Journal of Sustainable Solutions
ISSN : 3048-6947 | Vol. 2 | Issue 1 | Jan - Mar 2025 | Peer Reviewed & Refereed

Development of Secure API Gateways for Cloud Services

Laxmana Kumar Bhavandla
Independent Researcher, USA.

Accepted: 09/01/2025 Published: 11/01/2025 * Corresponding author

How to Cite this Article:
Sharma, A. (2025). Development of Secure API Gateways for Cloud Services, *Journal of Sustainable Solutions*, 2(1), 1-10.
DOI: https://doi.org/10.36676/j.sust.sol.v2.i1.53

Abstract
This paper proposes an approach to design a secure API gateway for cloud services to improve the security, scalability and performance of cloud-based applications. These components include: authentication component, role-based access control component, data validation component, encryption component and real time monitoring component. Since the use of a Zero Trust model and the application of threat detection at a much higher level the framework provides secure protection from threats, such as unauthorized access, data leakage, and DoS. The research also involves gap identification evaluation of API gateway tools and implementation strategies and testing protocols with a view of making a detailed solution to security threats in today's complex cloud infrastructure.

Keywords
API Gateway Cloud, Security, Authentication and Data Protection

Introduction

The recent adoption of cloud services has brought new risks into organizations especially in the use of the APIs. API gateways are part of software systems that act as a facade for exposing backend services to clients; therefore, APIs are often attacked. Preservation of these gates is something that is required if application that is based in cloud is to be protected. This paper presents a novel API gateway architecture to mitigate the security threats which include the following; Authentication, Authorization and data privacy. The framework is established for the purpose of safeguarding the data that is confidential, limit the intrusion to such data, and facilitate provision of services, all through the application of assessment and proactive methods for countering threats and effective monitoring means.

API Gateways in Cloud Services

API gateway is one of the significant components in cloud service that joins the clients and backend services as an interface that mediates the interaction to allow good control and security measures. As cloud computing environment has further developed, API plays an essential role in connecting different applications and help in sharing data between them.

However, as the number of APIs increases, most of them within multi-cloud and hybrid environments, extending their security and management is a major issue. API gateway protect from the unwanted traffic by enforcing policies for authentication, authorization, and request validation which lower down the risks of malicious traffic for API.

They are instrumental to centralizing API management since they enable organizations to have standards on policies on security, rate limiting and traffic across the different functions. In addition, they offer one client API that clients request services behind which can be complex back-end services and deliver them as fast as possible even in massive loads.

For example, rate limiting of incoming requests through API gateways helps to contain DDoS attacks, traffic redirection during outages of services, TLS to secure application data in transit, thus meeting GDPR, HIPAA or PCI DSS (Samira et al., 2024). In the case of cloud services where the system has to be highly scalable and elastic in nature, APIs helps to load balance the traffic towards the right microservices.

Microservices are typical for the architecture of contemporary clouds, where each of them carries out a particular operation API gateways act as a mediator between various microservices and protect them from possible cyber threats. Through this, the API gateway sit in front of the service and masks the backend services to the developers, allowing them to solely implement scalable, efficient applications while avoiding the pitfalls of poor security.

Furthermore, API gateways have options for monitoring and logging activities due to required utility for threat detection, performance analysis, and operation visibility. Applications embedded in the gateways for example real time analytics help in monitoring of API usage patterns and identify any shifts in patterns which might signify an attack.

