Aggregator Backend API With KrakenD

Fikri Muhaffizh Imani[1)]
[1]*Magister Terapan Teknik Komputer, Politeknik Caltex Riau, Pekanbaru, Indonesia*

E-mail: [*1)] fikri21mttk@mahasiswa.pcr.ac.id

Abstract: Implementation of API Gateway to secure and expose backend API as an endpoint that can be accessed by the client is ideal step in developing an integrated system. The client makes a request through an endpoint exposed by API Gateway to get the data needed to display a web page, but the server load will increase if the page requires data from several endpoints that require multiple requests to be made to the server. Vertical scaling can be applied to increase server resources in order to handle these requests, but this solution costs a lot of money. This research was conducted to overcome the problem of multiple requests by clients without having to do vertical scaling by implementing KrakenD as an aggregator backend API to combine several backend API into one and expose the combined results back into a new endpoint. Based on the results of the study, aggregation on several API backends was able to reduce 75% of the request load to the server. This solution can become novel consideration in building an information system that requires data from several different backend API.

Keywords: API, API Gateway, Endpoint, KrakenD

1. Introduction

The implementation of the api concept web service or backend API makes it easier for clients to access or change data [1]. API Gateway is a software that functions to facilitate the user interface with backend API as reverse proxy, authentication and authorization [2]. Kong is one of the open source API Gateways built on the Nginx high-performance web server [2] to manage and exposing the backend API to be a more reliable endpoint to be accessed by the client. However, the thing that needs to be considered in using the endpoint is the number of requests from the client side. If the client accesses a page that requires data from multiple endpoints, then as the client increases, there will be repeated requests to the server which can result in increased server load. This problem can certainly be solved by vertical scaling, with the adding faster hardware such as processors and RAM on available servers [3], however, this solution costs a lot of money. Therefore a technology stack is needed that can combine multiple API backends into 1 combined API backend, so that the client simply uses the endpoint without having to use the endpoints in it one by one. This can certainly reduce client requests to the server and the server load will be reduced.

In this study, the application of KrakenD is a stateless gateway software with declarative configuration that provide high performance [2] as a technology stack that can generate endpoints that can provide data responses from multiple API backends. The endpoint will be exposed by Kong API Gateway, so the client only needs to access 1 final endpoint to get data from multiple backend API.

2. Methods

2.1. Initial Configuration

The research was conducted by creating a local environment using Docker as a container to run the required technology stack. Local server used has the following specifications:

Table 1. Server Specifications

Hardware Model	Lenovo ThinkBook 14s G2 ITL
Processor	11th Gen Intel® Core™ i7-1165G7 @ 2.80GHz × 8
Memory	16.0 Gb
OS	Ubuntu 22.04 LTS 64-bit

The technology stack used will run inside a docker container with the following list:

Table 2. Technology Stack

Web Server & Application	Nginx & PHP 8.0	http://172.21.0.1.9003
Database	SQL Server 17	http://172.10.1.1:1433
API Gateway	Kong CE	http://172.13.2.1:6000
Aggregator Endpoint	KrakenD	http://172.14.2.1:6080

This research will use the Human Resource System of Politeknik Caltex Riau as a Web Application that has 4 backend API to be accessed:

Table 3. Backend API of Politeknik Caltex Riau Human Resource system

Data Pegawai	/pegawai/get?collection=pegawai-info
Data Cuti	/cuti/get?collection=list-tgl-cuti
Data Presensi	/presensi/get?collection=pegawai-info
Data Pelanggaran	/indisipliner/get?collection=pelanggaran-pegawai

The Backend API will be exposed by Kong API Gateway to become a public endpoint with a new URL:

Table 4. Public endpoint of the API backend

Data Pegawai	/api/pegawai
Data Cuti	/api/cuti
Data Presensi	/api/presensi
Data Pelanggaran	/api/indisipliner

Initial Scheme

In the initial schema there are 4 public endpoints that can be accessed by the client. The endpoint is an API backend that has been exposed by Kong API Gateway, so that the request made by the client will go through Kong and then forwarded to the backend according to the predetermined route. The initial scheme is described as follows:

