MuleSoft
from Salesforce

WHITEPAPER

Anypoint Flex Gateway Benchmarking Guide

How to select the right API gateway for your business.

Contents

Executive Summary

As more organizations transition from monolithic architectures to microservices-based architectures, organizations are increasing their reliance on APIs to enable smooth communications between disparate systems. As a result, IT teams need additional security, management, and control as APIs grow exponentially across diverse architectures and business priorities.

Industry best practices encourage IT teams to select a lightweight and ultrafast API gateway to control and secure any API deployed in the cloud or an enterprise data center.

Anypoint Flex Gateway is a high-performance, lightweight API gateway that empowers developers to build hyperscaling, highly available, and secure applications. With Flex Gateway, APIs can be managed and secured effectively.

Our benchmark performance analysis shows that Anypoint Flex Gateway provides high throughput and low latency with low memory and CPU usage. Additionally, Anypoint Flex Gateway scales to support the throughput required regardless of the request type or the underlying Kubernetes deployment architecture. The numbers shared prove that Anypoint Flex Gateway is an apt choice for enterprises building highly performant applications.

Introduction

APIs are fundamental to accelerating business innovation, touching everything from improving business agility to enhancing connectivity. As the use of APIs grows and the transition to microservices-based architectures accelerates, organizations increasingly face API sprawl.

Additionally, managing and securing these APIs is becoming exponentially difficult with varying environments and architectures. Gartner predicts that over 50% of APIs will go unmanaged by 2025.

MuleSoft Anypoint Flex Gateway allows IT teams to manage and secure APIs regardless of environment and architecture – an invaluable feature to counter the complications caused by API sprawl.

Adopting a flexible and highly performant API gateway enables IT teams to manage and secure APIs running across the organization's digital estate, whether on-prem or in the cloud. With consistent policies, organizations can have a unified, secure network that allows for scaling over time to support future innovations.

This whitepaper presents a high-level overview of the performance of Anypoint Flex Gateway. We'll cover performance benchmarking numbers based on several Kubernetes deployment options.

Managing and securing sprawling APIs is becoming exponentially difficult

Over

50%

of APIs are expected to go unmanaged by 2025

What Is Anypoint Flex Gateway?

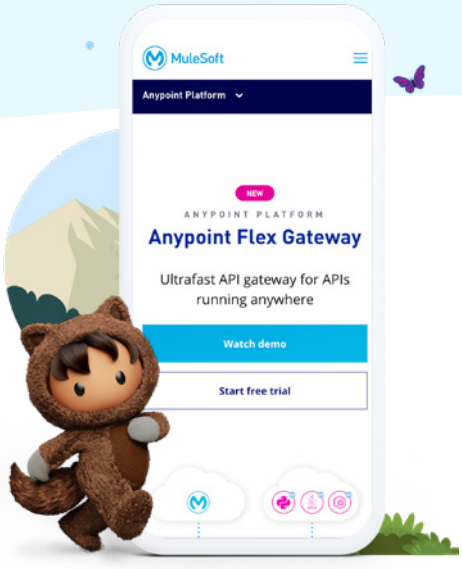
Anypoint Flex Gateway provides the ability to manage and control any API built and deployed anywhere. The gateway can be deployed in various environments, such as on-premise, cloud-native, or containerized environments, and can be easily integrated with CI/CD pipelines. To learn more, please consult the latest Anypoint Flex Gateway documentation for information on other deployment options.

Flex Gateway can run in two modes: *Connected Mode* and *Local Mode*.

1. **Connected Mode** fully connects to Anypoint Platform control plane for centralized management, observability, and security through a web user interface.
2. **Local Mode**: runs disconnected from the Anypoint Platform control plane. The gateway is managed locally with stored declarative configuration files, and is best suited for CI/CD deployment automation.

With these two options, organizations can manage the APIs based on the use case that best suits their needs.

Flex Gateway is built on Envoy, an open-source edge and service proxy, and uses a sophisticated threading model with worker threads. Multi-threading enables the gateway to run as a single process while achieving parallelism using every CPU available for optimal performance and scalability.



ANYPOINT PLATFORM

Learn more

Go to Anypoint Flex Gateway page on the MuleSoft website for more information.

Reading the Benchmarking Charts

This report shows the maximum possible throughput in a controlled environment. There are three different types of charts in this whitepaper showing the impact of response payload size on:

Queries per second (QPS) and latency

CPU and memory utilization

Latency and its latency distribution

There are also charts showing the impact of CPU resourcing on QPS and latency.

Definitions:

Latency

Latency describes the total time it takes to receive a response from a request. Latency distribution groups the request responses into percentiles. For example, 99th percentile latency indicates that the latency of 99% of the request is less than or equal to this value.

Throughput

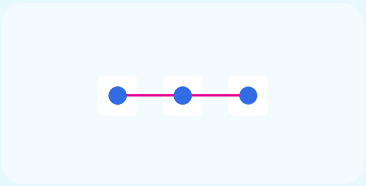
Throughput is the number of queries processed. We measure throughput through Queries per second metric (QPS).

Flex Gateway Performance in Kubernetes Deployments

An API gateway allows IT teams to add specific policies and control traffic flow. To determine the Flex Gateway deployment architecture for your organization, you'll need to consider infrastructure costs, API environment, and the number of APIs to manage.

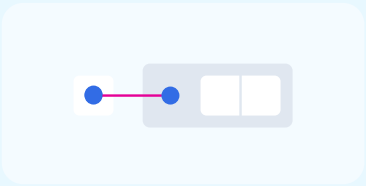
Out of multiple deployment options available, we tested Anypoint Flex Gateway deployed in Kubernetes, which allowed us to mimic production deployment since many organizations are moving toward containerized solutions for application scalability and portability.

Common Kubernetes deployment topologies used for Flex Gateway are:



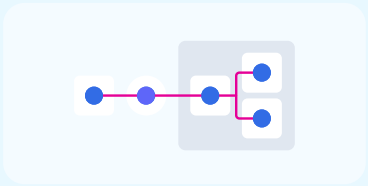
Standard Deployment

The backend server and the gateway are deployed on different nodes in the cluster. This deployment schema assigns a node for the gateway and another for the backend. Finally, the load generator deploys jobs to the remaining nodes on the cluster.



Sidecar Deployment

The backend server and the gateway are both assigned to the same node. The gateway is configured as a sidecar container within the same pod as the backend server. The load generator jobs are deployed on the remaining nodes in the cluster.



Ingress-Controller Deployment

This deployment schema assigns a node for the gateway and another for the backend within the same cluster. However, the load generator jobs are deployed outside the cluster and communicate via ingress rules. *Ingress-controller deployment* is ideal for routing to multiple APIs or services within the same cluster.

