Microsoft

# API Management in a Hybrid and Multi-Cloud World

# Contents

## API management today

Exposing applications through well-defined application programming interfaces (APIs) lets other software more easily access the services and data those applications provide. The best way to do this today is with *API management*, a technology that puts an intermediary between clients and the applications they access.

*Unified API management increases productivity and maximizes business value.*

In today's cloud-centric world, this intermediary typically runs on a particular cloud platform. Microsoft's *Azure API Management*, for instance, runs on Microsoft Azure. Yet the applications and data your APIs expose are probably spread across multiple on-premises and cloud environments. Managing these APIs consistently throughout your organization is a key part of increasing productivity and maximizing business value. To do this, your organization needs a unified approach to API management across your hybrid and multi-cloud world.

*Azure Arc enabled API Management*, which provides a self-hosted Azure API Management gateway, together with the other features of Azure API Management, addresses this problem. What follows looks first at the value of self-hosted gateways in general, then describes how this Microsoft offering allows unified API management.

### A quick overview of API management

To understand the value of a self-hosted gateway, you first need to understand the fundamentals of API management. Figure 1 illustrates the basic concepts of this technology.

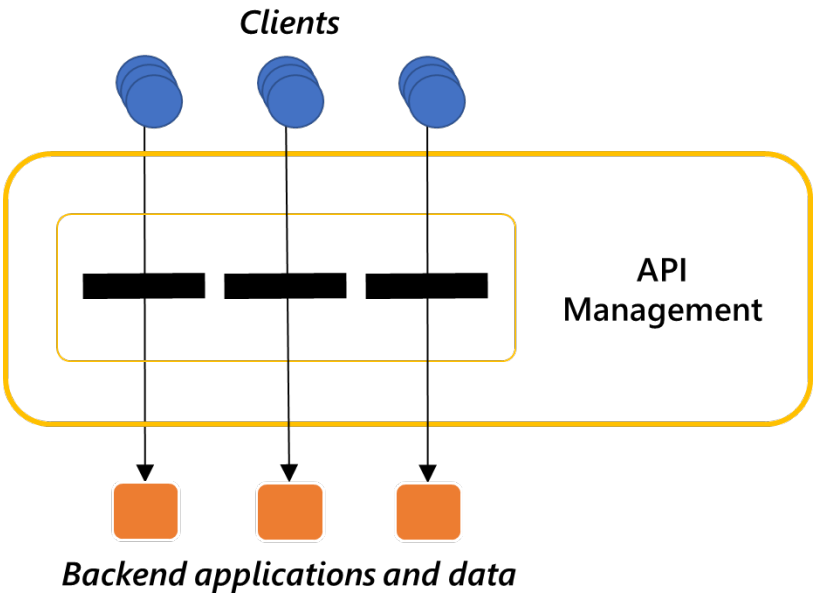


**Figure 1: API management sits between clients and backend applications and data.**

As the figure shows, API management provides an intermediary between clients and the backend applications and data they access. API management solutions offer services such as:

- **Providing self-service user onboarding.** This lets developers learn about APIs, including documentation, code samples, and more, then easily try those APIs. The result is a single place to catalog, discover, and share APIs across even a large organization.
- **Creating a façade that hides backends from clients.** This lets organizations move or re-architect backends without impacting the clients that use them. For example, your organization might exploit this capability to modernize legacy backend systems without affecting other applications that use them.
- **Establishing a single point of entry—a front door—for backend applications.** This allows the API management technology to route client requests intelligently, work with authentication and authorization, throttle clients by providing flow control, and more.

By providing services like these, API management solutions help make APIs more secure, more efficient, and easier to use.

## The problem: Providing unified API management everywhere

Today's most popular API management solutions run in public clouds. As mentioned earlier, for example, Azure API Management runs on Microsoft Azure. And while having a purely cloud-based API management service can work for pretty much all scenarios, it's not always the best choice. To see why, look at Figure 2.

