



# GBDi

Government Big Data Institute

สถาบันส่งเสริมการวิเคราะห์และบริหารข้อมูลขนาดใหญ่ภาครัฐ (สวช.)



Health Link



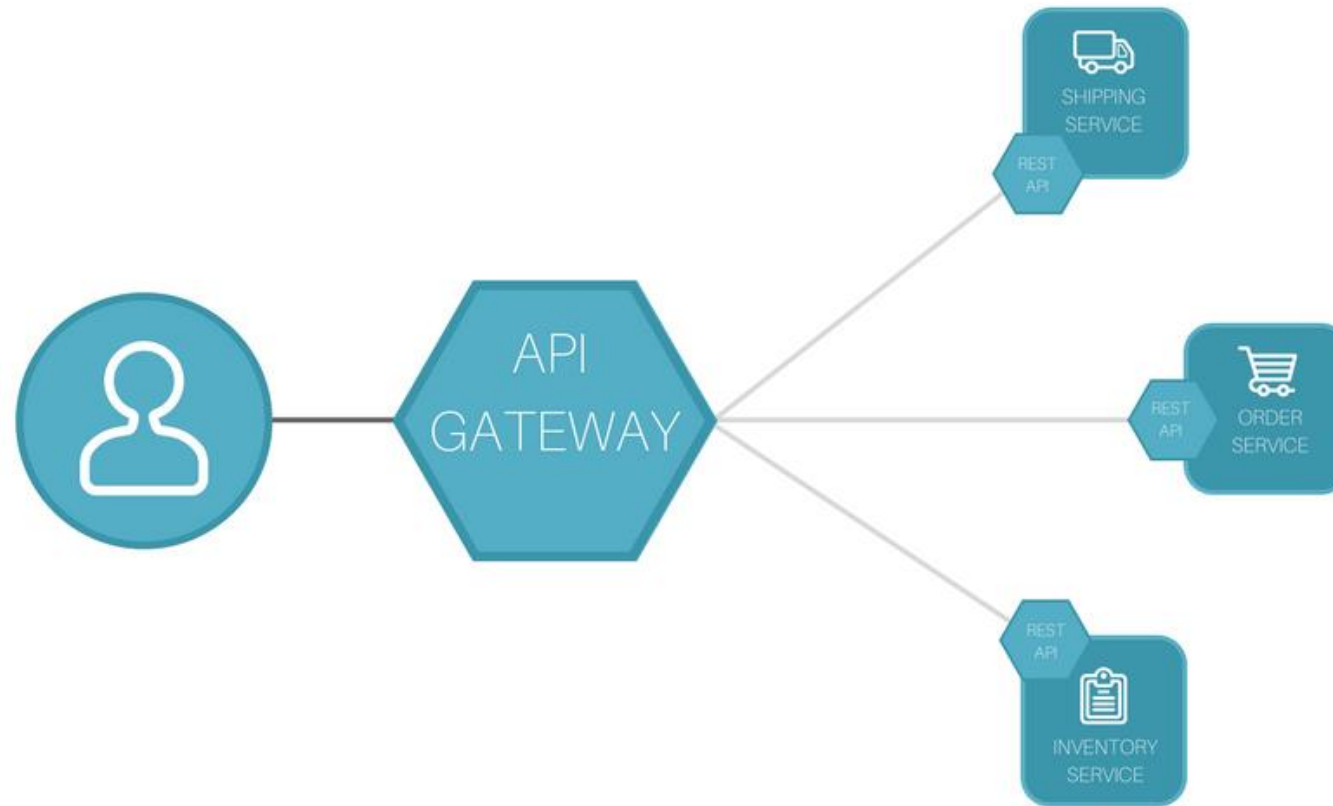
depa



[Kong](#) is an open-source API gateway and microservice management layer.

