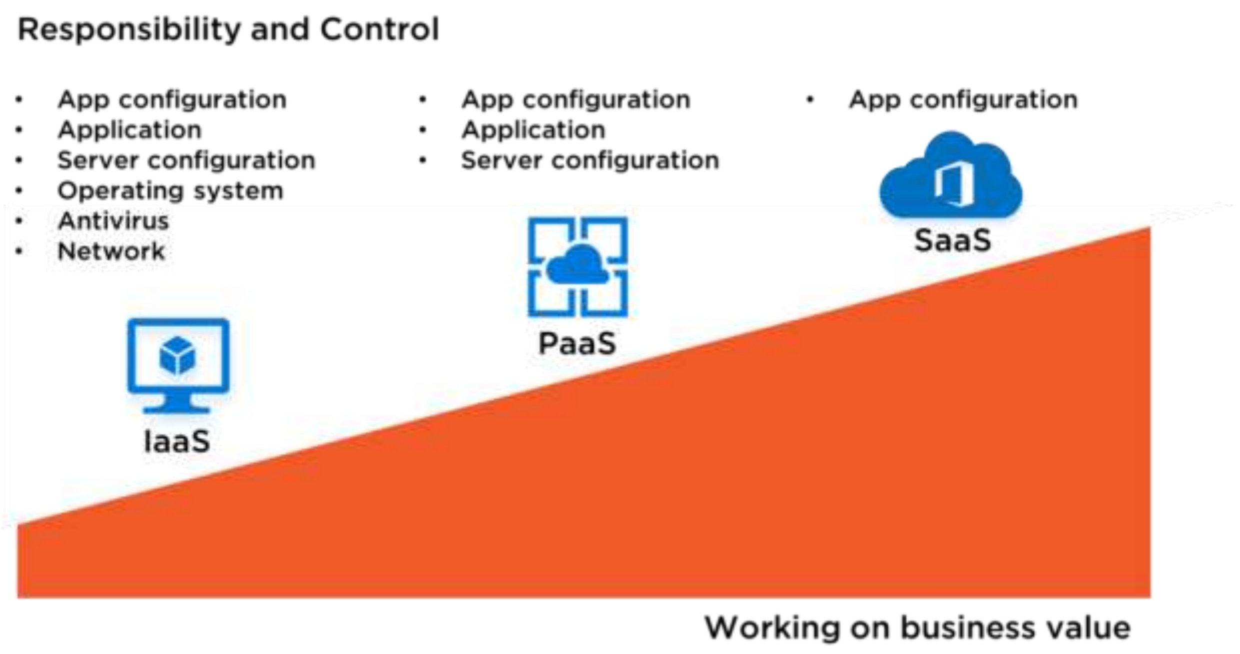
The Azure PaaS Services That Devs Love (and Why)

Let's break down all of Azure's PaaS offerings and see how they fit into the average developer's workflow so you can make informed decisions about your tooling The Microsoft Azure cloud offers a lot of services for almost every scenario that you might need. You can categorize these services into cloud computing types like IaaS, PaaS, and SaaS to determine the amount of control and responsibility you have versus the time you can spend on building things that matter.



Let me explain these types a bit further.

IaaS (Infrastructure-as-a-Service)

Typical IaaS services are Container Service and Virtual Machines. These allow you to have a lot of control over how you run them but also require you to be responsible for things like the OS, Antivirus and Load Balancing. Because of this, you and your team can spend less time working on adding business value.

PaaS (Platform-as-a-Service)

Examples of PaaS services are App Services, Azure Search and Azure CDN. You don't have to worry about the OS or even the server, you can just run your application. You are responsible for some server configuration, like scaling, although for some services, like Azure Functions, that happens automatically.

SaaS (Software-as-a-Service)

Examples of SaaS services are things like Azure IoT Suite and Office 365. SaaS is the highest abstraction level and allows you to just use the application, you don't even have to build it. You just configure it. No need to worry about the OS or even scaling the app. This allows you to work on business value, but offers you little control over your application.

For software developers, PaaS is a great cloud computing type to use. You have enough control to just work on your application and tweak it to be just right and you don't have to worry about all of the operations stuff. Things just run without worrying about OS patches or load balancing.

Azure has a very strong PaaS offering, which is very attractive for developers. This article will discuss the top Azure PaaS services for developers that can make your application better.

