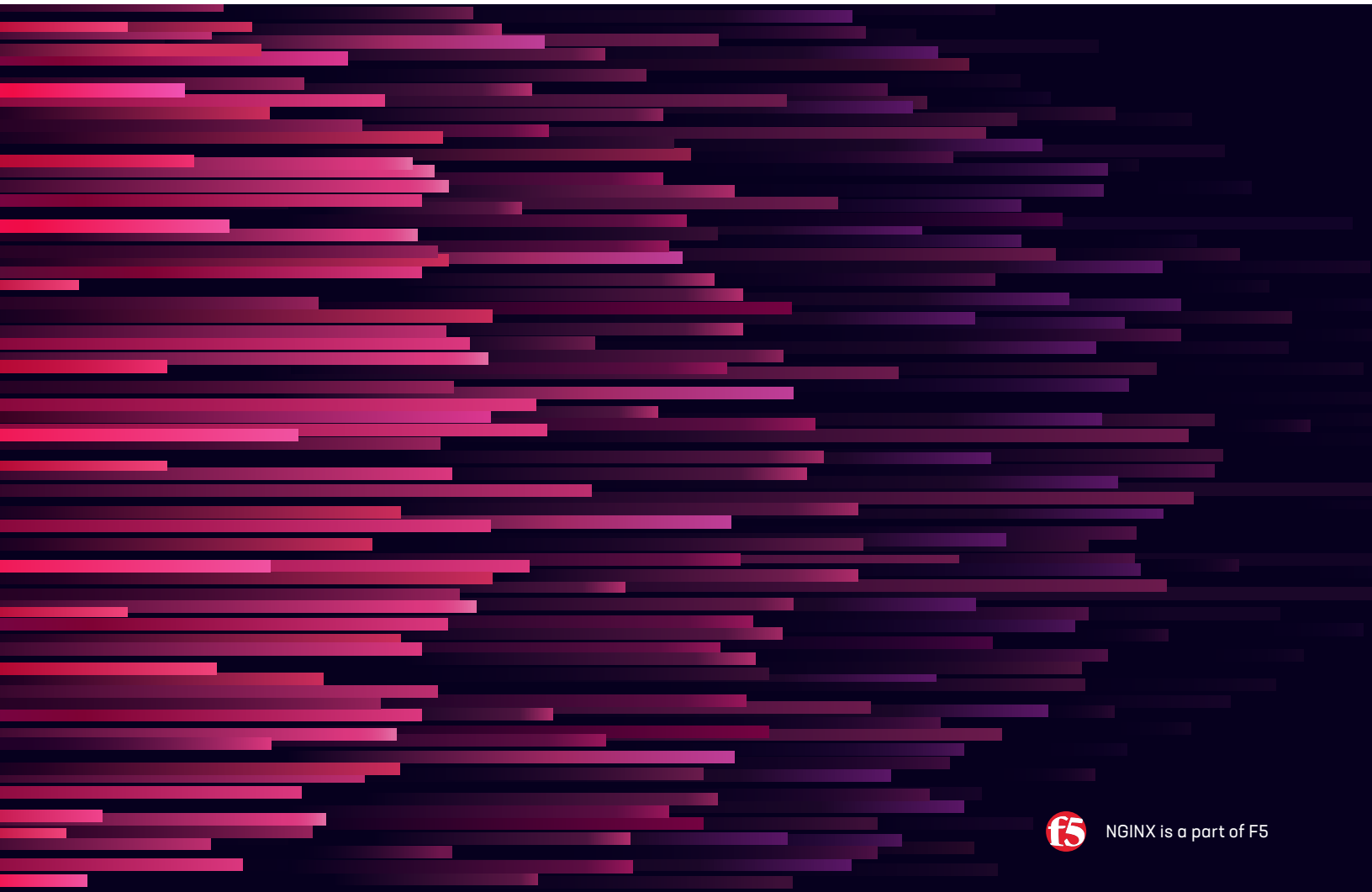
NGINX
Part of F5

EBOOK

The NGINX Real-Time API Handbook

by Karthik Krishnaswamy, Director, Product Marketing, F5, Inc.
and Alessandro Fael García, Technical Marketing Manager, F5, Inc.



NGINX is a part of F5

Table of Contents

Foreword

APIs play a central role in powering digital experiences. Whether you're ordering food on a delivery app or making payments online with your bank's mobile application, APIs are the key software component behind a successful outcome. According to ProgrammableWeb – a leading source of news and information about Internet-based APIs, including a directory of public APIs used in web and mobile applications – there has been extraordinary growth in the number of APIs over the last few years: on average 168 new APIs are added to the directory every month. In fact, APIs represent 83% of all web traffic according to the Akamai State of the Internet Security Report.

Developing APIs is often the first step in an enterprise's journey toward offering digital products and solutions. The performance of these APIs – delivering API responses rapidly – is crucial to the enterprise's ability to offer compelling digital experiences. High-volume and low-latency API transactions are increasingly becoming the standard. Just as with applications, poor API performance results in a poor experience for end users. Your brand gets associated with poor quality and your customers switch to your competitors – possibly just at the point these customers are switching online and embracing technologically innovative solutions.

This eBook shines a light on the criticality of delivering high-performance APIs – what we call real-time APIs. It starts with a brief but engaging tour of the role APIs play in our daily lives, then covers performance benchmarks of various API management solutions, a reference architecture, and a tool for measuring API performance. Our customers and prospects who are API practitioners in a wide variety of roles – including DevOps engineers, application developers, API architects, API product owners, and network engineers on Infrastructure & Operations teams – tell us this is just the sort of information they need to succeed and stay ahead of their competitors.

NGINX has a lot to offer to API professionals. NGINX software powers more than 450 million websites and, according to our 2019 user survey, more than 47% of NGINX Plus (and 35% of NGINX Open Source) users are already using NGINX as an API gateway. NGINX is also a popular component in many traditional API management solutions, providing the underlying gateway for Axway, Kong, MuleSoft, Red Hat 3Scale, and others. We have developed a radically innovative API management solution that is built for performance, with no runtime dependency between the data plane (NGINX Plus as the API gateway) and the control plane (the NGINX Controller API Management Module). This reduces complexity and maximizes performance by reducing the average response time to serve an API call.

We sincerely hope you enjoy this eBook as you embark on your API journey and plan to manage and offer real-time APIs.

Daniel Henley
RVP, Solutions Engineering, F5, Inc.

1. How Real-Time APIs Power Our Lives

The other day I went to dinner and it made me appreciate the need for fast application programming interfaces (APIs). Confused? Let me explain.

To get to dinner I used an app to hail a car from my smartphone. Most of us are familiar with this routine – you open the app, enter your destination, get a ride, step out of the car when you arrive, and pay for your trip automatically using the credit card on file. While you're waiting for the driver to pick you up, a map updates in real time to indicate the location of the car on approach. But on that day, the app did not update the map. I waited five increasingly uncomfortable minutes, not knowing if a driver was on the way or had even accepted my request. After 10 minutes, I got frustrated and switched to an alternative ride-hailing app! This time I was successful and watched in real time as my driver approached and picked me up. I made it to dinner with a few minutes to spare.

REAL-TIME INFORMATION IS CRITICAL TO TODAY'S DIGITAL EXPERIENCES.

The takeaway? Real-time information is critical to today's digital experiences. Let's look at another example.