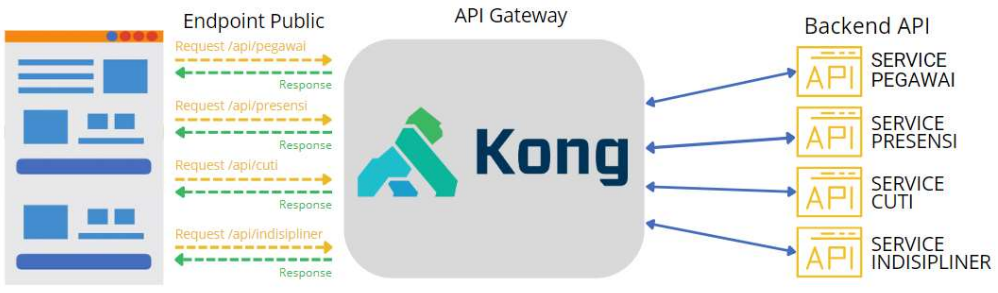


Figure 1. API Gateway initial schema in exposing Backend API

2.2 Aggregator scheme

In the aggregator scheme, KrakenD will be used as a new addition technology stack between Kong API Gateway and Backend API that will combine multiple Backend API into one and create a new aggregator endpoint.

Table 5. Endpoint Aggregator by KrakenD

Endpoint Aggregator	Backend API Source
/agg-pegawai-detail	- /pegawai/get?collection=pegawai-info - /presensi/get?collection=total-terlambat - /cuti/get?collection=list-tgl-cuti - /indisipliner/get?collection=pelanggaran-pegawai

This aggregator endpoint is then exposed by Kong API Gateway to become a public endpoint that can be accessed by the client

Table 6. Public Endpoint by Kong API Gateway

Endpoint Public	Forward To
/api/pegawai-detail	/agg-pegawai-detail

The Aggregator scheme is described as follows:

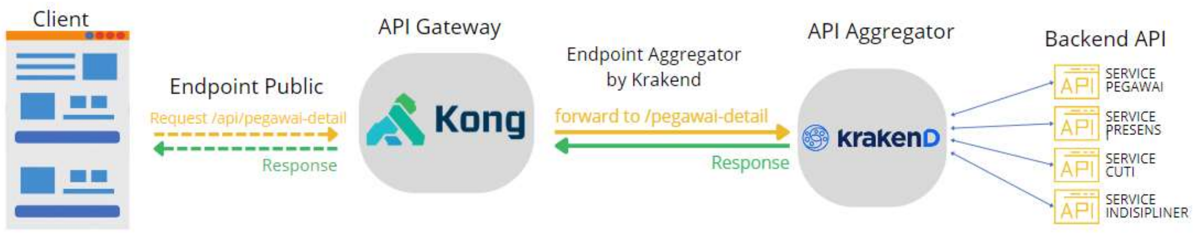
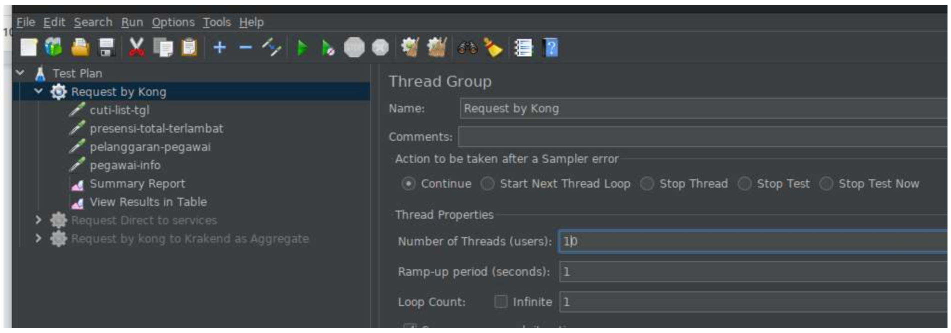


Figure 2. Schema Aggregator backend API by KrakenD

2.3 Test Models

Testing conducted with Apache Jmeter, a java-based open-source application for loading functional behavior tests and measuring web application performance [4] by making an HTTP request to a URL/endpoint with an adjustable number of threads/users, in this case it is an endpoint exposed by Kong API Gateway. Testing will be carried out with 2 schemes namely the Initial Scheme without KrakenD and the Aggregator Scheme with KrakenD.