Top Azure PaaS Services

One of the core Azure PaaS services is Azure App Services. Azure App Services provide multiple service types, each geared towards hosting your application or business logic for a specific use case. Additionally, the App Service types share common capabilities like auto-scaling, authentication and authorization and custom domains and SSL.

Let's go through the App Service types:

Web Apps

Azure App Service Web Apps are essential if you want to host a standard ASP.NET web application. Web Apps are an abstraction of a Web Server like IIS or Tomcat and can run applications that are written in .NET, PHP, Python, Node.js, Java and more. They are very easy to setup and provide you with lots of benefits out-of-the-box, like the fact that by default, they are available 99.95% of the time. No need to worry about downtime. App Services are now available for Windows and Linux both.

Mobile Apps

Azure can also help you when you are creating mobile applications. You can host a backend for your mobile app in Azure App Services Mobile Apps. You can easily connect to this backend using the SDKs for Azure Mobile Apps that are available for IOS, Android, Windows, Xamarin.IOS, Xamarin.Android and Xamarin.Forms.

The mobile backend provides you with some unique benefits. One of them is the ability to do offline sync. This enables a user to continue working with the app if he is offline and sync data back to the backend when he comes online again. Another capability is push notifications. This allows you to send notifications about your app to the user's device. Additionally, Mobile Apps has all of the same capabilities that Web Apps has, like auto-scaling, and high availability.

Logic Apps

Azure App Service Logic Apps are different from Web Apps and Mobile Apps in that you don't host an application in it, but orchestrate business logic with it. Think of Logic Apps as a way to automate a business process by just configuring it.

A Logic App is started by a Trigger. This can be a time (every 15 minutes) or an outside source, like a new message on a queue. The Trigger passes values into the workflow (like the contents of the queue message), that can be used throughout the Logic App. The rest of the flow of the Logic App consists out of calling Connectors, which are APIs to third party services, like Office365 or Twitter or your own APIs.

Logic Apps scale automatically and you only pay for them when they run. This is sometimes called "serverless" because it means that you can completely focus on your application or logic and not on the underlying infrastructure.

Azure Functions

Azure App Services Function Apps can host one or more Azure Functions. You use Azure Functions to host small applications, like background jobs or a microservice that only runs for a short period of time.

Azure Functions can be triggered by configurable timers, like on a schedule (every 15 minutes) or by an external service, like when a new Blob is added to Azure Blob Storage. When triggered, the code in the Azure Function can use the value from the trigger, like the Blob that was added. You can also add output bindings to an Azure Function to output a value to an external service, without writing any plumbing. This could, for instance, be a Blob Storage output where you just return a Blob without having to write code to connect to Azure Storage.

Just like Logic Apps, Azure Functions are "serverless", because they scale automatically and you only pay for them when they run.

Azure WebJobs

Another way to run background tasks is to run them in Azure WebJobs. WebJobs are part of App Services and run inside an App Service like a Web App or a Mobile App. You can write and host code in WebJobs that gets started by a trigger, like a timer (every 15 minutes) or an external service, like a new message in a queue.

WebJobs work similarly to Azure Functions in that they run small pieces of code that can be triggered by outside sources that don't require any plumbing code to set up.

They are different from Azure Functions in that you need to scale them manually. You pay for the App Service that hosts your Web Job, which is a monthly fee, regardless if the WebJob runs or not.

What to Use When

So which App Service should you use when? This table might help:

	Web Apps	Mobile Apps	Logic Apps	Functions	WebJobs
Host web applications and APIs	X				
Host backend for mobile apps		X			
Host business logic workflows			X		
Host background tasks				X	X
Scales automatically and only pay when it runs			X	X	
Control scaling and pay a monthly fee	X	X		X	X

App Services Shared Features

Azure App Services share some awesome features, that make them very compelling to use. Here are some of them:

- Easy authentication and authorization
- Continuous delivery
- Custom domains
- Hybrid connections
  - Enables you to connect to on-premises resources, like web services or databases
- (automatic)scaling and load balancing

- Deployment slots
    - These allow you to test the new version of your application and deploy it to production with no downtime. They even allow you to route some of your production traffic to the new version of your app, to see if it works as expected.
    - Learn more about deployment slots here