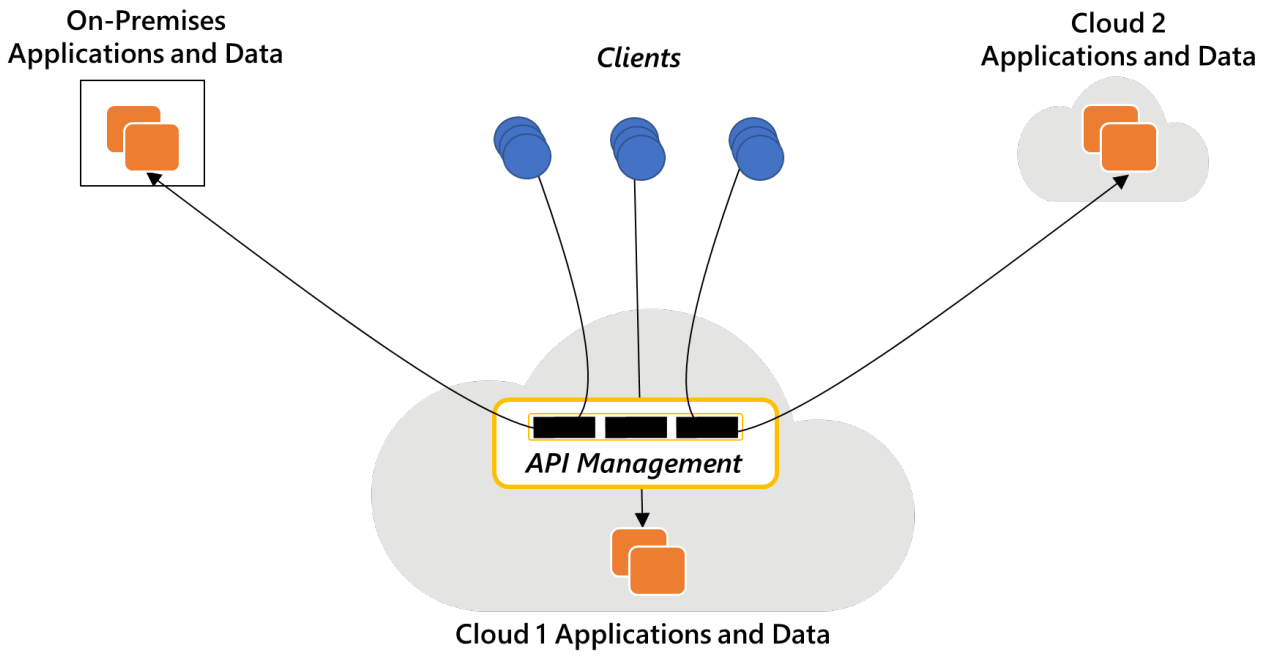


**Figure 2: A cloud-only API management solution requires all communication to go through whatever cloud this solution runs on.**

As this diagram illustrates, all requests from clients to backend applications and data go through the cloud that provides the API management service. For applications and data located in the same cloud as the API management service (called Cloud 1 in Figure 2), this is fine—those requests must access that cloud anyway.

*Running API management solely in one cloud has some limitations.*

But what about requests to backend applications and data that aren't located in Cloud 1? As the figure shows, calls made by clients to on-premises applications and data must also go through Cloud 1 if they're using API management; there's no alternative. Similarly, calls made by clients to applications and data located in other clouds, such as the figure's Cloud 2, are also required to go through Cloud 1 to get API management services. This raises three important issues:

- **What if a client is accessing on-premises applications and data within the same corporate environment, but none of the traffic is allowed to leave this environment?** Perhaps compliance requirements mandate that information must stay on the corporate network, or maybe accessing the cloud is prohibited by company policy. Whatever the reason, scenarios like this can't use an API management service running in any public cloud; the service must run on-premises. This is the most significant problem with API management in many organizations, so it's also the most important to solve.

- **When a client accesses applications and data in another public cloud, why pay for bandwidth twice?** Public clouds today typically charge for data sent out of them, so if a request to an application running in Cloud 2 must first go through Cloud 1 for API management, both clouds will charge you for the transferred data. These charges are small, but still, nobody wants to incur unnecessary costs.

- **If the client and the backend service are in the same environment, such as when they're both on-premises, why incur the delay of going through a public cloud for API management?** The reality is that this delay probably isn't especially significant—after all, most corporate networks today are built from LANs connected via slower wide-area links, and the client and backend service aren't likely to be on the same LAN. Because of this, adding a hop through a public cloud isn't likely to impact performance much. Yet adding needless latency is never a good idea.

*Self-hosted API management gateways can address these limitations.*

All three of these problems are real, and all are important in some situations. And today, it's common to use different solutions for managing APIs in clouds and on-premises. This diversity adds complexity and expense, along with making it hard to provide unified API discovery and governance. Fortunately, there is an approach that lets all these challenges be addressed in a common way: using Azure Arc enabled API Management, by leveraging a self-hosted API management gateway.

