Cloud Customer Architecture for API Management

Executive Overview

This paper provides an introduction to API Management and the architecture elements of an effective API Management platform that supports an enterprise API strategy and its roadmap for digital transformation. The architectural capabilities described in this document are an essential set of ingredients to instantiate an API runtime and management environment using private, public or hybrid cloud deployment models.

An Application Programming Interface (API) is a public persona for a company, exposing defined assets, data, or services for public consumption. An API is a way for services and products to communicate with each other through a documented interface. APIs allow companies to open up data and services to external third party developers, to business partners and to internal departments within their company. An application developer can leverage an API with ease and invoke it via a web browser, mobile application or device.

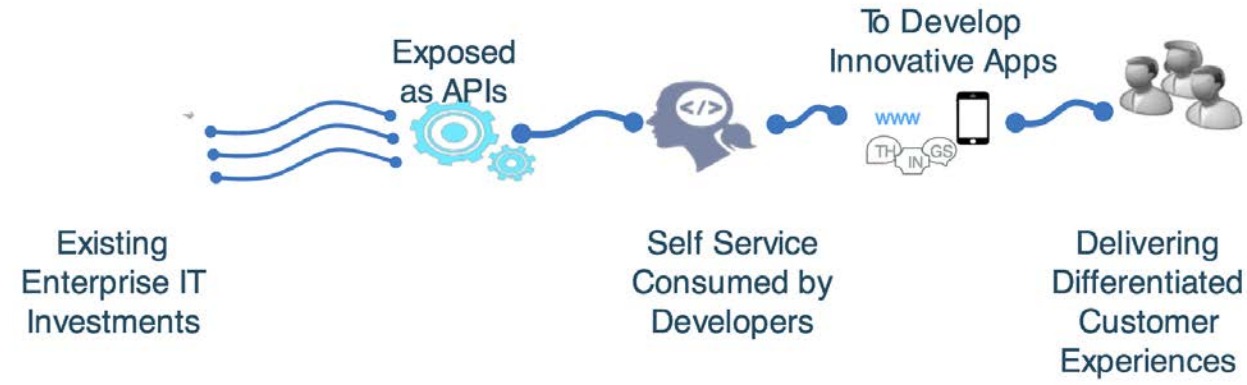


Figure 1: Enterprise Digital Transformation

An API management platform accelerates innovation by making it easier to open up new business assets in existing enterprise systems. Existing functionality can be exposed as APIs and published on a self-service portal that can be used by application developers who want to consume those APIs. This enables existing enterprise assets to be available to new channels and new audiences, with enriched customer experience in integrated omni-channel interactions. It allows for the support of new business models that may not be otherwise possible without API adoption. An API management platform provides that layer of controlled and secure self-service access to core business assets.

This paper will expand on the concept of API Management from various perspectives:

- **Why?** Highlight the value proposition of adopting a long term API strategy and embarking on the enterprise digital transformation journey.
- **Where?** Explain the principles and characteristics of selecting a solid API Management Platform.
- **How?** Illustrate the comprehensive lifecycle approach to creating, running, managing and securing APIs.
- **Who?** Identify the multiple personas and stakeholders in API Management and their use cases.
- **What?** Define the architectural components and capabilities that make up a superior API Management Platform.

It will also address runtime characteristics and deployment considerations.

APIs, Cloud Computing and Enterprise Digital Transformation

In today's market landscape, it is increasingly difficult for businesses to grow their revenue using their current systems as these are not flexible enough to dynamically adjust to constant external channel fluctuations. Exposing the business assets behind those systems via an API allows an external developer ecosystem to access existing enterprise core business assets to create innovative channel applications. Business APIs are a form of "crowdsourcing" – empowering digital disruption, opening the door to new types of solutions to grow the customer base, drive innovation, improve time-to-value, and open up new possibilities for creative business models.

Enterprises should consider five opportunities to include in their API strategy:

1. Accelerating in-house development to decouple / expose enterprise functionality as a reusable set of APIs for self-service consumption.
2. Innovating with digital applications on a cloud platform for rapid deployment and quick creation of a system of engagement to new channels.
3. Providing secure and controlled access to APIs from those digital applications in a *hybrid cloud* environment where the likes of mobile or IoT applications on a *public cloud* consume exposed APIs.
4. Joining or forming an ecosystem with a wider community of external developers and partners who will publish and consume APIs beyond enterprise boundaries.
5. Monetizing existing and new data and algorithms while enabling new business models.