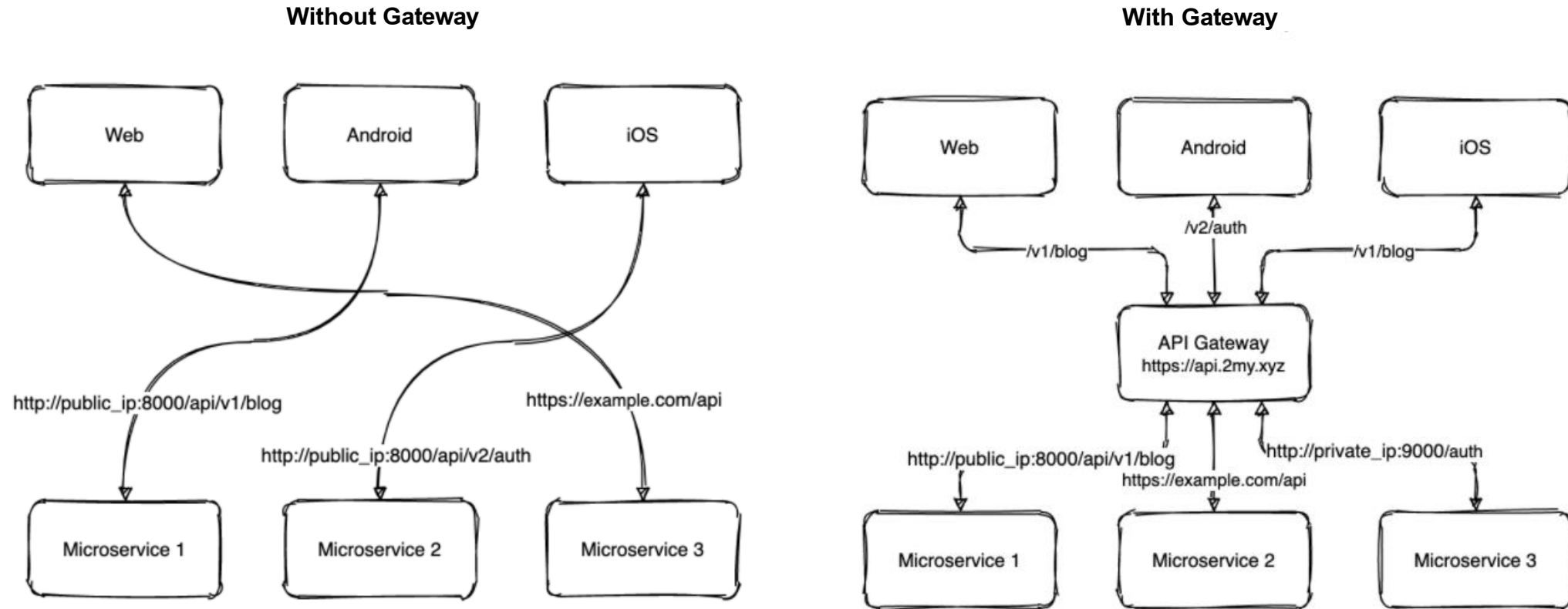
ÓöñiÄ âÿè ÂÒ âõ (Data Engineer)

# API Gateway?



<https://www.express-gateway.io/eg-vs-amazon-aws-api-gateway/>

# API Gateway Concept



How to deal with if ...

- Path/DNS change
- Test Pilot version

# Why do we need Gateway?

## Reasons to Use an API Gateway

Organizations are increasingly adopting microservices for the architecture's inherent flexibility and scalability, but to fully realize the benefits of a microservices approach, you need an API gateway.

A microservice-based system can consist of dozens or even hundreds of individual services communicating with each other via APIs. While it's possible for a client – be that a web browser, application or IoT device – to make requests to the relevant microservice directly, this approach has a number of disadvantages.

Direct client-to-microservice communication means exposing the APIs for each microservice. Developers wanting to interface with the system need to understand the network of services in order to identify the microservices they require. While this may be manageable for very simple applications, in many cases this will involve multiple requests to different microservices, some of which may not use web-friendly protocols. When changes are made to the system, such as combining or splitting services, consumers will be required to update their client-side requests. In addition, functionality such as authentication, rate limiting and monitoring must be applied to each microservice, which will often require the logic to be implemented in multiple languages.

The alternative to direct client-to-microservice communication is to use an API gateway. A gateway acts as an abstraction layer for your microservices and provides a single point of entry for consumers of your application.

## Why Use an API Gateway?

A key benefit of an API gateway is the abstraction of the backend microservices. An API gateway acts as a proxy for your application's microservices, exposing the public-facing API endpoints, routing incoming client requests to the relevant services, transforming them as required and aggregating the response data before sending the response to the client. An API gateway provides a clean interface for clients to interact with, making your system easier to use and therefore more attractive in a competitive marketplace.

Using a gateway also avoids overly "chatty" requests from clients. This is particularly relevant for remote client apps, where multiple roundtrips for requests can introduce high levels of latency and result in poorer performance. Being able to make a single request to an API gateway, which then routes the calls and collates the responses, is far more efficient.

Decoupling your system's public-facing API endpoints from the microservice architecture underneath allows you to make changes to the individual microservices without impacting the consumers of the public API. This consistency for consumers is not just important for existing microservice-based systems but also for organizations moving from a monolithic architecture to microservices. In the latter case, implementing an API gateway at the start of the migration process provides a consistent interface for clients to interact with while the architecture is broken down and re-built behind it.

As the single point of entry to your system, API gateways restrict access to your microservices from the outside world, reducing the potential attack surface compared to a direct client-to-microservice design. API gateways can be used to manage IP whitelists and blacklists and implement authentication and authorization. Not only does this ensure that only valid requests are allowed through, but it is also more efficient than implementing the logic in each microservice, which may mean replicating it for multiple languages and frameworks.

One of the many advantages of a microservice architecture is the ability to scale services independently according to load. An API gateway can provide load balancing to ensure even or weighted distribution of incoming requests across the available instances of a service.

Where high availability is required, load balancing can be combined with rate limiting and throttling to protect the system from unexpected spikes in traffic, including denial of service attacks. Implementing these features at the API gateway provides a central platform for managing this functionality. Again, this avoids the duplicate effort that would be involved in applying the functionality to individual microservices in multiple languages.

# API Gateway

## Best Practices When Using an API Gateway

As the interface that consumers of your system will interact with, an API gateway should be designed to meet their needs. If your system serves multiple types of clients, it may be appropriate to provide multiple API gateways based on those types. This design, known as “backends for frontends,” allows different endpoints to be exposed as well as different security and traffic management policies to be applied.