## The solution: A self-hosted API management gateway

Even though API management typically runs in the cloud, this doesn't need to be the only option. As Figure 3 shows, introducing the idea of a self-hosted API management gateway allows other possibilities.

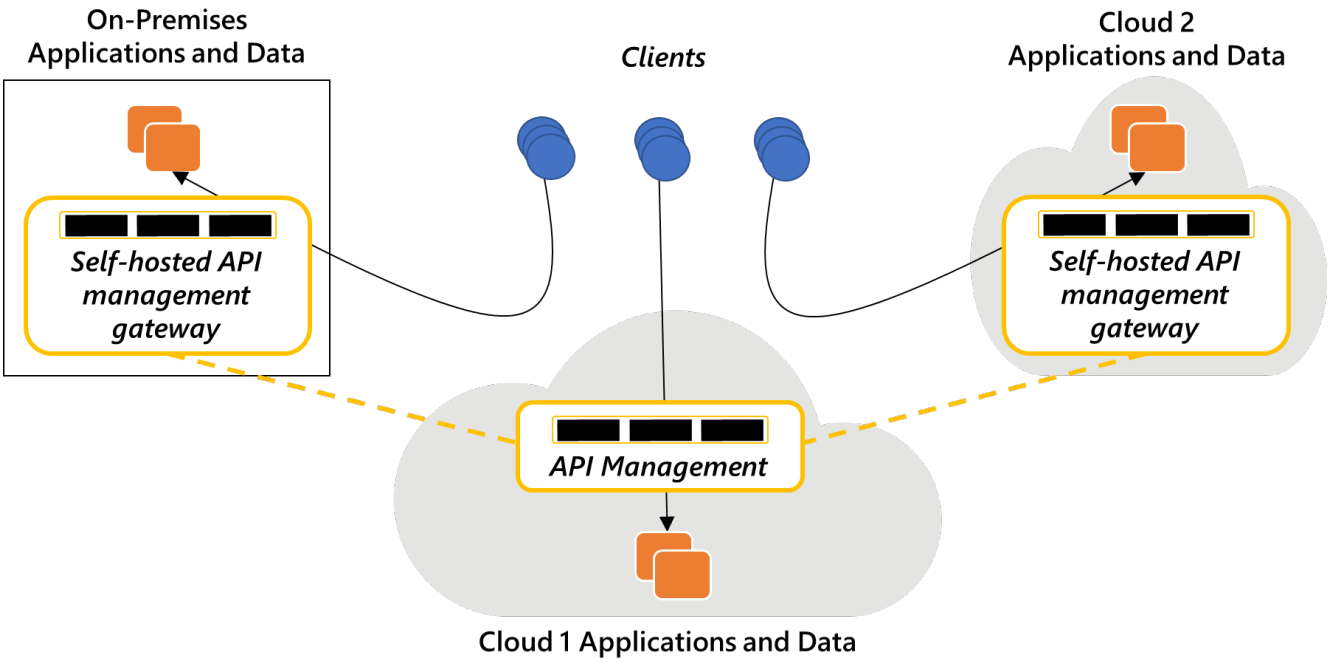


**Figure 3: A self-hosted API management gateway can run on-premises and in other clouds, while still being managed by a primary API management service.**

A self-hosted API management gateway doesn't replace the primary cloud-based API management service. Instead, it augments this service by providing the essential aspects of API management in software that organizations can run wherever they choose. As Figure 3 shows, the two most important scenarios for using this self-hosted gateway are:

- **Running the self-hosted API management gateway in your own on-premises datacenter.** This lets on-premises clients interact with on-premises backend applications without forcing this traffic to go through the public cloud, making it easier to meet compliance requirements. These interactions can still benefit from API management, but those benefits are now provided by the on-premises self-hosted API management gateway.
- **Running the self-hosted API management gateway in another cloud.** This lets clients access backend services in that cloud (such as Cloud 2 in Figure 3) through API management, but without forcing that traffic to go through the cloud that's running the primary API management service. Taking this approach eliminates both the delay and the extra cost of accessing two different clouds just to get API management.

Using a self-hosted API management gateway addresses all three of the problems described in the previous section. And as the dashed lines in Figure 3 suggest, you can do this while still maintaining centralized control: the primary API management service can connect to and manage your self-hosted API management gateways wherever they're running.