To adopt an API strategy, an enterprise needs to have a comprehensive API Management platform to support the API lifecycle. This includes creating and testing APIs and connecting their implementation code to backend systems. It also includes securing access to those APIs and managing them in production whether they are accessed from a system of engagement application, systems of record application, or other type of application. This is in addition to making them available on a self-service developer portal for application developers to use.

Understanding API Management Platforms

An API Management platform is the embodiment of an architectural layer that brokers the enterprise's core capabilities, data and services with the digital application ecosystem that channels those capabilities into new and novel business models.

A superior API Management platform should provide a comprehensive set of capabilities to address the entire lifecycle of an API from its creation to deployment and management. It should be an integrated creation, runtime, management, and security foundation for enterprise grade APIs to expose core business assets and microservices to power modern digital applications.

The key capabilities for an API Management platform include:

- **Automated, visual and coding options for creating APIs.** A set of tooling to rapidly design, model, develop, test and deploy APIs in an automated continuous delivery model.
- **Polyglot runtime support for creating microservices**. Polyglot runtime support is key to enabling innovation and agility within different programming models required by different use case scenarios. Support for Node.js and Java runtimes among others is essential.
- **Integrated enterprise grade clustering, management and security for polyglot runtimes**. API Management is backed by a platform that delivers strong non-functional characteristics such as monitoring, performance, stability, scalability, load balancing, service bandwidth priority control, and failover.
- **Lifecycle and governance for APIs, products and plans.** Productizing APIs, packaging and cataloging them, and tracking their lifecycle are activities that will help the effective management and control of APIs as they are deployed.
- **Access control over API's, API plans and API products**. A key security function is managing the access to APIs at various levels of granularity involving users and user groups in consumer or provider roles.
    - **Advanced API usage analytics.** Monitoring and analyzing API usage metrics from different user perspectives and roles helps in providing a feedback loop to support real-time bandwidth control and assists API owners and developers with future improvements.
    - **Customizable, self-service developer portal for publicizing APIs.** Publicizing and socializing APIs through a user friendly portal is crucial in promoting the value of your core business as well as the market reach of your brand.
    - **Support of self-service diagnostics to lower the adaptation barrier.**
    - **Policy enforcement, security and control.** A high performing and scalable API security gateway is imperative in any API Management platform mainly to protect access to your back-ends.
    - **Real-time analytics to provide runtime intrusion detection and response mechanisms.**

API Management Lifecycle

Figure 2: A Comprehensive API Platform for the Entire Lifecycle

There are four key aspects that support the lifecycle of an API, each of which requires a rich set of capabilities:

- **Create:** covers the development lifecycle: design, model, test, build and deploy. The capabilities include:
  - Rapid model-driven API creation
  - Data source to API mapping automation
  - Standards-based visual API spec creation in OpenAPI Specification 2.0 (i.e., Swagger) [1]
  - Local API creation and testing
  - On-cloud and on-premises staging of APIs and packaging them into Plans /Products for discovery and subscription
  - Logical partitioning of environments for development, testing, and production
  - Well-defined Function and Role security

- **Run**: covers the performance, scalability, load and resilience of the API runtime platform. The capabilities include:
  - Polyglot microservices runtime
  - Integrated runtime management for availability, load and performance
  - Enterprise high availability and scaling
  - On-cloud and on-premises staging of microservices applications

- **Manage**: covers the publicizing, socializing, management, governance and cataloging of APIs as well as the user management of API consumers and providers. It also covers the monitoring, collection and analysis of API metrics. The capabilities include:
  - API discovery model
  - API, Plan/Product policy creation
  - API, Plan/Product lifecycle management
  - API visibility via self-service, customizable, developer portals

    - Advanced analytics on API usage and performance metrics
    - Subscription and community management

- **Secure**: covers the runtime security enforcement of APIs in terms of authentication, authorization, rate limits, encryption and proxying of APIs. The capabilities include:
    - Dynamic API policy enforcement
    - Enterprise security and gateway capability
    - Quota management and rate limiting
    - Content-based routing
    - Response caching, load-balancing and offload processing
    - Message format and transport protocol mediation

These aspects are inter-dependent and their metrics will help refine the overall API management platform.

In order to optimize APIs throughout their lifecycle, the API platform should provide integration capabilities to external advanced analytics systems. Such systems may provide different aspects of analytics capabilities including predictive analysis, real time dashboards, and machine learning. The API platform may use the outcome from such a system to dynamically adjust processing priority based on predictive analytics, for example. Cloud service providers may take analytics further by introducing self-service diagnostics that allow clients to autonomously identify operational problems. That would be another case for using operational analytics to optimize the monitoring and support of an API Management platform.