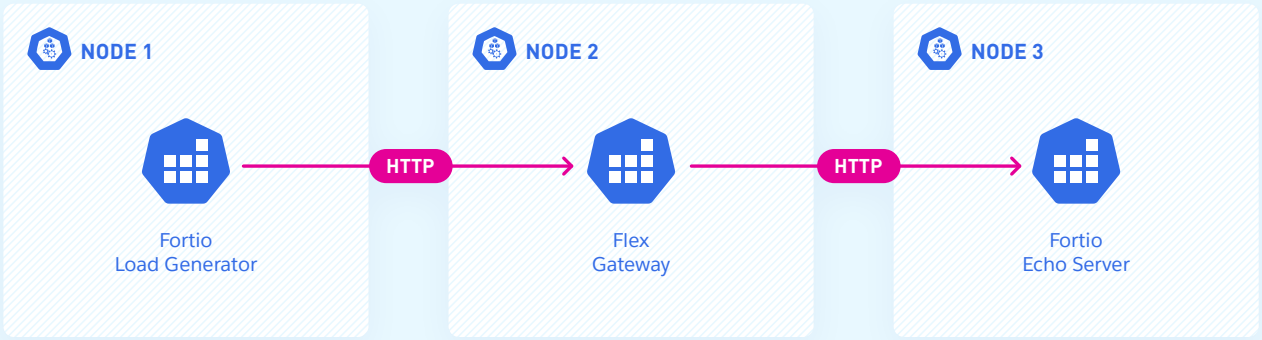


Diagram for *standard deployment*.

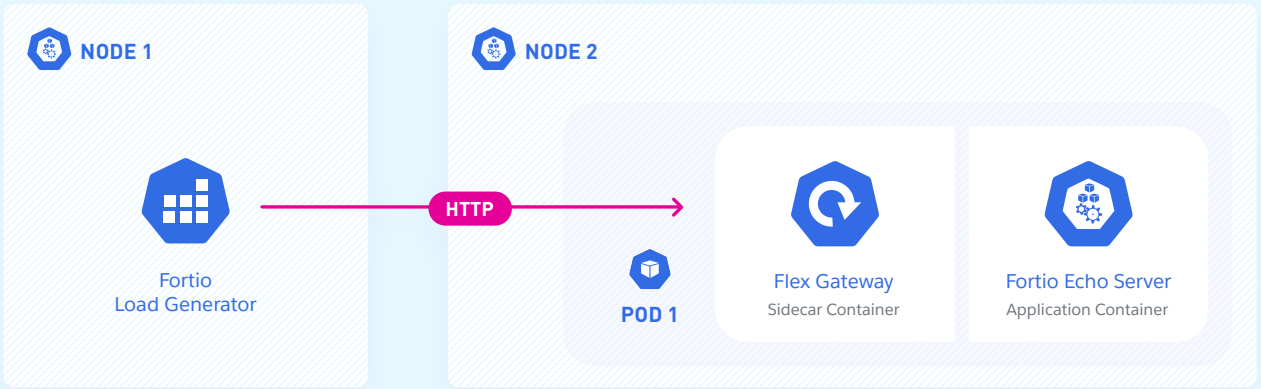


Diagram for *sidecar deployment*.

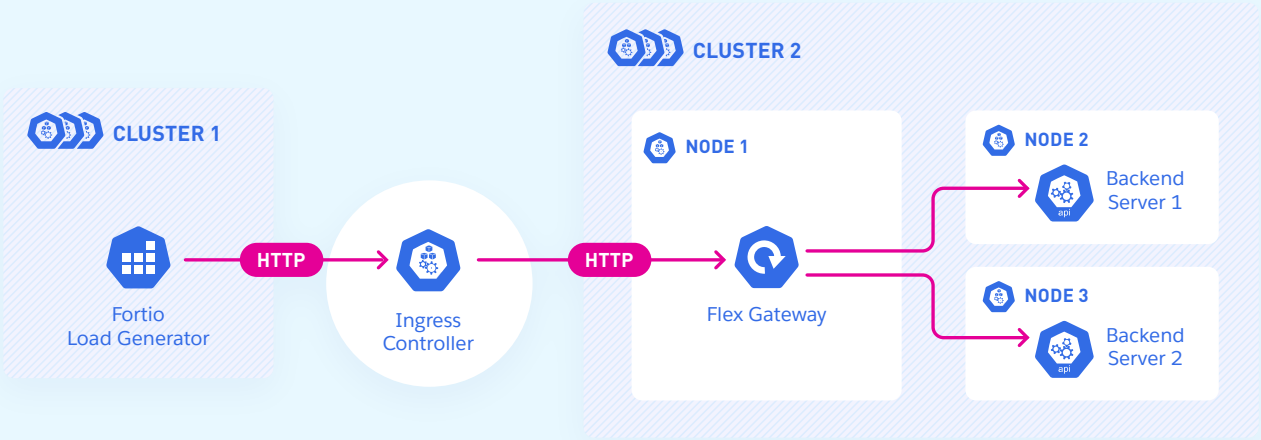


Diagram for *ingress-controller deployment*. Please refer to Appendix A for more information on the architecture and methodology.

This whitepaper will focus on *standard* and *sidecar deployments* where the load generator is within the cluster to minimize the effects of external factors such as a load balancer. It is safe to assume the performance of Flex Gateway in an ingress-controller deployment will have similar results as the *standard deployment*.

The scenarios tested include ***standard*** and ***sidecar deployments*** with HTTP GET and POST requests, where the effect of payload size on QPS, latency, memory, and CPU usage were measured. For each scenario, 200 concurrent threads requested 20 million queries over 20 minutes. Three different response payloads (1 KB, 10 KB, and 100 KB) were used to see the effects of response payload size on performance. One KB of request payload was used throughout HTTP POST requests. To minimize the number of dependencies in the testing, Flex Gateway was run on *Local Mode*. Please refer to Appendix A for more information on the methodology and other parameters used.

HTTP GET Requests

GET REQUESTS

Effect of Payload on QPS and Average Latency

Figure 1 shows that there is no significant difference between standard and sidecar deployment. Average latency increases with an increase in the response payload size, with the most significant difference between 10 KB and 100 KB.

However, consistency between 1KB and 10 KB response payloads indicates that Flex Gateway can process at optimal throughput between these two sizes. To achieve high performance with large payload size, the recommended best practice is to divide data into smaller sizes to be processed independently in parallel. In most modern application use cases, payload sizes are below 10 KB. In addition, HTTP caching may decrease latency as larger latencies only apply to new requests.

For developers, the high throughput and low latency with out-of-the-box Flex Gateway configuration mean saving time on the setup, allowing them to focus on higher-priority tasks like API development.