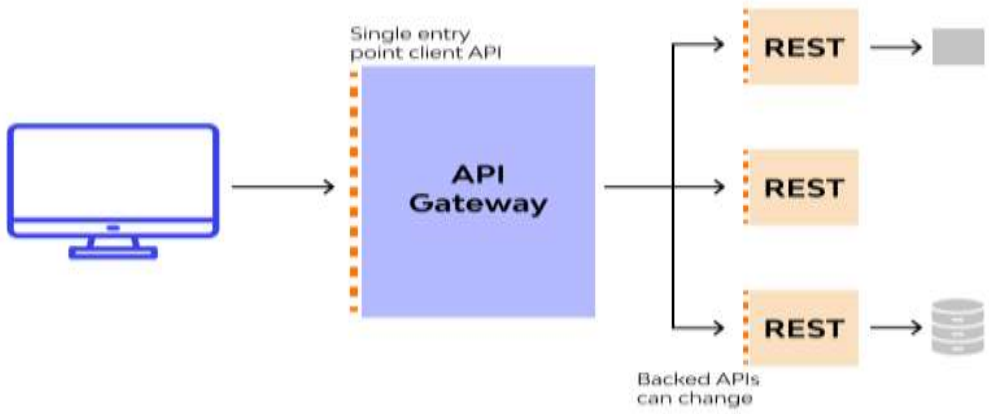

Figure 1 API Gateway (Wallarm, 2022)

For instance, the gateway can limit requests or even block a single source, for instance, if an API endpoint begins to receive many requests from a given IP address, possible downtime is prevented. API gateways also have a critical role in the organization's cloud services, particularly within organizations pursuing hybrid or multi-cloud initiatives.

Such strategies are usually applied to organizations to avoid being trapped in a long-term contract with a single vendor, cauterized costs agreement or to benefit from specific services, which different providers can offer. API gateways help to centralize the management of APIs in more than one cloud realm since all policies share a common touch point for API policies, traffic regulation, and security.

For instance, in a multi-cloud environment where backend services are with AWS Azure and Google cloud platform an API gateway can help in routing the incoming requests to the correct cloud service with equal efficiency and security. These are crucial particularly to companies that seek to build operational flexibility against disruptive events.

In addition, API gateways also support forward and backward compatibility ensuring that organizational APIs can integrate with old systems without escalating the technical debt. They function as a mediator between on-premise applications and cloud native solutions, converting protocols while integrating one system to another.

Out of all the responsibilities provided by an API gateway which is a crucial component of cloud services, security might be the most significant. APIs are always at the receiving end of attacks by malicious people who want to exploit a given vulnerability, get unauthorized access or compromise services.

These risks are handled successfully in API gateways through incorporating measures such as the token-based authentication and the IP white lists among others, data encryption. For example, most gateways offer OAuth 2.0 and JSON Web Tokens (JWT) as the means of safe API authentication and authorization.

Further, they can do input validation to block general scripts such as SQL injections, Cross-Site Scripting (XSS), or XML External Entity (XXE) attacks. In addition to these technical solutions, API gateways regulate the compliance with safety frameworks and prerequisites of information protection.

In industries such as healthcare and finance, where data security is of the utmost importance, gateways allow enterprises to meet high security requirements while data is both transferred across the Internet and stored. In addition to security, API gateways add value for developers by offering API management solutions during the API life cycle (Cao et al., 2024). These enable developers to develop, test, publish APIs effectively, and are regularly built into the developer CI/CD systems.

This automation is especially disadvantageous when an organization's operation is in a fast-moving business, because more time is gained to provide new features to the market. Documentation and version management are also important because, when helping clients build applications using APIs, gateways have to provide instructions on how to integrate API elements with each other and notify users of outdated versions and strategies for transition.

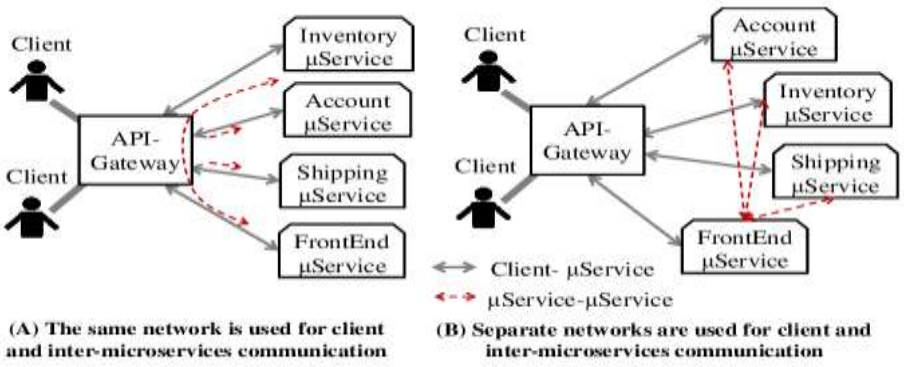

Figure 2 The API-Gateway architecture (ResearchGate, 2022)

