International Journal of Computer Science and Engineering (IJCSE)
ISSN(P): 2278–9960; ISSN(E): 2278–9979
Vol. 10, Issue 1, Jan–Jun 2021; 21–34


International Academy of Science, Engineering and Technology
Connecting Researchers; Nurturing Innovations

# ROLE OF API GATEWAY IN API DRIVEN ARCHITECTURE

***Saveetha Rudramoorthy***

*Senior Architect/Middleware and Cloud Expert, Amadeus Labs, Bangalore, Karnataka, India*

## *ABSTRACT*

*With the growth of microservices, IT industry is moving towards or rather already moved to API driven architecture. 'API First' design approach is followed by all the API architects and developer, which ensures API contracts are created before creating actual front-end or back-end microservices. With enormous growth of APIs, need for centralized control gates which manages these APIs are also growing. API Gateways offers such a compelling functionality, which ensures all the APIs spread across multiple domains and geographical regions are all created, published, maintained, monitored and secured efficiently.*

*This white paper aims in detailing about the role API gateway plays in order to efficiently manage APIs and their underneath services hence offering complete control and management of the same.*

***KEYWORDS:*** *API, API Gateway, ESB, Micro gateway, Microservices, Middleware, OpenAPI, Servicemesh, Services*

***Article History***

***Received:*** *06 Jan 2021* | ***Revised:*** *16 Jan 2021* | ***Accepted:*** *23 Jan 2021*

## INTRODUCTION

Middleware today evolved from simple Application connectivity via B2B, EAI, ESB and to multi-connected application and services on cloud platform over iPAAS. In the area of functionality and service offerings, middleware has grown from simple file exchange messaging to SOA services, Microservices, Service endpoints and APIs. All these growth factors emphasize one thing, how to simplify and access the service provided by providers on 'On-Premise' or over 'Cloud' across multiple regions under one common umbrella.

Early 2000's and up to 2010 one could think about 'SOA' which allows consumers to consume service offerings via loosely coupled, vendor agnostic, self-contained services working on the principle of 'set of services to provide common functionalities'. Post 2010, 'Microservices' come to light wherein services are simpler, lightweight, fine-grained, elastic, self-contained, loosely coupled working on the principle of 'independent service for specific function'. Be it SOA or Microservice, IT world always think about accessing this service in more normalized and uniform manner. APIs emerged to address the very problem mentioned. APIs offer defined methods of communication, which enables end consumer to interact with the services provided by service provider in seamless manner.

## EVOLUTION OF GATEWAYS

'Service proxy' is very popular in the world of ESBs and SOA. Origin of gateway happened there. Earlier provider services are abstracted with service proxies which ensures changes to the actual service does not impact consumer

interfacing service. It was later realised that simple service proxy is not enough. Other additional features like access control, load limiting, throttling, routing, registry etc. are required. That is when 'SOA Service Gateways' emerged. This takes care of all the aspects of service management, right from basic service proxying to service identification, monitoring, controlling, routing and management.

Once APIs emerge, we need the same controls to the APIs which will eventually be provided by API Gateways.

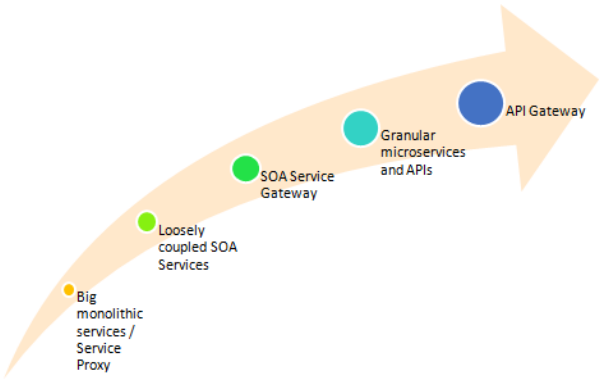



**Figure 1: Evolution of API Gateway.**

Both SOA Services and Microservices can be interfaced through SOAP and REST. Similarly, both SOA Service Gateway and API Gateway can host both SOAP and REST services endpoints.

## Why API Gateway

Knowing the importance of APIs, every company embraces API way of communication in all their business units to simplify and standardize their service communication model. A medium sized company delivering few business functionalities may generate more than 500+ APIs per month. APIs generated will grow in multitude of magnitude for a bigger company.