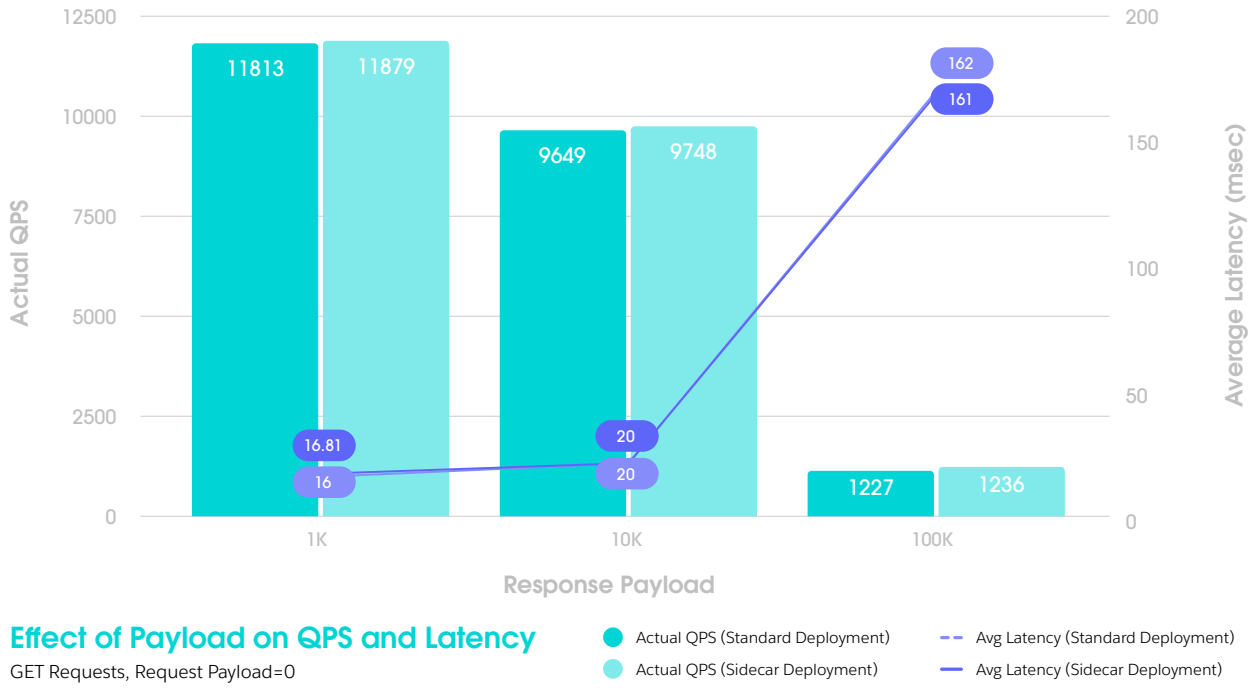


Figure 1: Effect of response payload on actual QPS & average latency for HTTP GET requests in *standard* and *sidecar deployments*.

GET REQUESTS

Effect of Payload on Latency Distribution

In Figure 2, both *standard* and *sidecar deployment* increased in latency with increased response payload. For 1 KB and 10 KB, the latency is almost negligible across the average latency distribution, indicating highly predictable response times. We notice that for larger response payloads, latency is affected. This is typical of all API gateways. As payload size increases, the gateway requires more time to read and respond with the payload.

Figure 2 shows the high reliability of Flex Gateway. Maximum latency changes are negligible across the distribution. This lower latency implies faster page loads for high-performance use cases such as fast-loading web applications, resulting in a better user experience for the end user.

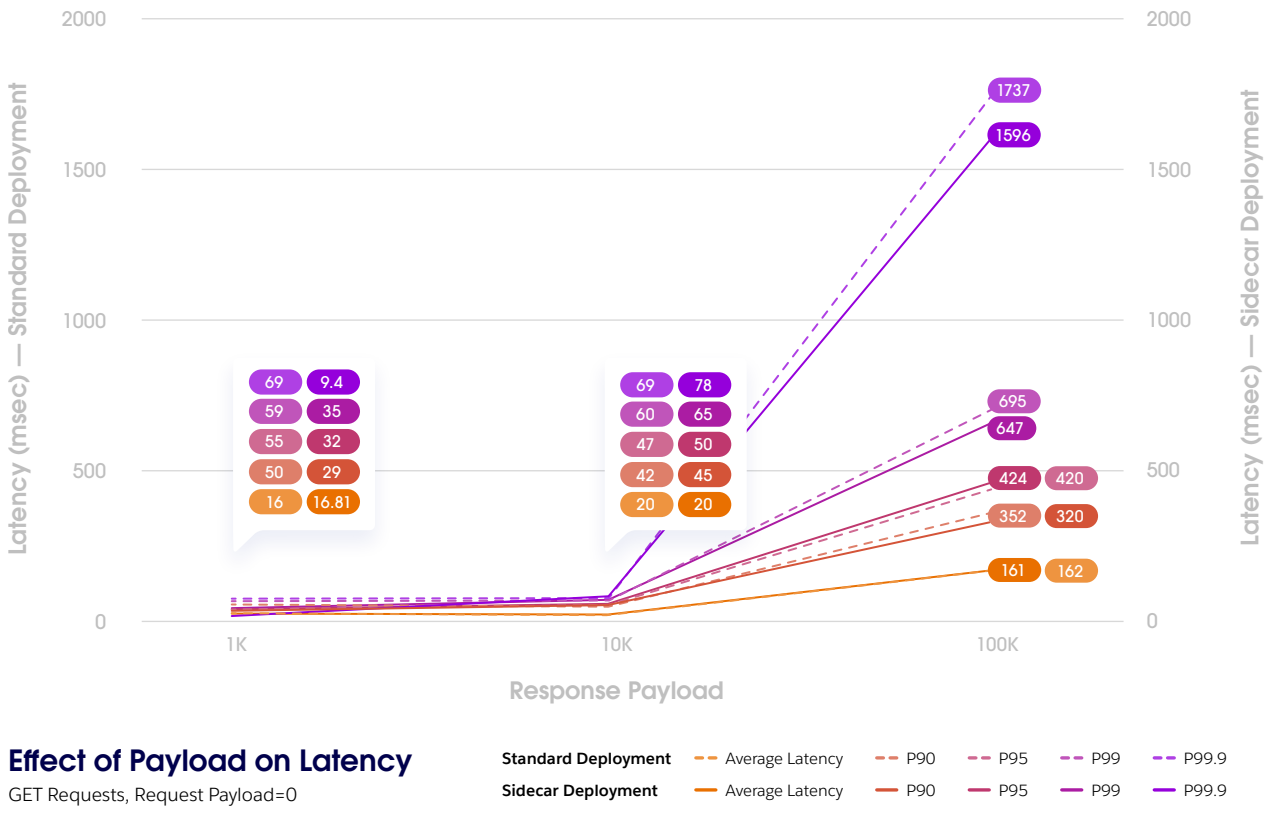


Figure 2: Effect of response payload on latency for HTTP GET requests in *standard* and *sidecar deployments*.