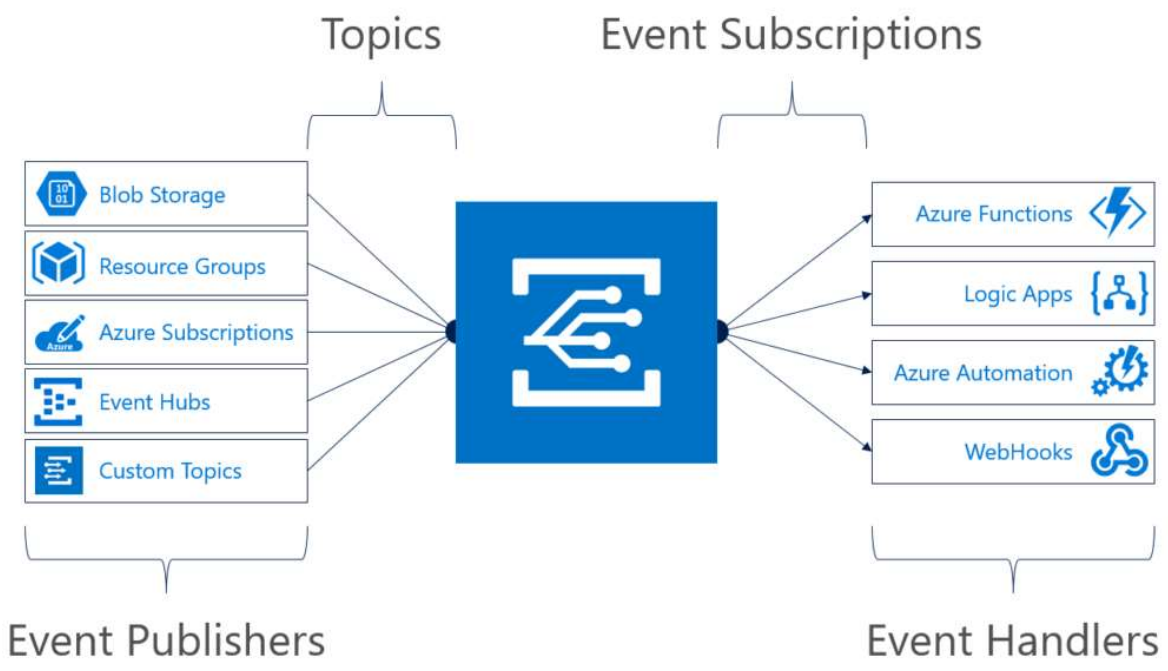
Deploying Services

Creating a new Web App or Logic App is simple through the Azure Portal, but you should really automate your deployments so that you can repeat them in different environments and run the exact same configuration. You can do that by creating Azure Resource Manager (ARM) templates that you deploy from Visual Studio or in a continuous delivery pipeline using a service like Visual Studio Team Services.

ARM templates describe your resources like Web Apps, Azure SQL Databases and resource groups in JSON documents. You can create them using the **Azure Resource Group project template** in Visual Studio and get inspiration by clicking on the **Automation Script** button on any resource in the Azure Portal.

Azure Event Grid

A relatively new service, Azure Event Grid, acts as the glue between services. Azure Event Grid can route events from any source to any destination.



Almost every Azure service can publish events that Azure Event Grid can receive and use to trigger your application with. You can also have your own applications and services publish events that Azure Event Grid can use to route to other services.

Azure Event Grid scales automatically and you only pay for the number of operations that you use.

Azure Event Grid eliminates the need for polling. Your applications can listen for and react to events from whatever service published events to Azure Event Grid.

You can, for instance, have Azure Event Grid listen to changes in your mailing list and trigger an Azure Logic App when something changes.

Azure Service Bus

One of the oldest and most used and reliable services in Azure is the Azure Service Bus. It consists out of multiple services, most notably Azure Service Bus Queues and Azure Service Bus Topics.