API Management Architectural Capabilities

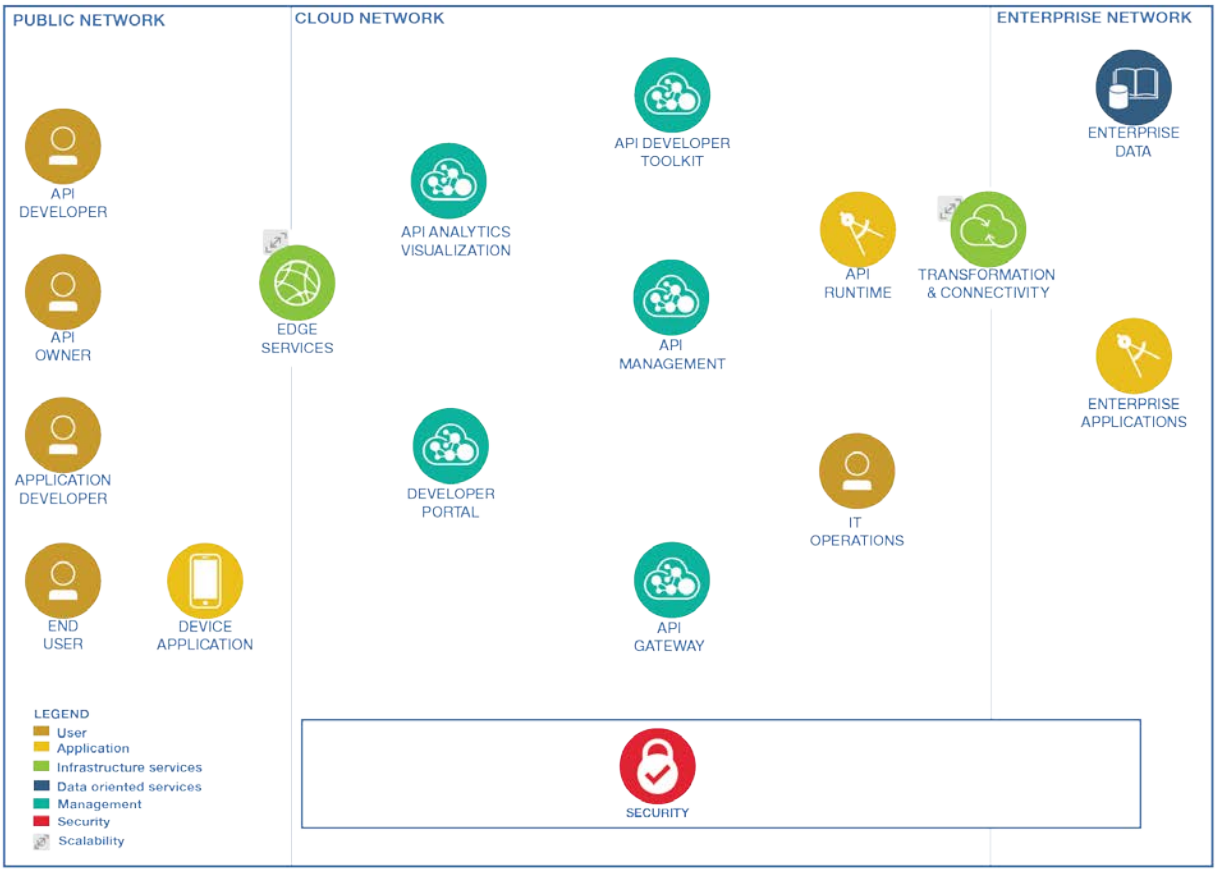


Figure 3: API Management Components

API Management Personas

A comprehensive API Management cloud offering should provide services that cater to all of the stakeholders in the API lifecycle. There are four major personas in an API lifecycle:

- **App Developers:** are consumers of APIs. They discover and subscribe to APIs that will be included in the business logic of their applications. They need to know:
  - Where do I access APIs?
  - How do I understand the APIs?
  - How do I measure success?

- **API Developers:** are the creators of APIs. They design and implement the logic behind the API to deliver proper data payloads from back-end business assets or services. They need to know:
  - How do I design, model and assemble APIs?
  - How do I manage security?
  - Will the infrastructure scale?
  - How do I measure performance?

- **API Owners/Product Managers:** are the designated owners of the API and the business asset that is exposed through that API. They need to know:
    - How can I rapidly release and update my APIs?
    - How do I publicize my APIs?
    - How do I measure success?