GET REQUESTS

CPU and Memory Utilization

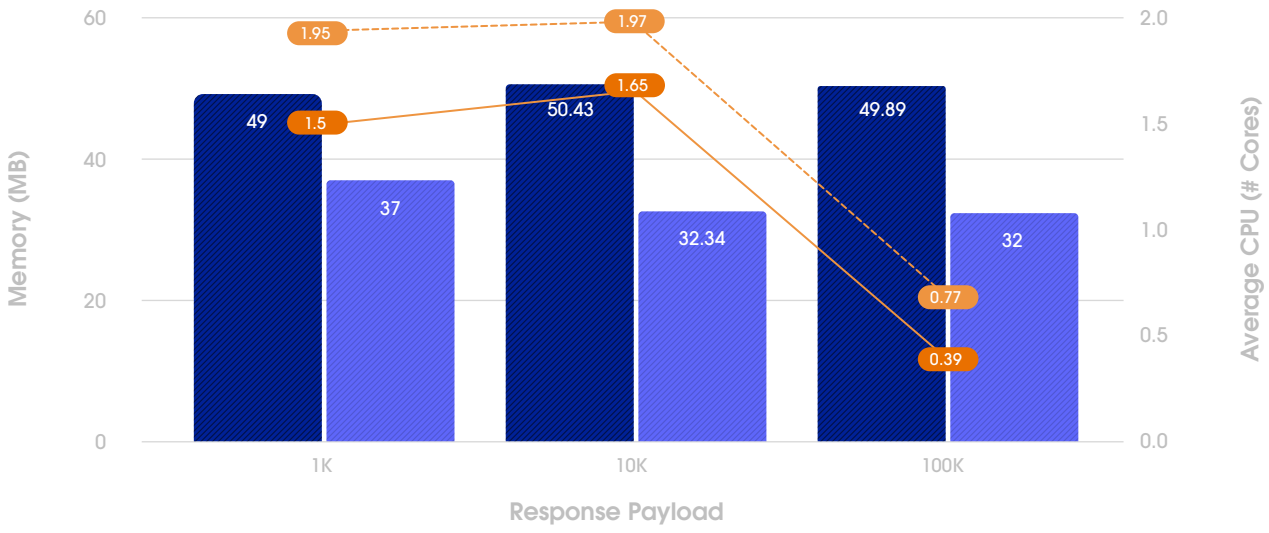
Figure 3 shows a negligible difference in memory utilization between payload sizes. The impact of CPU as request size increases is minimal. This indicates that Flex Gateway can scale to process queries without taxing the memory or CPU resources and proves that it has a smaller footprint resulting in lower costs overall.

In general, the underlying resource utilization determines the Flex Gateway's sizings and ultimately determines the costs to run Flex Gateway at optimal performance per the application needs.

On average, *sidecar deployment* has 30-40% lower utilization for memory than *standard deployment* because Flex Gateway is installed on the same pod as the backend application.

Lower CPU usage in *sidecar deployment* compared to *standard deployment* indicates that Flex Gateway is highly performant when placed directly with the backend server. This allows organizations to manage and secure APIs with Flex Gateway without creating additional endpoints.

Sidecar deployment for Flex Gateway is the best option for productions requiring faster response with minimal resource usage. However, ***standard deployment*** should be utilized when APIs are scattered across multiple nodes or pods, and one gateway is needed to manage them.



GET Requests: CPU Utilization and Memory Utilization

Figure 3: Effect of response payload on CPU and memory utilization in *standard* and *sidecar deployments.*

HTTP POST Requests

POST requests were sent with a constant request payload of 1 KB. We varied the size of response payloads. Based on our findings, the results for QPS, latency, and latency distributions were similar to GET requests.

POST REQUESTS

Effect on Payload on QPS and Average Latency

Based on the findings shown in Figure 4, there is no significant difference between *standard* and *sidecar deployments*. Actual QPS decreases with an increase in response payload size, similar to GET requests.

POST requests are often used to send data to the server, such as new user information, data updates, etc. The results show minimal delay and still a high throughput even though POST requests require the additional step of updating the backend server. And, at average latency below 30 milliseconds at 1 KB and 10 KB response, the user will experience real-time updates for most typical POST requests. There is a consistency between 1 KB and 10 KB response payloads in QPS and latency, indicating that Flex Gateway can process requests with high throughput for typical use cases.

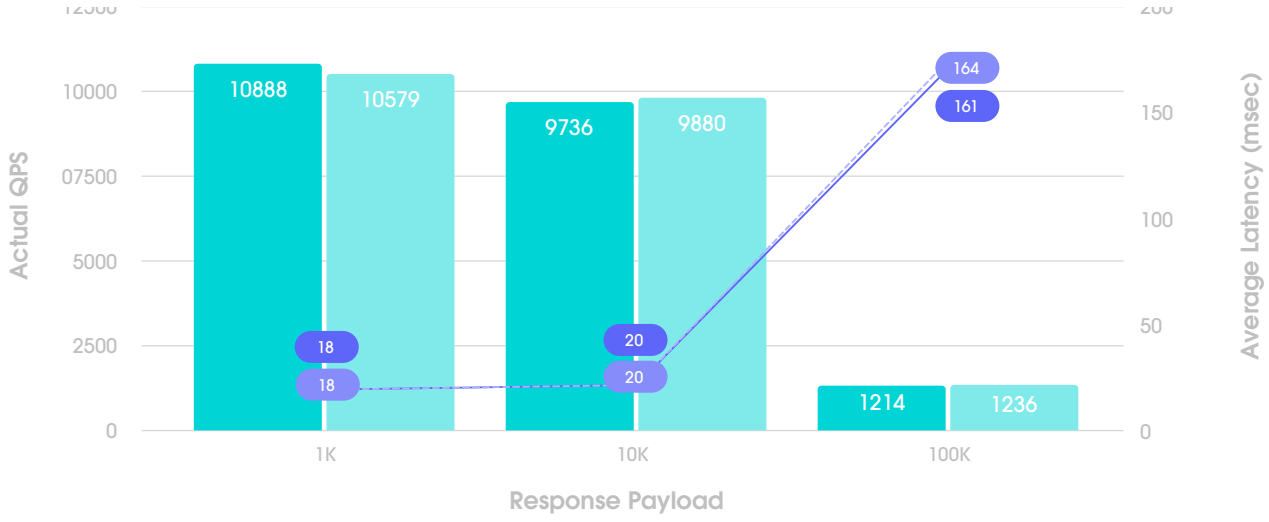


Figure 4: Effect of response payload on actual QPS & average latency for HTTP POST requests in *standard* and *sidecar deployments*.

POST REQUESTS

Effect of Payload on Latency Distribution

In Figure 5, both *standard* and *sidecar deployments* increase latency with increased response payload. For 1 KB and 10 KB, the latency is almost negligible across the average latency distribution, indicating highly predictable response times.

However, latency is affected at higher latency distribution for larger response payloads. This is typical of all API gateways.

As payload size increases, the gateway takes more time to read, process, and transmit the payload.

Overall, the difference in the type of request does not affect Flex Gateway latency. This stability means the gateway can reliably process both requests at a similar speed.