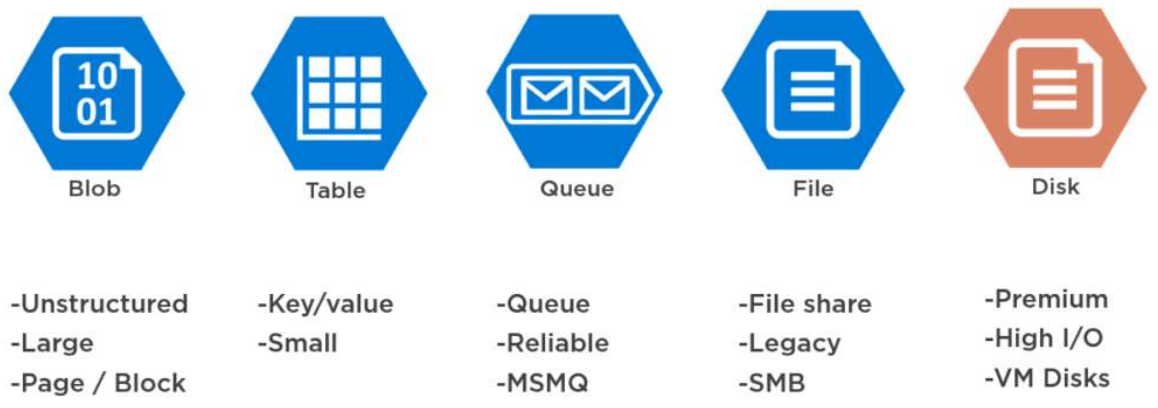
An application can put messages on an Azure Service Bus Queue. Each message is read from the queue and processed by one application. Azure Service Bus Queues have some advanced capabilities that you get out-of-the-box like duplicate detection and a dead-letter sub-queue where messages that can't be processed are moved to.

Azure Service Bus Topics are a bit different. Applications can put messages on an Azure Service Bus Topic and multiple applications can read and process the message. They receive the messages that have attributes that they are interested in. They subscribe to 'topics' on the queue. Azure Service Bus Topics have the same advanced capabilities as Queues do.

Both of these services allow you to decouple the application that puts messages on the queue from the application that processes the messages. This pattern enables the processing application to process tasks at its own pace without making the user wait on it. This is useful when processing can take a while and you don't want the user to have to wait on that, for instance when you are processing an order request that can take 5 minutes.

Azure Storage

A basic building block in any application is data storage. Azure Storage provides multiple options for storing data, like files or rows of data. It offers several different services that are all relatively inexpensive, scale almost infinitely and across geographic regions and offer features like encryption and authentication. You get all of this out-of-the-box, just by spinning up an Azure Storage Account. Let's discuss the various types of Azure Storage:



Blob Storage

You store large files, or 'blobs' in Azure Blob Storage. These can be things like VHD files or audio or video files or database backups.

You can choose a tier of Blob storage that determines your performance and costs. There is the **Hot** tier, for Blobs that you access frequently. There's also the **Cool** tier, that you use for Blobs that you don't access that often, maybe once a month. And then there is the **Archive** tier**,** for Blobs that you use almost never. When you request a Blob from the **Archive** tier, it could take hours before you can access it.

Table Storage

You can store semi-relational data in rows and columns using Azure Table Storage. This is perfect for things like logging data. You store data in tables that are similar to those in SQL Server, but are less strict in that they don't have relationships between them. Azure Table Storage is fast and relatively inexpensive, just like the other types of Azure Storage.

Queue Storage

You can store small messages that can be picked up and processed later in Azure Storage Queues, just like the Azure Service Bus Queues. This is a bit of a strange offering because it overlaps with the Azure Service Bus Queues. They are different in that Service Bus Queues have duplicate detection and Storage Queues don't. Also, Service Bus Queue messages remain available after 7 days, where Storage Queue messages can only be stored for 7 days.

File Storage

You can use Azure File Storage as an extra hard disk that you mount to your computer or a VM. This is a great service to get started with the cloud by migrating your files to it. You can have applications use it by just pointing them to Azure File Storage instead of their own hard disk. This allows you to start moving to the cloud.

Disk Storage

Azure Disk Storage is a premium feature and is highly performant and useful for when you need to perform I/O intensive work, like on a primary hard disk of a VM. It acts as a scalable, high-performance disk that you can mount to a VM.

Azure Cosmos DB

The new version and name of Azure DocumentDB is Azure Cosmos DB. Azure Cosmos DB is a database offering that fits the cloud perfectly. As a PaaS service, you just spin a Cosmos DB up and you're ready to go. If you were using Azure DocumentDB, you are now automatically using Azure Cosmos DB. Your code doesn't have to be changed.

There are a couple of things that make Cosmos DB extraordinary:

- You can program against it using different 'APIs', like SQL, JavaScript, MongoDB, Gremlin and Table Storage. You don't pick a type of database anymore, you just pick the way you want to talk to it, Cosmos DB takes care of the rest
- No need to create indexes anymore, Cosmos DB does this automatically for you
- Cosmos DB is highly performant. It even guarantees low latency in its SLA
- You get geographic scalability out-of-the-box. You just indicate where in the world you want your data to be and it gets replicated in real-time. This way, you can make sure that your data is close to your users

Learn how to get started with Azure Cosmos DB in these 5-minute tutorials.