- **IT Operations:** are part of the cloud provider organization offering both runtime and management API infrastructure services. They need to know:
    - How do I manage all the API environments that are being requested?
    - How can I scale each environment?
    - How can I easily find and fix issues?

End User

Users access applications on the cloud provider network using a browser or via a mobile native app. The users could be the end consumers or enterprise line of business users of the cloud applications. They could also be enterprise administrative users from the line of business 'Digital IT' team managing the components deployed within the cloud network.

Device Application

These are domain specific or device specific applications. The end user may use applications that run on smart phones, tablets, PCs or alternatively on specialized IoT devices including control panels. They access the backend business capabilities via APIs and are responsible for the user interface and overall experience.

Edge Services

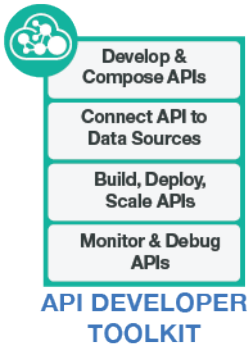
Edge services include service capabilities needed to deliver function and content to the users via the internet. These include:

- **DNS Server -** The Domain Name System (DNS) server maps the text URL (domain name) for a particular web resource to the TCP-IP address of the system or service that can deliver that resource to the client.
- **Content Delivery Network (CDN) -** Content Delivery Networks are geographically distributed systems of servers deployed to minimize the response time for serving resources to geographically distributed users, ensuring that content is highly available and is provided to users with minimum latency. Which servers are engaged will depend on server proximity to the user and where the content is stored or cached.
- **Firewall -** A Firewall is a system designed to control communication access to or from a system, aiming to permit only traffic meeting a set of policies or rules to proceed and blocking any traffic that does not meet these policies. Firewalls can be implemented as separate dedicated hardware, or as a component in other networking hardware, such as a load-balancer or router or as integral software to an operating system.
- **Load Balancer -** Load Balancers distribute network or application traffic across many resources (such as computers, processors, storage, or network links) to maximize throughput, minimize response time, increase capacity, and increase reliability of applications. Load balancers can

balance loads locally and globally. Considerations should be made to ensure that this component is highly available and is not a single point of failure.

API Developer Toolkit

The API Developer Toolkit is an SDK for API developers to model, create and test APIs locally and use cloud DevOps services to automate API build-deploy-publish tasks. The following capabilities are provided within this component:



- **Develop & Compose APIs -** Helps API developers create API definitions that invoke an existing API implementation that runs outside the API Management platform, or create API definitions for new API implementations to run within it. Multiple SOAP or REST services can be composed in a single API.
- **Connect API to Data Sources -** Provides connectors that connect APIs to a variety of back-end systems including:
    - Databases
    - SOAP or REST web services
    - E-mail
    - In-memory resources
- **Build, Deploy, Scale APIs -** Creates required artifacts to implement the API and associated definitions and policies.
- **Monitor & Debug APIs -** Provides functionality to test APIs both in an interactive manner and by enabling debugging information to be logged for each execution step.

The API Developer Toolkit should include a security mechanism for Function and Role authorization.

In addition, the toolkit provides the capability of creating APIs based on models and templates in order to accelerate the creation of APIs while enabling design standardization strategies. Such design governance can address aspects like naming, logging, versioning, prioritization, etc.

API Gateway

The API Gateway component enforces runtime policies to secure and control API traffic to existing enterprise data and services. The Gateway services also provide assembly functions that enable APIs to integrate with various endpoints, such as databases or HTTP-based endpoints.

The following capabilities are provided within this component:

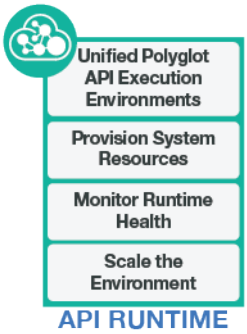
- **API Policy Enforcement -** The gateway provides a number of different types of policies, in addition to user-defined policies, to provide more processing control. A policy is a piece of configuration that controls a specific aspect of processing in the gateway during the handling of an API invocation at runtime.
- **Enterprise Security -** Performs actions that include schema validation, antivirus scanning, message filtering, authentication and authorization, token translation, message enrichment, encryption and decryption, digital signing, and validation of message transformation.