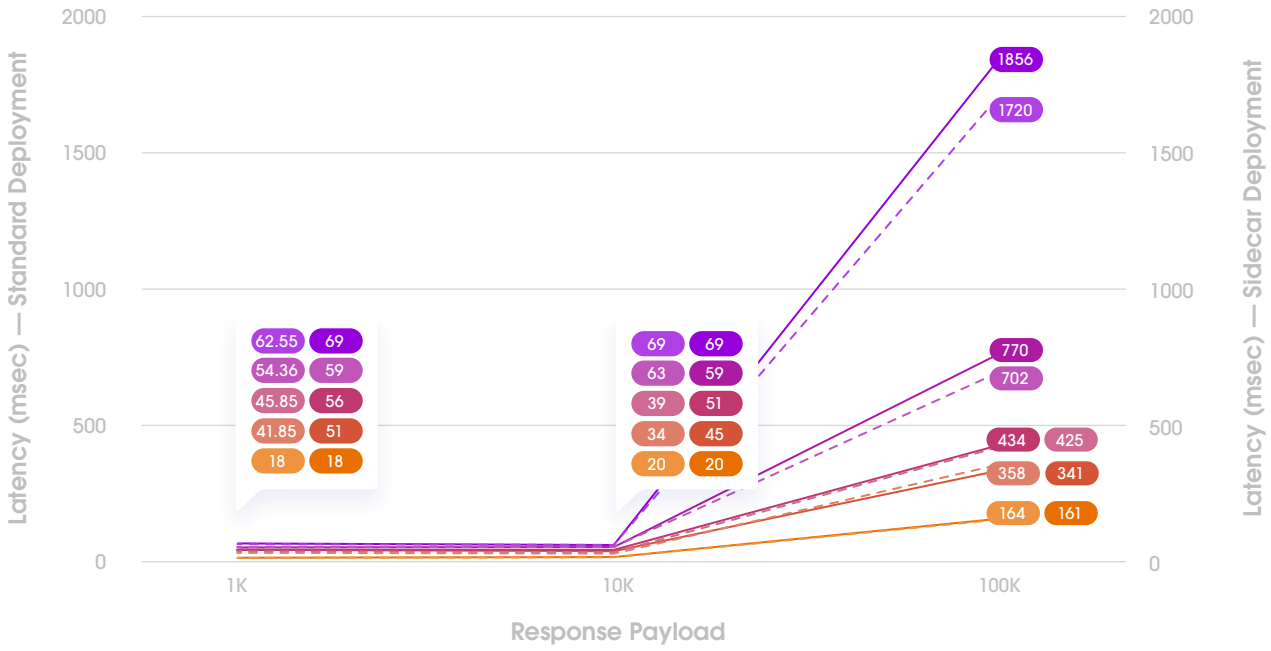


Figure 5: Effect of Response payload on latency for HTTP POST requests in *standard* and *sidecar deployments*.

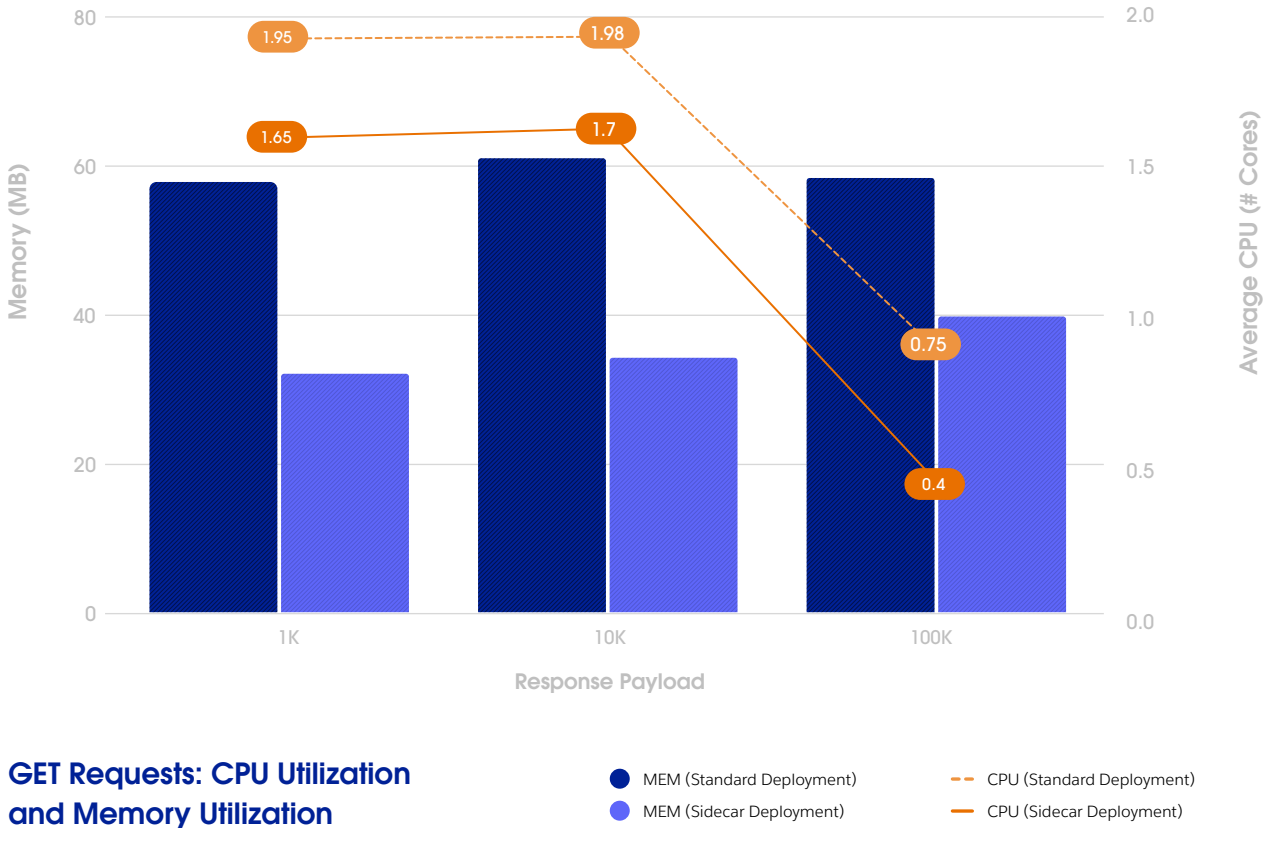
POST REQUESTS

CPU and Memory Utilization

Figure 6 shows that CPU and memory utilization for POST requests are the same for both *standard* and *sidecar deployments.* Similar to GET requests, as response payload increases, CPU usage decreases.

While CPU usage is overall higher for POST requests, there's a negligible difference in memory utilization between payload sizes and deployment types.

Flex Gateway in *standard deployment* used significantly less memory when compared to the GET requests result, given that response to POST requests does not require caching.



GET Requests: CPU Utilization and Memory Utilization

Figure 6: Effect of response payload on CPU and memory utilization in *standard* and *sidecar deployments.*

POST REQUESTS

Effect of CPU and Payload Sizes on QPS

In this section, we tested the impact on QPS and latency of POST requests with regard to:

→ Increase in request payload size
→ Increase in response payload size

In each scenario, we also varied the amount of CPU resource.