Initial Schema testing is performed by send request to 4 available public endpoints simultaneously to simulate when a client opens a page that retrieves data from multiple endpoints.

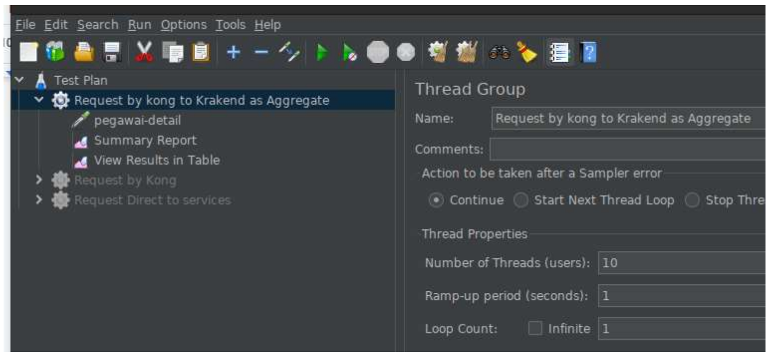


Test Plan
Request by Kong
cuti-list-tgl
presensi-total-terlambat
pelanggaran-pegawai
pegawai-info
Summary Report
View Results in Table
Request Direct to services
Request by kong to Krakend as Aggregate

Thread Group
Name: Request by Kong
Comments:
Action to be taken after a Sampler error
[✓ Continue] Start Next Thread Loop Stop Thread Stop Test Stop Test Now
Thread Properties
Number of Threads (users): 10
Ramp-up period (seconds): 1
Loop Count: Infinite 1

Figure 3. Initial Schema Request Simulation

Aggregator Schema testing is performed by send request to 1 public endpoint (/api/employee-detail) where this endpoint is the endpoint that exposes the KrakenD aggregator endpoint.



Test Plan
Request by kong to Krakend as Aggregate
pegawai-detail
Summary Report
View Results in Table
Request by Kong
Request Direct to services

Thread Group
Name: Request by kong to Krakend as Aggregate
Comments:
Action to be taken after a Sampler error
[✓ Continue] Start Next Thread Loop Stop Thre
Thread Properties
Number of Threads (users): 10
Ramp-up period (seconds): 1
Loop Count: Infinite 1

Figure 4. Aggregator Schema Request Simulation

3. Result and Discussion

The results of the Initial Schema test show that the total requests to server to be 4 times according to the number of endpoints accessed by the user. If 4 endpoints are accessed by 200 users, the server will handle 800 requests when all users open the page simultaneously. In the test results of the Aggregate Scheme, it can be seen that the total requests handled have decreased about 75%. Total Requests that were originally 800 became 200 after the implementation of the KrakenD endpoint aggregator.

