How IBM leads building hybrid cloud solutions: Implementing the CSCC Customer Cloud Architecture for API Management

Table of Contents

1 Introduction .....................................................................................................................................2
2 APIs, cloud computing, and enterprise digital transformation ................................. 3
3 Understanding API management platforms ................................................................... 4
   3.1 API management lifecycle ......................................................................................... 5
   3.2 API management personas ...................................................................................... 7
   3.3 Introducing IBM API Connect .................................................................................. 7
4 API management architectural capabilities .................................................................. 8
   4.1 API developer toolkit ............................................................................................... 9
      4.1.1 Capabilities ........................................................................................................ 9
      4.1.2 Solution feature: IBM API Connect toolkit .......................................................... 9
   4.2 API gateway ............................................................................................................ 10
      4.2.1 Capabilities ........................................................................................................ 10
      4.2.2 Solution feature: IBM API Connect Gateway ....................................................... 11
   4.3 API runtime ............................................................................................................ 12
      4.3.1 Capabilities ........................................................................................................ 12
      4.3.2 Solution feature: IBM API Connect Runtime ....................................................... 12
   4.4 API management service ......................................................................................... 13
      4.4.1 Capabilities ........................................................................................................ 13
      4.4.2 Solution feature: IBM API Connect Manager Server ........................................... 14
   4.5 Developer portal ........................................................................................................ 17
      4.5.1 Capabilities ........................................................................................................ 17
      4.5.2 Solution feature: IBM API Connect Developer Portal ......................................... 17
5 API management runtime flow ...................................................................................... 20
6 API management deployment considerations .......................................................... 21
   6.1 Security ....................................................................................................................... 21
   6.2 Cloud topology view ................................................................................................. 22
   6.3 Isolation considerations ............................................................................................. 23
   6.4 Scalability and load balancing ................................................................................... 23
   6.5 High availability ........................................................................................................ 24
7 Summary .......................................................................................................................... 24
The objective of this paper is to show how IBM products can be used to support the functional characteristics described in Customer Cloud Reference Architecture for API Management, a paper published by the Cloud Standards Customer Council (CSCC). The architectural capabilities described in the CSCC paper are an essential set of ingredients needed to instantiate an API runtime and management environment using a private, public, or hybrid cloud deployment model.

This paper introduces API management and the architecture elements of an effective API management platform that supports the enterprise’s API strategy and its roadmap for digital transformation. We review the requirements and characteristics for a robust and comprehensive cloud-based API management foundation as explained in the CSCC paper. We then show how IBM supports the capabilities and components with scenarios illustrating the application of IBM products.

1 Introduction

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 reveal data 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.

An API management platform accelerates innovation by making it easier to discover new business assets in current enterprise systems. Existing functionality can be exposed as APIs and published on a self-service portal, which can be used by developers of digital applications who want to consume those APIs. This makes existing enterprise assets available to new channels and new audiences, with enriched customer experience in integrated omnichannel 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.
The concept of API management must be expanded from several perspectives:

- **Why**: The value proposition of adopting a long-term API strategy and embarking on the enterprise digital transformation journey.
- **Where**: The principles and characteristics of selecting a solid API management platform.
- **How**: The comprehensive lifecycle approach to creating, running, managing, and securing APIs.
- **Who**: The multiple personas and stakeholders in API management and their use cases.
- **What**: The architectural components and capabilities that make up a superior API management platform, including runtime characteristics and deployment considerations.

2 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. These systems are not flexible enough to dynamically adjust to constant external channel fluctuations. Exposing the business assets behind those systems via an API interface allows an external developer ecosystem to access existing enterprise core business and to create innovative channel applications. Thus, APIs are a form of “crowdsourcing” that empowers digital disruption, opening the way to new kinds of solutions to grow the customer base, drive innovation, improve time-to-value, and expose more possibilities for a flexible future.