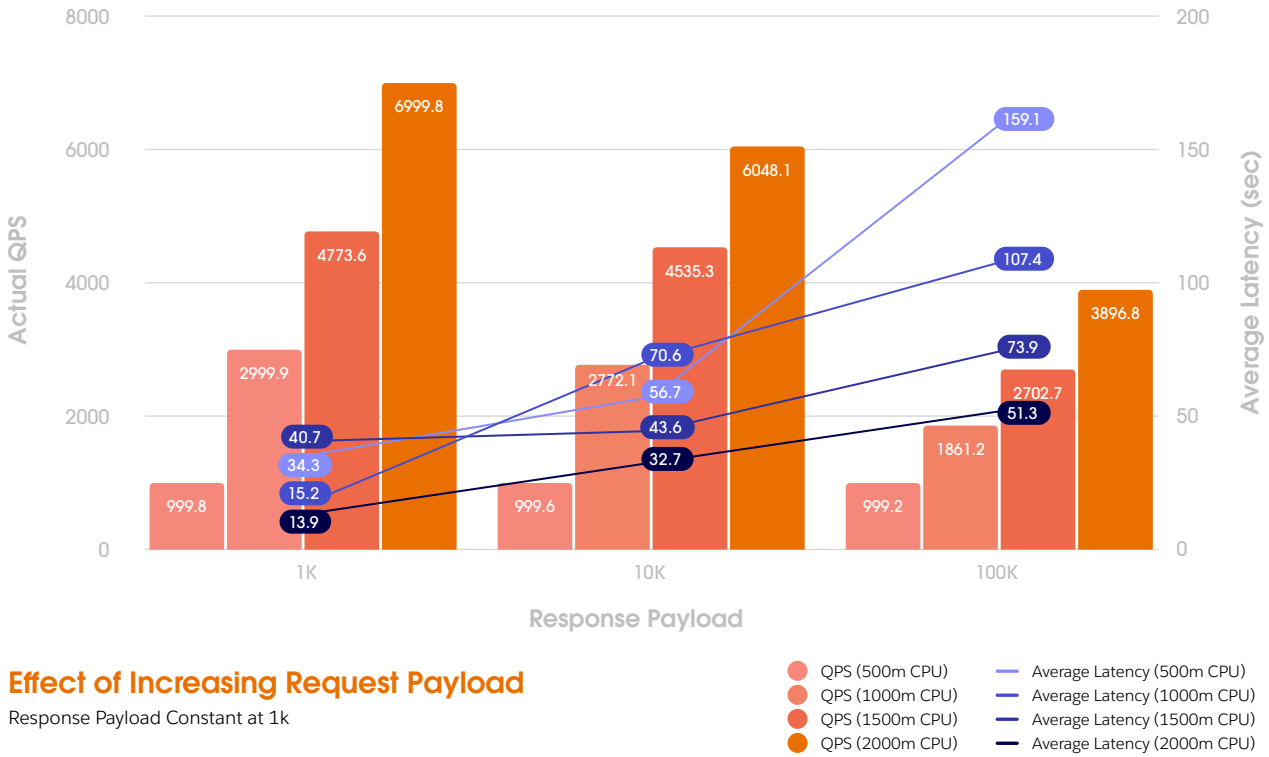
For both scenarios, the performance of the POST requests is heavily CPU-bound. Since there was a negligible effect on memory in our previous scenarios, the focus will be on the impact of payload sizes on CPU resources and queries per second processed.

For more information on Flex Gateway sizing, please see Appendix B: Flex Gateway Sizing.

Increase in Request Payload Size

In this scenario, the response payload size is kept constant at 1 KB. As payload size increases from 1 KB to 200 KB, queries are processed on average 60% less speed with a constant CPU size (Figure 7). There is the least QPS change between 1 KB and 10 KB response payload size, which is the typical payload size range in many use cases.

As illustrated, Flex Gateway's performance correlates highly with CPU resources. For example, at 100 KB payload size, we see a 250% increase in queries processed when the CPU was increased from 500 millicores to 2000 millicores.



Effect of Increasing Request Payload

Response Payload Constant at 1k

Figure 7: Effect of an increasing payload request.

Increase in Response Payload Size

In this scenario, the request payload size is kept constant at 1 KB. The results are similar to the increase in request payload size. As response payload size increases from 1 KB to 200 KB, queries are processed on average 70% less speed (see Figure 8).

CPU resources still correlate to the throughput with increasing response payload size. Flex Gateway's ability to process queries increases by about 85% as CPU resource increases – even with a higher response payload.

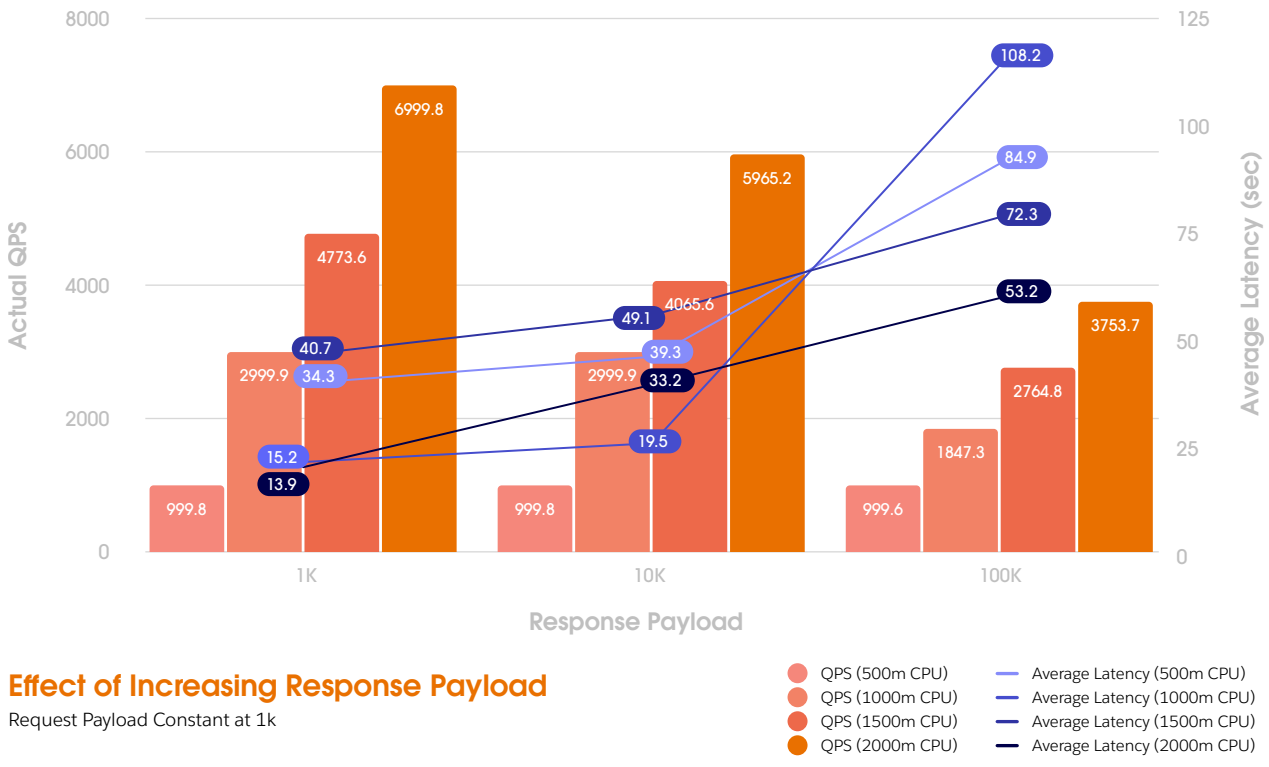


Figure 8: Effect of an increasing response payload.

