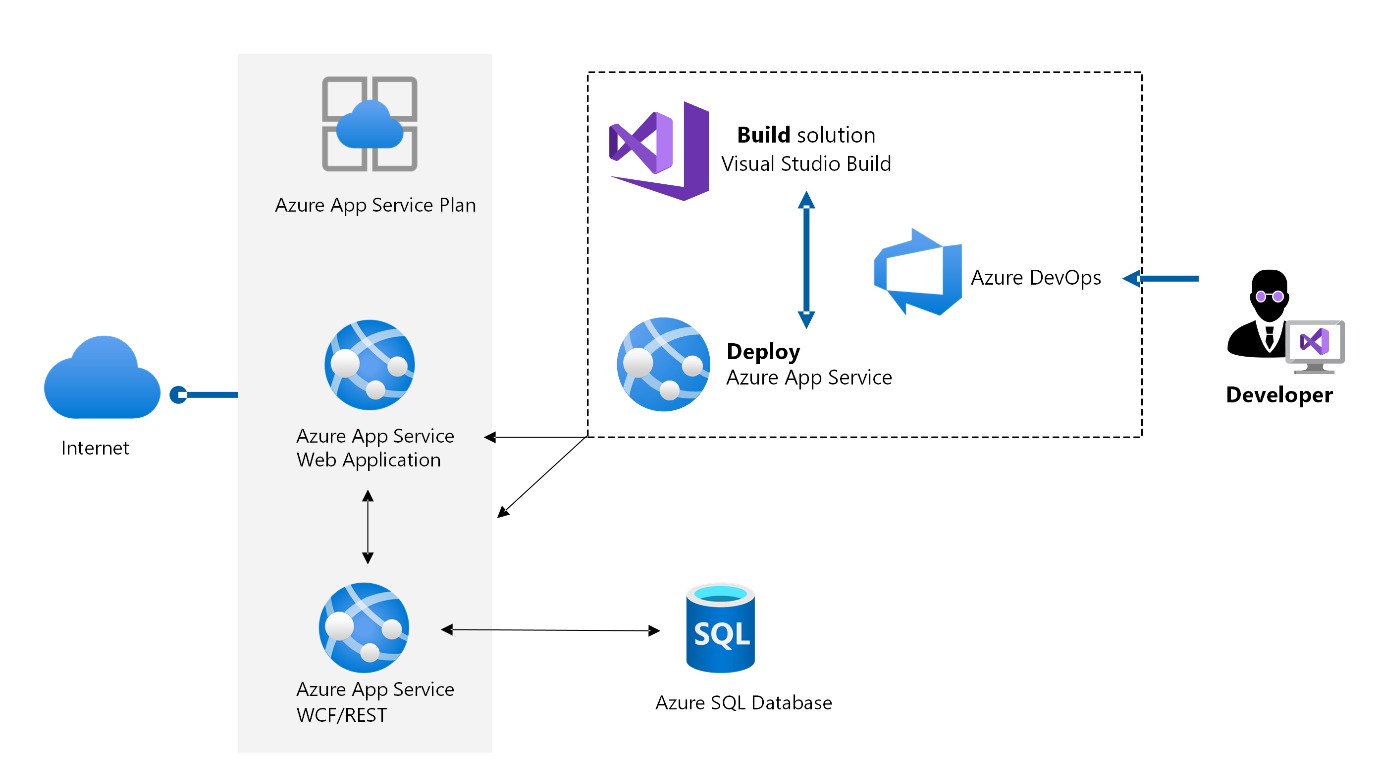
**Azure App Service**

Microsoft’s Platform as a Service (PaaS) offering is called App Service. We use it to host backend services for mobile applications, REST APIs, and web applications.

Which programming language or framework you have used doesn’t really matter. The azure app service can host web applications and services created with any of the following programming languages or frameworks. It might be:

* .NET
* .NET Core
* Java
* Ruby
* Node.js
* PHP
* Python



**Azure Kubernetes Service (AKS)**

The alternative option is to deploy an application in Azure using the increasingly well-liked Kubernetes platform.