For example, a new version of an API may provide enhanced features and at the same time support backward compatibility so that the API gateway is able to handle the transition because it directs traffic to the version appropriate for the client. The increasing trends in serverless computing and edge computing also support the centrality of API gateways to cloud services. API gateways in serverless architectures function as the interface that initiates a function where the server is not owned or managed by the developer.

They include requests, authentication, and throttling, which allow owners of serverless applications to expand them while ensuring they remain safe and efficient. In edge computing situations where data processing happens closer to the source of data generation, API gateways help in enhancing security and efficiency of the APIs in distributed computing conditions.

They can allow efficient communication between edge devices and the central cloud services that overcome latency issues and quality of the user experience. API gateways are critical in cloud services where they serve as tools for security, optimization, and management. They prevent modern architectures of cloud applications from falling apart, allowing for communication between clients, microservices, and backend systems while protecting against new threats.

With increasing usage of cloud computing within the organizational environment, the concept of API gateways is set to grow, as are the requirements for the API gateway to be adaptive to the added

complexity and distribution of workload. API gateways enable organizations to fully leverage their Cloud solutions through features such as management, security, and scalability in the face of emerging problems.

Threat and Vulnerabilities

In general, API gateways which have become critical to cloud services are exposed to an increasingly diverse class of threats since the availability of such systems inherently provides opportunities to attackers. Since APIs are the main means of communication in the cloud ecosystem, they are being attacked more often.

There is a severe concern arising from defective access control; weak authentication and authorization can result in leakage of data or give rogue applications access to backend services. For instance, using of rather poor or improperly configured API keys, hardcoded auth credentials, and insufficient token verification can result in privilege ramps or data leak.

These are exploited by the attackers through, Brute force attack, credential stuffing or misconfigured OAuth workflows to gain access to the resources. In shared cloud environments where various clients have access to similar resources, the risk is extended, as an attacked API may implicitly share information across multiple clients (Kurniawan et al., 2024).

Two of the most common attacks that plague APIs are injection, including SQL injection and command injection. In such cases, an attacker interferes with APIs inputs to either run unauthorized code or gain access to prohibited data. Lack of proper input validation make it easy for attackers to craft requests that will cause damage to the integrity of the underlying database or the application's logic.

Likewise, APIs that disclose too much data or do not use filtering mechanisms also become liable to be targeted by a malicious user through mass assignment where the user fetches a lot of data or over-fetching where he/she acquires more data than he/she requires. These are exacerbated by lack of strict compliance to the principle of least privilege or lack of strict schema validation.

Also, in cases where APIs have been documented poorly or are outdated, this becomes another breeding ground for security issues, where organisations do not decommission any endpoints that are no longer in use and can in turn be vulnerable. These are targeted as they are usually barbs in a network security as they are frequently ignored during audits giving a hostile entry into what otherwise may be a very secure network.

DoS and DDoS attacks are another big threat for API gateways that can be used to bring a service down. Simply put, they overwhelm an API with numerous requests in hopes that the system's resources available to the actual clients will be depleted or entirely unavailable. These attacks interfere with the continuity of cloud services leading to violation of service level agreements and therefore have a negative impact on the customers.

Due to their position as the entry point to all traffic, API gateways are highly vulnerable to volumetric attacks if rate limiting as well as throttling measures are not put in place. Furthermore, many threat actors such as advanced persistent threats (APTs) and botnet-driven attacks prefer to attack APIs more and more, utilizing various sophisticated approaches to deflect detection. Some of these threats work by masking themselves as similar to the typical traffic flow or using multiple vectors of attack that put incredible pressure even on the most sophisticated defence structures.