Typical eCommerce systems have interactions between multiple capabilities like Product Listing, Order Management, Supplier management, Pricing and Payment management etc. There may be multiple vendors involved in providing these services over APIs to big eCommerce platform. But how will all the actors involved communicate each other so that entire API life cycle (depicted below) is managed.

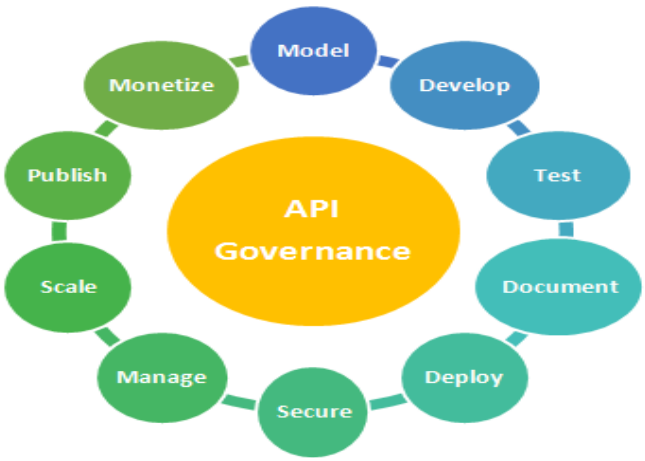



**Figure 2: API Lifecycle.**

APIs are complex and this complexity is compounded when these APIs are communicating with each other and with external clients. 'API Gateway' acts as a gateway between APIs and clients interested in those APIs.

Where does API Gateway Fit?

API Gateway serves as 'Central' interface for all external communications into an enterprise. API Gateway is even more important in microservices world as whole network of offerings consists of 'N' number of microservices offering different endpoints, operations and Http methods.

Following diagram depicts how an API Gateway fit in overall landscape of microservices:

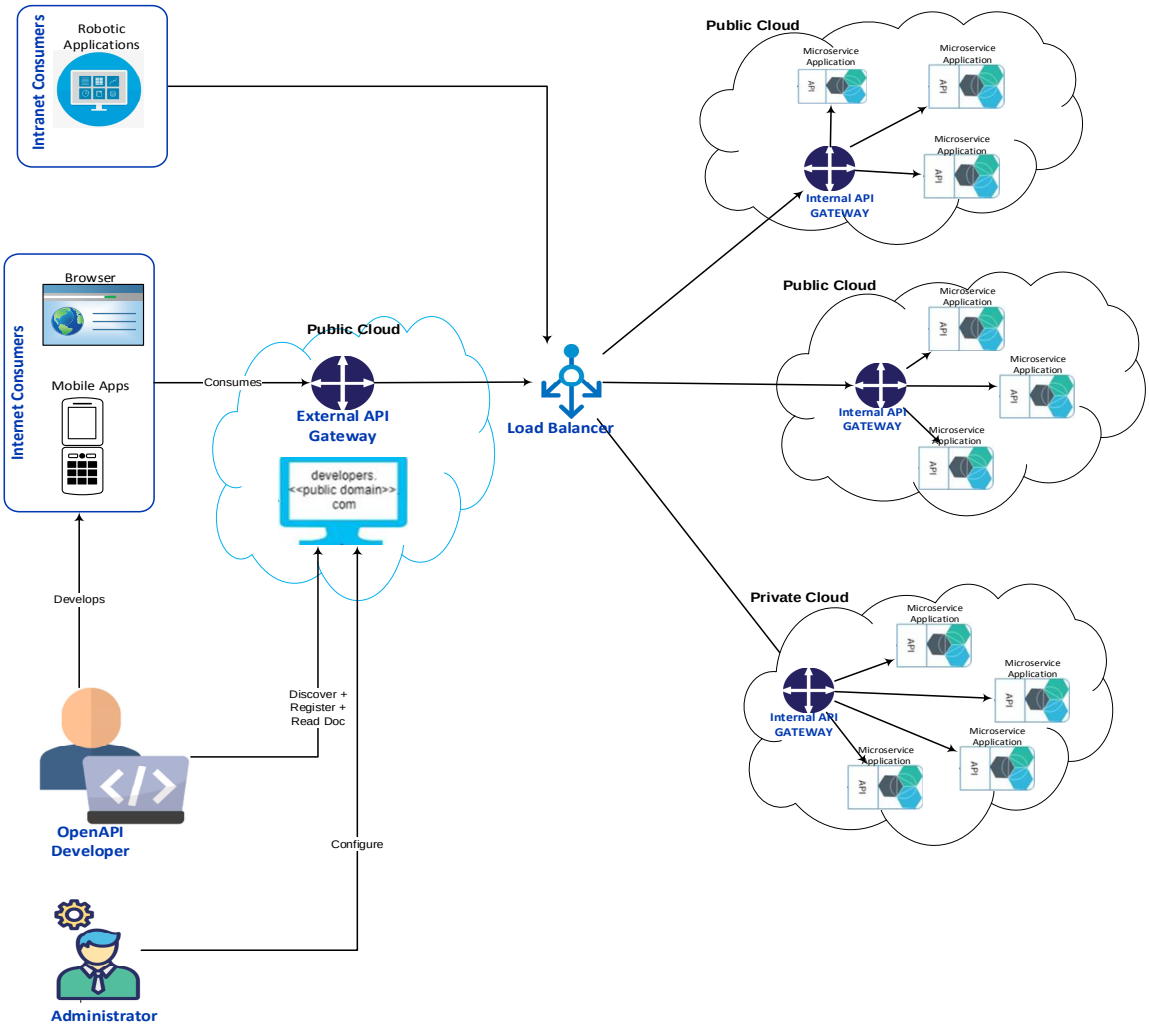



**Figure 3: API Gateway in Microservice Landscape.**

APIs exposed to internet community can be offered over **external API gateways**, whereas ones consumed by intranet applications can be offered over **internal API gateways**. An Organization can limit the APIs exposed to internet community over external gateway. OpenAPI developer community/internet applications can access only those APIs exposed over external API gateway.

Typically, an enterprise will deploy cluster of internal API Gateways ensuring high availability, whereby all the API configurations are synchronized across the cluster. Any call from intranet application will be load balanced via load balancer and reaches API Gateway cluster. Internal API gateway ensures APIs are maintained private to the organization

and are not necessarily exposed to internet community. Communication over internal API gateway is quicker as internal gateways are typically hosted closer to service providers. Also, security options and controls provided by internal gateway can be different from those offered on external gateway.

APIs exposed to public will be accessible over 'Developer portal' through which API developer community can discover or register to an API leveraging API documentation provided.

## Role of API Gateway

## Summary

Following is the summary of API Gateway features detailed inbelowsections:

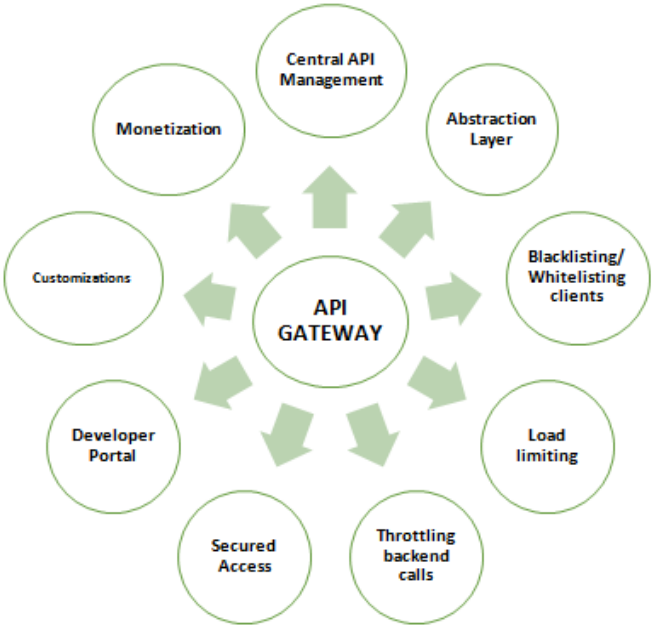



**Figure 4: Role of API Gateway.**

## Details

Following are detailed functional features expected out of an API gateway

## Centralized API Management

API Gateway is heart of any enterprise which offers 'Single point of Entry' to access APIs offered across multiple divisions and business units. Any private and partner APIs can be hosted on API gateway. API gateway is single source of truth or registry of all APIs in an Organization.