- **Traffic Control -** Acts as a proxy that receives inbound API traffic, routes and prioritizes requests to the relevant endpoints within an organization's firewall.
- **Workload Optimization -** Optimizes delivery of workloads across multiple channels such as mobile, API, web, SOA, B2B and cloud.
- **Monitoring/Analytics Collection -** Real-time filters, sorts, aggregates and predicts API event data to influence service priority. Presents the results within correlated charts, tables, and maps to help manage service levels, set quotas, establish controls, set up security policies, manage communities, and analyze trends.

API Runtime

The API runtime executes API and microservices business logic in different programming models (e.g., Node and Java). This runtime usually includes a UI console for IT operations staff to perform unified operations and management across the runtime instances.

The following capabilities are provided within this component:



- **Unified Polyglot API Execution Environments -** Provides a set of runtimes to execute an application in the language of choice (Java, Node.js, Ruby, etc.). It also provides a large set of platforms for mobile devices: iOS 8, Android, hybrid or JavaScript, etc.
- **Provisions System Resources -** Creates and binds system resources required to run APIs.
- **Monitor Runtime Health -** Monitors different aspects of the health of the environment: server availability, processor usage, memory usage, and disk space usage.
- **Scale the Environment -** Scales microservice components at runtime independently of other microservice components, enabling efficient use of resources and rapid reaction to changes in workload.

API Management

API owners, API developers, and business users use the API Management component to catalog, package, and publish APIs as well as to obtain API usage metrics for monitoring and analytics purposes.

The following capabilities are provided within this component:

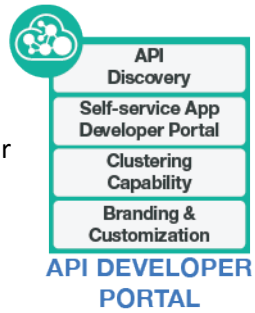
- **API, Plan, Product, Policy Creation -** Groups APIs into subscription plans and discoverable API products and controls their availability and visibility. It also defines policies that control a specific aspect of processing in the Gateway server during the handling of an API invocation at runtime associating them with APIs or plans.
- **API Product Versioning & Lifecycle Management -** Defines multiple versions of a product. These versions can occupy any of the lifecycle stages, which facilitate development.
- **API Monitoring & Analytics -** Creates custom analytics to influence system behaviors and dashboards for catalogs, which consist of default or user created visualizations such as tables, graphs, and maps.

Subscription & Community Management - Manages requests sent by application developers to subscribe to a plan or a product. It also manages the developer organizations that access APIs and plans when their users sign up to use the Developer Portal.

API Visualization & Analytics

This component is generally part of the API Management Services and provides API monitoring and analytics functionality. It enables the creation of custom analytics dashboards for catalogs, which consist of default or user created visualizations such as tables, graphs, and maps (see API Management Services section above).

API Developer Portal



The Developer Portal is a web site where APIs are made public to the application developer communities to discover the APIs and subscribe to their usage. The Developer Portal enables API providers to build a customized developer portal for their application developers. It is a portal where APIs are published to encourage the development of new applications that extend the value of core enterprise assets.

The following capabilities are provided within this component:

- **API Discovery -** Allows application developers to discover and use published APIs to which they have access.
- **Self-service App Developer Portal -** Provides self-service sign up and service diagnostic for rapid on-boarding of application developers from enterprise, business partner and third party developer communities.
- **Clustering Capability -** Provides the ability to cluster the portal over multiple nodes and make sure that services scaled over multiple nodes could behave as one single entity.
- **Branding & Customization -** Customizes the theme and the appearance of the Developer Portal.

Transformation and Connectivity

The Transformation and Connectivity component enables secure connections through to the enterprise systems. This component includes the following capabilities:

- **Enterprise Secure Connectivity** - Integrates with enterprise data security to authenticate and authorize access to enterprise systems.
- **Transformation** - Transform and enrich data in message headers and payload as they go through different network domains and heterogeneous platforms.
- **Enterprise Data Connectivity** - Provides the ability for cloud components to connect securely to enterprise data. Examples include VPN and gateway tunnels.

Enterprise Application

Enterprise Application represents applications that run enterprise business processes and logic within existing enterprise systems. This also includes enterprise services that represent modular and reusable logic that provides simple or complex enterprise business functions which can be linked up by BPM to quickly create various enterprise applications.

Enterprise Data

Enterprise Data represents the one or more systems of record, for example, transactional data or data warehouses that represent the existing data in the enterprise.