Being the single point of entry for your system does not mean an API gateway should become a bottleneck or a single point of failure. For applications requiring high availability, setting up a cluster of API gateways with requests load balanced across them ensures a more resilient system.

In order to protect your organization’s assets, API gateways should be designed and configured with security in mind. This includes being mindful of what data is made available via public-facing endpoints and response headers, using secure communication channels, and implementing authentication, rate limiting and throttling.

API gateways provide a central platform for managing these cross-cutting concerns efficiently, ensuring a microservice architecture doesn’t result in duplicated effort. With Kong Gateway, configuring your public-facing endpoints is simple. Kong Gateway includes support for high-availability clusters and includes an extensive range of plugins to address cross-cutting concerns, including authentication, security, rate limiting, throttling, transformations, analytics and monitoring.

Secure

Scale

Dynamic

HA with Load Balancer

Decoupling

Encapsulation

Monitoring



## It's not magic (but it might feel like it)

See why Kong is placed as a Leader having the furthest completeness of vision in the Gartner® Magic Quadrant™ for the second year in a row.

[Read the Report](#)



Source: Gartner (September 2021)

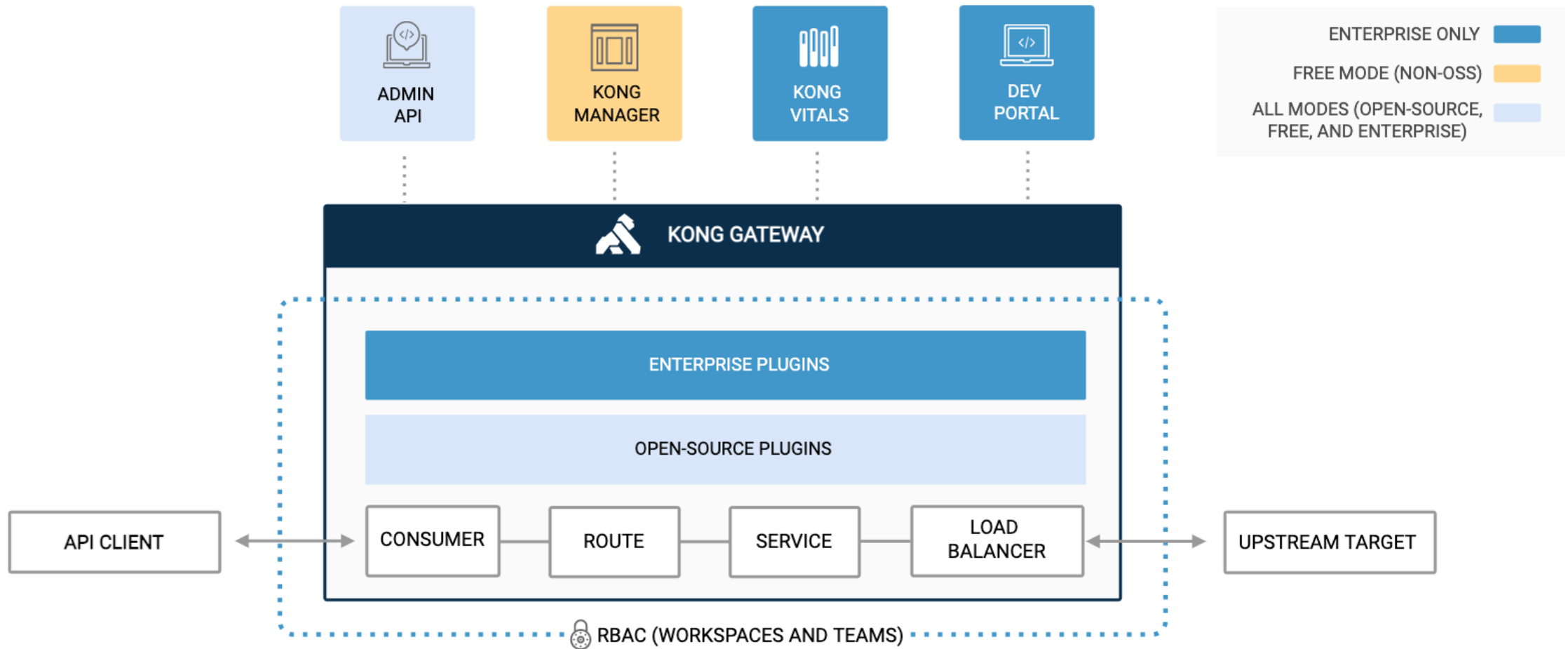
Kong API Gateway คือ Open-source Software โดยเป็นเครื่องมือที่ทำหน้าที่เป็น API Gateway ที่เป็นตัวกลางในการจัดการ Microservices ต่าง ๆ มาไว้ที่เดียว เพื่อควบคุม และจัดการให้ง่ายขึ้น ไม่ว่าจะเป็นเรื่องความปลอดภัย สิทธิ์การเข้าถึง การ Monitoring การกำหนด Rate Limiting เป็นต้น

- Open source or proprietary
- Self-hosted or cloud
- Community
- Plugins and integration support
- Technologies used