## Abstraction Layer

This is a typical requirement within any enterprise solution wherein service contract shared by service provider is abstracted to service consumer. API Gateway behaves as central component providing service abstraction to service consumers so that backend applications providing services (and APIs) are completely flexible to change their technical implementation provided API endpoint shared to consumers remain same i.e. decoupling client interface from backend implementation.

This is especially important in scenarios where the client applications are built by developers outside of the organization that cannot be forced to update to the greatest and latest APIs every time APIs are updated.

API Gateway also abstracts one or more methods/operations from clients. For example, in order to handle 'FlightSchedule' information, a microservice application can provide various functionalities and can expose following REST APIs and Operations:

GET flight schedule of passenger flight: HTTP GET https://<DNS Name>/flightschedules/passenger/flights

Create new flight schedule of passenger flights: HTTP POST https://<DNS ame>/flightschedules/passenger/flights

GET flight schedule of cargo flight: HTTP GET https://<DNS Name>/flightschedules/cargo/flights

UPDATE flight schedule of cargo flight: HTTP PUThttps://<DNS Name>/flightschedules/cargo/flights

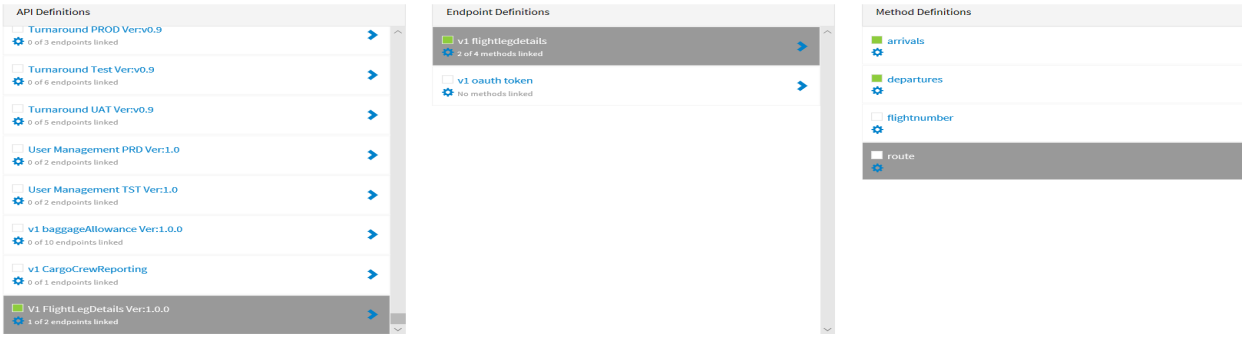
Though the microservice exposes many endpoints, operations and HTTP methods, API Gateway can expose only limited endpoints or operations or even HTTP methods. Which means consumers see only Search on passenger flight schedule.

Following configuration snippet shows how API gateway limits number of HTTP methods exposed:


## Access Control

All the APIs hosted on gateway is not necessarily to be shared with all types of end users. With various access control options, an API gateway can decide to expose or shield set of APIs to various users like internal applications, partner applications, internet applications etc. Each application within internal/partner applications/internet can as well have different level access to APIs.

Following screenshot shows how one could allow or disallow access to specific endpoint or a method on an API gateway:




Also, consumer application should have agreed security mechanisms with API Gateway in order to access the APIs. Various authentication/authorization mechanisms could be api_key, subscription key, combination keys or Reference tokens etc.

## Blacklisting and Whitelisting Requests

API Gateway can provide various set of policies which will ensure only specific consumer can access APIs. Following policies comes in handy here:

Restrict caller IPs - Filters (allows/denies) calls from specific IP addresses and/or address ranges.

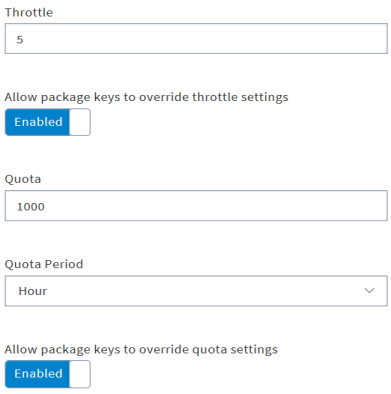
Check HTTP Header – Allows calls into API only with specific HTTP header or header value.