I recently checked out an Amazon Go store in San Francisco. What a marvelous experience that was! With the Go app downloaded to your smart phone, you just approach the door and it unlocks automatically. As you walk around the store, any item you pick up is automatically added to your virtual cart, and automatically removed if you put it back on the shelf. When you are done, you just walk out! Your card on file is charged and you receive a receipt by email. No lines, no checkout counter, no delays, and no cashiers. It's an easy, effortless in-store shopping experience. Amazon Go accomplishes this by connecting a series of real-time systems, including ones for computer vision, sensor fusion, and deep learning technologies.

Yet again, we see real-time information is critical to a good experience. And that, evidently, I don't like taking my credit card out of my wallet.

APIs ARE THE CONNECTIVE TISSUE OF GOOD DIGITAL EXPERIENCES

What's the technology powering such convenient, and thus satisfying, consumer experiences? APIs! Specifically, real-time APIs.

In the case of the ride-hailing app, being able to track the approaching car's location in real time has benefits that are somewhat intangible – it assures the impatient (and aren't we all?) rider that progress is being made, gives you something to watch as you wait (if you're bored), and let's you know when arrival is imminent so you can start looking for the car. On a more practical note, you can tell when the driver has taken a wrong turn and get in touch to help course correct (this has actually happened to me). During the actual journey, both you and the driver can see exactly where you are and can extrapolate how much longer the trip will take.

And finally, paying with a card on file is much more convenient than paying a cab driver (by either cash or card), and you don't have to calculate a tip.

In the case of the Amazon Go store, the entire experience is predicated on real-time processing: as you enter the store you're identified immediately and your credit card info accessed so it's ready even if you just grab one item and dash out. The Go app keeps a running tally as you take or replace items on the shelves, so you don't have to wait for it to calculate the total cost before you leave the store. You don't even have to get out your wallet (or worse, wait while the person in front of you watches idly as items get rung up and bagged, and only then starts rummaging through his or her backpack looking for the wallet). Any processing delay that interferes with this frictionless shopping experience, and you might as well walk down the block to a traditional store.

There's a lot riding (pun intended) on consumers having good, real-time experiences. The barrier to switching to a competitor in the digital world is very low. That's why companies like Uber and Amazon invest significantly in ensuring they retrieve information in real time. In fact, Uber relies on Google Maps to provide directions for its drivers and render visualizations for its riders. What does Uber pay to deliver this value? Over a two-year period they paid Google $58M.

WHAT DOES "REAL TIME" MEAN? LESS THAN 30 MILLISECONDS

RESEARCH SUGGESTS REAL TIME HAS TO BE LESS THAN 30 MILLISECONDS.

Real-time transaction processing and analytics have become an important building block for compelling digital experiences. How do we characterize what real-time is? Research suggests real time has to be less than 30 milliseconds (ms).

Consider these proof points:

- **13ms for humans to process images.** According to a study by neuroscientists at MIT, the human eye can process and identify entire images in as little as 13ms. Researchers asked subjects to identify a series of six to twelve images, each presented for between 13 and 80ms. The fastest rate at which subjects identified these images was 13ms.
- **20ms to synchronize video.** An IEEE paper about using 'Media fingerprinting' technology to prevent loss of synchronization between image and sound when delivering video content states that the tolerable limit for latency is between 6 and 20ms.
- **30ms to deliver wireless data.** Ubiquitous connectivity is on the horizon with the introduction of 5G technologies, which promises peak speeds of up to 1 Gbps and latency of less than 30ms. Why such low latency? Because 5G needs to be that fast to replace in-home WiFi and wireline broadband like fiber and cable modems.

Real time is all about minimal delay, and the tolerance for latency is getting lower and lower. It's fair to say that to be perceived as real time, processing has to take no more than 30ms, and in some cases as low as 13ms.

More specifically, the 30ms latency threshold must be maintained all the way up to the 99th percentile of processed requests (meaning that on average only 1 request in 100 takes longer than 30ms). To learn why this is important, see next chapter.

REAL-TIME EXPERIENCES REQUIRE REAL-TIME APIs

I opened by saying my dinner made me think of APIs. Let's connect those dots.

Real-time experiences rely on API connectivity. Uber retrieves Google Map data via an API call. Amazon connects in-store Go infrastructure with sensor, vision, and analytics capabilities via API calls.

A REAL-TIME EXPERIENCE IS HEAVILY RELIANT ON REAL-TIME API CALLS.

It stands to reason that a real-time experience is heavily reliant on real-time API calls. That means your API infrastructure needs to process API calls in 30ms or less. For some use cases, you need as little as 6ms!

That might not sound difficult, but let's consider that API infrastructure has to:

- **Route APIs.** Ensure the API consumer – like your ride-hailing app – is correctly directed to the right backend resource, like the mapping service.
- **Authenticate APIs.** Is this API consumer a valid user allowed to access this backend resource? You have to authenticate the user to ensure it is.
- **Secure APIs.** Encryption is table stakes these days. APIs are the gateway to what is arguably your most critical application capital, so they better be secure.
- **Shape APIs.** Not all API calls are equal. You need a way to shape the traffic to avoid resource contention, provide proper bandwidth, and prioritize certain API calls.
- **Cache APIs.** Many companies process billions of API calls per day. How do you handle that volume? Caching API responses is one way to boost performance.

Thus, we define a real-time API as: ***One that can process an API call end-to-end in less than 30ms.***

And we define real-time API infrastructure as: ***The technology to route, authenticate, secure, shape, and cache APIs in less than 30ms.***

EVERYDAY USE CASES FOR REAL-TIME APIs

There are plenty of activities in the digital world that harness the benefits of real-time APIs. Let me go through some of these.

Chat

Why real-time APIs are needed: We have all used chat apps. Nowadays chat seems to be the preferred customer-service platform for many enterprises, especially in B2C verticals. Chat has replaced 1-800 customer-service numbers! Many chat applications are site plug-ins that require API calls to a third-party chat tool or bot.

How not having real-time APIs affects your business: Low latency for your chat apps is a crucial factor in the customer experience. Without real-time API calls you will lower customer satisfaction, which can impact revenue and decrease your net promoter score (NPS).

Fraud Detection

Why real-time APIs are needed: The financial services sector employs fraud detection technologies on credit card transactions *at the point of sale*. These enterprises process large amounts of data and use predictive/forensic analytics to detect any outliers. For example, are there a large number of high-value transactions occurring within a short span of time? Is your card being used far away from where you usually shop? Such transactions are flagged as deviations from the cardholder's usual purchase pattern and the card processor might either block the transaction or call the cardholder for explicit confirmation of the purchase. All this needs to happen in real time, when the customer is at the point of sale.

How not having real-time APIs affects your business: Chances are your cardholders have more than one credit card. If your transaction takes too long, or gets denied, then they'll simply use another card. You need real-time APIs to prevent customers from using someone else's products and services.

IoT

Why real-time APIs are needed: IoT is changing our lives at incredible speeds! Let me illustrate a few examples where APIs are used in our everyday lives, and must be done in real time:

- **Home automation:** Do you have a remote that you can talk to? What about a home automation device powered by Alexa or Siri? A good experience requires real-time voice calls to control everything from streaming video to smart blinds that automatically adjust for optimal sunlight and privacy depending on the time of day.

- **Medical device monitoring:** IoT is increasingly being used to monitor patient health – from cancer to diabetes. A variety of vital parameters such as glucose levels or blood pressure are recorded on a daily basis. Updates are then sent to the patients' physician who administers appropriate treatment. Accomplishing this in real time is paramount for improving patient outcomes.
- **Driverless cars:** They're coming! This technology uses a variety of sensors and software to control, navigate, and drive the vehicle. Key decisions about the best route to take and when and where to stop to avoid or minimize collisions all have to be taken in an instant by analyzing large amounts of data collected by the sensors.