*You are responsible for installing and running your self-hosted API management gateways.*

It's worth emphasizing what "self-hosted" means here: your organization will be responsible for installing and running self-hosted API management gateways wherever you need them. You're also responsible for keeping them running, so they do require some extra effort. Still, in scenarios where compliance or cloud data transfer costs or delay are a concern, using a self-hosted API management gateway is likely to be the best approach.

## Understanding the Azure API Management solution

Azure Arc enabled API Management provides a self-hosted API management gateway. To understand this feature, however, it's useful to know a little bit more about the structure of Azure API Management itself. The next section takes a closer look at this cloud service.

### How Azure API Management is structured

Azure API Management, the service running on Microsoft Azure, can be thought of has having three distinct parts. Figure 4 illustrates this idea.

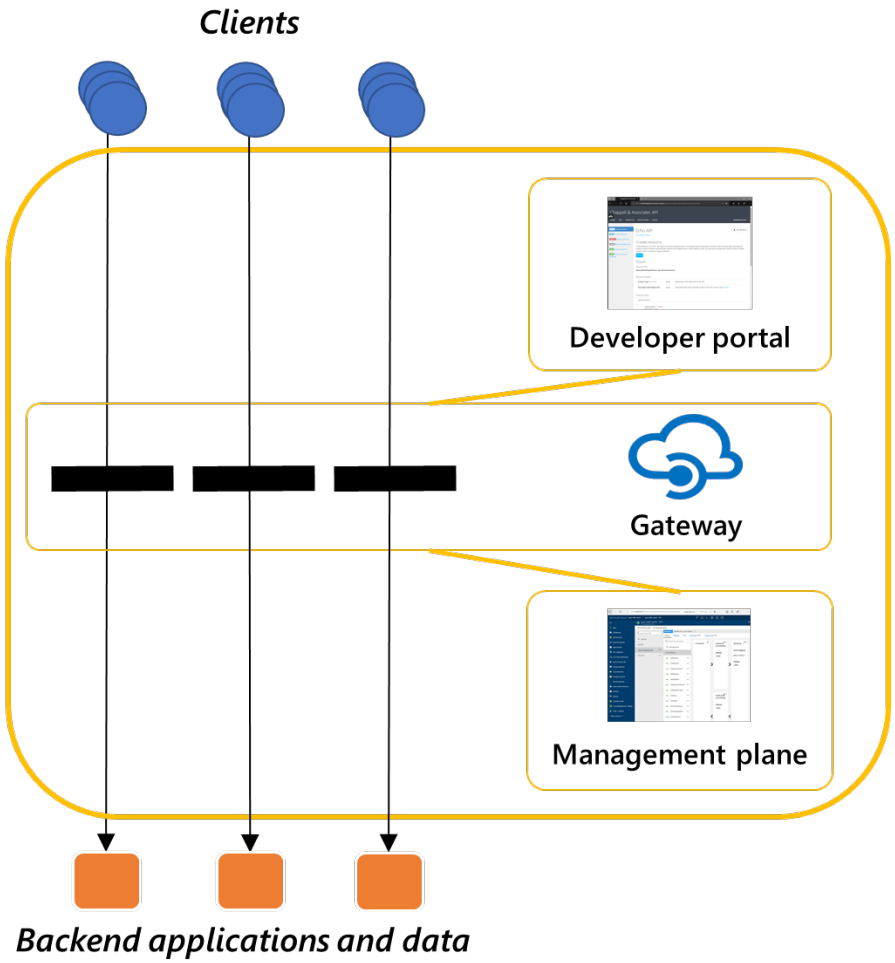


**Figure 4: Azure API Management has three distinct components.**

Those three components are the following:

- **Developer portal***:* People who want to use an API, typically developers, need some way to learn about that API. What calls can it accept? What are the parameters for those calls? The Azure API Management developer portal makes this possible. It can contain documentation describing the API—its operations and parameters—along with sample code for calling the API from various languages and more.

*API management creates facades for accessing backend applications and data.*

- **Gateway:** This part of Azure API Management provides facades—proxies—for the real APIs in backend applications. These proxies intercept all calls from clients, process those calls, then pass them on to backend applications. The gateway can expose REST and

SOAP APIs from all kinds of backend software, including applications running in the cloud and applications running on premises. The technology stack used to build that software—.NET, Java, or anything else—is irrelevant; Azure API Management works with anything.