By blacklisting requests, backend resources can be protected from common attacks and abuse by users. For example, if a malicious user misuses the system, all requests received from that user or specific application can be completely blocked. In contrary to Blacklisting, it is possible to whitelist set of ips/users/applications which can access APIs.

## Load Limiting/Rate Limiting

API Gateway not only ensures only specific consumer can access APIs and but also at specific load limits. Mission critical applications will have 'High' load limits on an API compare to public usage or to an ordinary application. Whenever consumer application makes more calls in comparison to allotted number of calls, API gateway will restrict making further calls and will allow only when consumer application considerably reduces the load until agreed load limit.

Different API Gateways have different mechanism and policies to handle. Following screenshot show how load level can be defined on API gateway so that consumer application can consume only within defined limit:




Here 'Throttle' shows how many parallel calls specific application can make within a second. Quota shows upper limit of calls specific application can make in agreeable Quota period.

## Throttling calls to Backends

Implementations of backend service may have limitations to handle parallel calls beyond certain limit. In such cases, API gateway can throttle the calls hitting backend across all the consumers hence protecting backend services from unwanted surge of load.

## Securing the Endpoints

APIs and Endpoints exposed on gateway can be made accessible only for reliable users. These users share the acceptable contract with API gateway so that APIs can be accessed. An API Gateway can have one or more following means of securing an endpoint:

### *api_key*

This is simplest form of authentication model on API which authenticates any user, developer or calling program to an API. This is one time key generated for a specific user and usually never expires unless this key is been revoked by gateway itself. API keys are passed while invoking a service endpoint over transport header or as URL parameter. Following URL shows api_key being part of URL parameter as query string:

HTTP GET https://<DNS Name>/flightschedules/passenger/flights?api_key=xyz1234589

'Api-key' is typically used as authentication mechanism for robotic users than human users. This is not considered as most secure mechanism as there is possibility of 'Man-in-middle' attack and key can be stolen.

### *OAuth2.0*

The API Gateway can use the **OAuth 2.0 protocol** for authentication and authorization. The API Gateway can act as both OAuth 2.0 Authorization Server and Resource Server (as it is proxying the provider resources). With the support for OAuth2.0 and several grant types, API gateway can authenticate and authorize various users like Brower based application, native mobile application, user agents, web server etc. Various grant types supported by APIgateway are

- Authorization Code
- Implicit
- Password
- Client Credentials
- JWT etc.

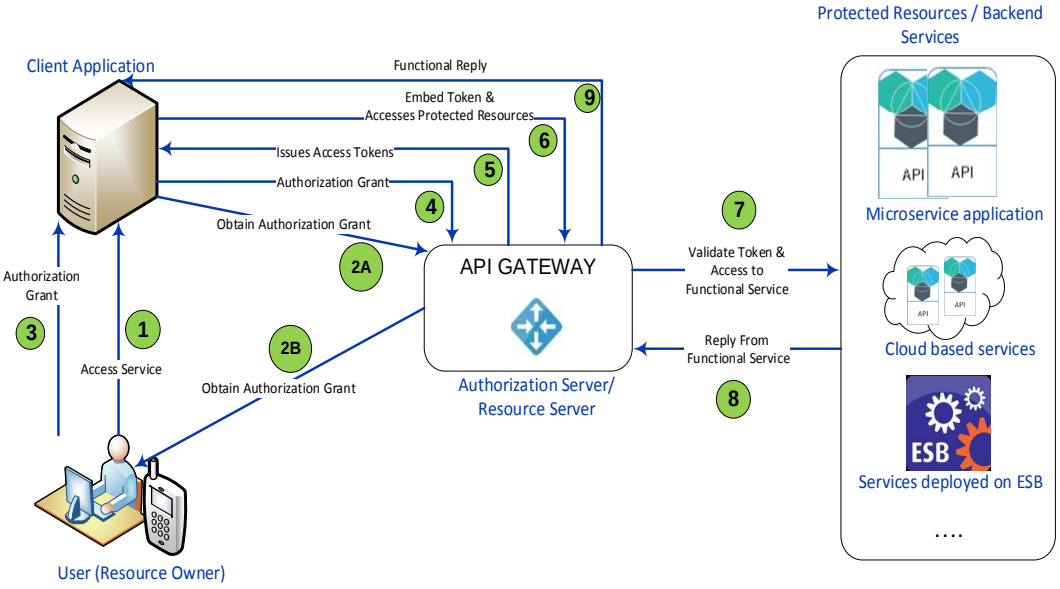



**Figure 5: Authorization using OAuth 2.0.**

As depicted in the above picture, a resource owner typically an end user connects to any client application for a service. For example, above scenario can be explained as follows:

- An end user connects to 'chatbot' for enquiring flight schedules. Chatbot application acts as client application here.
- Client application sends request to authorization server (part of API Gateway) that will in turn ask the user/Resource owner to identify and give consent so that the client application can act on user's behalf.
- End user authenticates himself and grants authorization to client application.
- Upon receiving 'Authorization Grant' from user, client application requests an access token by authenticating with the authorization server and presenting the authorization grant.
- The API Gateway ('Authorization server') authenticates the client and validates the authorization grant, and if valid, issues an access token (OAuth2.0).
- Client application will then embed issued access token in the request and makes API call to protected resource hosted on Api gateway i.e. flight schedule service here
- APIgateway ('Resource Server' here) validates token and in turn will communicate with actual backend service i.e. flight schedule service and re-directs the response to 'chatbot' client.

### *Key+Secret*

Gateways also provide Hashed Secret Keys to authenticate API users

### *Subscription Key*

API Gateways on cloud like Azure supports 'Subscription Key' based authentication and authorization. Client applications that need to consume the published APIs must include a valid subscription key in their HTTP requests. This model mandates user to have cloud subscription with the provider.

### *Basic Authentication*

Basic authentication is a simple HTTP authentication scheme in which the request will contain an authorization header with a valid base64 encoded username and password. API Gateway validates requests with base64 encoded user details which is sent over "Authorization" header and allows calls to hit backend services.

### *Custom*

Any custom way to authenticate a client call before accessing an API. This can be leveraged if organization has specific legal restrictions and users to be authenticated against corporate AD/LDAP solutions or over special certificates.

## Caching

In order to reduce the latency and strain on service provider, an API gateway provides 'Caching' which could

- **Cache Replies** getting out of gateway ensuring that client will get response from gateway rather than hitting backend every time. This feature is very useful when static content or content which is rarely changing is provided by backend. 'TimeToLive' (TTL) set on APIgateway ensures how often cached contents should be refreshed.
- **Cache Headers** which will ensure only specific headers are cached.
- **Custom Caching** enabling user to define their own caching elements and controls.

### Support for Multiple API Versions and Revisions

API Gateway should also seamlessly support hosting multiple versions of an API and should control access on API version level as well. This enables few clients to stick to older version of APIs wherein clients will to make change and obtain newer features can use new version of API.

Revisions are used when APIs are already in use but still provider want to activate new revision of same API. In such cases, new revision will be made active and others will be made in-active.

### Developer Portal

#### *OpenAPI*

As APIs are becoming primary interface for services, it is eminent that these APIs are opened for public usage. **OpenAPI**s are publicly available APIs that provides developers with programmatic access to a proprietary software application or web service. OpenAPIs are typically backed by open data and are openforprivate/public/partner usage. Also, OpenAPIs are based on Open standard and specification.

Following are the benefits an organization can gain from OpenAPI:

- Faster innovation as more people can contribute to organization success
- Companies can create better products while standing out from the competition
- OpenAPIs are used and backed by many big companies and are their condensed knowledge of building thousands of APIs over the years. This knowledge is a great asset for anyone before starting to write their first API
- As OpenAPIs are based on standards, it is easy for anyone to obtain any shared API and leverage them from day 1.