How not having real-time APIs affects your business: A lack of real-time APIs can prevent adoption of disruptive services like voice-controlled smart devices, in-home medical care, and driverless cars. Preventing these new services from reaching their full potential stalls revenue and market expansion.

NGINX IS THE PROVEN FOUNDATION FOR REAL-TIME API INFRASTRUCTURE

DELIVERING TRANSFORMATIVE EXPERIENCES REQUIRES A HIGH-PERFORMANCE API MANAGEMENT SOLUTION.

Delivering transformative experiences requires a high-performance API management solution. NGINX powers more than 450 million websites and, according to our 2019 user survey, more than 47% of NGINX Plus (and 35% of NGINX Open Source) users are already using NGINX as an API gateway, the foundational layer in any API management solution. That doesn't even include all of the solutions where NGINX is a key component, such as traditional API management providers Axway, Kong, MuleSoft, Red Hat 3Scale, and others.

Why is NGINX foundational to all this API infrastructure? We reliably see NGINX process API calls in less than 30ms, with customers telling us 6ms is common. It's the only real-time API solution on the market.

2. Benchmarking API Management Solutions from NGINX, Kong, and Amazon: Do They Deliver APIs in Real Time?

Speed – it's key in today's digital landscape, where consumers can easily switch to a competitor if your app's performance is too slow. App speed is ultimately driven by responsive, healthy, and adaptable APIs, and one of the critical factors in API responsiveness is the latency introduced by your API gateway. But not all API gateways perform at the same level.

That point was brought home to us last fall when an NGINX customer – a major player in the consumer credit industry – told us about the increasing importance of "real-time" API performance as more and more apps and other components need to communicate to provide the digital experience users expect. We were highly gratified to learn that NGINX Plus was the only API gateway that achieved the super-fast API latencies – as low as 10 milliseconds (ms) – the customer needed. Many other customers, such as Capital One, have also shared with us how they cut latency and improved throughput with NGINX Open Source or NGINX Plus as their API gateway.

We decided to look deeper into the API ecosystem and try to figure out what makes an API "real-time". Based on a number of factors, we determined that a real-time API must process API calls end-to-end in less than 30ms at every percentile through the 99th (meaning that only 1 call in 100 takes longer than 30ms).

COMPARING API MANAGEMENT SOLUTIONS

Our own testing consistently shows that it's easy to achieve real-time API performance with our API management solution, which combines NGINX Plus as the API gateway for processing API calls and the NGINX Controller API Management Module for managing both NGINX Plus instances and your APIs as you define, publish, manage, and monitor them across their full lifecycle.

But we understand that you might not want to take our word for it. So we commissioned GigaOm, an independent technical research and analysis firm, for objective and transparent benchmarking of our API management solution and other popular solutions on the market: two solutions that like NGINX can be deployed on premises or in the cloud, Apigee* and Kong Enterprise, and two fully managed cloud offerings, Amazon API Gateway and Kong Cloud.

NGINX PLUS DELIVERED APIs IN REAL TIME IN EVERY TESTED CONDITION, AND THE OTHER SOLUTIONS DIDN'T.

In this chapter we summarize the results of GigaOm's testing (spoiler: NGINX Plus delivered APIs in real time in every tested condition, and the other solutions didn't). For all the details on the solutions, the testing methodology, and the results, download the free GigaOm report.

***Note: The Apigee end user license agreement (EULA) prohibits publication of testing results without Google's express permission, so unfortunately neither the report nor this chapter includes information about Apigee.**

BENCHMARK OVERVIEW

GigaOm used the Vegeta HTTP load-testing tool to generate requests (API calls), and measured how much latency – the amount of time it took to return the response to an API call – the API gateway introduced at various numbers of requests per second (RPS), which GigaOm calls "attack rates". GigaOm performed test runs with attack rates starting at 1,000 RPS and scaling up through 5,000, 10,000, 20,000, and so on until Vegeta reported error status codes. Each test run lasted 60 seconds and was repeated 3 times. As shown in the graphs below, GigaOm captured the latencies at the 50th, 90th, 95th, 99th, 99.9th, and 99.99th percentiles and also recorded the longest latency observed during the test run (**Max** in the graphs).

RESULTS: NGINX VS. KONG ENTERPRISE

GigaOm conducted two benchmarks comparing NGINX Plus (as deployed using NGINX Controller) and Kong Node (as deployed using Kong Enterprise). In the first benchmark, there was a single worker node (one instance of NGINX Plus or Kong Node). In the second, there were three worker nodes load balanced by NGINX Open Source in Round-Robin mode. (GigaOm stresses that using NGINX Open Source as the load balancer didn't create an advantage for NGINX Plus, and even Kong recommends it as the load balancer to use for clustered Kong Node instances.)

AT THE 99.99th PERCENTILE KONG'S LATENCY IS TRIPLE OR DOUBLE NGINX'S.

As shown in the following graphs, the difference in latency between NGINX and Kong is negligible up to the 99th percentile, at which point Kong's latency starts to grow exponentially. At the 99.99th percentile Kong's latency is triple or double NGINX's in the two benchmarks respectively.

The 99th percentile is a decent minimum value for defining an API as real-time, but GigaOm notes that in real-world deployments it's "extremely important" to maintain low latency at higher percentiles like the 99.9th and 99.99th. The report explains:

> Latency results tend to be multi-modal over time, with the tops of the spikes representing "hiccups" in response times.
>
> These hiccups matter. If the median response time or latency is less than 30 milliseconds, but there are hiccups of 1 second or greater latencies, th[ere] is actually a cumulative effect on subsequent user experience. For example, if you visit a fast food drive through where the median wait time for food is 1 minute, you would probably think that was a good customer experience. However, what if the customer in front of you has a problem with their order and it takes 10 minutes to resolve? Your wait time would actually be 11 minutes. Because your request came in line after the hiccup, the delayed 99.99th percentile's delay can become your delay too.

The negative effect of unusually large latencies at high percentiles becomes even more significant in distributed applications where a single client request might actually spawn multiple downstream API calls. For example, suppose that 1 client request creates 10 API calls to a subsystem with a 1% probability of responding slowly. It can be shown mathematically that as a result there is almost a 10% probability the 1 client request will be affected by a slow response. For details on the math, see Who moved my 99th percentile latency?

Figure 1 depicts the results for the 30,000 RPS attack rate with a single worker node. At the 99.99th percentile Kong's latency is more than 3 times NGINX's, and exceeds the 30ms threshold for real-time APIs. In contrast, NGINX Plus achieves real-time latency at every percentile – even its highest recorded (**Max**) latency of 13ms is less than half the real-time threshold.