With its Azure Kubernetes Service, Azure offers a fully managed Kubernetes solution. If you are familiar with Kubernetes, it works fairly well and you can get up and running quickly.

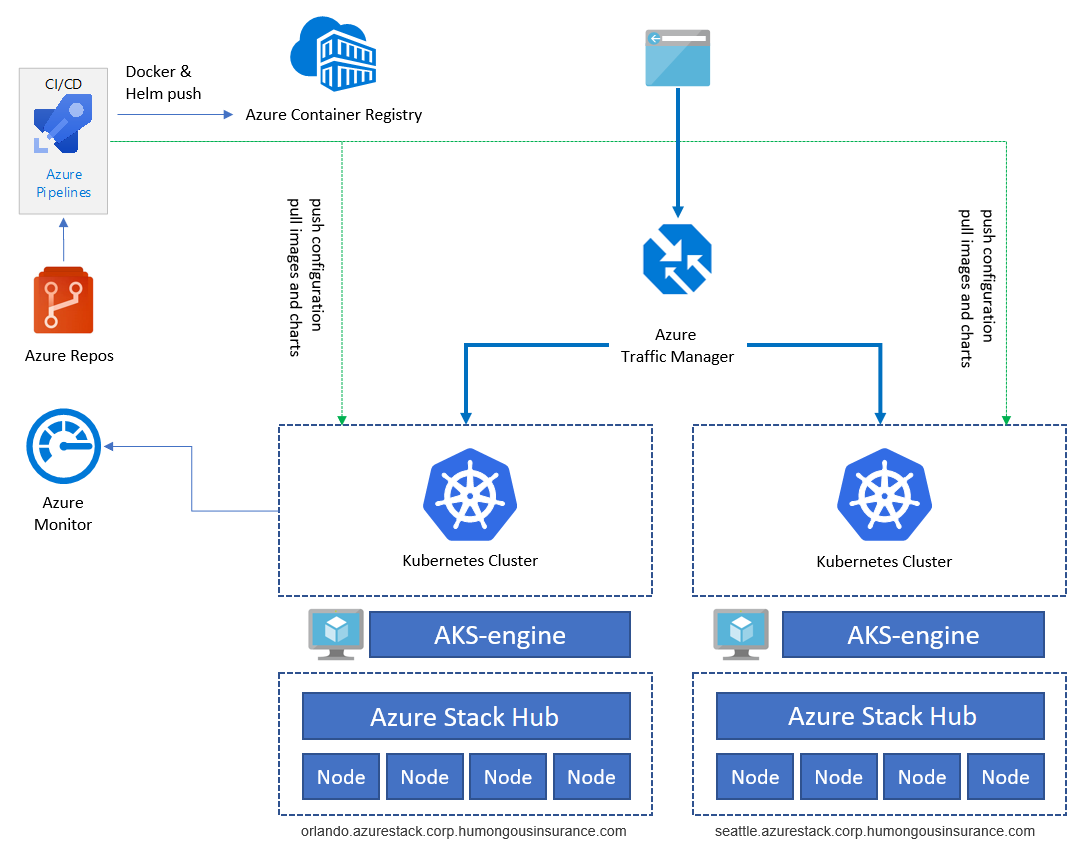
Currently, Kubernetes is an open-source industry standard for automatically deploying, scaling, and managing containerized applications.  
It is widely used and has all the features needed to create a website that is both highly scalable and resilient.

Let’s now compare the two options quickly while keeping important requirements in mind.

**High Availability**

A crucial requirement for resilient applications is high availability.

High availability is natively supported in a single region in AppService and AKS. At least three availability zones are present in a single region. A customised solution utilising Azure Traffic Manager can also be used to achieve multi-region availability if that is required.



**DNS**

**AppService:** Each AppService under \*.azurewebsites.net is given a special domain name based on the app service name.

**AKS:** A service can be given a fully qualified public domain hosted under \*.location>.cloudapp.azure.com if it is exposed via a public load balancer.  
Using DNS configuration is another option if a non-Azure DNS is needed.