#### *Portal*

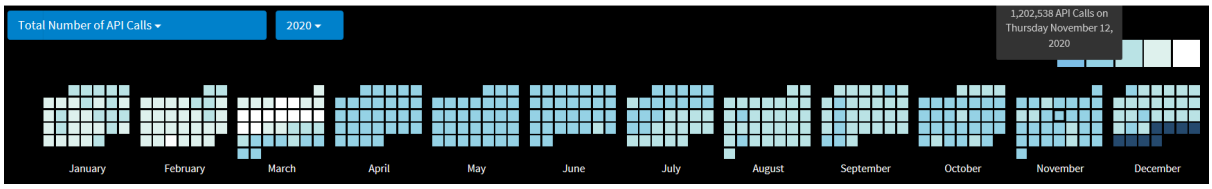
Developer portal is the face of APIs provided by an Organization which enables any internal, partner and third-party developers to access the APIs. Any application developer should connect to portal to learn about offered APIs. From developer portal, one could

- Register their application.
- Create token required to access APIs.
- View all the exposed APIs and Documentation associated with APIs, which is adhered to OpenAPI/Swagger specifications.
- Can view their usage of APIs in terms of traffic and latency.
- Play with exposed APIs before using it in their applications.
- Obtain programmatic stubs to connect to API in various programming languages.
- Can connect to Blogs, Webinars, Forums around APIs.
- Can connect to communities exposed where they can ask questions or provide inputs.

## Monitoring and Reporting

As APIs are exposed to different types of users, it is vital to analyse all the traffic handled by the gateway. Gateways typically provide multiple views on report data:

- **High Level Executive View**: Summary view of APIs with metrics and trends which can be shared with stakeholder audiences.




**Report View:** These reports provide second level detailed view by which summary of all Successful and Unsuccessful API calls are shown. Apart from that traffic distribution across all the APIs, Endpoints and Operations can be explored. Also, it would be interesting to obtain details up to user or application or api_key level. These reports can be exported into excel file or other apps through which other dimensions can be created.


- **Low Level Call Details:** Gateways also provides means to trace every API call detail entering into the system through which issues can be quickly troubleshot. It gives detailed information about how requests and responses looked as they pass through different points. Ideally this is recommended to be enabled only on Non-Production environments and for shorter periods as this feature will drastically impacts performance of APIgateway.

## Customizations

Other than passing through the calls from clients, an APIgateway can involve in 'Mini-ESB' activities like validation, transformation, caching, content-based routing, retry customizations etc.

**Validations:** Validate header elements to enforce specific header or value of header. Also, validate authorization token present in the request.

**Transformations:** Transform Request or Reply data before it hits backend or client. Transformations could be for

- Converting JSON to XML or vice-versa
- Convert SOAP to REST or vice-versa
- Content replacement based on XSLTs
- Set additional HTTP header or query parameter or to the extent of mocking entire message structure

**Content Based Routing:** Route to specific backend URL based on HTTP header or body structure.

## Monetize

As an API provider organization might want to monetize their APIs so that they can generate revenue for the use of their APIs. With Gateways capabilities to control the access of APIs, organization can share their enterprise APIs only to the users who are willing to pay.

## Beyond API Gateway

API Gateways have evolved from 1st generation Service proxies through 2nd generate API Gateways into 3rd generation 'Micro Gateways'.

Microgateway (also called as 'Edge' gateway) is a lightweight, distributed API proxy that is designed to enforce policies and business logic at or near the service endpoints. Microgateways are specially designed for microservices architecture, focusing on API gateway per microservice or set of microservices. Here team responsible for delivering microservices will also manage their microservices over microgateways. In contrary to API Gateways, micro gateways are de-centralized, scalable and light weight.