Figure 1: NGINX Controller and Kong Enterprise at 30,000 RPS with 1 Node

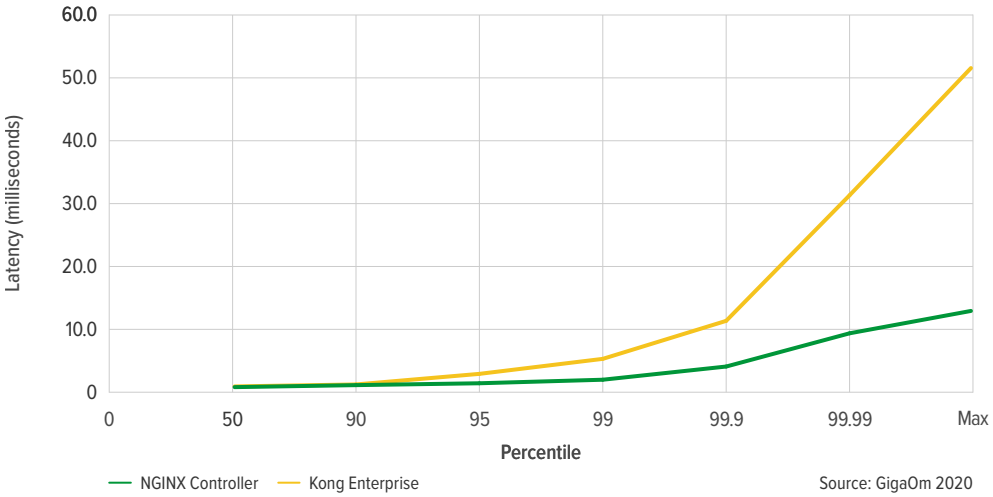
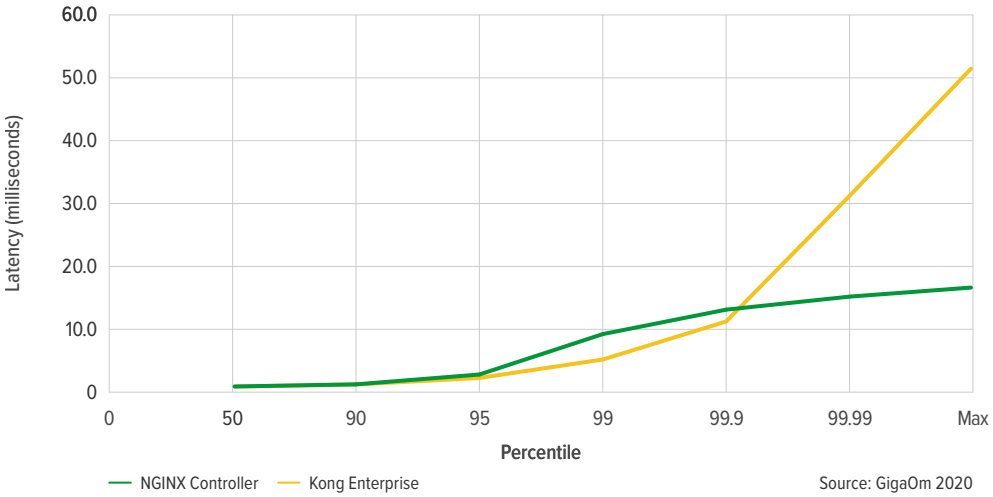


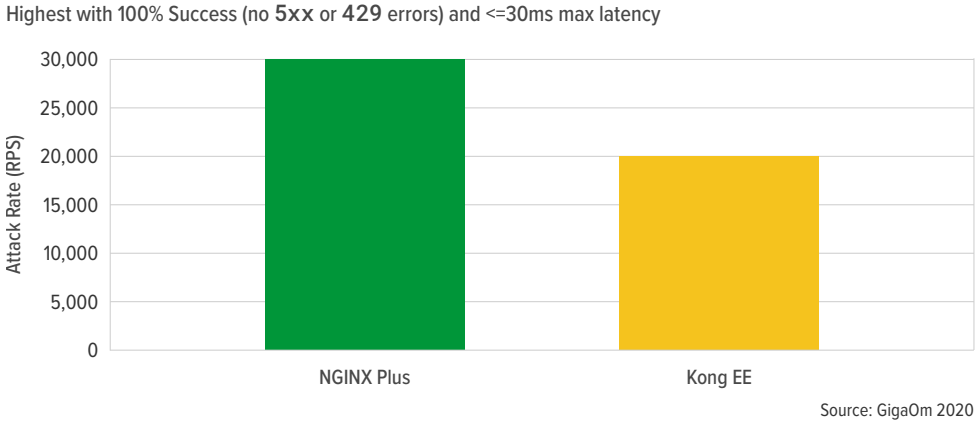
Figure 2 shows very similar results for the benchmark with three worker nodes, also at the 30,000 RPS attack rate. Interestingly, Kong outperforms NGINX at the 99th and 99.9th percentiles, but once again suffers a huge latency spike at the 99.99th percentile, this time hitting a latency that's about double NGINX's. As in the first benchmark, NGINX stays below the 30ms real-time threshold at all percentiles.

Figure 2: NGINX Controller and Kong Enterprise at 30,000 RPS with 3 Nodes



Another way to gauge an API gateway's performance is to determine the maximum RPS it can process with 100% success (no `5xx` or `429 [Too Many Requests]` errors) and latency below 30ms at all percentiles, for both the single-node and three-node configurations. Figure 3 shows how by this measure, NGINX sustains 50% higher RPS than Kong: 30,000 versus 20,000.

Figure 3: Maximum throughput achieved without errors



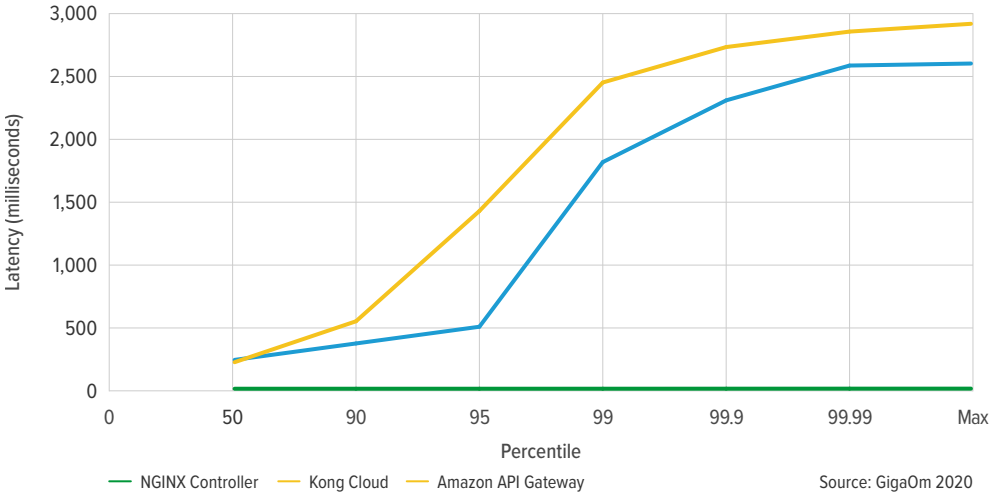
Source: GigaOm 2020

RESULTS: NGINX VS. KONG CLOUD AND AMAZON API GATEWAY

In the third set of benchmarks, GigaOm compared NGINX Plus to Kong Cloud and Amazon API Gateway. GigaOm stresses that a direct comparison is highly problematic because Kong Cloud and Amazon API Gateway are fully managed SaaS offerings, while NGINX Controller is a PaaS offering and not currently available as SaaS. In particular, neither SaaS offering reveals the instance type, compute power, memory, or networking capabilities it uses. GigaOm therefore had to make a best guess at the comparable settings to make for NGINX Plus.

Even setting aside comparison with NGINX Plus, it's immediately clear that the SaaS offerings cannot deliver APIs in real time at any of the tested percentiles, even at the second-lowest attack rate (5,000 RPS) shown in Figure 4. At the 50th percentile, the SaaS offerings' latency is already more than 7 times the 30ms threshold; at the 99.99th percentile, it exceeds that threshold by more than 8000%. The obvious implication is that there are no conditions under which Kong Cloud or Amazon API Gateway achieves 100% success with latency under 30ms.