Although environment conditions can vary significantly across deployments and use cases, IT teams can make educated decisions based on the findings in the two charts on how much CPU resources are needed.

Conclusion

From the various tested scenarios above, we found that Flex Gateway scales regardless of deployment architectures or the type of requests made. Flex Gateway provides a reliable and small-footprint gateway option with high throughput, low memory usage, and low latency predictability without fine-tuning. And with the CI/CD integration capabilities, Flex Gateway setup can be automated, saving developers additional time.

Our benchmark performance analysis finds that Anypoint Flex Gateway provides high throughput and low latency with low memory and CPU usage. Anypoint Flex Gateway scales to support the throughput required regardless of the request type or the underlying Kubernetes deployment architecture. The numbers shared demonstrate that Anypoint Flex Gateway is an apt choice for enterprises building highly performant applications.

With Flex Gateway, businesses can provide real-time user experiences with:

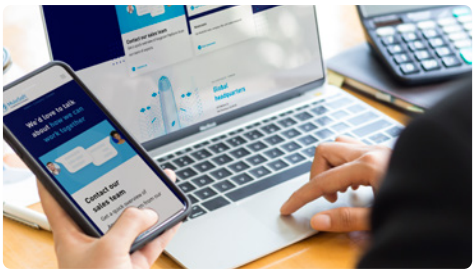
- → **Lower infrastructure and maintenance costs**
- → **Less time spent on fine-tuning the gateway for performance**
- → **The ability to service numerous concurrent requests**

The additional resources of time and money can then be spent on innovation and to support faster development cycles.

Anypoint Flex Gateway is a solution that allows developers to focus on building innovations to drive seamless user experiences. Flex Gateway also helps the organization meet business goals by enabling them to securely and quickly access the data they need and keeping the application lean.

It's time to enable your team with an API gateway solution that allows developers to focus on building innovations while making it easier for the organization to meet business goals by securely and quickly access the data they need. With MuleSoft Anypoint Flex Gateway, your organization can achieve success quickly and make an impact on your business now.

Learn more about Flex Gateway



CONTACT US

Start Your Security Journey Today

See how MuleSoft helps IT teams secure every API in the digital estate.

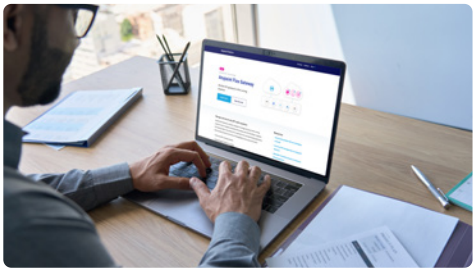
Get to know MuleSoft

ANYPOINT FLEX GATEWAY

Powerful, Quick, And Flexible

Manage and secure any API built anywhere with Anypoint Flex Gateway.

Start securing

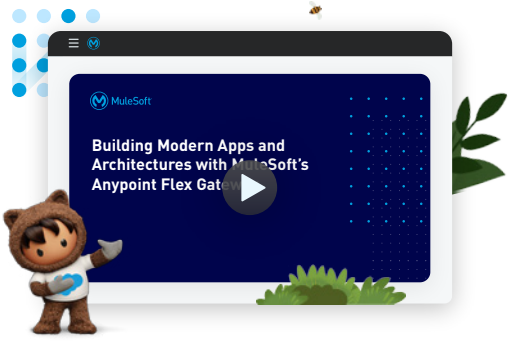


BLOG

Blog: See How Anypoint Flex Gateway Works

Follow along with this tutorial post to see Flex Gateway in action.

Read on



WEBINAR

Webinar: Build Modern Apps And Architectures With Mulesoft's Anypoint Flex Gateway

Watch and learn the benefits of a lightweight and ultrafast gateway to control that can secure any API deployed anywhere.

See the webinar

APPENDIX A

Deployment Architectures

The common setup elements for the benchmarks related to the deployment topologies section were run on AWS. Figure 9 shows the *standard deployment,* and Figure 10 shows the *sidecar deployment*.

To test the performance of Flex Gateway, a K8s cluster was provisioned in AWS EKS. The EKS was deployed in US-WEST-2 region with 3 r5.xlarge nodes, 4vCPU, and 32 GB. Anypoint Flex Gateway 1.0.1 was used for these particular tests with limits of 2000 millicore CPU and 4 GiB memory. Fortio 1.38.0 comes with both a load generator and an echo server which were installed on separate nodes. The echo server had limits of 2000 millicore CPU and 1 GiB memory.

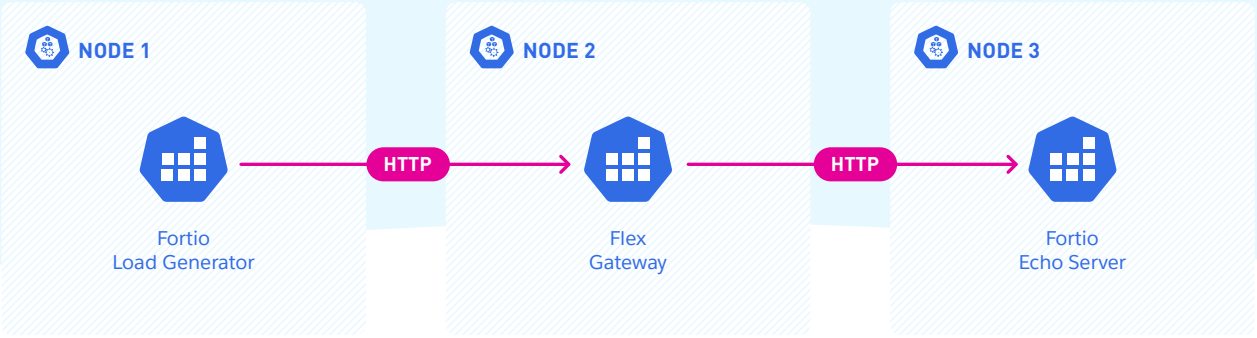


Figure 9: *Standard deployment* architecture.

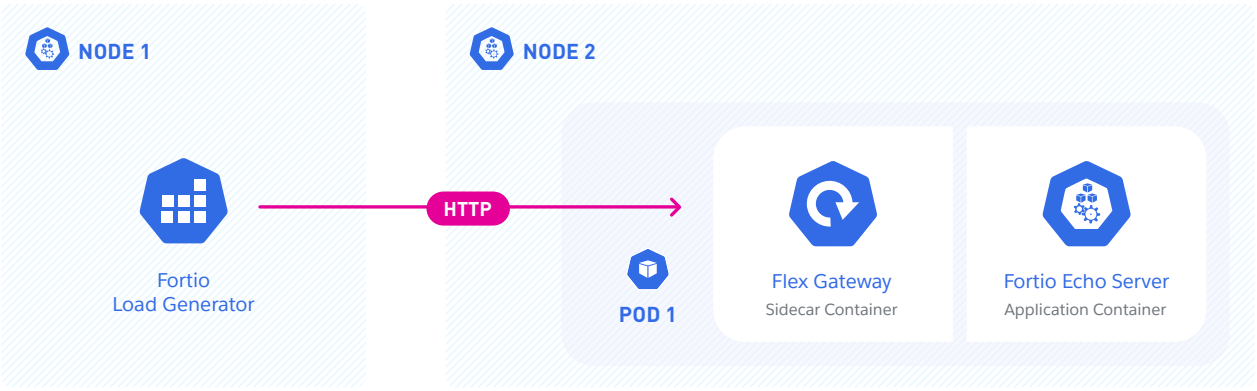


Figure 10: *Sidecar deployment* architecture.