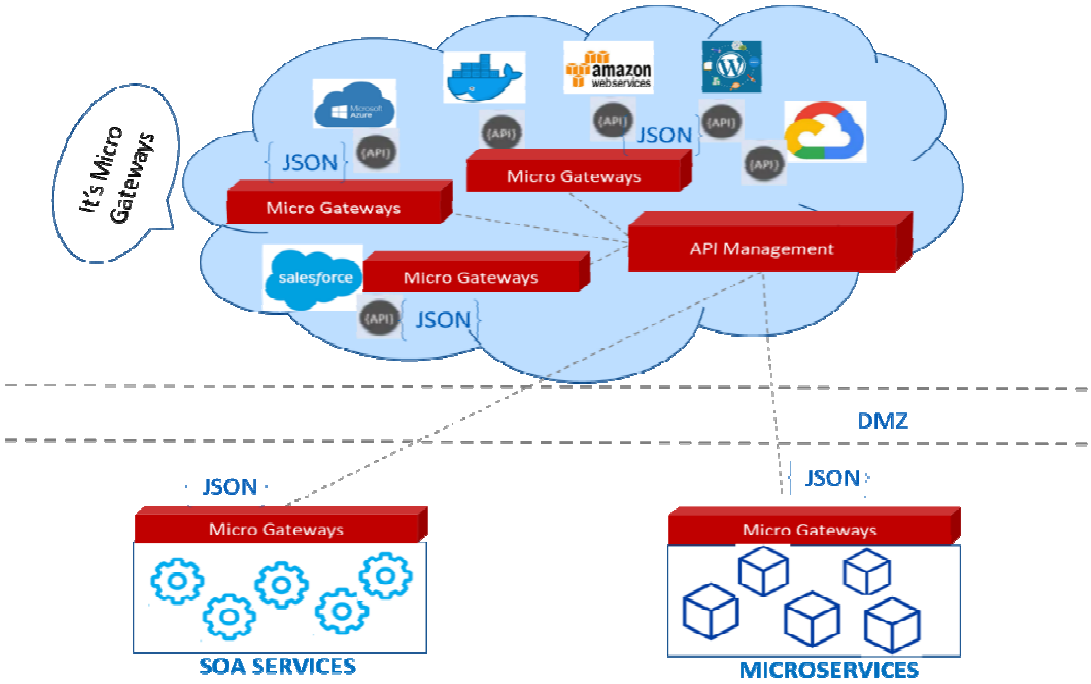



**Figure 6: Micro Gateway.**

'Service Mesh' is another component very popular in API world. Service mesh is used primarily in service-to-service communications. Security, Observability, Error handling, Load Balancing, Service discovery, Routing etc. are handled by service mesh as a side-car implementation of actual service.

**Table 1**

| Service Mesh | API Gateway | Micro Gateway |
|---|---|---|
| Service Mesh is the layer of communication between microservices. Service mesh facilitates service to service network communication using standard methodologies. | Acts as a central gateway between APIs and clients directly requesting services | API microgateway is a proxy which sits close to the microservice. |
| Decentralized sidecar deployment model that can be adopted and enforced on every service | Heavy weight API proxy where all APIs are centrally managed | Light weight API proxies, typically distributed and deployed closer to service endpoints |
| Focusing on brokering between internal resources | Maps external traffic to internal resources | Mapstraffic received to internal resources |
| Sits between network and application. No real business notion of the solution. | Exposes APIs to service specific business function | Serve specific business function like API Gateway |
| Service connectivity over HTTP and other TCP traffic like MQ/JMS/Web socket etc. | Supports connectivity primarily over Http | Supports connectivity primarily over Http |
| Supports both Layer4 and Layer 7 traffic policies | Supports Layer 7 traffic policies | Supports Layer 7 traffic policies |
| Does not meant to manage API Lifecycle | Product is meant for complete API life cycle management | Supports API life cycle |

## CONCLUSIONS

Gateways are the best deployment option for larger businesses and teams that have more existing resources to expose. Rolling own management tool for every API become over-whelming for any Organization. Hence Organization should leverage API Gateway which acts as 'Single point of entry' for all consumer communications.

Based on detailed analysis, it is very evident that API Gateway acts as central entry point into any enterprise API ecosystem. APIs are grown multi-fold and it is MUST for us to leverage API Gateways and reap complete benefits provided by API Gateways.

## REFERENCES

1. *https://www.gartner.com/doc/reprints?id=1-244J49X4&ct=200909&st=sb*
2. *https://www.tibco.com/api-led-integration*
3. *https://docs.microsoft.com/en-us/azure/api-management/api-management-key-concepts*
4. *https://tools.ietf.org/html/rfc6749#page-6*
5. *https://medium.com/api-integration-essentials/api-gateway-vs-microgateway-microservices-gateway-e8fbbd8ba9c0*

## AUTHOR DETAILS

Saveetha Rudramoorthy received the B.E. degree with specialization on Computer science from the University of Madras, India, in 2000. Saveetha is a Senior Architect, Enterprise Integration and Middleware Expert with Amadeus Labs based out of Bangalore, Karnataka. She is AWS cloud certified and Tibco, WebMethods certified with over 20 years of experience in cloud, enterprise integration and middleware across domains like Telecom and Airline.

You can find Saveetha on LinkedIn: https://www.linkedin.com/in/saveetha-rudramoorthy-06aa296