Figure 4: NGINX Controller, Kong Cloud, and Amazon API Gateway at 5,000 RPS with 1 node



VALIDATING NGINX AS THE ONLY REAL-TIME API SOLUTION

NGINX IS THE ONLY SOLUTION TESTED BY GIGAOM THAT MET THE STANDARDS OF REAL-TIME API PROCESSING.

In summary, NGINX is the only solution tested by GigaOm that met the standards of real-time API processing, achieving less than 30ms of latency at every percentile. Kong Enterprise attains real-time performance at the 99th percentile, but its latency spikes significantly at higher percentiles, making it ill-suited for production environments that require even a moderate volume of real-time API processing. Neither of the tested SaaS solutions can be categorized as real-time.

The GigaOm report validates both our previous benchmarks and what we hear from our customers. NGINX Plus is the fastest API gateway in the market and the only API solution capable of sustaining a latency of less than 30ms at all percentiles. And if you pair it with NGINX Controller, you gain access to a uniquely architected API management solution where because of careful decoupling there is no impact whatsoever on the performance of the API gateway data plane (NGINX Plus) from the API management control plane (NGINX Controller).

Here we've summarized the high-level results from the GigaOm report. For complete details, download the complete report for free.

3. A Real-Time API Reference Architecture

In Chapter 1 we showed how real-time APIs play a critical role in our lives. As companies seek to compete in the digital era, APIs become a critical IT and business resource. Architecting the right underlying infrastructure ensures not only that your APIs are stable and secure, but also that they qualify as real-time APIs, able to process API calls end-to-end within 30 milliseconds (ms).

API architectures are broadly broken up into two components: the data plane, or API gateway, and the control plane, which includes policy and developer portal servers. A real-time API architecture depends mostly on the API gateway, which acts as a proxy to process API traffic. It's the critical link in the performance chain.

API gateways perform a variety of functions including authenticating API calls, routing requests to the right backends, applying rate limits to prevent overburdening your systems, and handling errors and exceptions. Once you have decided to implement real-time APIs, what are the key characteristics of the API gateway architecture? How do you deploy API gateways?

THE REAL-TIME API REFERENCE ARCHITECTURE IS BASED ON NGINX'S WORK WITH OUR LARGEST, MOST DEMANDING CUSTOMERS.

This chapter addresses these questions, providing a real-time API reference architecture based on NGINX's work with our largest, most demanding customers. We encompass all aspects of the API management solution but go deeper on the API gateway which is responsible for ensuring real-time performance thresholds are met.

THE REAL-TIME API REFERENCE ARCHITECTURE

Our reference architecture for real-time APIs has six components:

1. API gateway. A fast, lightweight data-plane component that processes API traffic. This is the most critical component in the real-time architecture.
2. API policy server. A decoupled server that configures API gateways, as well as supplying API lifecycle management policies.
3. API developer portal. A decoupled web server that provides documentation for rapid onboarding for developers who use the API.
4. API security service. A separate web application firewall (WAF) and fraud detection component that provides security beyond the basic security mechanisms built into the API gateway.
5. API identity service. A separate service that sets authentication and authorization policies for identity and access management and integrates with the API gateway and policy servers.
6. DevOps tooling. A separate set of tools to integrate API management into CI/CD and developer pipelines.

Of course, there are elements like the API consumer, the API endpoint, and various infrastructure components like routers, switches, VMs, containers, and whatnot. These are included in the reference architecture where needed but we're not discussing them in detail as they're assumed to be common infrastructure in the enterprise. Instead, we're focusing on these six components, which are needed to create a comprehensive real-time API architecture.

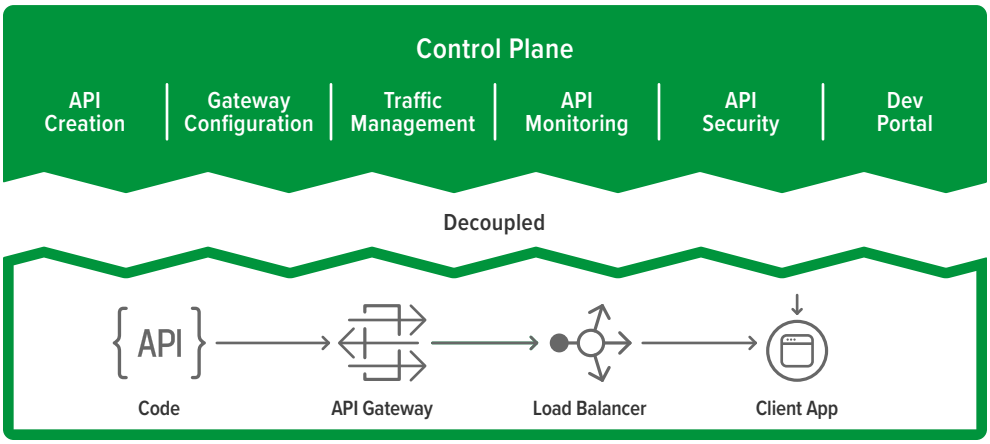


Figure 5. Decoupled architecture which isolates the data plane (API gateway) from the control plane to eliminate administrative overhead from API call processing.

API Gateway

The functions of an API gateway include request routing, authentication, TLS termination, rate limiting, and service discovery when deployed in a microservices architecture. To ensure high availability, a cluster of API gateways needs to be deployed to manage traffic for each application that exposes APIs. To efficiently distribute API traffic, a load balancer is deployed in front of the cluster. The load balancer selects the gateway based on location (nearness to the client app) or capacity to process APIs. The gateway then forwards the request to the backend. For best performance, implement the best practices detailed in Real-Time API Gateway Characteristics and Architectural Guidelines below.

API Policy Server

This is the control plane that allows developers and DevOps teams to define, publish, and secure APIs, and IT Operations to monitor and analyze APIs. This is where you configure request routing to backend services and fine-grained access control policies that specify what's allowed on a published API (read-only versus read-write, for example) and the kinds of users or client apps that are permitted to consume resources.

The policy server also configures the API gateways with all the APIs that need to be exposed for use. To achieve real-time performance, this component must be completely decoupled from the API gateway data plane that mediates API traffic (see Figure 6). If the data plane relies on the policy server for authenticating, rate limiting, and routing each API call to the appropriate backend, then API calls initiated by the API consumer traverse through this control plane – and in some cases associated databases – resulting in extra latency due to the additional overhead.

API Developer Portal

This component enables developers who consume your APIs to onboard in an efficient manner. A developer portal (or *dev portal*) provides a catalog of all the published APIs, documentation for each API, and sample code. To improve performance and availability, the developer portal is decoupled from the control plane and hosted separately on its own web server. A distributed developer portal enables multiple instances to be located across different clouds, geographies, or availability zones. This can improve performance for developers accessing it, but because it's decoupled form the data plane, it does not affect the efficiency of API call processing.