Another threat that continues to grow is exploitation of API misconfiguration. For instance, APIs that inadvertently expose sensitive data through verbose error messages or overly permissive Cross-Origin Resource Sharing (CORS) policies can become a goldmine for attackers.

Misconfigured security headers or a failure to enforce HTTPS can also leave APIs vulnerable to man-in-the-middle (MITM) attacks, where attackers intercept and manipulate data in transit. Furthermore, APIs integrated with third-party services introduce additional attack surfaces (Kondam, 2024). If third-

party APIs are not secured to the same standard, they can become a weak link in the security chain, allowing attackers to compromise the entire system through supply chain attacks.

Figure 3 API Security Best Practices to Protect Data (Simform, 2022)

The shift towards microservices and serverless architectures, while enhancing scalability and agility, has also introduced new vulnerabilities. APIs in such environments often have multiple interconnected components, increasing the attack surface and making it challenging to implement consistent security policies.

Attackers can exploit these interdependencies, targeting the weakest link in the chain to compromise the entire architecture. Additionally, shadow APIs those created without proper oversight or governance pose a growing threat (Balachandar et al., 2024). These undocumented and unmonitored APIs are particularly susceptible to exploitation since they often lack adequate security controls, making them a favourite target for attackers seeking low-hanging fruit.

Insecure API logging and monitoring practices further exacerbate the threat landscape. Without comprehensive logging, it becomes difficult to detect anomalies or trace the source of an attack. This lack of visibility can allow attackers to operate undetected for extended periods, increasing the likelihood of a successful breach.

Compromised logs, if not adequately secured, can also provide attackers with valuable intelligence, such as API usage patterns or sensitive request payloads. The absence of real-time monitoring tools can hinder an organization's ability to respond quickly to threats, prolonging downtime and amplifying the potential damage.

Year	API Calls (in Millions)	DDoS Attacks Mitigated	Latency (ms)	Data Breaches Detected	Blocked IPs	Traffic Throughput (GB/s)	Error Rate (%)	Cost Savings ($)
2020	1,200	150	120	4	12,500	5.8	1.2	1,200,000
2021	1,500	210	110	5	15,800	6.5	0.9	1,500,000
2022	1,800	320	105	3	18,300	7.2	0.7	1,800,000
2023	2,100	400	98	6	20,000	8.1	0.5	2,100,000
2024	2,500	450	92	2	22,500	9.0	0.4	2,500,000

For instance, APIs that reveal sensitive data due to detailed messages when an error occurs or through leaky CORS policies turn into gold mines for the attackers. Insecure nonprofits or incomplete secure safety headers may also open targeting APIs to MITM assaults, where the assailants hijack information in motion (Kondam, nd).

Moreover, such APIs present third-party services, which means additional entry points for attackers. As for third-party APIs those resources can be unbelievably loosely secured and thus become a hardly

protected point that chain attack can use and take full control over the entire system and through supply chain attacks.

While scaling the applications and making them more adaptable with the advent of microservices and serverless, we have come across newer forms of threats. APIs under such settings typically comprise of several interrelated modules, and this expands the attack surface and affords little opportunity for conforming to standardized security policies or protocols.

These interdependencies are open for exploitation and the enemy can go for the biggest vulnerabilities in the architecture if the smaller links have been breached. Further, shadow APIs the result of creating them without proper governance or in more derivative terms, oversight is becoming a real risk. These undocumented and unmonitored APIs are owed to insecure situation since they commonly lack the sufficient security controls, hence preferred targets of the attackers who seek easy tasks.

Insecure API logging and monitoring practices compound the risks seen in this domain even further. This makes it hard to note a potential problem area or follow the root of an attack to its source (Oyeniran et al., 2024). Insufficient awareness may lead to longer unauthorized entry to system or network, growing the probability of achievements.

