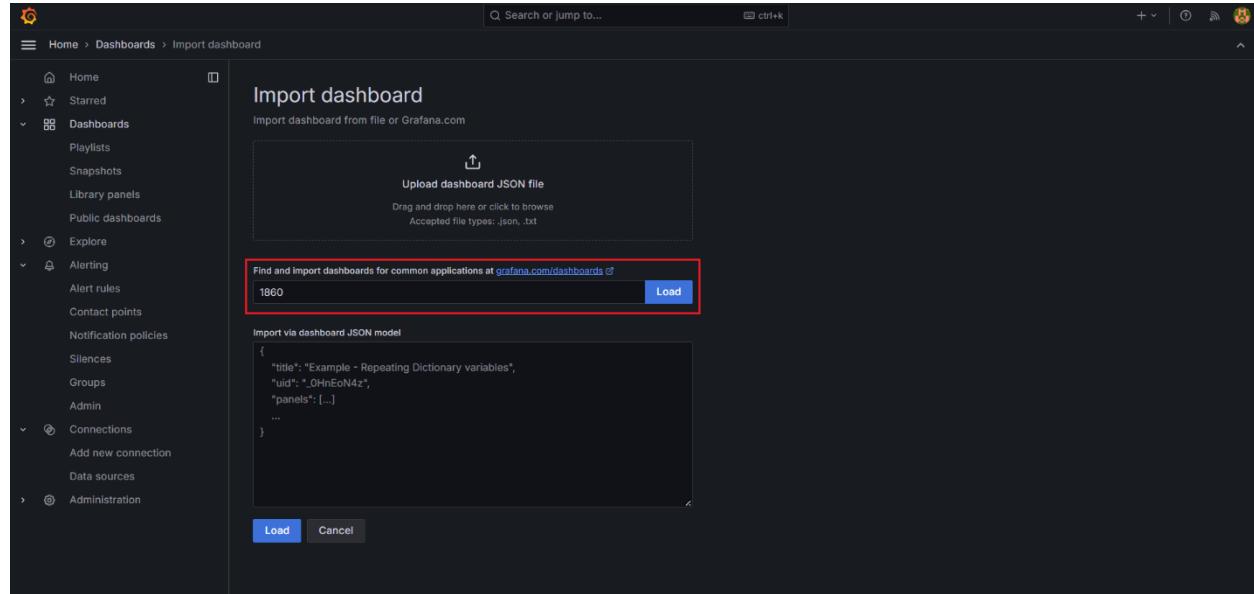
ต่อมาไปที่หน้า Dashboard เราจะ Import Dashboard เข้ามาใช้งาน

จะกดตรง + มุมขวาบน และกด Import dashboard หรือตรง New --> Import ก็ได้เหมือนกัน

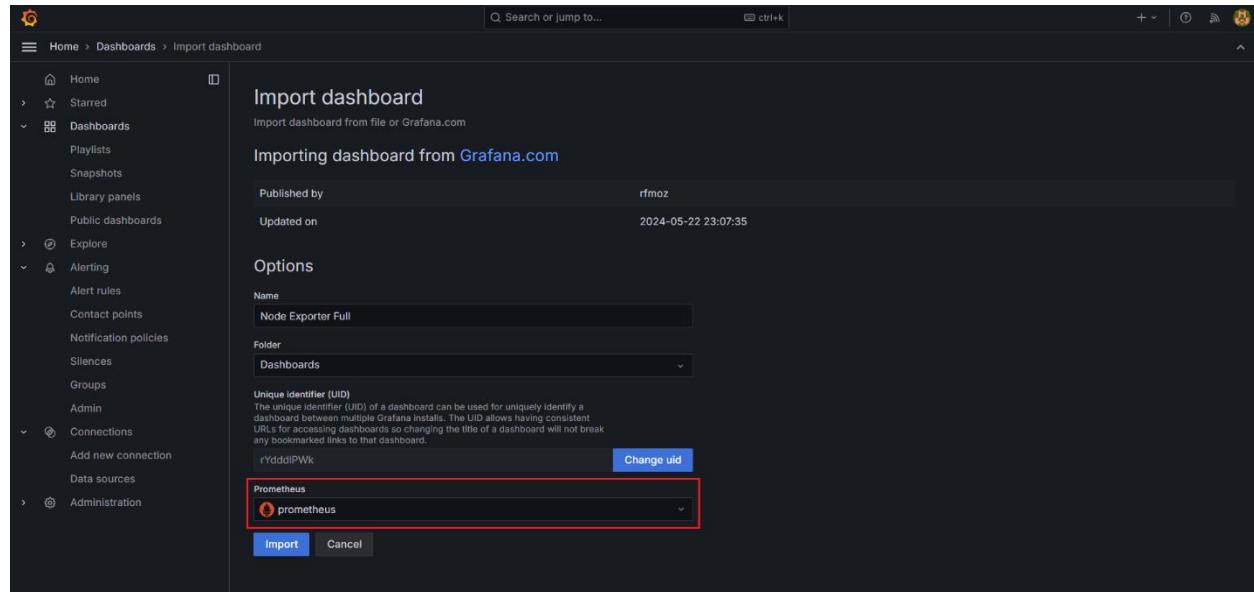
The screenshot shows the Grafana interface with the sidebar navigation open. The 'Dashboards' section is selected. In the top right corner, there is a button labeled 'Import dashboard' with a red box around it. Other options like 'New dashboard' and 'Create alert rule' are also visible.