Cosmos DB is most suited for non-relational or semi-relation data.

Azure Cognitive Services

Adding intelligence to your applications has never been as easy as with the Azure Cognitive Services. These are a set of APIs that provide almost magical abilities, powered by AI and Machine Learning. There are about 30 Cognitive Services and more are coming. Here are some examples:

- Emotion API, which analyzes faces in photos and videos to detect emotions like happiness, sadness, disgust and so on
- Language Understanding Intelligent Service (or LUIS), which can actually understand language context in more than 12 languages. You can use this to create a smart bot that actually understands what you are saying to it
- Speaker Recognition API, which identifies speakers based on speech. You can use this to authenticate using voice or to identify people based on their voice
- Computer Vision API, which can detect information about visual content found in images, like that the background is water and sky or the jacket that the person is wearing is black

The Cognitive Services are very easy to use as you just use them from the cloud and you don't need to manage any infrastructure or configuration.

You can use your own data to train the services to enhance your results. Doing this, the Face API could, for instance, return the names and ages of your coworkers. Additionally, the Cognitive Services learn on the fly, by the data you feed them. This makes them smarter and more accurate.

To use a Cognitive Service, you simply call the API endpoint, like https://westcentralus.api.cognitive.microsoft.com/face/v1.0/detect with a subscription key that you get when you sign up and feed it the parameters it needs, like a Byte Array of an image, in the case of the Face API. This returns a JSON response like this (some lines are omitted to save space):

Face detection result:

```
[
  {
    "faceId": "a030596b-28dd-443f-80d2-99a978a93003",
    "faceRectangle": {
      "top": 124,
      "left": 459,
      "width": 227,
      "height": 227
    },
    "faceAttributes": {
      "hair": {
        "bald": 0.01,
        "invisible": false,
        "hairColor": [
          {
            "color": "brown",
            "confidence": 1.0
          },
          {
            "color": "blond",
            "confidence": 0.69
          },
          …
        ]
      },
      "smile": 0.826,
      "headPose": {
        "pitch": 0.0,
        "roll": -16.9,
        "yaw": 21.3
      },
      "gender": "female",
      "age": 23.8,
      "facialHair": {
        "moustache": 0.0,
        "beard": 0.0,
        "sideburns": 0.0
      },
      "glasses": "ReadingGlasses",
      "makeup": {
        "eyeMakeup": true,
        "lipMakeup": true
      },
      "emotion": {
        "anger": 0.103,
        "contempt": 0.003,
        "disgust": 0.038,
        "fear": 0.003,
        "happiness": 0.826,
        "neutral": 0.006,
        "sadness": 0.001,
        "surprise": 0.02
      },
      …
  }
]
```

You can see in the output that this particular service returns a lot of data. It even detects where facial features are located in the picture and if the person is wearing makeup. I think that's really amazing.

Azure CDN

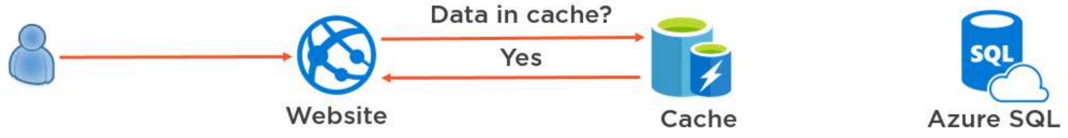
You can speed up your applications by offloading traffic to Azure CDN. Azure CDN (Content Delivery Network) acts as an endpoint for your static content like video, image, JavaScript and CSS files. By having Azure CDN serve these files, your application doesn't have to and has more resources to handle more requests. Additionally, Azure CDN replicates the static content to Points-of-Presence (PoPs) all over the world, thereby placing static content close to where your users are and limiting latency, which is good for performance.

You can use Azure CDN by creating an Azure CDN Endpoint, coupling that to a source, like Azure Storage and upload files to the Azure Storage. Azure CDN automatically replicates the files to the PoPs all over the world and you can link to the files through an URL like https://custom.azureedge.net/cd/myimag.png. You can also use a custom domain name for the URL.