Thus, if compromised logs themselves are not protected, the log data can disclose the attacker's additional intelligence, for example API usage patterns or the content of certain requests. Marketers and other associated professionals state that if such real-time monitoring tools don't exist in an organization, it is hard to respond to threats fast and limit the losses.

The risks that stem from the use of the API gateways in the containerized services show that security cannot be an afterthought and should be implemented in layers (Su et al., 2024). These challenges must be managed by organizations through introducing strong and efficient methods of user authentication, restrictive measures involved in rate limiting and access control and effective monitoring and logging.

Given the current threat landscape and new threat vectors daily emerging API security cannot be a one and done proposition but rather must be a continuous process for adapting to new threats and protecting API in an increasingly connected cloud world. In the following section, the key areas of concern will be disclosed, which allows fortifying API gateways and, thus, safeguarding the stability and security of cloud services an organization offers.

Framework

Developing a secure API gateway for cloud services; must have well-defined solutions to the following authentications; authorization; data validation; encryption; and monitoring requirements since cloud services are attackers' chronicle. The proposed framework is designed to interface the clients with the backend services and it is built to be a secure mediator that can handle client and service request securely and effectively without compromising the privacy of the former or the security and integrity of the latter.

At the core of the framework is the application of the Zero Trust principle where every request is verified all the time no matter the source. This eliminates providing authorization based on trust by location and status of a network. The authentication mechanism uses MFA alongside JWT for the validation of users to enhance security.

In most cases each JWT is signed with a private key to minimize replay attacks and normally has a limited lifespan. The token validation process takes place at the gateway level, and this makes it easy for the framework to veto rogue requests before getting to the backend level. One more relevant aspect of the framework is its role-based access control system that guarantees users and services, for example, can activate only those applications with which they are permitted to work.

These access policies are implemented dynamically, where the gateway identifies a user's role, their location and the security of the device being used. To determine the policy and the most appropriate

access rules relevant for its formation, the policy engine connects with IAM systems (Jani et al., 2024). Furthermore, the framework involves rate limiting and request throttling as measures of preventing volumetric attacks including Dods attacks.

Such mechanisms include setting maximum users per November, maximum IPs per November, or maximum endpoints per November, thus serving its clients and blocking volume spammers. Data validation and sanitization are two specific parts of the proposed framework which deserve mentioning.