API Security Service

API security most commonly entails an advanced web application firewall (WAF) to detect a variety of attacks. It must provide high-confidence signatures and be able to prevent

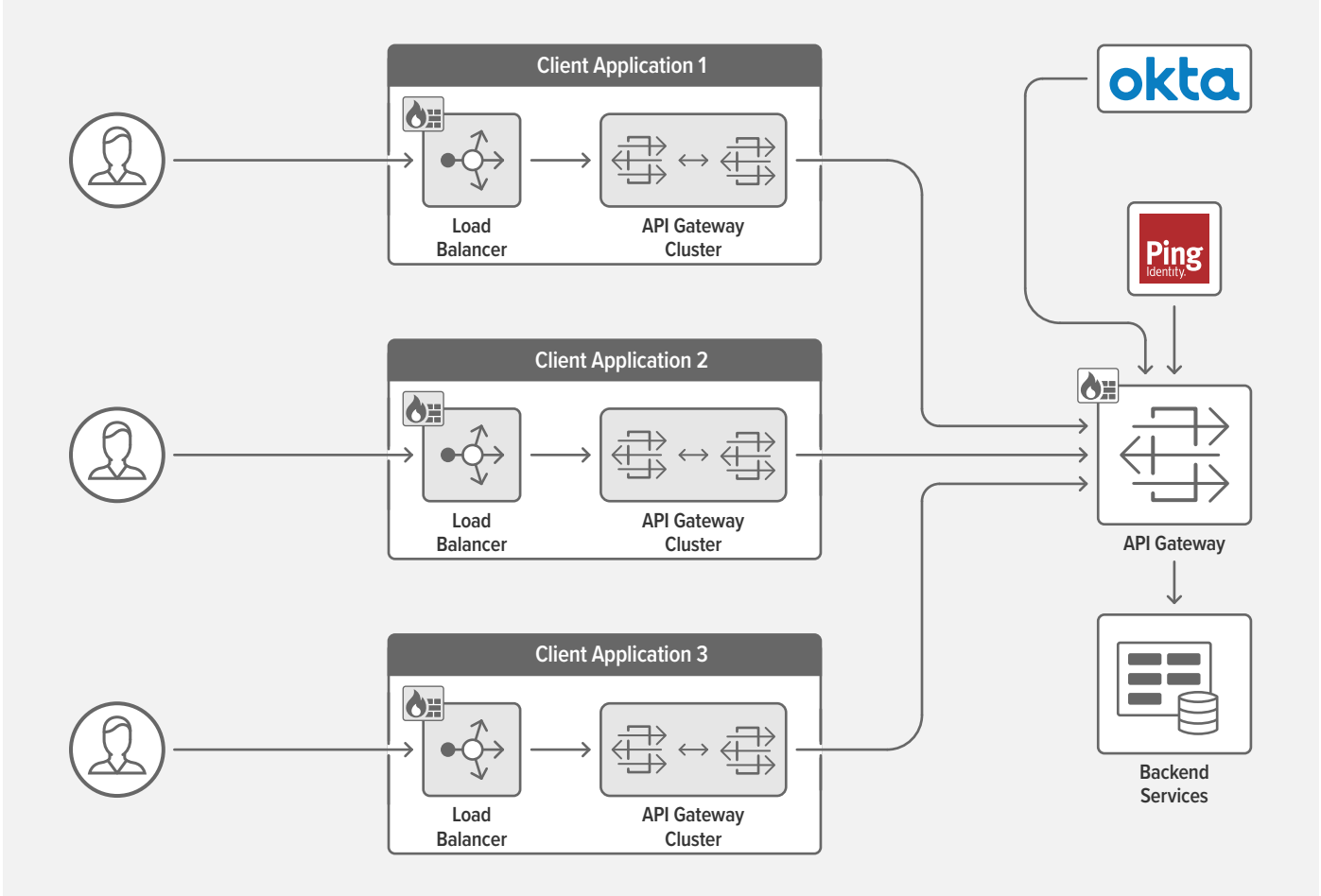


Figure 6. Reference architecture for real-time APIs at runtime: clusters of API gateways fronted by load balancers and protected by WAF.

breaches due to malformed JSON, null requests, or requests that do not comply with the gRPC protocol. It needs to support advanced API protection including path enforcement, method enforcement, data type validation, and full schema validation. This component needs to protect both the cluster of gateways and the load balancer.

Many organizations also deploy API fraud detection, which involves looking deeper into the logic of the API call to see if it is malicious or abnormal. Fraud detection is often a separate function but may be tied into the API gateway or WAF layer for enforcement.

It's important to note that API security almost inevitably introduces latency, which may result in API processing that exceeds the real-time API performance threshold of 30ms. We encourage organizations to determine if API security or absolute performance is more important. We include API security in the reference architecture to show how it differs from edge or perimeter security which may not provide API-specific protections.

API Identity Service

This component provides access and authorization policies that secure APIs and protect backend resources. Recommended best practice is to integrate with leading identity providers such as Okta and Ping Identity to ensure secure access to APIs. These solutions enable you to create and manage access policies that key off of end-user and client application attributes.

For the purposes of this reference architecture, we are assuming such identity services are already in place and mention them only for completeness of the overall API architecture.

DevOps Tooling

A declarative API interface must be provided to accomplish all aspects of API lifecycle management – creation, publication, gateway configuration, and monitoring. This enables automation of API creation and gateway configuration changes. You can accelerate API release velocity by seamlessly integrating with an automation platform such as Ansible and a CI/CD toolkit such as Jenkins.

Although this form of automation does not impact API call processing performance per se, it is critical to ensuring changes can be made quickly and as part of the development process. If a problem occurs that does impact performance – such as an DDoS attack or misconfigured API client – CI/CD integration ensures that you can quickly resolve the problem and reconfigure API gateways to restore performance back to an acceptable level.

REAL-TIME API GATEWAY CHARACTERISTICS AND ARCHITECTURAL GUIDELINES

NGINX has worked with some of the largest organizations in the financial services, retail, entertainment, and software industries to build their API architectures. These customers have scaled to handle hundreds of billions of API calls per month, all with less than 30ms of latency. From our work with these customers we've distilled the following six API gateway best practices that underpin our real-time API reference architecture.

Deploy High Availability API Gateways

In order to ensure rapid responses, first and foremost the API gateway needs to be operational. You must employ a cluster of API gateways to boost availability. A cluster of two or more API gateways improves the reliability and resilience (note, not resiliency) of your APIs. There must be a mechanism for sharing operational state (such as rate limits) among all the gateways so that effective controls can be applied in a consistent manner.

Authenticate but Do Not Authorize at the API Gateway

Authentication is the process of verifying that a user or calling entity is who it claims to be. Authorization is the process of determining which privileges or access levels are granted to a user.

Authorization entails additional processing – typically using JSON Web Tokens (JWTs) – to determine whether a client is entitled to access a specific resource. For instance, an e-commerce app might grant read-only access to product and pricing information to all clients, but allow read-write access only to select users. Because of this granularity, authorization is best performed by the backend service that processes the API call itself, because it has the necessary context about the request. With authorization delegated to the business-logic layer of the backend, the gateway doesn't have to perform lookups, resulting in very fast response times. API gateways are certainly capable of performing this function, but you potentially sacrifice real-time performance.

Enable Dynamic Authentication

Pre-provisioning authentication information (whether using API keys or JWTs) in the gateway minimizes additional lookups at runtime. Authentication is thus almost instantaneous.

Fail Fast with Circuit Breakers

Implementing circuit breakers prevents cascading failures. Let me illustrate with an example. Assume an API call is handled by a backend consisting of several microservices. One of the microservices, microservice A, performs database lookups. When a database lookup fails, microservice A takes a very long time to return an error. This adversely impacts the API response time for not only the current API call, but also subsequent API calls.

Circuit breakers address this issue. You set a limit on the number of failures that can occur within a specified amount of time. When the threshold is exceeded, the circuit trips and all further calls instantly result in an error. No client application or user is left hanging due to resource exhaustion.