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

1. Accelerating in-house development to 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 mobile or Internet of Things (IoT) applications on a **public cloud** consume exposed APIs.
4. Joining or forming an ecosystem with a wider community of external developers and partners, which allows for publishing and consuming 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 comprehensive API management to support the lifecycle of APIs. This includes creating and testing APIs and connecting their implementation code to back-end systems. It also includes securing access to those APIs and managing them in production, whether they are accessed from System of Engagement types of applications, Systems of Record, or other types of applications. This is in addition to making them available on a self-service developer portal for application developers to use.
3 Understanding API management platforms

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

A superior API management platform should provide a comprehensive set of capabilities to cover 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 both 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**: You need 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 enable innovation and agility within the 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**: The service level agreement in API management is backed by platform characteristics such as performance, scalability, load balancing, and failover.
- **Lifecycle and governance for APIs, products, and plans**: Productizing the APIs, packaging and cataloging them, and tracking their lifecycle are all activities that help provide effective management and control of the APIs as they are deployed.
▪ **Access control over APIs, API plans, and API products**: Another key function for security is managing the access to APIs at various levels of granularity, involving users and user groups in a consumer or a provider role.

▪ **Advanced API usage analytics**: Monitoring and analyzing API usage metrics from different user perspectives and roles provides a feedback loop to the API owners and developers for future improvement.

▪ **Customizable, self-service developer portal for publicizing APIs**: Publicizing and socializing the APIs through a user-friendly portal is crucial in promoting the value of your core business and the market reach of your brand.

▪ **Policy enforcement, security, and control**: A high performing and scalable API security gateway is imperative in any API management platform to protect access to your back-end systems.

### 3.1 API management lifecycle

There are four key phases in the lifecycle of an API, each of which requires a rich set of capabilities.

1. **Create phase**: covers the development lifecycle, design, model, test, build, and deploy.
   - Rapid model-driven API creation
     Create a model representing a resource that API management can use in a variety of ways, such as exposing it via a set of REST endpoints, persisting it to a data source (for example, in-memory or MySQL), or manipulating it programmatically as a JavaScript object.
   - Data source to API mapping automation
     Create models by discovering fields from existing database tables on various types of databases, such as MySQL, Oracle, PostgreSQL, and SQL Server.
   - Standards-based visual API spec creation in Swagger 2.0
     Create OpenAPI (Swagger 2.0) definition files for created APIs.
   - Local API creation and testing
     Empower developers to create and test APIs locally on their laptops and stage them to an on-premises or cloud deployment.
   - On-cloud and on-premises staging of APIs and packaging them into plans and products
     Create products that include plans of created APIs and copy all files to the target, whether it is on premises or on cloud, without running the application (which only happens after publishing).

2. **Run phase**: covers the performance, scalability, load, and resilience of the API runtime platform.
   - Polyglot microservices runtime
     Create the microservices by using unified Java and Node operations and management.
   - Integrated runtime management for availability, load, and performance
     Configure runtimes to meet runtime requirements.
- Enterprise high availability and scaling
  Deploy multiple management servers and multiple gateway servers to achieve high availability, scalability, or resilience.
- On-cloud and on-premises staging of microservices applications
  Create microservice implementation to the target whether it is on premises or on cloud before making it available to application developers.

3. **Manage phase**: 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.
   - API discovery model
     Publish APIs for application developers to find and use in their applications.
   - API, plan, and product policy creation
     If not already done during “Create” phase, create plans, products, and associate policies to them.
   - API, plan, and product lifecycle management
     Manage the lifecycle of APIs, plans, and products.
   - API visibility via self-service, customizable developer portals
     Control access to APIs, plans, and products so that they can only be accessed by authorized application developers.
   - Advanced analytics on API usage and performance metrics
     Monitor the usage and performance of published APIs.
   - Subscription and community management
     Manage user accounts of application developers who can access the developer portal.

4. **Secure phase**: covers the runtime security enforcement of APIs in terms of authentication, authorization, rate limits, encryption, and proxying of APIs.
   - Dynamic API policy enforcement
     Dynamically associate a loosely-coupled policy to an API without needing to restart it or use late binding to associate policies at runtime.
   - Enterprise security and gateway capability
     Control and secure access to endpoints against threats and unauthorized usage.
   - Quota management and rate limiting
     Block access to APIs when an application behaves suspiciously, exceeds rate limits, or becomes compromised.
   - Content-based routing
     Configure the gateway to route based on a protocol such as HTTP, specific information, or a URL. Alternatively, program how matching happens using a stylesheet.
   - Response caching, load balancing, and offload processing
     Cache response from API calls, load balance over multiple back ends, and perform processing on requests or responses to offload back ends using policies.
3.2 API management personas

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

1. The **application developer** is the consumer of APIs. He will discover and subscribe to the API that he includes in the business logic of his application. He wants to know:
   - Where do I access APIs?
   - How do I understand the APIs?
   - How do I measure success?