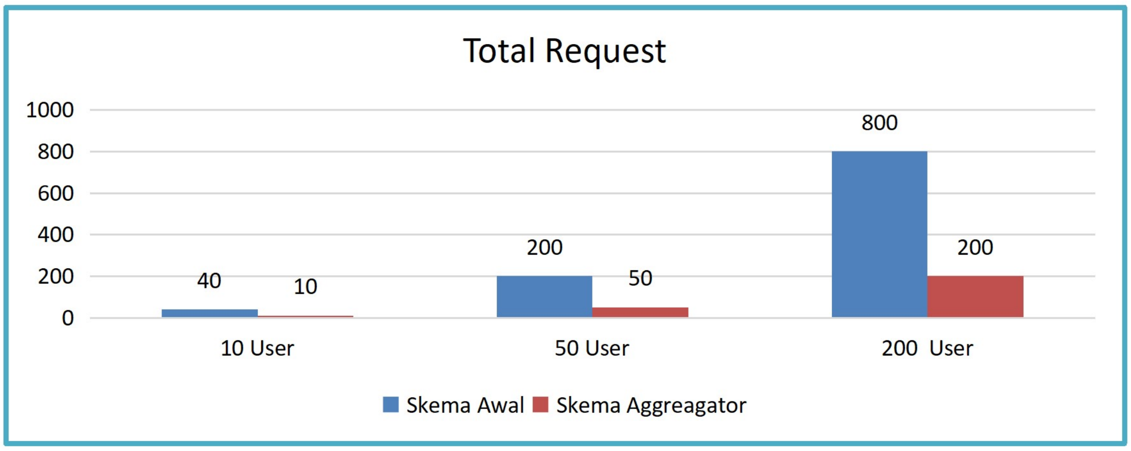


Figure 5. Total Request test results

Initial Schema testing of execution time shows an increase in the time required to complete the request according to the number of users who request simultaneously. In the Aggregator Scheme execution time looks stable in both 10, 50 and 200 users. This shows KrakenD consistency as an aggregator endpoint is quite good at handling requests, even at 200 user execution time the Aggregator Scheme was much faster down about 91% which was originally 29712ms to 2553ms.

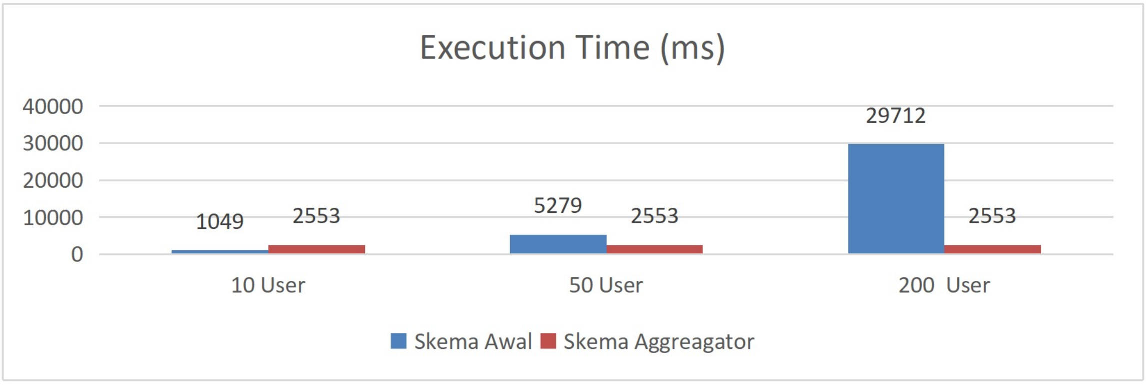


Figure 6. Execution Time test results

4. Conclusions

The addition of KrakenD as an aggregator endpoint between Kong API Gateway and Backend API has proven to be able to reduce server load in overcoming multiple requests to the server and increase page load time due to faster execution time.

Merging Kong API Gateway with KrakenD can be a solution in building pages that require data to come from multiple API backends. KrakenD can complement Kong API Gateway as an API backend aggregator and Kong API Gateway will be in charge of consumer management, authentication, authorization, route and caching.

Reference

[1] B. Baharuddin, H. Wakkang, and B. Irianto, “IMPLEMENTASI WEB SERVICE DENGAN METODE REST API UNTUK INTEGRASI DATA COVID 19 DI SULAWESI SELATAN,” J. Sintaks Log., vol. 2, no. 1, pp. 236–241, Feb. 2022, doi: 10.31850/JSILOG.V2I1.1035.

[2] R. Trebichavský and H. Olesen, “API Gateways and Microservice Architectures Title: API Gateways and Microservices”, Accessed: Aug. 04, 2022. [Online]. Available: http://www.aau.dk

[3] A. H. Ali and M. Z. Abdullah, “A survey on vertical and horizontal scaling platforms for big data analytics,” Int. J. Integr. Eng., vol. 11, no. 6, pp. 138–150, 2019, doi: 10.30880/IJIE.2019.11.06.015.

[4] D. I. Permatasari, “Pengujian Aplikasi menggunakan metode Load Testing dengan Apache JMeter pada Sistem Informasi Pertanian,” J. Sist. dan Teknol. Inf., vol. 8, no. 1, p. 135, 2020, doi: 10.26418/justin.v8i1.34452.