Azure Redis Cache

Another way to speed up your applications is by caching data using Azure Redis Cache. When you get data from a cache instead of from another data store, like a database, you speed up your application.

That is because a cache like Azure Redis Cache holds data in-memory and stores data in a simple key/value format. Because of this, data can be served quicker as it doesn't have to be retrieved from disk and it doesn't have to be retrieved by executing a complex query.

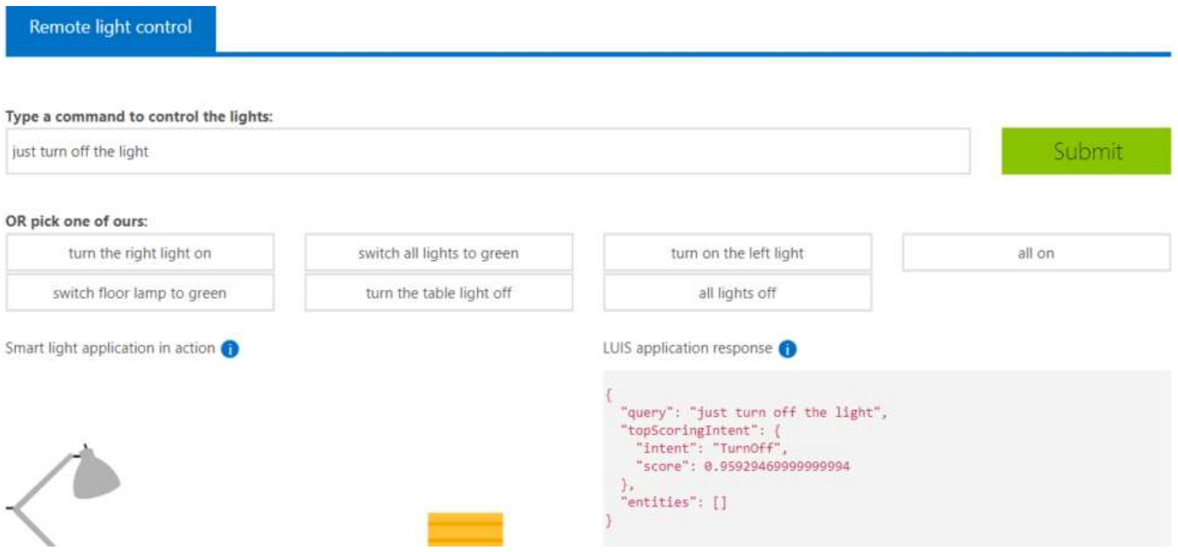


Azure Redis Cache provides a cache-as-a-service and provides advanced capabilities, like clustering and geo-replication. It is based on the popular open-source Redis Cache and is now backed by Microsoft SLAs and enterprise support. Azure Redis Cache is *the* caching option for applications in Azure.

Azure API Management

You can use Azure API Management to enhance your APIs or third-party APIs that you use. Azure API Management acts as a gateway between your API and the outside world. This allows you to enhance your API.

With API Management, you can create a portal where users of your API can go to manage their subscription(s) and to test the application. An example of such a developer portal is the website of one of the Cognitive Services, that also uses API Management:



```
{
  "query": "just turn off the light",
  "topScoringIntent": {
    "intent": "TurnOff",
    "score": 0.959294699999999994
  },
  "entities": []
}
```

You can control the usage of the API by setting limits per user or subscription type. This is a good way to monetize your API by for instance offering a free usage tier up to 10 requests per day, and if you need more, you start paying. If users try to use more, without the right subscription, their requests get throttled.

Additionally, you can enhance APIs by making them more performant. You can cache API responses so that they don't have to get the responses by doing calculations or composing data from a data source.

And you can protect your APIs with keys, passwords, certificates and IP filtering. On top of that, you can transform the input and output to and from your APIs. You could, for instance, transform incoming XML to JSON or do the same with responses.

Azure API Management is really powerful and increases the control that you have over your APIs (and third-party APIs that you expose through API Management). And it allows you to do that without changing the API.

Conclusion

The services in this article are all examples of Azure PaaS services. All of them are very easy to spin up and require minimal maintenance. They will stay up and running, are self-healing and basically 'just work'.

The value that these services can provide is enormous and easy to incorporate into your applications. This truly takes care of a lot of the plumbing and repetitive tasks that we used to build and do ourselves.

Azure PaaS lets us focus on building things that matter!