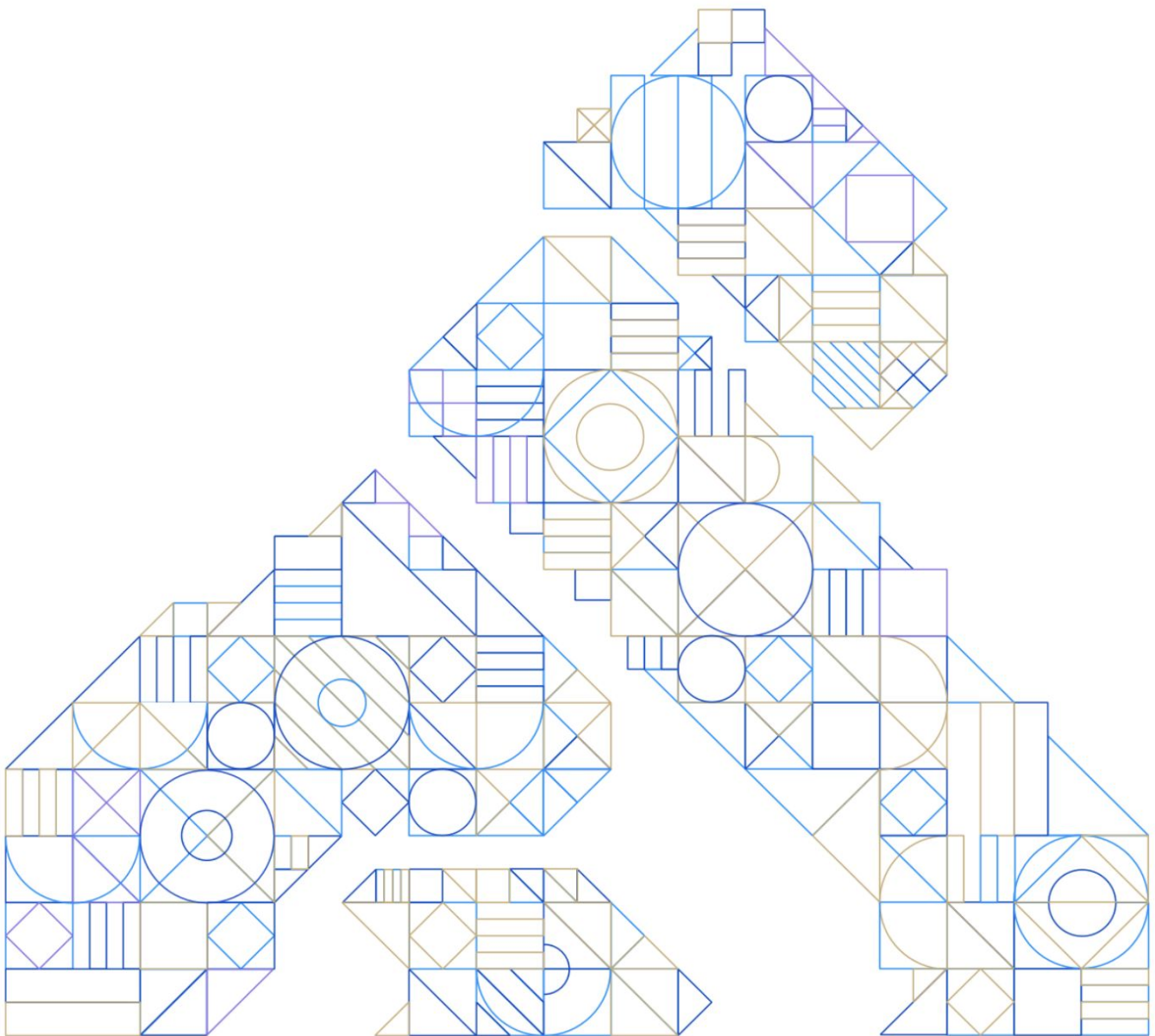




THE CLOUD CONNECTIVITY COMPANY

# Modern API Management With Kong

Using Kong's Service Connectivity Platform to fully automate the API lifecycle in any environment



## Business Summary

API management has become exponentially more important in the last few years. This has increased the need to select a modern API management solution that can run across any cloud or deployment environment, allowing the decentralization of API deployments within an organization and providing the appropriate level of control and governance from a central IT perspective.