2. The **API developer** is the creator of APIs. She designs and implements the logic behind the API to deliver the proper data payload from the back-end business assets or services. She wants to know:
   - How do I design, model, and assemble APIs?
   - How do I manage security?
   - Will the infrastructure scale?
   - How do I measure performance?

3. The **API owner or product manager** is the designated owner of the API and of the business asset that is exposed through that API. He wants to know:
   - How can I rapidly release and update my APIs?
   - How do I publicize my API?
   - How do I measure success?

4. The **API IT operations lead** is part of the cloud provider organization offering the API infrastructure services both runtime and management. She wants 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?

3.3 Introducing IBM API Connect

IBM provides first class product support for APIs in a cloud computing architecture. IBM API Connect™ helps customers create, run, manage, and secure APIs at scale. It is designed for cloud and is a service in IBM Bluemix®, IBM’s Platform as a Service (PaaS) offering. API Connect, combined with IBM Secure Gateway and the products in IBM’s integration and messaging portfolios, help support safe, managed, and scalable hybrid deployment capabilities in enterprise business solutions.
API Connect is a comprehensive end-to-end API lifecycle solution that enables the automated creation of APIs, simple discovery of systems of record, self-service access for internal and third-party developers, and built-in security and governance. Using automated, model-driven tools, create new APIs and microservices based on Node.js and Java runtimes—all managed from a single unified console. It helps ensure secure and controlled access to the APIs using a rich set of enforced policies. API Connect can help drive innovation and engagement with the developer community through the self-service developer portal. API Connect provides streamlined control across the API lifecycle and can enable businesses to gain deep insights around API consumption through its built-in analytics.

The concept behind API Connect is that APIs are small data applications, but they are applications nonetheless. APIs need to be created, tested, deployed, debugged, clustered, scaled, monitored, managed, and administered. The actions in the lifecycle occur before organizations can use API management offerings to add and enforce policies to APIs for security. API Connect enables organizations to perform the actions in the API lifecycle efficiently, while avoiding the issues that commonly surround creating, running, updating, or scaling APIs separately from API management offerings.

For additional monetization, organizations can publish their APIs to select business partners or developer communities using varying rate limit plans. API Connect can analyze API usage and performance to help improve the efficiency of the offerings. API Connect can promote visibility and consistency alongside development, operations and line-of-business teams, because it acknowledges that helping the organization to better align its technology strategy with its business strategy is vital for success.

4 API management architectural capabilities

There are at least five major components in an API management architecture to address the set of capabilities. Two are resilient runtime components and the other three are UI components that interface with the personas.

1. The API developer toolkit is an SDK for API developers to model, create, test APIs locally, and use cloud DevOps services to automate API build-deploy-publish tasks.

2. The API gateway component enforces runtime policies to secure and control API traffic.

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

4. The API management service is used by API owners, API developers, and business users to catalog, package, and publish APIs and to obtain API usage metrics for
monitoring and analytics purposes.

5. The **developer portal** is a website where APIs are made public to the application developer communities to discover the APIs and to subscribe to their usage.

### 4.1 API developer toolkit

The API developer toolkit is the development environment for creating APIs and defining the characteristics of the API exposure. You can create and configure your APIs with the developer toolkit.

#### 4.1.1 Capabilities

- **Develop and compose APIs**
  Creates API definitions that invoke an existing API implementation that runs outside the API management platform, or creates 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 types including:
  - Databases such as Cloudant®, DB2®, MongoDB, and MySQL
  - SOAP or REST web services
  - Email
  - In-memory

- **Build, deploy, scale APIs**
  Creates required artifacts to implement the API and associated definitions and policies.

- **Monitor and debug APIs**
  Provides functionality to test APIs both in an interactive manner and by enabling debugging information to be logged for each execution step.