## API GATEWAY COMPARISON

API Gateway	Open source/proprietary	On premises/cloud	Community	Plugins and integration support	Technologies
Kong Gateway	Open source (Free tier) Proprietary (Enterprise tier)	On premises (free) Cloud (enterprise)	Large	Yes, a lot	NGINX, Lua
AWS API Gateway	Proprietary (Free tier available)	Cloud	Small	Not much	-
Tyk Gateway	Open source (MPL license)	On premises Cloud	Growing	Yes, a lot	Golang, Python, JavaScript, and more
KrakenD	Open source (Free tier) Proprietary (Enterprise tier)	On premises Cloud Hybrid	Small	Not much	Go, Lua
Gloo Edge	Open source (Free tier) Proprietary (Enterprise tier)	On premises Cloud Hybrid	Small	Not much	C++

# โครงสร้างองค์ประกอบของ Kong Gateway



Kong gateway จะมีรูปแบบการติดตั้ง 2 แบบ คือ แบบมีฐานข้อมูล และไม่มีฐานข้อมูล (เรียกว่า with Database และ DB-less)



## Install Kong Gateway



Docker



Kubernetes



Helm



OpenShift



CentOS



Ubuntu



Amazon Linux 2



RHEL



Debian



MacOS  
OSS ONLY

# Install Kong Gateway

- [With a database](#): Use a database to store Kong entity configurations. Can use the Admin API or declarative configuration files to configure Kong.
- [Without a database \(DB-less mode\)](#): Store Kong configuration in-memory on the node. In this mode, the Admin API is read only, and you have to manage Kong using declarative configuration.

Let's go 

<https://docs.konghq.com/gateway/latest/install-and-run/docker/>

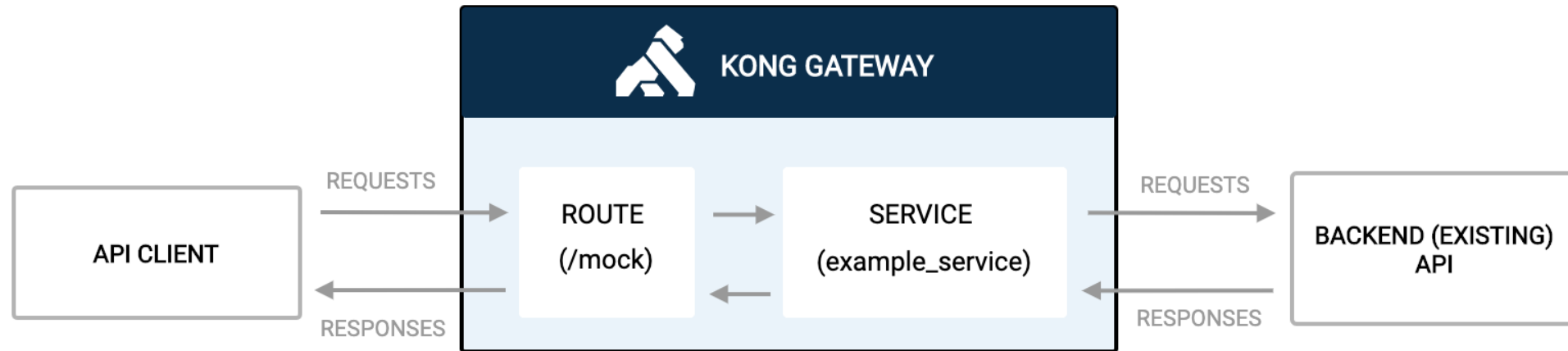
<https://docs.konghq.com/gateway/latest/get-started/quickstart/>

## Kong default ports

By default, Kong listens on the following ports:

- `8000` : listens for incoming `HTTP` traffic from your clients, and forwards it to your upstream services.
- `8001` : [Admin API](#) listens for calls from the command line over `HTTP` .
- `8443` : listens for incoming `HTTPS` traffic. This port has a similar behavior to `8000` , except that it expects `HTTPS` traffic only. This port can be disabled via the configuration file.
- `8444` : [Admin API](#) listens for `HTTPS` traffic.

# Kong Entity (basic)



# Service and Route

- Service is Host or Domain name URL
- Route is Path or Specific request header value
  - Host name based route ใช้ตัวแปร hosts (เป็น Listing จึงเขียนเป็น hosts[])
  - Path name based route ระบุ path ให้ตรงตามที่กำหนด

Let's go 

<https://docs.konghq.com/gateway/latest/get-started/quickstart/configuring-a-service/>

<https://docs.konghq.com/gateway/latest/get-started/comprehensive/expose-services/>

```
$curl -i -X POST \
  --url http://localhost:8001/services/ \
  --data 'name=example-service' \
  --data 'url=https://mockbin.org'
```

กรณี Hostname based route

```
$curl -i -X POST \
  --url http://localhost:8001/services/example-service/routes \
  --data 'hosts[]=example.com'
  --data 'name=backend-api'
```

กรณี Pathname based route

```
$curl -i -X POST \
  --url http://localhost:8001/services/example-service/routes \
  --data 'paths[]=backend-api' \
  --data 'name=backend-api'
```