When a multi-cloud and multi-line business solution is used correctly, it helps reduce the total cost of ownership of API platforms at the organizational level as well as improving developer productivity and time to market. This allows an organization to capture business value by innovating and iterating faster, rather than spending a disproportionate amount of capital and time focusing on IT governance. A truly modern platform will allow consistency to be applied across an entire API portfolio, regardless of where the APIs are hosted.

## Technical Summary

The move to ever smaller deployment units (e.g., containers and serverless functions) and ever more automated ways of testing, deploying and managing them (e.g., CI/CD pipelines and Kubernetes) has increased the complexity of a typical application landscape manyfold. To counter the complexity of ever more decentralized business logic, centralizing common functionality and policy is essential to reducing the overhead of security and governance that exists for every deployment unit, as well as to increase the speed of developing new features and functionality.

At the same time, this centralization should not happen somewhere far away from where the implementations are running. Be it on virtual machines, containers or even bare metal, API management should sit right next to the implementations to increase performance, reduce latency, and support the same deployment processes and tooling.

Many organizations are adopting Kong as the modern API management solution in a fully automated, cloud-agnostic world – gaining significant benefits in time to market and overall cost of ownership of their API deployments whilst benefiting from a horizontally scalable and extensible platform. This paper will explain how and why.

# Addressing Business Challenges

## Time to Market and Developer Productivity

Modern API approaches need to be customer-focused and transform IT into an enabler for the business. One of the major benefits of using an API gateway is to apply cross-cutting concerns to APIs and microservices rather than bottlenecking development by manually coding logic into each individual service. Coupled with a design-first approach to reduce time spent in development cycles, API platforms are a core technology component needed to accelerate time to market, as well as secure and govern APIs and services.

As API adoption has matured, these productivity gains have hit a new limiting factor due to the significant increase in the number of APIs and distributed nature of their deployment. This leads to a degradation of efficiency as developers and API owners apply these practices at scale. Without driving the end-to-end API lifecycle declaratively, continuous integration cannot be fully continuous, causing manual effort to deploy APIs and decide which policies should be applied to which endpoints. It can also lead to drift between the API specs and the corresponding API implementations as changes are made. The cost of this human-driven approach to the API lifecycle will increase with the number of APIs and the rate of change of each, creating new bottlenecks that decelerate the overall time to market.

## TCO and ROI

Modern API platforms nearly always span multiple deployment platforms and technologies. It is rare for a large enterprise to only use a single cloud provider, as it is rare to find an organization that has fully migrated away from its legacy technologies. This means for there to be a single logical approach to an API platform, the software must be able to run on any deployment target, be it on different cloud providers, containers or even on-premise bare-metal. As this proliferation of API management occurs as a company grows its API economy, it leads to four main factors driving up the overall cost of the platform:

1. The cost of compute to manage a larger estate and the traffic following through it
2. Operational complexity that arises by having different approaches for different deployment targets – both in terms of the skill sets required to support the disparate technology and also in the context of duplicating functionality so that a piece of logic can be executed on different runtime targets

3. Architectural complexity – by not having a standard blueprint across all deployment platforms, the risk of instability through operator error greatly rises
4. Lack of flexibility in deployment options, extensibility and scalability that slow down innovation

## Digital Transformation

Digital transformation is underpinned by three core pillars: people, process and technology. A well thought out API platform approach touches all three of these core tenants.

1. **People** – Digital transformation is the shift from IT as the bottleneck to IT as the enabler. The only way IT teams can stay in lockstep and even anticipate business needs is by shifting to a culture of re-use and self-service, and to a governance model more focused on decentralized empowerment than centralized enforcement. This means supporting the varying needs of the key API personas in the full API lifecycle across federated groups operating at different paces, whilst supporting a developer ecosystem in which internal and external APIs are treated as products and services.
2. **Process** – Digital transformation requires the adoption of different approaches from traditional software development, such as DevOps and agile delivery. This means modern platforms must have excellent support for DevOps tooling, must expose easy-to-use interfaces to integrate into automated code delivery pipelines and must support being treated as immutable infrastructure in small units of compute that can be easily destroyed or scaled.
3. **Technology** – Most businesses are now moving away from selecting a single technology partner to an approach of selecting best-in-class technology vendors that are suitable for all use cases. It is therefore critical that each vendor supports seamless, standards-based integration with the other solutions to co-exist, rather than attempt to force proprietary sub-optimal solutions. Common use cases for this approach are ensuring that an organization has a single pane of glass for observability across its entire estate, and also that open data and standards are adhered to.