APPENDIX B

Flex Gateway Sizing

The following section describes the maximum queries processed based on CPU resources and the effect on CPU resources based on response and request payload sizes.

We tested in *standard deployment* using CPU resources, memory size, and throughput variables. Please see Appendix C for the environment architecture utilized to get the results in this section.

Maximum Processed Queries Based on CPU Resources

To test the maximum capacity of Flex Gateway, GET requests of 1 KB response payload size were utilized. The maximum processed queries per second can be seen in Table 1 for each of the different CPU resources given to Flex Gateway. For every CPU resource increase by 500 millicores, the processing capacity increases by, on average, 92%. Assigning different memory capacities does not impact the gateway's performance; therefore, the focus was on CPU resources.

Table 1: *Maximum Achieved Throughput based on CPU Resource*

CPU Resource in Millicores	Maximum Processed Queries per Second	Capacity Improvement
250	948	Base – 100%
500	1784	188%
1000	3441	363%
1500	5206	549%
2000	7000	738%

For every CPU resource increase by 500 millicores, the processing capacity increases by, on average, 92%.

Note: For a high throughput scenario, the user might choose to enable multiple replicas with lower CPU millicores or choose a high CPU millicore with fewer replicas. A higher policy load will reduce throughput as the replica will need to process each REST call, and it will impact throughput.

APPENDIX C

Architecture for Sizing Performance

The common setup elements for the benchmarks related to the sizing performance section were run on AWS with a structure similar to Figure 11.

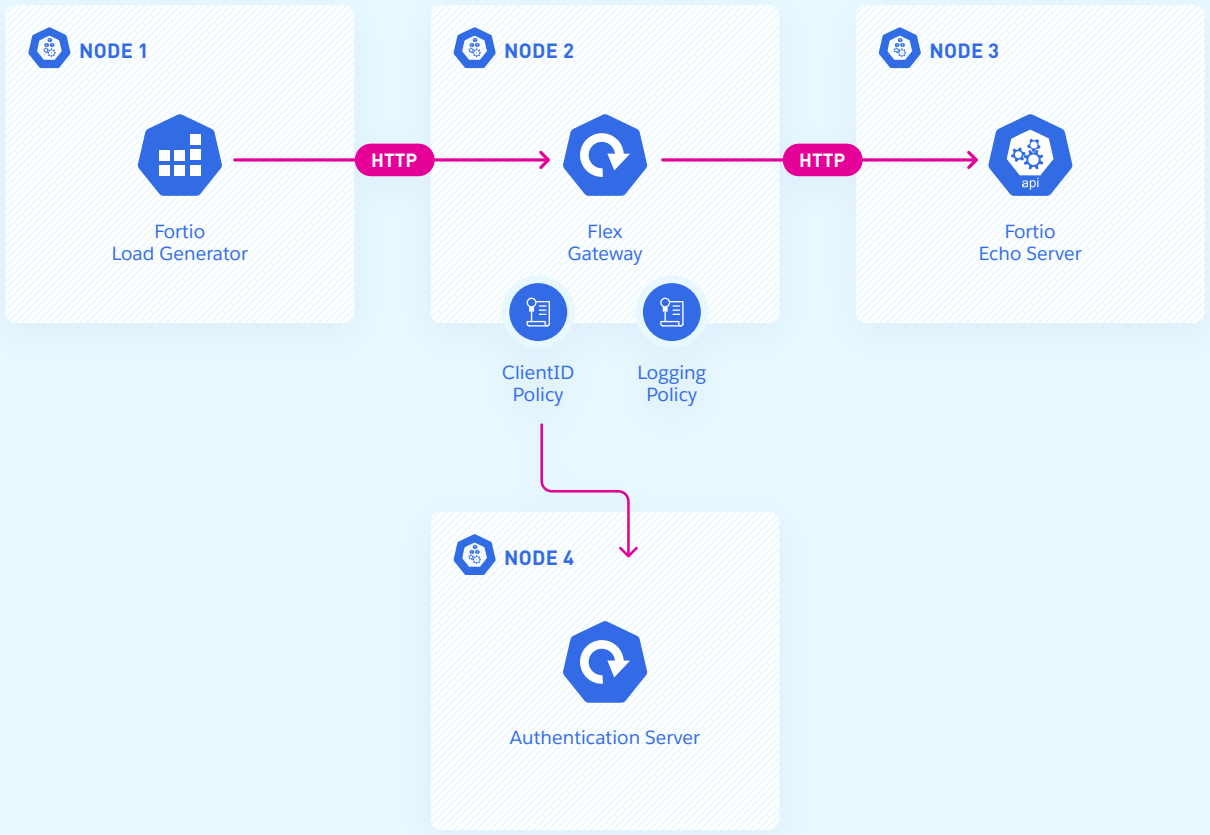


Figure 11: Sizing performance architecture.

To test the performance of Flex Gateway, a K8s cluster was provisioned in AWS EKS. The EKS was deployed in US-WEST-2 region with 4 r5.xlarge nodes, 4vCPU, and 32 GB. Anypoint Flex Gateway 1.0.1 was used for these particular tests with resource limits varying from 250m to 2000 millicore CPU and from 250 MB to 4 GB of memory. Fortio 1.38.0 comes with both a load generator and an echo server which were installed on separate nodes. The echo server had limits of 2000 millicore CPU and 1 GiB memory.

Different from deployment topologies tests, clientID and access log policies were enforced. Therefore, an authentication server was added to verify the client information. Access log policy allowed us to track the metrics.

APPENDIX D

Disclaimer

The MuleSoft Performance Engineering team collected the numbers presented in this report. The metrics were collected in a controlled environment and should not be explicitly used to size deployments. Flex Gateway performance will vary based on:

- → Type of deployments such as containers, VM, or bare metal
- → Number of CPU Cores
- → Number of applied policies
- → Size of response payload
- → Backend service response time

Before using Flex Gateway in production, organizations should run tests based on their own criteria and use cases to consider the trade-offs. If you need help with Flex Gateway sizing, we recommend getting in touch with a MuleSoft representative.

Salesforce, the global CRM leader, empowers companies of every size and industry to digitally transform and create a 360° view of their customers. For more information about Salesforce (NYSE: CRM), visit salesforce.com.

Any unreleased services or features referenced in this or other press releases or public statements are not currently available and may not be delivered on time or at all. Customers who purchase Salesforce applications should make their purchase decisions based upon features that are currently available. Salesforce has headquarters in San Francisco, with offices in Europe and Asia, and trades on the New York Stock Exchange under the ticker symbol "CRM."

For more information please visit salesforce.com, or call 1-800-NO-SOFTWARE.