#### 4.1.2 Solution feature: IBM API Connect toolkit

The IBM API Connect developer toolkit provides commands for working with APIs, products, and applications, and for launching the API Designer graphical tool. It features a command-line tool, `apic`, for creating and testing APIs that you can then run, manage, and secure with API Connect. The API developer can also use the command-line tool to script tasks such as continuous integration and delivery.
The API Connect API Designer tool features a model generator that allows the API developer to create a data model for the API, add business logic, customize the model, attach it to back-end data sources, and expose the model over REST.

If there is an existing back-end schema (like a database), the API developer can create models based on that schema. This process is called discovery and is supported by data source connectors for MySQL, Oracle, PostgreSQL, and SQL Server.

For unstructured data such as that in NoSQL databases and REST services, the developer can create API models using instance introspection, which creates a model from a single model instance from MongoDB, REST, and SOAP data sources.

Additionally, API developers can rapidly create, proxy, assemble, and secure APIs through the API Designer user interface (UI). The API Designer assembly feature allows API developers to create complex REST or SOAP API operations that transform data, perform multiple service calls, aggregate data, and apply policies.

The IBM API Connect API Manager interface has API monitoring and debugging functionality that allows the developer to test and ensure that the API is defined and implemented correctly. The interface has an integrated test tool. As part of the testing process, the test tool stages and publishes an API product for the developer. The test tool then calls the API and displays the result of that call.

### 4.2 API gateway

The API gateway processes and manages security and protocols, and stores relevant user and appliance authentication data. The gateway servers also provide assembly functions that enable APIs to integrate with various endpoints, such as databases or HTTP-based endpoints.

#### 4.2.1 Capabilities

- **API policy enforcement**
  The gateway provides several 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 processing aspect 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 and message transformation.

- **Traffic control**
  Acts as proxy that receives inbound API traffic and routes the 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 and analytics collection**
  Filters, sorts, and aggregates API event data, then 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.

### 4.2.2 Solution feature: IBM API Connect Gateway

The IBM API Connect Gateway server processes and manages security protocols and stores relevant user and appliance authentication data. It constitutes the enforcement point for runtime policies to control API traffic. It also provides assembly functions that enable APIs to integrate with various endpoints, such as databases or HTTP-based endpoints.

The IBM API Connect Gateway comes in two flavors: Micro Gateway and the DataPower Gateway. Both gateways provide the runtime component where APIs execute depending on the application needs.

- The Micro Gateway is a gateway that is built on Node.js and provides enforcement for the authentication, authorization, and flow requirements of an API. The Micro Gateway is deployed on an API Connect collective and has a limited number of policies available to it.

- The DataPower Gateway is a gateway that is deployed on a virtual or physical IBM DataPower® appliance. DataPower Gateway has more policies available to it than the Micro Gateway and can handle enterprise-level complex integration.

API Connect offers a variety of options for securing APIs through these categories of security definitions.

<table>
<thead>
<tr>
<th>Security Definition</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Basic authentication</td>
<td>Use a basic authentication security definition to specify a user registry or an authentication URL to be used to authenticate access to the API.</td>
</tr>
<tr>
<td>API key</td>
<td>Use an API key security definition to specify what application credentials are required to call an API.</td>
</tr>
<tr>
<td>OAuth</td>
<td>Use an OAuth security definition to specify settings for OAuth token-based authentication.</td>
</tr>
</tbody>
</table>
In addition, API Connect allows the creation of schema validation, message filtering, and enrichment via default policies, as well as custom policies to add specific API encryption and decryption, digital signing, and validation of messages. A policy is a set of configurations that controls specific aspects of processing in the IBM API Connect Gateway component during the handling of an API invocation at runtime. API Connect provides several different types of policies by default and allows the creation of user-defined policies for more processing control.

4.3 API runtime
This is a collection of polyglot runtime environments that execute the API code.

4.3.1 Capabilities

- **Unified polyglot API execution environment**
  Provides a set of runtimes to execute an application in the language of choice, such as Java, Node.js, Ruby, and others. It also provides a large set of platforms for mobile such as iOS 8, Android, hybrid, or JavaScript.
- **Provision system resources**
  Creates and binds system resources required to run APIs.
- **Monitor runtime health**
  Monitors different aspects of the health of the environment including 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.