# Service and Route (Ready to call)

## Host name based

```
curl -i -X GET \  
--url http://kongip:8000/ \  
--header 'Host: example.com'
```

## Path name based

```
$curl -i -X GET \  
--url http://kongip:8000/backend-api
```

# Route (strip path attribute)

ATTRIBUTES	DESCRIPTION
<code>strip_path</code>	When matching a Route via one of the <code>paths</code> , strip the matching prefix from the upstream request URL. Default: <code>true</code> .

```
$curl -X POST http://localhost:8001/services/example-service/routes \
--data 'strip_path=false' ...
```

Backend API: `http://example.com/view/customers`  
 Kong's service: `http://example.com`  
 Kong's route path: `/view`

`http://kongip/view <=> http://example.com/view`

```
$curl -X POST http://localhost:8001/services/example-service/routes \
--data 'strip_path=true' ...
```

Backend API: `http://example.com/view/customers`  
 Kong's service: `http://example.com/view`  
 Kong's route path: `/callview`

`http://kongip/callview <=> http://example.com/view`

# Admin API

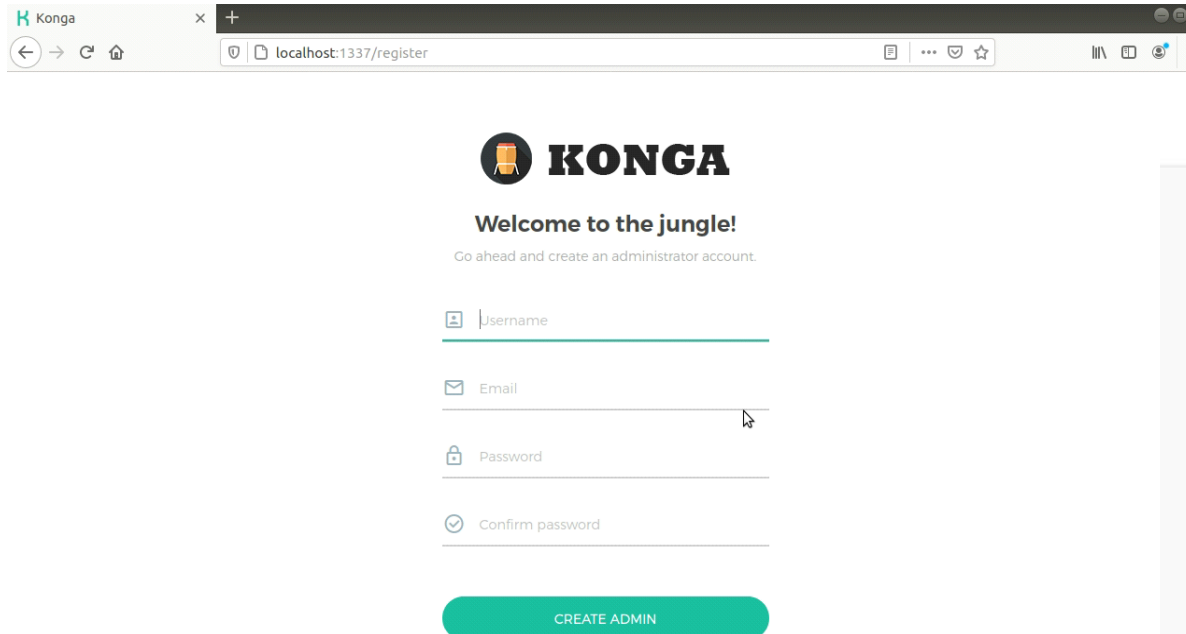
- Kong Gateway comes with an **internal** RESTful Admin API for administration purposes.
  - 8001 is the default port on which the Admin API listens.
  - 8444 is the default port for HTTPS traffic to the Admin API.

Create Service	
POST	/services
List All Services	
GET	/services
Retrieve Service	
GET	/services/{service name or id}