- **Management plane***:* Many aspects of APIs need to be configured and managed. To allow this, Azure API Management relies on a management plane, a set of tools accessed through the Azure Portal. This part of the service lets an API's owner publish that API, for example, specifying its operations and parameters, then creating a façade in the gateway for the API.

*Azure API Management provides a variety of services for APIs.*

The management plane does more than just publish APIs, however. It provides analytics that let you monitor the usage and health of your published APIs. It also lets you set policies that control various aspects of how each API functions. There are dozens of policies; here are a few examples of what they allow:

- Restricting how many calls can come from a single source in a defined period. This kind of flow control prevents client applications from overwhelming a backend application or data source with requests.
- Blocking calls from specific IP addresses.
- Turning data caching on and off. Caching frequently requested results in Azure API Management can improve an API's performance while minimizing the load on a backend application.
- Converting data from XML to JSON and vice-versa.

The self-hosted Azure API Management gateway is another feature of the Azure API Management offering. It doesn't include everything just described, however. Instead, as described next, the self-hosted gateway provides only the essentials required for its function.

## Understanding the self-hosted Azure API Management gateway

*The self-hosted Azure API Management gateway lets you discover, use, and manage your APIs across all environments.*

Using the self-hosted Azure API Management gateway lets you discover, use, and manage diverse APIs across your on-premises and multi-cloud

environment. Yet the gateway itself need only provide core API management services, such as flow control and caching. Because of this, the self-hosted Azure API Management gateway includes only one of the three parts of the main service: the gateway itself. Figure 5 shows how this looks.

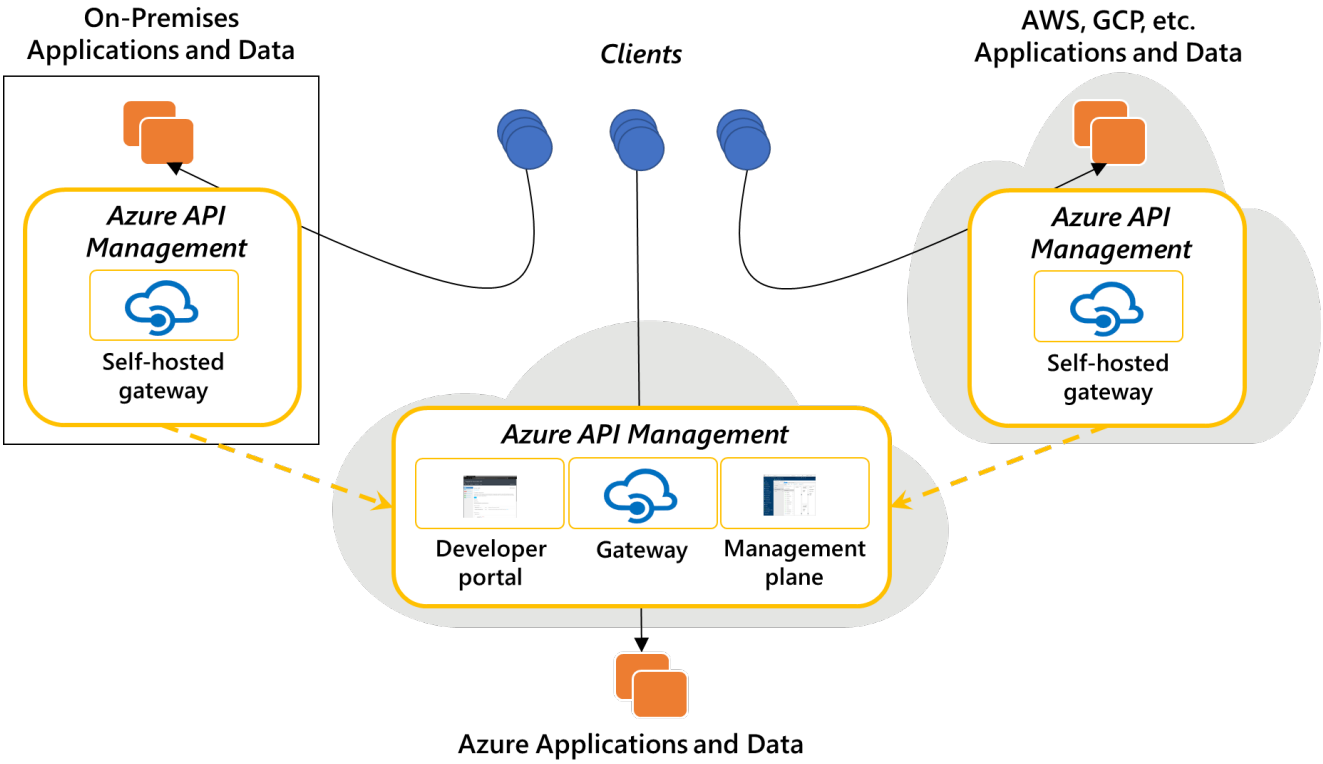


**Figure 5: The self-hosted Azure API Management gateway includes only the gateway functionality of the full Azure API Management service.**