Do Not Transform at the API Gateway Layer

Data transformation, such as translating an XML-formatted request payload to JSON, tends to be computationally intensive. Assign this work to another service. Performing transformations at the gateway adds significant latency to API calls.

Use gRPC If Possible

gRPC is a modern open source remote procedure call framework introduced by Google. It is gaining popularity because of its widespread language support and simple design. gRPC relies on protocol buffers which, according to Google, are a "language-neutral, platform-neutral, extensible mechanism for serializing structured data". Because gRPC uses HTTP/2 as its transfer protocol, it automatically inherits all the benefits of HTTP/2, such as data compression and multiplexed requests over TCP connections. Multiplexing allows the client and server to send multiple requests and responses in parallel over a single TCP connection. HTTP/2 also supports server-push which allows a server to pre-emptively push resources to a client, anticipating that the client may soon request them. This reduces round-trip latency as resources are sent even before they are requested. All these features result in faster response to the client app and efficient network utilization.

HOW CAN NGINX HELP?

NGINX Plus is built for delivering APIs in real time. It supports:

- High availability clustering that allows sharing of state information such as rate limits across all instances in a cluster.
- Authentication using both API keys and JWTs.
- In-memory SSL/TLS certificate storage in the NGINX Plus key-value store. This technique can be used to pre-provision authorization information such as API keys or JWTs.
- gRPC.

A high-performance ethos is a defining feature of NGINX's API management solution as well. Unlike traditional API management solutions, the data plane (NGINX Plus as the API gateway) has no runtime dependency on the control plane (the NGINX Controller API Management Module). A tightly coupled control and data plane adds significant latency as the API call must traverse databases, modules, and scripting. Decoupling the data and control planes reduces complexity and maximizes performance by reducing the average response time to serve an API call.

The API gateways and the load balancer can be secured by NGINX App Protect – a DevOps-friendly WAF that provides enterprise-grade security. NGINX Plus being an all-in-one load balancer, reverse proxy, and API gateway ensures high availability and high performance while reducing complexity and tool sprawl.

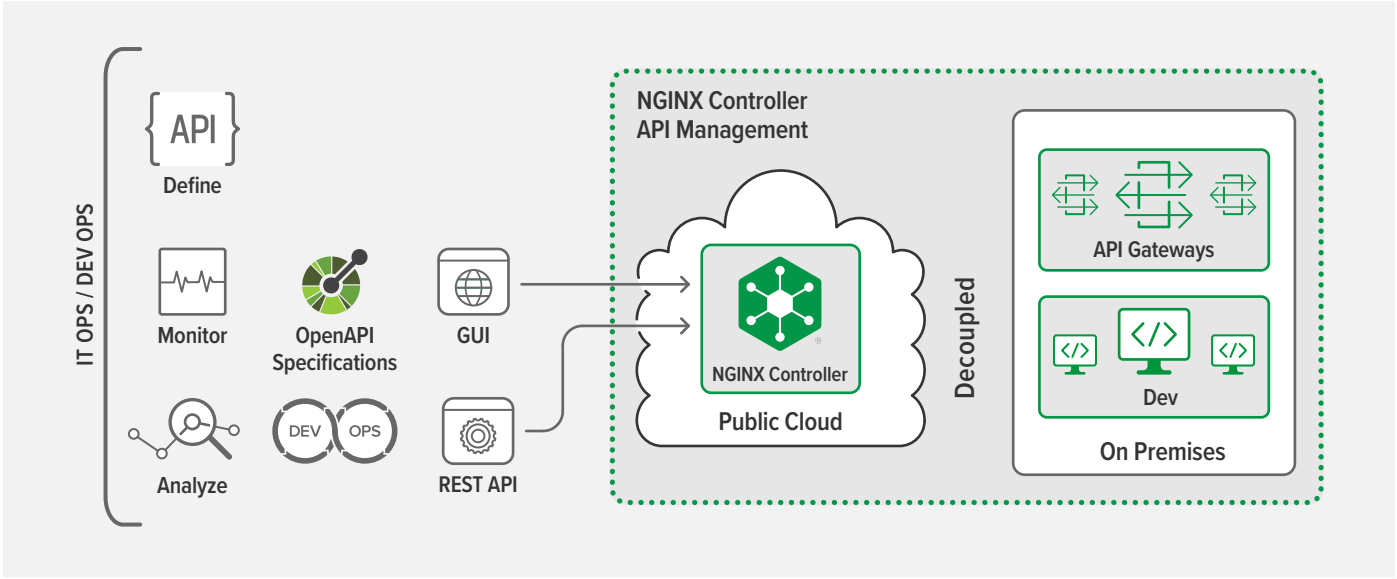


Figure 7. Illustration of an NGINX API management deployment. In this scenario, NGINX Controller is deployed in the public cloud. NGINX API gateways and developer portals are deployed on premises. The portability of both NGINX Controller and NGINX Plus as the API gateway maximizes flexibility. Please refer to Figure 6 to understand how API calls are processed at runtime.

4. Is Your API Real-Time?

TEST ITS LATENCY WITH THE `rtapi` TOOL FROM NGINX

SPEED IS HIGHLY VALUED, AND IS ULTIMATELY DRIVEN BY RESPONSIVE, HEALTHY, AND ADAPTABLE APIs.

APIs lie at the very heart of modern applications and evolving digital architectures, with API calls representing 83% of all web traffic, according to the Akamai State of the Internet Security Report. In today's landscape, where it's so easy for consumers to switch to a digital competitor, it is of the utmost importance for them to have a positive experience with your site or app. Speed is highly valued, and is ultimately driven by responsive, healthy, and adaptable APIs. If you get this right – your API is faster than your competitor's – developers will choose you.

According to the IDC report APIs — The Determining Agents Between Success or Failure of Digital Business, over 90% of organizations expect a latency of less than 50 milliseconds (ms), while almost 60% expect latency of 20ms or less. (Latency is defined as the amount of time it takes for your API infrastructure to respond to an API call – from the moment a request arrives at the API gateway to when the first byte of a response is returned to the client.)

We've used this data, together with some end-to-end analysis of the API lifecycle, to define a real-time API as one with latency of 30ms or less. Further, the 30ms latency threshold must be maintained all the way up to the 99th percentile of processed requests (meaning that on average only 1 request in 100 takes longer than 30ms). For details, see Chapter 2

It's a major challenge for most businesses to process API calls in as near to real time as possible, especially when using architecturally complex API gateways as the entry points to their API endpoints. So, how do your APIs measure up? Are they already fast enough to be considered real-time, or do they need to improve? Does your product feel a bit sluggish, but you can't quite place why that is? Maybe you don't know for sure what your API latency looks like?

Enter `rtapi` – a real-time API latency benchmarking tool created by NGINX that tests the responsiveness of your API gateways and endpoints, and generates reports that you can easily distribute and socialize among your peers. Two report formats are available: a PDF with a summary graph and an ASCII-formatted table of detailed metrics.

RUNNING `rtapi`

1. Download the `rtapi` binary from GitHub, using one of two methods:
   - If you have Golang installed, run this command:

     ```
     $ go get github.com/nginxinc/rtapi
     ```

   - If you don't have Golang installed, navigate to github.com/nginxinc/rtapi/releases, download the binary for your platform, and make it executable.

2. Specify one or more API endpoints (targets) that you wish to query, in JSON or YAML format as in the following examples. We recommend saving the JSON or YAML in a file called **endpoints.json** or **endpoints.yml**. (For JSON you can also include the data on the `rtapi` command line in the next step, as the parameter to the `--data` flag.)