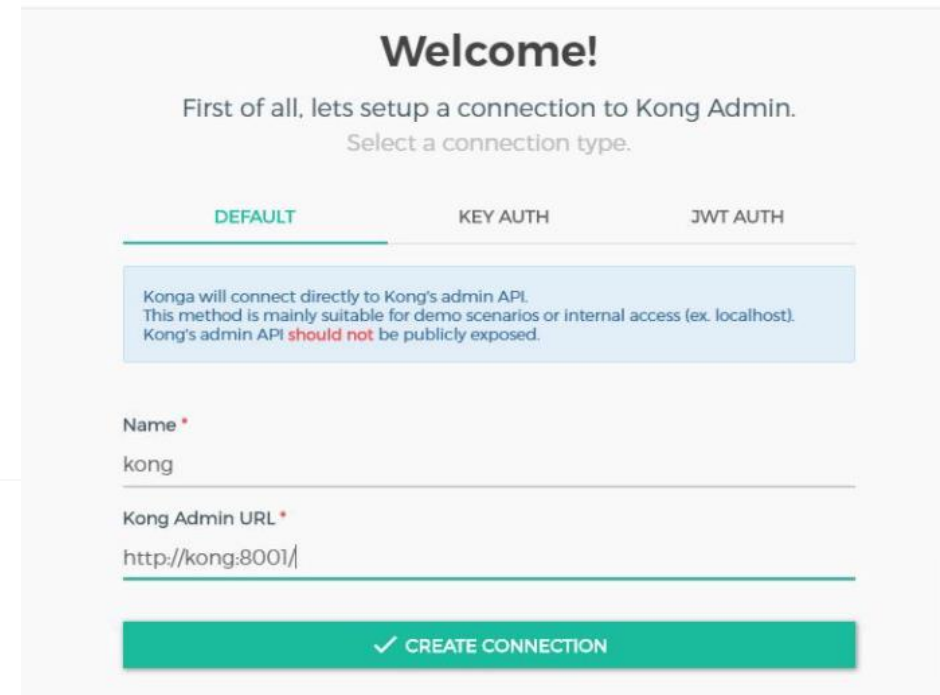




# Konga GUI for Kong gateway

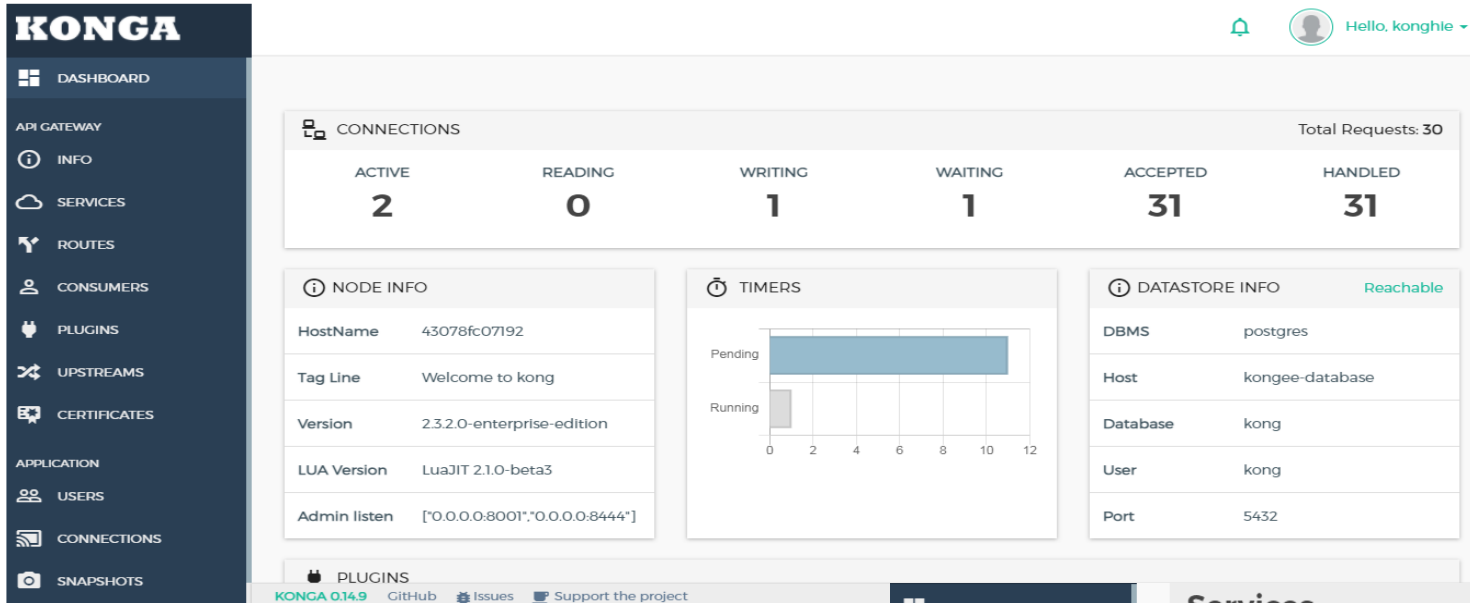


A screenshot of a web browser showing the Konga registration page. The browser's address bar displays 'localhost:1337/register'. The page features the Konga logo (a stylized orange and black icon) and the text 'Welcome to the jungle!'. Below this, a subtitle reads 'Go ahead and create an administrator account.' The registration form includes four input fields: 'Username', 'Email', 'Password', and 'Confirm password', each with a corresponding icon (person, envelope, lock, and checkmark respectively). A green 'CREATE ADMIN' button is positioned at the bottom of the form.



A screenshot of the Konga GUI connection setup page. The page has a light gray background and features the text 'Welcome!' in a large, bold font. Below this, it says 'First of all, lets setup a connection to Kong Admin.' and 'Select a connection type.' There are three tabs: 'DEFAULT' (highlighted in green), 'KEY AUTH', and 'JWT AUTH'. A blue information box contains the text: 'Konga 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.' Below the tabs, there are two input fields: 'Name' (with a red asterisk) containing the text 'kong', and 'Kong Admin URL' (with a red asterisk) containing the text 'http://kong:8001/'. A green 'CREATE CONNECTION' button with a checkmark icon is at the bottom.