**Scalability**

**AKS**

**Automatically:** By using the horizontal pod autoscaler to increase the number of pod replicas or the cluster autoscaler to increase the number of cluster nodes. By examining CPU and memory usage by pods using the vertical pod autoscaler, Kubernetes can also adjust resources up or down per pod. Unfortunately, AKS does not yet support this feature.

**Manually:** Increase node CPU/Memory usage to scale up.

**AppService**

**Automatically:** VM instances running inside of AppService Plan can be scaled. (Scale-Out)

**Manually:** Upgrade to a higher tier of the app service plan.

**Resource Control**

**AKS:** Capable of managing the CPU and memory for every pod’s container

**AppService:** There is no way to restrict how much memory and CPU are used by the app service.

**Environments**

**AppService:** Having multiple AppServicePlans is the only way to create multiple environments in AppService. Sharing computing resources between environments is not possible because each AppServicePlan runs on its own hardware.

**AKS:** We can create distinct namespaces inside the same Kubernetes cluster or use separate clusters to create distinct environments. Additionally, it is simple to create new environments because Kubernetes makes it simple to create a namespace. Workload isolation is also possible by using different node pools.

**Cloud Vendors**

**AppService:** This service is only offered by Azure. Even though Azure AppService makes use of Docker containers, Azure training is necessary to use it.

**AKS:** The majority of cloud providers offer Kubernetes. Workloads running on Azure Kubernetes Service can, if necessary, be transferred to other Kubernetes providers such as Google Kubernetes Engine (GKE) or Amazon EKS.

**Running Background Jobs: AKS vs App Services**

**AKS**

You can create a Job in AKS that can be triggered on-demand or one that runs on a regular schedule.

There are a tonne of different ways you can go about managing a [background job](https://docs.microsoft.com/en-us/azure/architecture/best-practices/background-jobs). You could, for instance, create a job that runs a number of Pods concurrently until a predetermined number of completions. Alternatively, you could create a Job that runs only until a Pod successfully completes it. There are numerous configuration options as well to handle failure. For example, we can specify a restart policy that determines whether to restart the Job if it fails, up to a predetermined number of tries, etc.

One really great feature of Kubernetes is that you can set up your jobs to run on different node pools, which is especially useful if your job calls for a lot of CPU or memory. Additionally, you can place CPU and memory restrictions on a job to prevent it from evicting other pods from the cluster.

**App Service**

You can create two different types of WebJobs in AppService:

* Continuous WebJob
* Triggered WebJob

In every instance of an AppService Plan, a Continuous WebJob runs indefinitely. By limiting the [WebJob](https://docs.microsoft.com/en-us/azure/app-service/webjobs-create) to run on a single instance, this can be changed.

A Triggered WebJob, on the other hand, only runs when it is triggered or in response to a schedule or event. A schedule-driven trigger or an event-driven trigger are the two different types of triggers for WebJobs. In contrast to Kubernetes Jobs, WebJobs cannot be restricted or separated from other workloads.

**Access Tools**

Access to tools is a crucial factor to take into account when deciding between Azure AppService and Azure Kubernetes Service.  
Kubernetes is extremely powerful because of its thriving community and abundance of excellent tools. ArgoCD for CI/CD with GitOps and Helm makes managing and deploying Kubernetes applications much simpler. Additionally, keep in mind that the majority of important frameworks, including Elastic Search, Grafana, Nginx, MongoDB, etc., have helm charts that make it incredibly simple to deploy applications to them.

**Cost Comparison: AKS vs App Services**

AKS outperforms Azure AppService in terms of pricing, with an average cost savings of 30%.

A free container service called Azure Kubernetes Service (AKS) makes it easier to deploy, manage, and run Kubernetes as a fully managed Kubernetes container orchestrator service.

AKS is the most effective and economical container service available because users only pay for the virtual machines and related storage and networking resources that they actually use.