4.3.2 Solution feature: IBM API Connect Runtime

The API Connect Collective provides a runtime environment for deploying applications to the collective member hosts. The collective is a collection of servers (hosts) that provides automatic scaling (via scaling policy) of applications. The collective supports multiple runtime languages (polyglot) and manages the deployment, upgrading, high availability, and management of the runtimes.
A collective controller manages one or more collective members. Requests to multiple members are managed by the On-Demand Router (ODR) plugin which is in turn managed by the controller. The controller tracks member server states and maintains a list of active servers for routing. This figure shows a typical topology setup of API Connect.

### 4.4 API management service

This service manages the operations of the various servers in the API management platform. The management service also provides analytics functions that collect and store information about APIs and API users. The analytics are used in providing feedback to the API owner, the API developer, and IT operations in terms of API usage and performance.

#### 4.4.1 Capabilities

- **API, plan, product, policy creation**
  Groups APIs and plans in 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 and associates them to APIs or plans.

- **API product versioning and lifecycle management**
  Defines multiple versions of a product. These versions can occupy any of the lifecycle stages, which facilitates development.

- **API monitoring and analytics**
  Creates custom analytics dashboards for catalogs that consist of default or user created visualizations such as tables, graphs, and maps.

- **Subscription and community management**
  Manages requests sent by application developers to subscribe to a plan or a product. Also manages the developer organizations that access APIs and plans when their users sign up to use the developer portal.

### 4.4.2 Solution feature: IBM API Connect Manager Server

With the API Connect API Manager, the API developer can configure the availability of their APIs. For example, availability options include:

- Publishing products
- Configuring subscription options
- Managing API versions

API development in IBM API Connect is done using the IBM API Connect Developer Toolkit, the API Designer GUI, or both. When development is complete, the toolkit also lets the API developer publish APIs to the API Connect runtime and the API Manager Server. The developer publishes APIs by including them in a product and publishing the product. The developer defines APIs and products by creating and validating YAML definition files in his local file system. He can then interact with the IBM API Manager Server by using either the API Designer or the toolkit commands.

In API Connect, APIs are managed through **products** and **plans**. A **product** is the consumable packaged offering to be made public to the application developer; a **plan** determines which APIs are included and how they are to be used. Plans and APIs are grouped together in products, through which the API owner manages the availability and visibility of those APIs and plans. The API Designer is used to create, edit, and stage the product, and the API Manager manages the lifecycle of the product.
The following diagram demonstrates how products, plans, and APIs relate to one another.

Create a product to collect a set of APIs and plans into one offering that is made available to application developers. A plan includes rate limit settings that can be applied to the entire plan or specified for each operation in an API. To make an API available to an application developer, you must include it in a plan. Plans can only be created within products, and these products are published in a catalog. Using products and plans, API Connect provides greater control over what APIs the application developers have access to.

The example below shows how the “Free” plan allows the use of read-only APIs in the “Account” product and the “Customer” product.

After creating products and plans, a lifecycle manager can control the availability and visibility of APIs and plans through the API Manager UI. Using the API Manager, the API owner can specify the plans that an application developer can subscribe to through the developer portal. The application developer can only subscribe to one plan from a specific product. Multiple plans within a single product are useful because they can fulfill similar purposes but have differing levels of performance. For example, you can have a "Demo Plan" that makes a single API available and a "Full Plan" that makes several APIs available.
Additionally, you can use different plans to implement different rate limits. A rate limit determines how many times the API can be called by a particular developer. You can implement a rate limit as a default rate that is shared across an entire plan, or you can set a rate limit for specific operations of an API within that plan, exempting the operations from the shared plan rate limit. Different plans can have differing rate limits, both between operations and for the overall limit. This is useful for providing differing levels of service to customers. For example, a "Demo Plan" might enforce a rate limit of ten calls per minute, while a "Full Plan" might permit up to 1000 calls per second.

APIs become accessible when you publish a product and make it visible on the IBM API Connect Developer Portal for use by application developers. A product can be published to selected communities of application developer organizations, and the plans within the product can be used to further tailor access and visibility.

API Connect gives you the ability to perform subscription and community management for controlling the consumption of APIs.