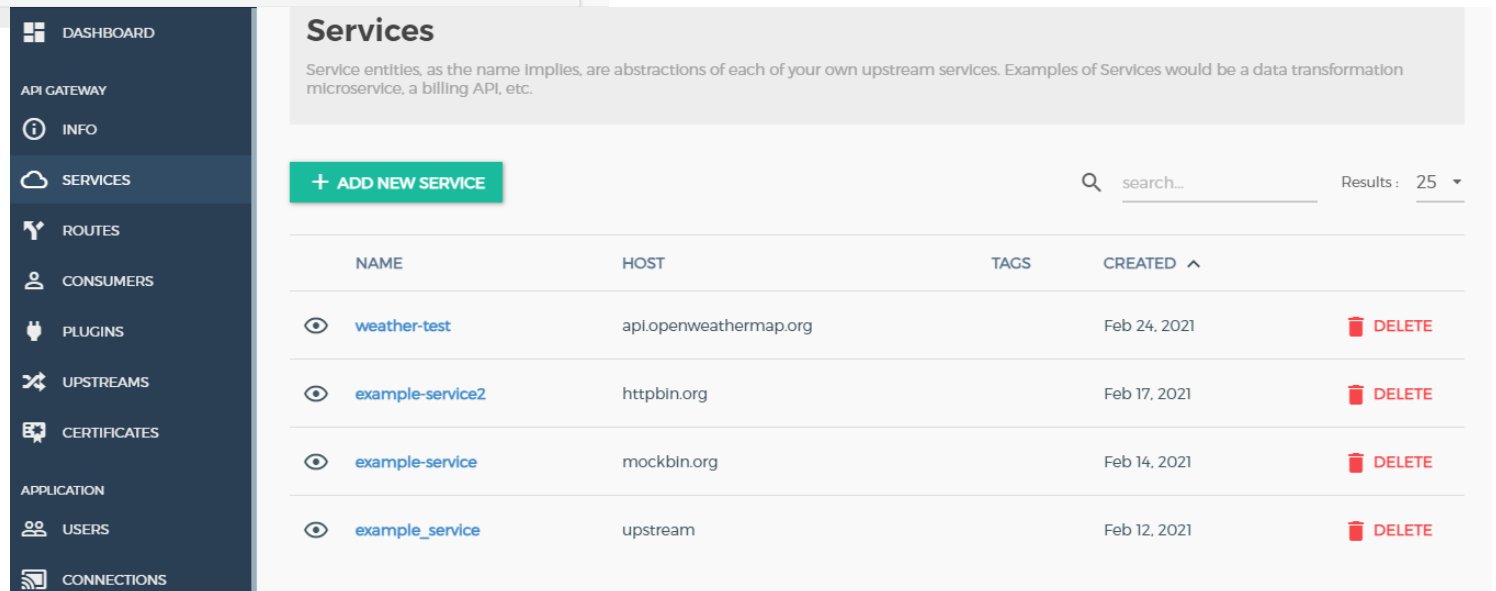
## Dashboard



## Service





Let's go 

<https://hub.docker.com/r/pantsel/konga>



The Services page displays a list of configured services, each with a unique ID, name, host, tags, and creation date. A search bar and a results count are provided at the top right. A green button allows adding new services.

**Services Table:**

NAME	HOST	TAGS	CREATED	
<a href="#">weather-test</a>	api.openweathermap.org		Feb 24, 2021	 DELETE
<a href="#">example-service2</a>	httpbin.org		Feb 17, 2021	 DELETE
<a href="#">example-service</a>	mockbin.org		Feb 14, 2021	 DELETE
<a href="#">example_service</a>	upstream		Feb 12, 2021	 DELETE

# Kong plugins

- Plugins provide advanced functionality and extend the use of the Kong Gateway, which allows you to add new features to your implementation.

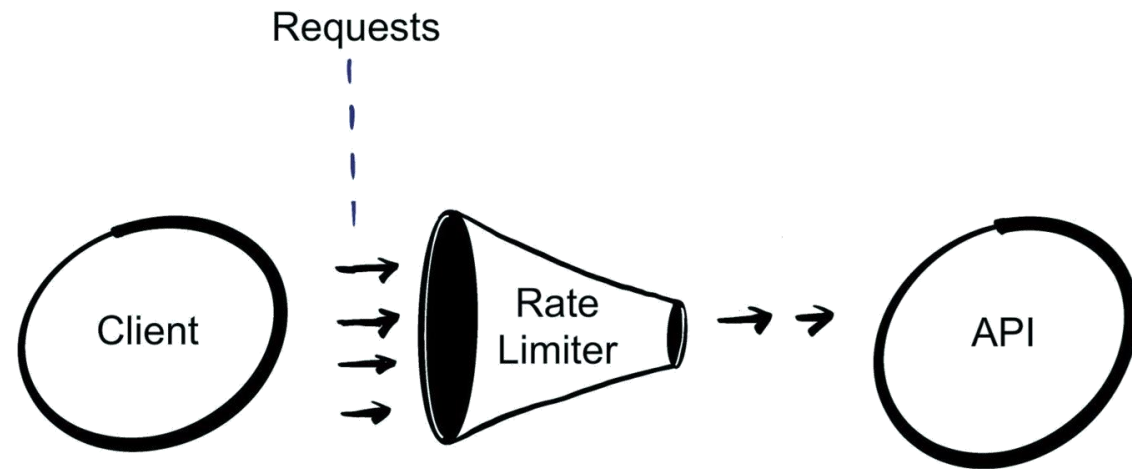
CAPABILITIES	KONG	OTHER PLATFORMS
Extensible	 Easily add advanced traffic management functionality leveraging 95+ out-of-the-box <u>plugins</u> , <u>community plugins</u> and custom <u>plugins</u> .	 Limited ability to extend and customize.
Security	 Implement consistent and fine grained traffic and security policies through advanced out-of-the-box security <u>plugins</u> .	 Limited security features across all use cases.