   The only required parameters for each endpoint are **`target.url`** and **`target.method`**. If you don't specify **`target.body`** and **`target.header`**, they remain empty. If you don't specify a parameter in the **`query_parameters`** object, `rtapi` uses the (default) value for it shown in the following examples.

   ### Sample JSON Input

   ```
   [
     {
       "target": {
         "url": "https://www.example.com",
         "method": "POST",
         "body": "{\"id\":\"0\"}",
         "header": {
           "Content-Type": [
             "application/json"
           ]
         }
       },
       "query_parameters": {
         "threads": 2,
         "max_threads": 2,
         "connections": 10,
         "duration": "10s",
         "request_rate": 500
       }
     }
   ]
   ```

Sample YAML Input

```
- target:
    url: https://example.com/api/id
    method: POST
    body: '{"id":"0"}'
    header:
      Content-Type:
        - application/json
  query_parameters:
    threads: 2
    max_threads: 2
    connections: 12
    duration: 10s
    request_rate: 500
```

3. Run the following command, where **`--file`** names the file containing the JSON/YAML-formatted list of API endpoints and **`--output`** names the PDF report to be generated by **`rtapi`**. (As noted in Step 2, you can provide JSON input on the command line: for **`--file endpoints.json`** substitute **`--data`** followed by the JSON string.)

```
$ drtapi --file endpoints.json --output report.pdf
```

To display the ASCII-formatted table of metrics in the terminal, include the **`--print`** flag, instead of (or in addition to) the **`--output`** flag.

```
$ rtapi --file endpoints.json --print
```

INTERPRETING THE PDF REPORT

The PDF report generated by `rtapi` contains an HDR histogram of response latency like the following (from an NGINX Plus API gateway configured using the NGINX Controller API Management Module). The Y axis shows latency in milliseconds, and the X axis shows the proportion of calls processed in less than a given latency, expressed as a percentile. (In the following example, we've compressed the Y axis for the sake of space.)

To highlight the results for your API, the dashed lines that are parallel to the axes intersect to show your API's measured latency at the 99th percentile (**99%** on the graph). For your API to be considered real time, the latency at the intersection needs to be less than 30ms.

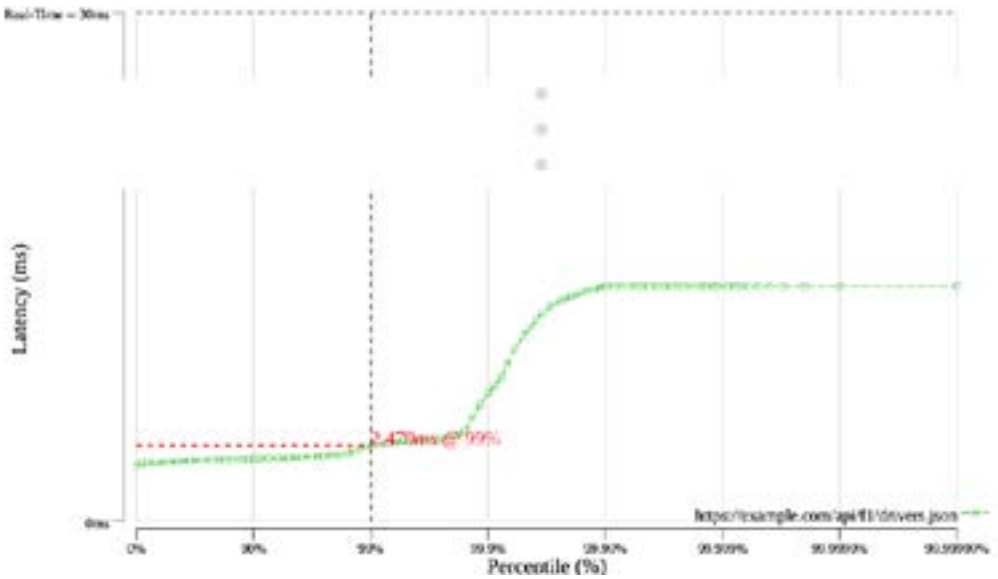


Figure 8. Sample PDF report generated by `rtapi`.

HOW CAN NGINX HELP?

Is your API's latency below 30ms?

If you're like many companies we talk with, chances are it's not. With many API gateway and API management solutions, latency of 100ms is common, and even 500ms is not that unusual.

We can help you improve your API performance – NGINX and NGINX Plus are the fastest API gateway in the industry, and together with NGINX Controller can help you route, authenticate, secure, shape, and cache API calls in less than 30ms.

5. Real-Time APIs: Stories from the Real World

Real-time APIs have become part of our daily lives. From ordering food to buying and selling stocks using our mobile phones, many day-to-day activities entail interaction with real-time APIs. But what does this look like from the perspective of the API publisher? What market demands are driving companies to serve APIs in real time? How are NGINX customers achieving real-time performance? Let's look at some examples.

Managing API Traffic at Scale for a Microservices-Based Application

Situation

African Bank was founded in 2016 in South Africa to provide digital banking services to underserved and rural customers. They built a modern application from the ground up using microservices.

Challenge

A sophisticated microservices architecture based on a reference architecture from NGINX provided speed-to-market and cost advantages, but introduced complexity around API scalability and inter-service communication.

Solution

The bank replaced Apigee, a legacy point solution for API management, with NGINX. The NGINX Controller API Management Module complements the existing NGINX microservices deployment with CI/CD integration and monitoring for SRE teams.

Processing a Large Volume of Credit Card Transactions in Real Time

Situation

A leading U.S.-based financial services corporation is adopting the Open Banking Standard, which provides API specifications for sharing of customer-permissioned data and analytics with third-party developers and firms that are building applications and services. This is driving the need to support increased API volumes and provide API-Management-as-a-Service.

Challenge

The incumbent API management solution added 500 milliseconds (ms) of latency to every API call. To avoid adverse impact to revenue as it transitions to Open Banking, the customer needed a solution that processed API calls with latencies under 70ms.

WITH NGINX, API PROCESSING TIME IS CONSISTENTLY BELOW 10ms, EXCEEDING REQUIREMENTS BY 85%.

Solution

The customer deployed the NGINX Controller API Management Module. API processing time is consistently below 10ms – exceeding their performance requirements by 85%.

Powering Digital Transformation with Real-Time APIs

Situation

A leading telecom organization in the Asia-Pacific region embarked on digital transformation initiatives that require processing 600 million API calls per month for 800 different APIs.

Challenge

The incumbent API management solution was too expensive and slow for the high volume of internal API traffic. The customer also wanted to augment and empower their DevOps teams with self-service API management.

Solution

The NGINX Controller API management solution sits “beside” the legacy API management solution. The NGINX solution reduced latency by 70%, while also enabling self-service for DevOps to create, publish, and monitor APIs. DevOps teams were able to increase API release velocity by integrating API management tasks into their CI/CD pipelines. They achieved significant savings in time and effort by automating routine tasks such as API definition and gateway configuration using APIs.

THE NGINX SOLUTION REDUCED LATENCY BY 70%, WHILE ALSO ENABLING SELF-SERVICE.

WHAT'S YOUR REAL-TIME API STORY?

Many other enterprises from various sectors have expressed interest in managing real-time APIs. Contact our API experts to request a free assessment of your API's performance and find out how we can help make your APIs real-time. In the meantime, read this report from IDC to learn about API trends and best practices for API lifecycle management.

NGINX
Part of F5