Security

Security for hybrid integration addresses the following needs of security:

- Integrity - Both cloud and enterprise data is not tampered with
- Threat management - Cloud components are up despite security threats
- Compliance - Addresses any industry or regulatory compliance needs

Capabilities include:

- **Identity & Access Management** - Capabilities to identify and authorize the user providing role-based access to cloud applications. It also enables single sign-on, user lifecycle management, and audit logging. The user types and their levels of access for cloud applications need to be managed. This could include business users (customer, vendor, 3rd party, staff users), or IT users (administrators, privileged users, application users). Identity and access management could leverage the enterprise user directory.
- **Data & Application Protection** - Capabilities that help identify vulnerabilities and prevent attacks targeting sensitive data. It provides protection to cloud components against many malicious threats right from the beginning of the development cycle. In addition, it monitors privileged access to sensitive data. It also protects the integrity of sensitive data in transit and at rest and provides network isolation. Firewalls in the public network component tier help protect the network level flows to application and data.
- **Security Intelligence** - Capabilities to monitor the cloud components for security breaches to provide visibility. It provides actionable intelligence to detect and defend against threats using event, trend, traffic pattern, and log analysis that enables real-time responses and feeds to a corporate incident management system.

API Management Runtime Flow

The components of the API Management platform will interact with one another to support various use cases involving the four types of actors: App developer, API developer, API owner and IT operations.

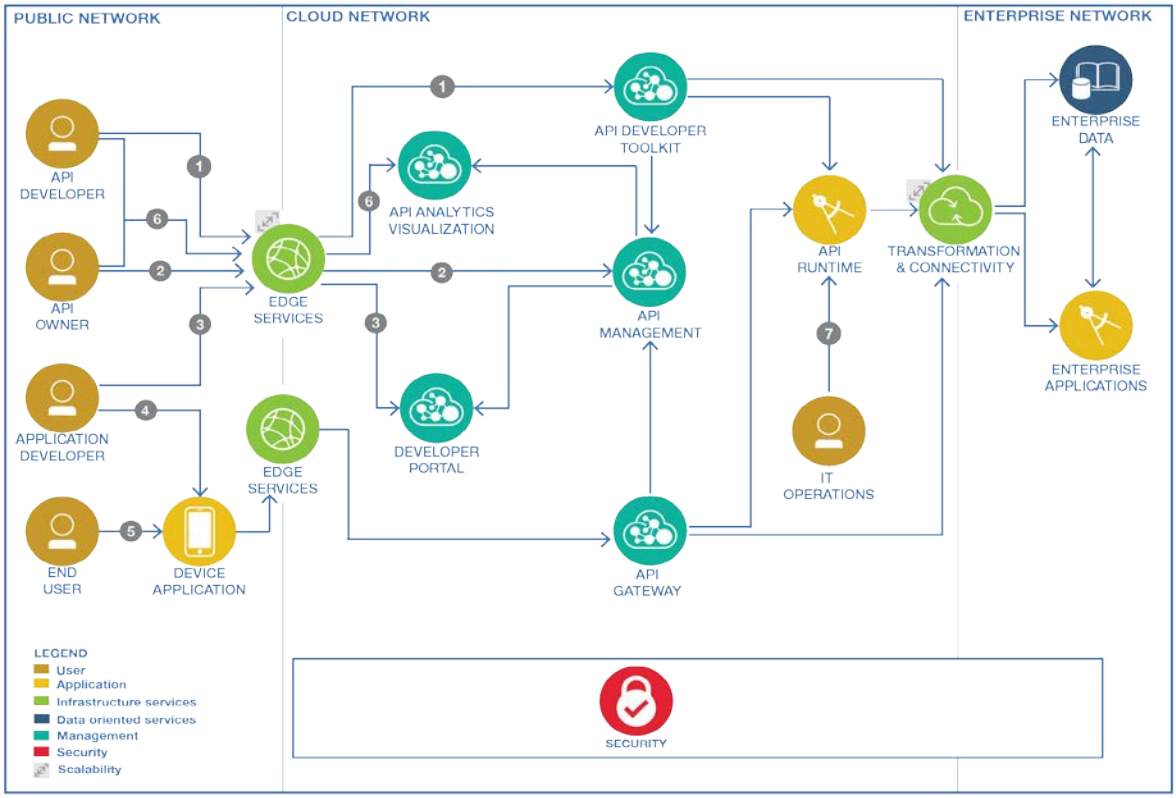


Figure 4: API Management: Component Interaction

Interaction Flow

1. API developer signs on to the API Management cloud services account. He/she accesses or downloads the API Developer Toolkit to:
   - Create the API and implement business logic.
   - Map and integrate the API data model to the backend schema through the transformation and connectivity service.
   - Test the API on the test environment.
   - Deploy the API to the runtime on the production environment.
   - Publish the API through the API Management component.
2. API owner signs on to the API Management cloud services account. He/she accesses the API Management component to:
   - Include the API endpoint in existing API products and plans, and specify access control.
   - Publish the API to the Developer Portal for external discovery by application developers.
3. Application developer accesses the Developer Portal. He/she searches and discovers the API.
4. Application developer uses the API in his/her app and deploys the app to an end-user device.
5. The device end user opens the app which issues the API request:
   - The request is handled by the API Gateway which performs load balancing and security validation for all API requests.