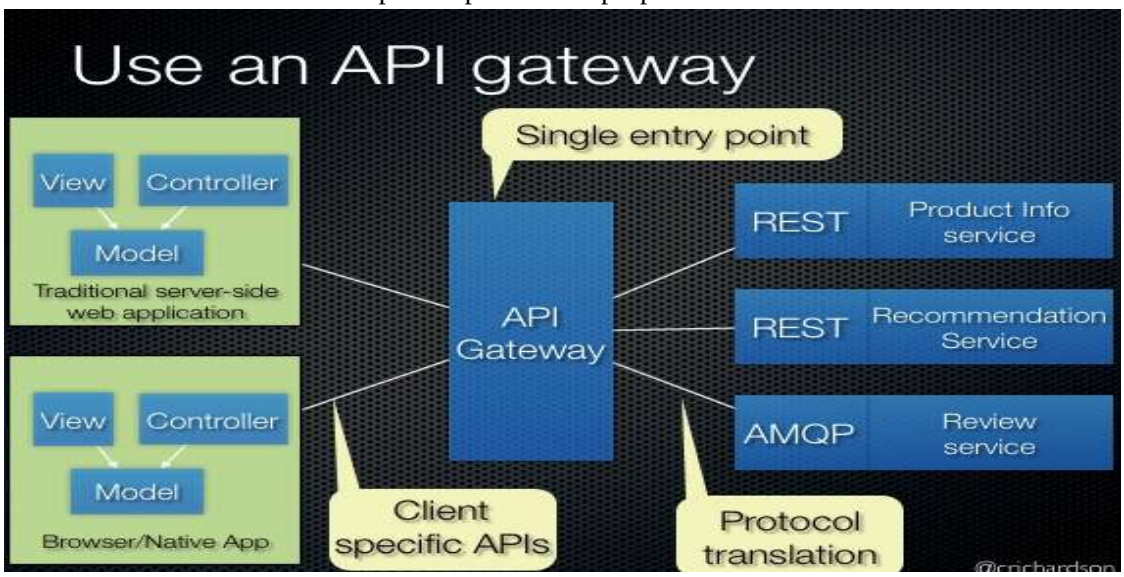

Figure 4 Pattern of API Gateway (Microservices.io, 2022)

Each new request is checked to conform to prescribed forms to enforce pre-set patterns. It also helps to do away with injection attacks which are common with the likes of SQL injection or cross site scripting (XSS). Moreover, communications in transit are also encrypted with such protocols as TLS 1.3 to make it impossible for breaches and retaliation of the conveyed information.

It also applies encryption to some data stored in logs to minimize the chance of being invaded in case the Oracle log is invaded by any unauthorized personnel (Barua et al., 2024). Therefore, the real-time monitoring and, the detection of deviations from normal behaviour are critical in managing security threats.

The used framework applies machine learning techniques to traffic data to recognize abnormality which may prove to be an attack. For example, large number of requests per minute, non-standard request payloads, or if a user profile shows different behaviour than other users of the same type are considered suspicious.

The monitoring system links with an SIEM solution, centralising its visibility, allowing an organisation to swiftly respond to security breaches. The framework also provides the capability to automatically resolve incident related tasks such as possibly locking MSIPs for a certain period, nullifying compromised tokens, among others.

Code snippet implementing token validation and role-based access control:

```python
from flask import Flask, request, jsonify
import jwt
from functools import wraps

app = Flask(__name__)
SECRET_KEY = "your_secret_key"

# Mock role-based access control
user_roles = {
    "user1": "admin",
    "user2": "viewer"
}

# Token validation decorator
def token_required(f):
    @wraps(f)
    def decorated(*args, **kwargs):
        token = request.headers.get('Authorization')
        if not token:
            return jsonify({"message": "Token is missing!"}), 401
        try:
            data = jwt.decode(token.split(" ")[1], SECRET_KEY, algorithms=["HS256"])
            user = data["username"]
            if user not in user_roles:
                return jsonify({"message": "Invalid user!"}), 403
            request.user_role = user_roles[user]
        except jwt.ExpiredSignatureError:
            return jsonify({"message": "Token expired!"}), 401
        except jwt.InvalidTokenError:
            return jsonify({"message": "Invalid token!"}), 401
        return f(*args, **kwargs)
    return decorated

# Role-based access control decorator
def role_required(required_role):
    def decorator(f):
        @wraps(f)
        def decorated(*args, **kwargs):
            if request.user_role != required_role:
                return jsonify({"message": "Access denied!"}), 403
            return f(*args, **kwargs)
        return decorated
    return decorator

@app.route("/secure-data", methods=["GET"])
@token_required
@role_required("admin")
def secure_data():
    return jsonify({"message": "This is secure data accessible only to admins."})

if __name__ == "__main__":
    app.run(debug=True)
```

Implementation

The general procedure of the framework of the proposed secure API gateway at the implementation level includes the use of a highly modular design that incorporates the essential components that include; Authentication, RBAC, data validation and encryption. Scalability and fault isolation are possible by using the microservices based approach (Cadet et al., 2024). Such a gateway is deployed by a light-weight, high performance API gateway solution like Kong, or Envoy, and includes custom plugins which allow the validation of JWT tokens and streaming of real-time statistics to a visualization tool like Grafana.

Sub-tests are carried out in discrete parts: unit testing of one object, coupling and integration testing to confirm proper interaction, and load testing to confirm the work of an object under intensive load. Security assessments, including penetration testing, are conducted to define risks, guarantee compliance with a high level of security within the framework.