# Why Are Enterprises Transforming With Kong?

## Business

### Developer Productivity and Time to Market

With the abstraction of policy concerns, developers can focus on business logic when developing new functionality, while standardization of APIs promotes reuse rather than redevelopment. Supporting declarative configuration throughout the API lifecycle further accelerates time to market and reduces errors.

### TCO and ROI

Migrating away from older, expensive and less flexible technologies (e.g., ESBs/JVM gateways) significantly reduces the overall hardware and software spend and often delivers savings within the first year.

### Enabling the Platform Team

Many of Kong's customers strive to separate a central platform team from other development teams. The intention is to isolate common concerns (e.g., security, observability, networking) in the platform team and provide such functionality "as a service" to others.

### Self-Service Training

## Technical

### Platform Agnostic

Kong can be deployed in any major cloud platform, including AWS, Azure, GCP and Alibaba. Its single binary architecture makes it easy to deploy in an automated fashion. The most common deployment targets include virtual machines, Docker containers and Kubernetes (including a native Kubernetes Ingress Controller).

### Decentralization

As many instances of Kong can be deployed as desired – in non-prod and production environments. This allows distributing the workload and building in high availability from the start.

### Automation

Kong allows automating the entire end-to-end API development process, e.g., as part of a CI/CD pipeline, starting with the automatic generation of declarative config files from the API design stage.

### Performance

A single Kong instance can achieve over 25,000 TPS per node while maintaining latencies of less than 4ms. This is typically significantly higher than JVM-based

Kong provides over 70 on-demand courses as part of its Kong University program. This allows customers to train developers at scale and constantly take advantage of new content.

### **Talent Acquisition and Retention**

With more than 27,000 Github stars, Kong is the most widely adopted open source API gateway, used by developers as their preferred tool of choice for API management. For many companies, Kong is part of a wider open source strategy that helps attract and maintain the best developer talent.

gateways and can unlock large cost savings. Kong's lightweight footprint (40 MBs) allows the platform to auto-scale to demand.

### **Extensibility**

Kong's flexible plugin architecture allows customers to extend the platform to any specific needs that are not covered by any of the out-of-the-box functionality.

### **Observability**

Having a central place where all traffic flows through enables a "single pane of glass" across all deployments and environments (e.g., multiple clouds).

### **Discoverability**

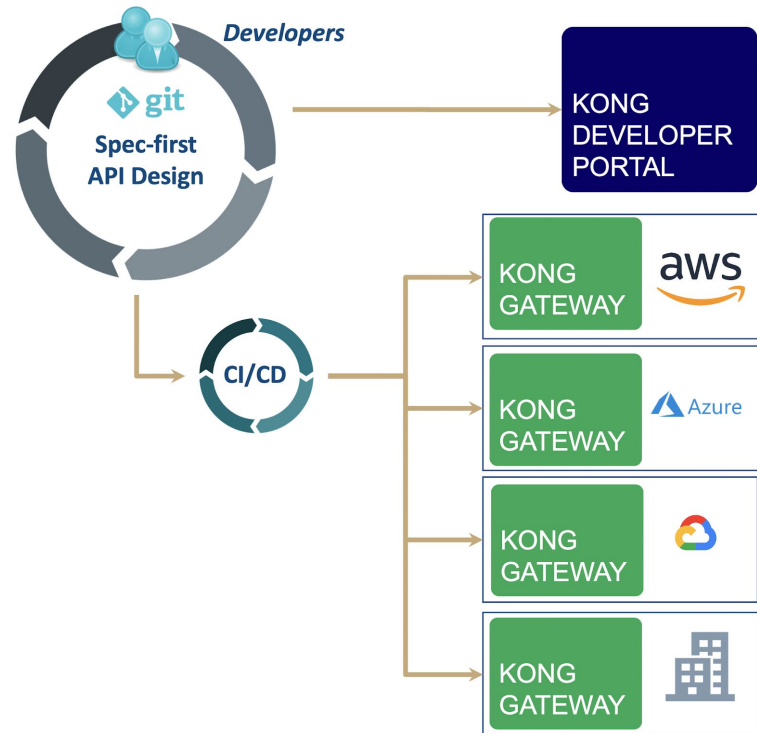
A central place and uniform standards allow documenting all APIs in a single place. This is a significant enabler for reusing existing functionality and saving the cost of developing more services than necessary.

# Kong Multi-Cloud/Hybrid-Cloud Architecture



For customers with a multi-cloud or hybrid-cloud strategy, a set of separate Kong instances will usually be deployed in each cloud provider, allowing local management of traffic without incurring the added latency and transfer costs associated with a central deployment.