A community is a collection of developer organizations. You can use communities to control which organizations have access to products and plans without having to assign access on an individual basis. You can use communities to restrict the visibility and accessibility of APIs to specific business partners, internal organizations, or other groups of application developers, for example.

Publishing a product to selected communities means that only the application developers within the organizations that are contained within the community can see the product on the developer portal and can obtain application keys to access it. Alternatively, you can publish a product to all communities.

After publishing an API, applications that belong to users in the communities can subscribe to a product and to the API contained in the product.

API Connect also provides many features in API monitoring and analytics. The API owner can filter, sort, and aggregate API event data, then present 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.

The analytics feature in API Connect is built on the Kibana V4.3 open source analytics and visualization platform, which is designed to work with the Elasticsearch real-time distributed search and analytics engine. The user can export dashboards so that they can be imported by other users or into other catalogs on the system. Dashboards are exported as .json files, which can then be imported. During the export, the filters that are currently applied are preserved in the exported file.

From the API Manager UI, the user can view and export the raw analytics data that is returned in configured visualizations. The user can also view the individual API event records that are
generated for the aggregated data sets in the visualizations and can collectively export all the API event records that relate to all visualizations in a dashboard. Any exported analytics or API event data is saved to a comma-separated values (CSV) file.

4.5 Developer portal
The developer portal enables API providers to build a customized developer portal for their application developers. It is a portal where you can publish your APIs to encourage the development of new applications that extend the value of your core enterprise assets.

4.5.1 Capabilities

- **API discovery**
  Allows application developers to discover and use published APIs to which they have access.

- **Self-service application developer portal**
  Provides self-service sign up for rapid onboarding of application developers from enterprise, partner, and third-party developer community.

- **Clustering capability**
  Provides the ability to cluster the portal over multiple nodes so that developer portal users will always have access to the site.

- **Branding and customization**
  Customizes the theme and the appearance of the developer portal.

4.5.2 Solution feature: IBM API Connect Developer Portal

Users of the API Manager UI can publish products and APIs to the IBM API Connect Developer Portal for application developers to access and discover the APIs. The developer portal provides additional features including forums, blogs, comments, ratings, and an administrative interface for customizing the portal. From the developer portal, the application developer can view interactive analytic information about all the APIs within an organization. They can learn more about an organization's APIs and register applications that use those APIs.

The IBM API Connect Developer Portal can accelerate APIs time to market with ease of onboarding and fast enterprise branding and customization.
• Self-service onboarding
When self-service onboarding is enabled in the developer portal (which is the default), users can join a developer organization without being invited by the organization’s owner.
  o Users: After you log in to the developer portal site, you have a user record in the developer portal database. User ID 1 is always reserved for the administrator account. Regardless of whether other user accounts are using remote authentication such as LDAP, the administrator account will always be a local account to enable administration of the developer portal site.
  o Roles: A role is a collection of permissions that define the actions that a user can perform in the developer portal site. You can assign users one or more roles. By default, a user who is logged in to the developer portal will be in the authenticated role. Other roles that can be assigned to a user include:
    ▪ Administrator
    ▪ Forum moderator
    ▪ Content author
    ▪ New custom roles can be created
  o Permissions: Permissions define the actions that a user can or cannot perform in a developer portal site. Permissions are additive. If a permission that enables a user to perform an action is not assigned, the user cannot perform the action. If a user has multiple roles and any of them contain a specific permission, the user will be able to perform that action.

• Branding and customization
Branding and customization can be achieved easily in the IBM API Connect Developer Portal through its many configurable features, including customizing the front page banner image, integrating Twitter data into the developer portal’s social feed, and customizing themes. In addition, the user can customize nodes, fields, content types, regions, blocks and modules, and more.

The IBM API Connect Developer Portal offers branding and customization at multiple granularity levels to accommodate the changing needs of enterprise customers. These are all components of the developer portal that supports customization:
  o Nodes: A node is a piece of individual content such as a page, poll, article, forum topic, or a blog entry. Each piece of content in a developer portal site is a node. You can apply new features or changes to all content of a single type.
  o Fields: A field is a type of data that can be added to an element. The following types of data are examples of fields:
    ▪ Title
    ▪ Body
    ▪ Comment body
    ▪ Tags
    ▪ Image
    ▪ Different types of fields can be created
- **Content types:** A content type is a predefined collection of fields. Content types define the default fields that content editors use to add content in a developer portal site, and help structure the authoring and developing of content. Content types can be displayed in the developer portal. You can control which content types are displayed as well as the order and format in which they are displayed in the developer portal.