- The API Gateway validates access policies with API Management and invokes the API.
- The API Polyglot Runtime executes the API and obtains the data payload from the back-end system. The API response is sent back to the API Gateway. Alternatively, APIs exposed by enterprise applications can be executed on that enterprise application runtime.
- The API Gateway forwards the response to the calling app.

6. The API Gateway reports usage metrics and analytics to the API Management component. API developers and API owners can log on the to the API Analytics Visualization component to view dashboards on API usage metrics and other analytics.
7. Cloud provider IT operators log on to the polyglot Runtime to monitor and manage the API runtime environments.

Cloud Deployment Considerations

Cloud environments offer tremendous flexibility with less concern for how components are physically connected. The need for advanced planning is reduced but still important. This section offers suggestions for better provisioning of data and computing resources.

Initial Criteria

- Elasticity
- CPU and Computation
- Resilience
- Security
- Optimized provisioning

Elasticity

Elasticity is the ability for a cloud solution to provision and de-provision computing resources on demand as workloads change. Public clouds have a distinct advantage since they generally have larger pools of resources available. You also benefit by only paying for what you use. Private clouds and dedicated hardware can make up some of the difference with higher bandwidth data paths.

In the API economy environment, enterprises are expanding their channels and ecosystems via APIs. The number of API calls can fluctuate as can the associated number of transactions. This is exemplified in cases of seasonal surges such as the holiday season for the retail industry. An API platform needs to provide scalable processing capabilities for a fast performing runtime, quick access to data resources, real-time security enforcement, and prompt collection of usage metrics.

CPU and Computation

The availability of inexpensive commodity processors means the private and hybrid cloud server farms are more viable than in the past. Modern development environments using Hadoop, Spark and Jupyter (iPython) take advantage of these massively parallel systems.

Streams and high speed analytics are an emerging area where cloud applications leverage more powerful processor pools to enable real-time, in-motion data solutions. Dedicated hardware allows for faster development and testing prior to migration towards hybrid and public environments.

A successful API platform might require the deployment of multiple environments to support development lifecycle requirements, regional compliance for data location and isolation or simply

dedicated runtimes or data repositories. A cloud infrastructure provides this flexibility in provisioning resources and the integrity that all resources are functioning as one entity.

Resilience
Resilience and fault tolerance are critical to a successful API platform. API management platforms should not depend on one single component at any point and should tolerate the failure of a single component, such as the API gateway or the API developer portal. Components in the provider cloud can be made resilient through clustering and the use of multiple instances of programs and cloud services combined with data replication and redundancy on multiple storage systems.

The networks should also be resilient, for example with multiple paths and multiple providers in the public network. There is no silver bullet to make the entire network available all the time but it should be highly available and resilient. It is important to ensure that the connectivity capabilities can support resilience.

Security
As more data about people, financial transactions, and operational decisions is collected, refined and stored, the challenges related to information governance and security increase. The data privacy and identity management of devices and individuals is very important from a cloud computing point of view. The simple fact is that more people have access to data calls for better monitoring and compliance strategies. The cloud generally allows for faster deployment of new compliance and monitoring tools that encourage agile policy and compliance frameworks. Tools that monitor activity and data access can actually make cloud systems more secure than standalone systems. Hybrid systems offer unique application governance features: software can be centrally maintained in a distributed environment with data stored in-house to meet jurisdictional policies.

Security related to APIs is tightly coupled with the access control to the data they are delivering and consequently API security policies can be more complex and granular especially when brokering authentication and authorization credentials across domains. An API platform will be able to take advantage of the many standard gateway services and network security capabilities provided by cloud providers to run within a secure perimeter.