Conclusion

It is important to create secure API gateway that would ensure that cloud based applications remain secure from new emerging threats. The proposed framework pays emphasis on a number of security aspects namely: token-based authentication, RBAC, data validation and encryption hence offering a suitable security solution for cloud services. Pre and post testing, along with security audits confirm its performance in practical scenarios. Therefore, as cloud computing advance the require for a powerful security approach to API becomes a concern. The framework proposed in this paper provides organizations with a solution that is both scalable and robust for the provision of efficient cloud protection.

References

Balachandar, S. K., Prema, K., Kamarajapandian, P., Shalini, K. S., Aruna, M. T., & Jaiganesh, S. (2024). Blockchain-enabled Data Governance Framework for Enhancing Security and Efficiency in Multi-Cloud Environments through Ethereum, IPFS, and Cloud Infrastructure Integration. Journal of Electrical Systems, 20(5s), 2132-2139. https://www.proquest.com/openview/a15bb8d498c6f13874e5c138ed1849fe/1?pq-origsite=gscholar&cbl=4433095

Barua, B., & Kaiser, M. S. (2024). Cloud-Enabled Microservices Architecture for Next-Generation Online Airlines Reservation Systems. https://doi.org/10.21203/rs.3.rs-5182678/v1

Cadet, E., Osundare, O. S., Ekpobimi, H. O., Samira, Z., & Wondaferew, Y. (2024). Cloud migration and microservices optimization framework for large-scale enterprises. https://doi.org/10.53022/oarjet.2024.7.2.0059

Cao, X., Zhang, H., & Shi, H. (2024). Load Balancing Algorithm of API Gateway Based on Microservice Architecture for a Smart City. Journal of Testing and Evaluation, 52(3), 1663-1676. https://doi.org/10.1520/JTE20220718

Jani, Y., & Jani, A. (2024). Robust Framework for Scalable AI Inference Using Distributed Cloud Services and Event-driven Architecture. https://doi.org/10.21203/rs.3.rs-4909036/v1

Kurniawan, D., & Dzikri, A. (2024, January). Development of a Prototype for Microservices Architecture and API Gateway Integration in an eLearning Platform. In Proceedings of the 6th International Conference on Applied Engineering, ICAE 2023, 7 November 2023, Batam, Riau islands, Indonesia. https://books.google.co.in/books?hl=en&lr=&id=-vEMEQAAQBAJ&oi=fnd&pg=PA190&dq=API+Gateways+for+Cloud&ots=Lb1qvpmHMw&sig=jZN0idPtPGRt-AvLm7z7VVGiJww&redir_esc=y#v=onepage&q=API%20Gateways%20for%20Cloud&f=false

Oyeniran, C. O., Adewusi, A. O., Adeleke, A. G., Akwawa, L. A., & Azubuko, C. F. (2024). Microservices architecture in cloud-native applications: Design patterns and scalability. Computer Science & IT Research Journal, 5(9), 2107-2124. https://www.researchgate.net/profile/Adebunmi-Adewusi/publication/383831564_Microservices_architecture_in_cloud-native_applications_Design_patterns_and_scalability/links/66e620922390e50b2c8d762d/Microservices-architecture-in-cloud-native-applications-Design-patterns-and-scalability.pdf

Samira, Z., Weldegeorgise, Y. W., Osundare, O. S., Ekpobimi, H. O., & Kandekere, R. C. (2024). API management and cloud integration model for SMEs. Magna Scientia Advanced Research and Reviews, 12(1), 078-099. https://doi.org/10.30574/msarr.2024.12.1.0148

Su, B., Yongcong, Z. H. U., Pang, J., Wang, C., Li, J., & Liu, X. (2024, May). Design of security protection framework for power grid cloud application API based on zero trust. In Eighth International Conference on Energy System, Electricity, and Power (ESEP 2023) (Vol. 13159, pp. 2035-2043). SPIE. https://doi.org/10.1117/12.3024219