# Rate Limiting

KONNECT COMPATIBLE

- จำกัดจำนวนการ Request ในระยะเวลาที่ระบบมีให้เลือก ได้แก่ วินาที นาที ชั่วโมง วัน เดือน และ/หรือ ปี
- Request per (`second`, `minute`, `hour`, `day`, `month`, `year`) และสามารถกำหนดรวมกันได้ เช่น เปิดให้ 5 req/minute และ 1000 req/hour




Let's go



[https://miro.medium.com/max/2635/1\\*rIXhA437Q14GYy5dzA0ePg.jpeg](https://miro.medium.com/max/2635/1*rIXhA437Q14GYy5dzA0ePg.jpeg)

<https://docs.konghq.com/hub/kong-inc/rate-limiting/>

# Kong plugins hub

 Kong | Docs

Search the

WE'RE HIRING!

Docs

Plugin Hub

Support

Community

Kong Academy

Request Demo

**INTRODUCTION**  
Plugin Overview  
Compatibility

**FILTERS**  
**All Plugins**  
Konnect  
Third-Party

**SUBSCRIPTION TIERS**

**ENTERPRISE**  
  
**Portal Application Registration**  
Allow portal developers to register applications against Services

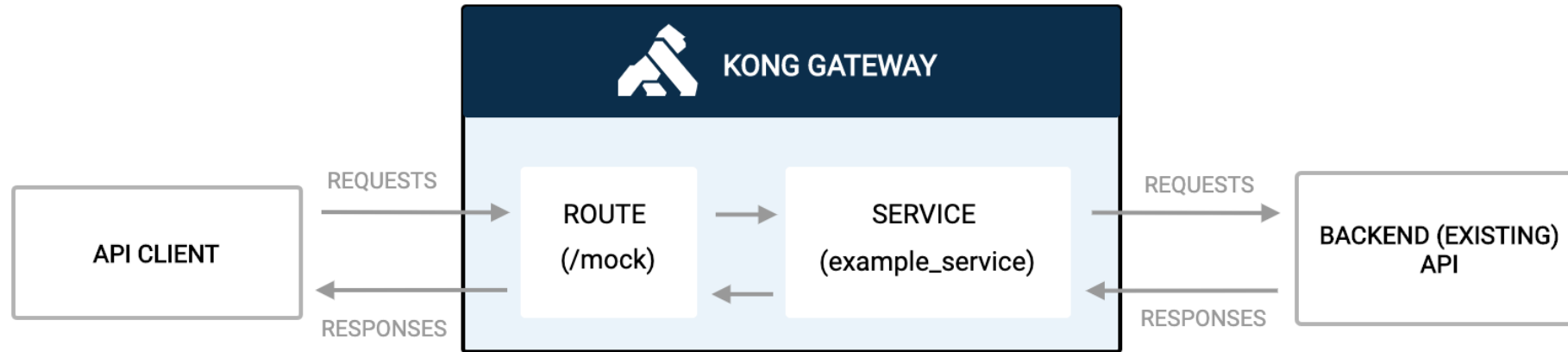
  
**Basic Authentication**  
Add Basic Authentication to your Services

  
**HMAC Authentication**  
Add HMAC Authentication to your Services

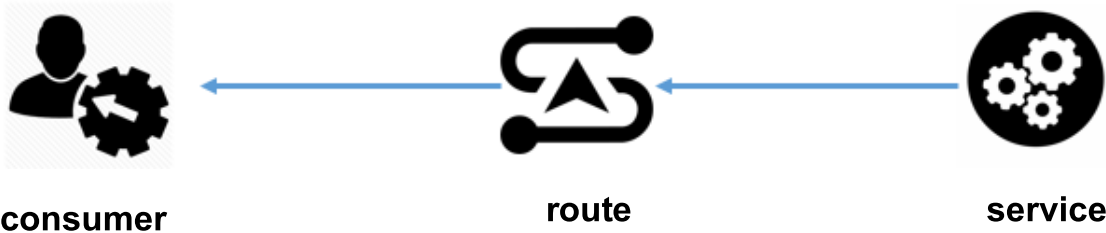
more

<https://docs.konghq.com/hub/>

# How Authentication API



# Consumer component and Authentication plugin



## Key Authentication

KONNECT COMPATIBLE

Let's go 

<https://docs.konghq.com/hub/kong-inc/key-auth/>

# Kong tool and document references

- Kong Gateway (official document) <https://docs.konghq.com/gateway/latest/>
- Kong Gateway docker [https://hub.docker.com/\\_/kong](https://hub.docker.com/_/kong)
- Kong plugins <https://docs.konghq.com/hub/>
- Konga GUI docker <https://hub.docker.com/r/pantisel/konga>

# Thank You