Hybrid Cloud and APIs

An enterprise routinely needs a combination of public cloud, private cloud, and on-premises components that when linked, create a hybrid cloud. Hybrid cloud computing is a deployment model which involves combining the use of multiple cloud services across different deployment models – in particular, combining the use of public cloud services with private cloud services and existing on-premises enterprise systems. See the CSCC's *Practical Guide to Hybrid Cloud Computing* for more information about hybrid cloud. [2]

Businesses implementing hybrid cloud solutions are looking for flexibility and agility in delivering new capabilities. Some examples that explain the need for hybrid cloud deployment for API Management:

- Mobile workforce using mobile applications deployed on the public or private cloud and invoking APIs that access data and transactions located in on-premises data centers.
- Market channel expansion where the enterprise digital platform is hosted on a cloud environment and exposes core business capabilities from backend systems residing on-premises via a set of APIs.

- Enterprise B2B integration where inter-enterprise collaboration is enabled through a set of APIs hosted on a cloud platform and brokers B2B capabilities from on-premises back-ends, enterprise private clouds or external cloud services such as commercial SaaS applications.

API Management Deployment Considerations

Security

Security is enforced by the API Gateway component. In that role, the API Gateway applies configured policies to all requests: authentication, authorization, traffic management, routing, and other types of policies.

- **Encryption Support -** To increase mobile and API security for protecting mission-critical transactions, an API Gateway provides JSON Web Encryption, JSON Web Signature, JSON Web Key, and JSON Web Token. It also protects mission-critical applications from security vulnerabilities.
- **Policy Authoring -** To simplify policy authoring, an API Gateway's pre-configured policies can be used to enable quick delivery of gateway capabilities without any custom policy authoring or coding.
- **Open Standards -** From an openness standpoint, an API Gateway provides flexible user authentication for Single Sign-On (SSO) to web, mobile and API workloads using social or enterprise identities based on OpenID connect.
- **OAuth authorization standard -** An API Gateway should support OAuth. Creating an OAuth security definition in an API provides settings for controlling access to the API operations through the OAuth authorization standard. OAuth is a token-based authorization protocol that allows third-party websites or applications to access user data without requiring the user to share personal information.

Isolation considerations

Deployment can be planned to cover varying degrees of isolation among different environments:

- **Platform level -** Provides a completely separate installation of the environment.
- **Organization level -** An organization is a logical partition within an API management platform. If an API were to be included within two organizations, the same process would be used across the platform.
- **Catalog level -** An API catalog is a logical partition of the Gateway and the Developer Portal. The URL for both the API calls and the Developer Portal are unique to a particular catalog.

Within the API Management platform, customers will want to create environments that correspond to the software development lifecycle, such as development, functional test, system test and production. Depending upon the partitioning/isolation requirements between these environments, (development, test, etc.) they will map to either catalogs, organizations, or a separate API Management platform.

For example, catalogs can match the software development environments such as development, test or production; or they can correspond to different API catalogs within environments (e.g., internal and external). Each catalog has:

- A Gateway service associated with it that handles API requests. (Note a Gateway service can be used for more than one catalog).
- Its own Developer Portal.

Scalability and Load Balancing

The scalability of each service (Management, Gateway, API Runtime or Portal) is provided by adding additional instances which can also be done to a service at runtime for scalability and/or resilience. The scalability of an entire platform has an additional dimension of prioritizing services or a group of services within a time period.

The Management service receives requests from the Gateway, Developer Portal, Developer Toolkit, and the admin user interface. External load balancing is required to receive requests from these different services. The Gateway and Developer Portal can provide their own load balancing mechanism to the management servers.

The Gateway service can use the internal load balancer to process incoming requests if it has that capability, otherwise, it can use an external load balancer.

Developer Portal Servers are clustered together to provide scalability. An external load balancer is always required for the Developer Portal when high availability and scalability requirements exist.

The API Runtime can be clustered and managed via a cluster controller. It can be composed of multiple clusters depending upon the scalability and high availability requirements. An external load balancer is always required to balance the traffic across the available instances of the application running on the clustered environment.

The API Runtime can also allocate more CPU bandwidth to certain requests within predefined time slots or based on the Predictive Analytics which project a short fall before critical deadlines. Cloud service providers could achieve this by feeding metrics from the API Management platform into predictive analytics capabilities from other external advanced analytic services on the cloud.

References

[1] OpenAPI Specification. https://www.openapis.org/

[2] Cloud Standards Customer Council 2016, Practical Guide to Hybrid Cloud Computing. http://www.cloud-council.org/deliverables/practical-guide-to-hybrid-cloud-computing.htm

Acknowledgements

The major contributors to this whitepaper are: Ahmed Abbass (IBM), Hussain Chinoy (Bespoke Systems), Chris McCarthy (State Street), Tien Nguyen (IBM) and Chandra Singh (Honeywell).