ใส่ ID ของ Dashboard ที่เราจะติดตั้งแล้วกด Load

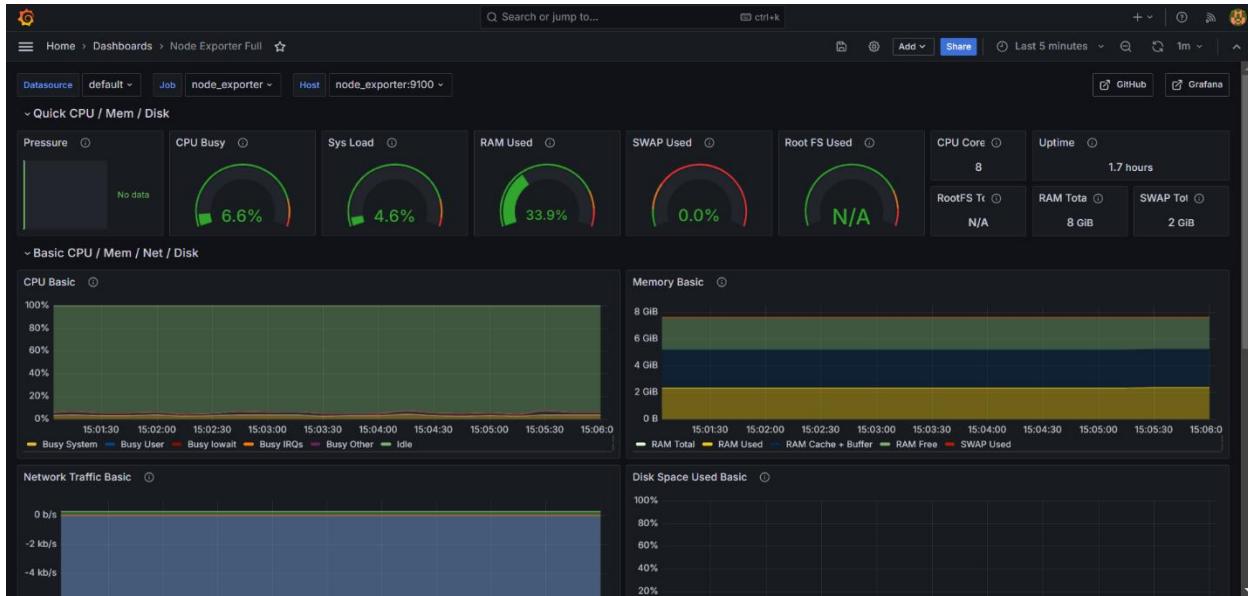
(ในตัวอย่างนี้จะใช้เป็น ID : 1860 จะเป็น ID ของ Node Exporter Dashboard)



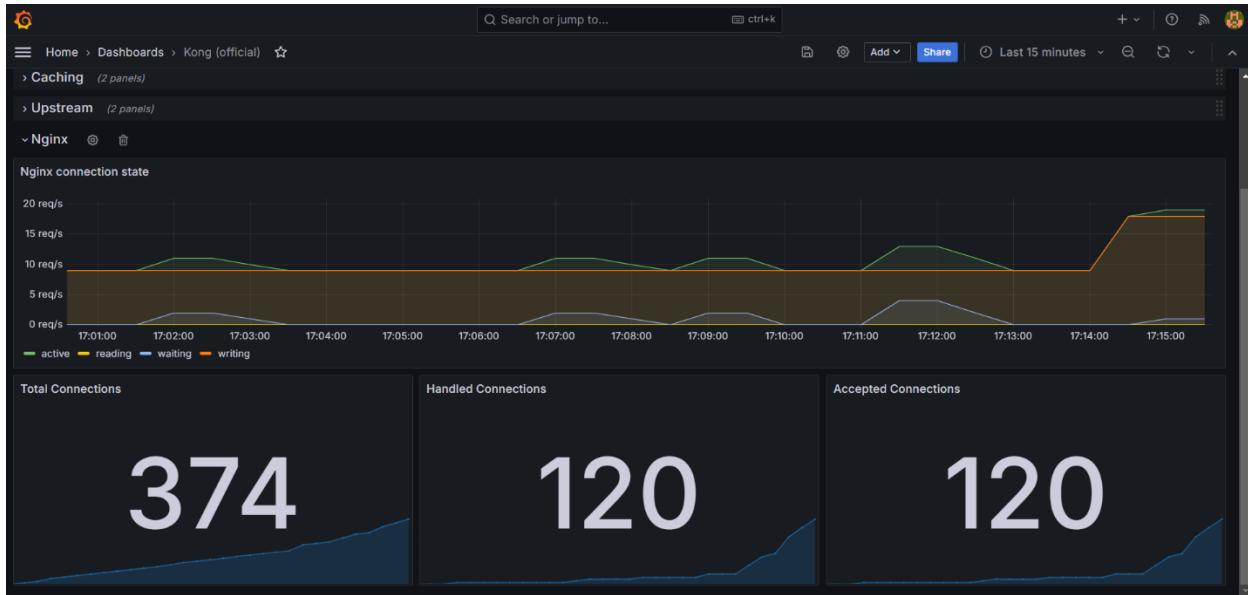
เมื่อกด Load แล้ว ในส่วนของ Prometheus ให้เราเลือก Data Source Prometheus ที่เราได้สร้างไว้ตอนแรกแล้วกด Import



เราจะได้หน้าตาของ Dashboard ของ Node Exporter มาใช้งาน



สำหรับ Dashboard ID ของ Kong APIs Monitoring แบบ Official จะเป็น 7424



จะเห็นว่าใน Request Rate, Latencies Bandwidth, Upstream จะขึ้นว่า No Data ก็เนื่องจากว่าเรายังไม่มีข้อมูลพวน์นมาแสดง จากตัวอย่างที่ได้ทำจะแสดงแค่ส่วน Nginx และ Caching

โดยเราสามารถค้นหา Dashboard ได้จาก <https://grafana.com/grafana/dashboards>