- **Themes:** Themes enable you to control the appearance of your developer portal site. You can modify the appearance of the theme in the following ways:
  - Identify and use a theme that is provided by the Drupal community or third-party website and modify the theme settings.
  - Modify or extend the code of an existing theme.
  - Create a completely new theme.

  You can also create a sub-theme that inherits all the settings of its parent theme.

- **Regions:** Regions are the specific areas of a developer portal site in which content can be placed. Regions are customized and styled in the theme.

- **Blocks:** Blocks are boxes of content that can be displayed in regions on your developer portal site. Blocks can be made available to your developer portal site by enabling specific modules. After creating a block, you can modify its appearance, shape, size, and position. You can also define on which developer portal page or pages blocks appear. Some modules can provide multiple blocks when they are enabled, while other modules might not define new blocks.

- **Modules:** Modules are similar to the concept of plugins as they extend the core functionality of the developer portal site. A set of modules is implemented by default with the Drupal core, and there are additional modules that can be enabled to extend the default functionality. You can find and add more modules to your developer portal site from the Internet.

- **Views:** Views enable you to manipulate the content that is displayed on a page, block, and other visual elements in the developer portal site. Use views in conjunction with content types to format the site’s appearance to your specifications.

- **Panels:** Panels are modules that allow a site administrator to create customized layouts for multiple uses. It is a drag-and-drop content manager that allows you to visually design a layout and to place content within that layout. Integration with other systems allows you to create nodes and landing pages that use this and to override system pages such as taxonomy and the node page. Panels enable you to customize the layout of your site with detailed permissions.

- **Pages:** Panels and other appearance-modifying features of the developer portal use pages to customize the appearance of the portal. Pages can be customized to be as specific as you require, and can be configured to satisfy the context of the situation in which they are used.

- **Templates:** Template files define how the output of a given component can look. They are formatted as PHP template files. There are template files for each section of the developer portal visual layout, such as body of HTML pages, content nodes, comment nodes, search results, product previews, and more.
From a dynamic collaboration view at runtime, the components of the API management platform interact with one another and support the various use cases involving the four types of actors: application developer, API developer, API owner, and IT operations.

**Figure 2 - API Management: Component interaction**

**Interaction flow:**

1. API developer signs on to the API management cloud services account. He accesses the API Designer/Developer Toolkit.
   a) He creates the API and implements business logic.
   b) He maps and integrates the API data model to the back-end schema through the transformation and connectivity service.
   c) He tests and deploys the API to the runtime and publishes to API management.

2. API owner signs on to the API management cloud services account. She accesses the API management component.
   a) She includes the API endpoint to existing API products and plans, and specifies access control.
   b) She publishes the API to the developer portal for external discovery by application developers.
3. Application developer accesses the developer portal. He searches for and discovers the API.

4. He uses the API in his application and deploys his application to the device.

5. The device user opens the application which issues the API request.
   a) The request is handled by the API gateway, which performs load balancing and security validation for all API requests.
   b) The API gateway validates access policies with API management and invokes the API.
   c) The API polyglot runtime executes the API and obtains the data payload from the back end. The API response is sent back to the API gateway. Alternatively, APIs exposed by enterprise applications can be executed on that enterprise application runtime.
   d) The API gateway forwards the response to the calling application.
   e) The API gateway reports usage metrics and analytics to API management.

6. API developers and API owners can log on to the API analytics visualization component to view dashboards on API usage metrics and other analytics.

7. Cloud provider IT operations log on to the polyglot runtime to monitor and manage the API runtime environments.

6 API management deployment considerations

6.1 Security

The API gateway component enforces security. In that role, the API gateway applies configured policies to all traffic including authentication, authorization, traffic management, routing, and other types of policies.