The self-hosted gateway is functionally equivalent to the gateway in the Azure API Management cloud service; it's not a subset. Wherever the self-hosted gateway is running, however, it must be federated with—connected to—the main cloud service. Each self-hosted gateway is attached to and managed by a single Azure API Management cloud service.

Because of this, the federated self-hosted gateways share the developer portal and management plane of the Azure API Management cloud service. This lets your organization have a common approach for these aspects of API management across your entire environment, including on-premises, Azure, and other clouds, such as Amazon Web Services

(AWS) and Google Cloud Platform (GCP). All your managed APIs can be discovered through a single developer portal, and all can be managed from the same management interface. The result is a single unified view across your APIs.

Understanding what the self-hosted gateway does is important, but it's also useful to know a bit about how it's implemented. Some of the most important aspects of this are the following:

*The self-hosted Azure API Management gateway is provided as a Linux-based Docker image.*

- **The gateway functionality of Azure API Management is packaged in a Linux-based Docker image.** All that's required to run the self-hosted gateway is the ability to run a Linux Docker image. You can deploy this container image on Kubernetes if you choose, letting this popular platform handle scaling, upgrades, and more. You can also deploy the image directly on Docker or use other platforms such as Swarm—Kubernetes isn't required.
- **The self-hosted Azure API Management gateway requires only an outbound connection to the main cloud service it's federated with.** This avoids the potential security risk of opening an inbound connection while still allowing two-way interaction between the self-hosted gateway and the Azure cloud service it's connected to. For example, the gateway can be configured to send logs to Azure API Management as needed.
- **By default, the self-hosted gateway doesn't have any local storage.** The cloud service provides all the information a self-hosted gateway needs, such as the flow control policy for each API, and the self-hosted gateway holds this in memory. If the self-hosted gateway fails and loses its connection to the cloud service, however, the information it holds can get lost. To avoid this, the self-hosted gateway can optionally have local persistent storage that holds a backup copy of this data. This storage is just a file, though—it's not a database system—so using this option doesn't create a local management burden. It's a good solution in scenarios that lack reliable cloud connectivity.

Finally, it's worth reiterating that using a self-hosted gateway imposes more of a management burden on your organization. While Azure API Management provides a guaranteed service level agreement (SLA), the reliability of your self-hosted gateways depends on you. You're also required to apply updates to these gateways as needed. Using self-hosted gateways brings some costs along with the benefits they bring. If you don't need them, you're better off sticking with just the cloud-based Azure API Management service.

**The rise of multi-cloud solutions**

The self-hosted Azure API Management gateway wraps an Azure cloud service in a container, then lets that container run outside Azure. If your organization is pursuing a multi-cloud strategy, such as using both Azure and AWS, this lets you have common API management across both clouds.

Could this same approach work for other Azure services? Sure. In fact, there are plenty of situations in which having common services across your entire multi-cloud environment can make your life easier. While the self-hosted Azure API Management gateway is one of the first Azure service to be made available in this way, don't expect it to be the last. It's more correct to view it as a harbinger of the future.

## What to do now

Exposing applications and data through managed APIs is a fundamental part of digital transformation. Doing this often requires spanning multiple clouds and on-premises environments. Rather than incur the complexity and expense of deploying different API management solutions in different places, it makes sense to use a single cloud service combined with self-hosted gateways where needed.

*Azure API Management can provide a unified solution to API management across your environments.*

Azure Arc enabled API Management provides exactly this kind of solution. By providing the same gateway technology wherever you need it, this Microsoft offering lets you put all your diverse APIs behind a

single management service while minimizing your costs. It also gives you a common view of your exposed applications and data, wherever they run.

The result is a unified and modern approach to API management, even in a multi-cloud world. If you're looking for an API management solution, Azure API Management is the place to start.

## For further reading

To learn more about Azure API Management, go to https://azure.microsoft.com/en-us/services/api-management/.