The associated API specifications and Kong deployment configuration will typically be version controlled in a central Git repository and published to a central Developer Portal.

This uniformity helps reduce the operational burden of running across multiple environments.



## Customers

	<p>Verifone, powering 46% of the world's non-cash transactions, is lowering TCO and time to market by consolidating 20 separate API gateways into Kong, as well as further increasing productivity by using it to automate the end-to-end API lifecycle. Security, scalability and availability are Verifone's core pillars, and Kong Enterprise is a core component to achieving these in its adoption of microservices at a global scale.</p>
	<p>"Kong plug-ins were able to seamlessly integrate with our CI/CD tools [...], cut[ting] application deployment times by over 90%."</p> <p>- <i>Discovery Communications</i></p>

## Why Kong Enterprise Over a Cloud Gateway

**Single-pattern multiple clouds** – By adopting Kong as a gateway provider, organizations can provide a single architectural blueprint that can be applied across any cloud and also on-premise, avoiding vendor lock-in and duplicating effort in applying the same set of policies across different clouds.



**Open and extensible** – Cloud gateways are a black-box managed solution and do not allow the rich customization of both the runtime as well as the plugins and policies applied to the gateway. Kong Enterprise is built on top of an open core, allowing customers to know exactly what they are deploying and how they can tune it for their unique requirements.

**API management is not just a gateway** – The Kong platform offers more than just a gateway solution. Cloud vendors typically do not cover the entire API management lifecycle, which should start with an API design tool to help developers build, test and mock consumable APIs, then subsequently publish them to a developer portal where they can be socialized and reused internally and externally. Finally, when an API is running in production, it must be observable to support not just debugging but also the capture of MI data (e.g., logs, metrics and traces) to inform decisions related to treating APIs as products.

## Why Kong Enterprise Over a JVM Gateway

**Lightweight** – Kong is a lightweight, single 30MB binary, whereas JVM-based gateways will usually be hundreds of MB. This has a significant impact on the overall infrastructure footprint when running at scale. In most cases, Kong will be able to proxy 10-40 times the number of backend services and API calls using the same set of hardware as a JVM gateway.

**Runs anywhere** – Kong is a single, self-contained binary based on Nginx that can be run anywhere, be it in a virtual machine, a Docker container/Kubernetes pod or on a developer machine. Typical hardware requirements start at 1 core and 2GB RAM, and go up to 8 cores and 16GB RAM, at which point Kong will typically be able to proxy up to 25,000 transactions per second. Being able to run on a developer machine greatly increases developer productivity by allowing end-to-end tests of the gateway and backend code to be run constantly throughout day-to-day development.

**Configuration over code** – Kong's entire configuration can be dynamically adjusted at runtime, including dynamic certificate management. JVM-based gateways will typically require modifications to the on-disk Java keystore and a subsequent restart of the gateway. Kong supports and encourages full declarative configuration, allowing the gateway to be driven entirely from a Git repository.

**Extensibility** – Kong ships with over 60 built-in plugins for common use cases (e.g., AuthN/AuthZ, rate limiting, caching and transformations). In addition, hundreds of additional plugins are available in the community.

## Key Kong Differentiators at a Glance

Capability	Kong
<b>Portability</b>	Any cloud, on-premises, Kubernetes
<b>Automated configuration</b>	API, JSON, YAML
<b>Single binary</b>	Yes
<b>CI/CD integration testing (*)</b>	Yes
<b>Built-in policies</b>	60+ official and 100+ community
<b>Custom policy plugins</b>	Yes –, Lua or Go
<b>GraphQL, gRPC</b>	Yes
<b>Observability (****)</b>	HTTP, TCP, UDP, Syslog (e.g. Splunk, ELK)
<b>Pricing</b>	Multi-model(*****)

(\*) Can the gateway configuration be tested together with the API implementation as part of integration or system testing?.

(\*\*) JVM-based gateways are typically too heavyweight and have too many dependencies to be part of automated CI/CD-based integration or system testing. In contrast, Kong would recommend incorporating API gateway testing into the CI/CD pipeline and test against an actual ad-hoc Kong deployment running in a Docker container.

(\*\*\*) There does not seem to be a central place in the AWS documentation listing all available policies.

(\*\*\*\*) Out-of-the-box export targets of full request/response logs, if available.

(\*\*\*\*\*) Kong offers multiple different pricing models.