**Encryption support:** To increase mobile and API security for protecting mission-critical transactions, the API gateway provides JSON Encryption, JSON Signature, JSON Key, and JSON Token. It also protects mission-critical applications from security vulnerabilities with enhanced TLS protocol support using elliptic curve cryptography (ECC), Server Name Indication (SNI), and perfect forward secrecy (PFS).

**Policy authoring:** To simplify policy authoring, the API gateway 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, the API gateway provides flexible user authentication for single sign-on (SSO) to web, mobile, and API workloads using social (such as Google) or enterprise identities based on OpenID connect.
**OAuth authorization standard:** The API gateway supports OAuth. When you create an OAuth security definition in an API, you provide 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.

### 6.2 Cloud topology view

![Deployment View: Server Communications](image)

**Figure 3 - Communications between servers in the API Management deployment instance**

1. Between API gateway and API management:
   - Gateway pulls configuration data from API management.
   - Gateway pushes usage analytics data to API management.
   - API management deploys APIs to the gateway.
2. Between API gateway and API runtimes:
   - Gateway invokes APIs.
3. Between API management and API runtimes:
   - API management deploys APIs to the runtime.
   - API runtimes pull configuration data from API management.
   - API runtimes push usage analytics data to API management.
4. Between API management and API developer portal:
   - API management publishes APIs to the portal.
   - Portal pulls configuration data from API management.
6.3 Isolation considerations

You can plan your deployment to cover varying degrees of isolation among different environments.

- **Platform level**: Completely separate installation of the environment.
- **Organization level**: An organization is a logical partition within an API management platform. If an API is included within two organizations, the same process used across the platform is used.
- **Catalog level**: An API catalog behaves as 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 platform, customers want to create environments that correspond to the software development lifecycle, such as development, functional test, system test, and production. Depending on the partitioning and isolation requirements between these environments, they will map to catalogs, organizations, or a separate API platform.

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

- An associated gateway service that handles any API requests. Note that a gateway service can be used for more than one catalog.
- Its own developer portal.

6.4 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, resilience, or both.

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 those 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. The developer portal requires an external load balancer when high availability or scalability requirements exist.

The API runtime can be clustered and managed via a cluster controller. It can be composed of multiple clusters depending on 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.

### 6.5 High availability

The management service controls two types of data:

- **API configuration data** (users, API, and plans)
  At any time, only one management server (the primary) is allowed to write the data. The other servers (secondary) maintain a complete local cache for performance and high availability reasons.

- **API analytics data** (API usage and performance)
  All data that is written to the analytics system is replicated asynchronously on the back-end server. When analytics data is sent from the gateway to a management server, that data is balanced across the available management servers. If there are two or more management servers in the management service, at least two of the management servers hold each analytics data record for redundancy purposes. If a server fails, the others keep running as normal and the whole system can still be used. Internally, one of the management servers is designated as the primary while the others are secondary. If the primary is the one that fails, one of the others will assume the responsibility of being the primary.

The developer portal holds its own data that is replicated to all nodes within the service. At least three nodes are required for high availability because a majority of the nodes must always be functioning. For example, if two of the three nodes are working, that is 66%.

If you are considering multi-datacenter topologies, three data centers are recommended where DC 1 and DC 2 have the same number of nodes and DC 3 has at least one node. This means that a majority of working nodes is always possible even with a single data center failure. If nodes are shut down gracefully (for example, during a scheduled maintenance window), the remaining nodes will continue to function even if they are in a minority.

For the gateway service, if a server fails using internal load balancing, the inflight resource calls fail but any subsequent calls are routed to the gateways that are still operational. For external load balancing, the behavior depends on the external load balancer configuration.

### 7 Summary

The API economy introduces many organizations to new ways of doing business. Whether it’s about improving internal application development, expanding channel access, or enabling partners and external vendors to use enterprise capabilities, the idea of API adoption is essential to support that kind of digital transformation.
API adoption requires a capable API management platform, which provides enablers for an organization’s API strategy throughout the entire API lifecycle. The platform includes capabilities for creating APIs, managing and securing them, and providing the runtime that hosts them. The API management platform consists of key components that provide the capabilities mentioned. Those components are offered in a range of different deployment options to suit every organization’s needs and environment along with underlying nonfunctional requirements.