**What is a App Service:**

**Azure App Service** is a fully managed **Platform-as-a-Service (PaaS)** offering from Microsoft Azure designed for building, deploying, and scaling web and mobile applications. It supports multiple programming languages (like .NET, Java, Node.js, Python, and PHP) and integrates seamlessly with DevOps tools for CI/CD pipelines.

### ****Key Features:****

**1. Multi-language & Framework Support:**

* + Host web apps (ASP.NET, Node.js, Python, etc.), REST APIs, and mobile backends.
  + Supports Windows & Linux environments.

**2**. **Auto-scaling & High Availability:**

* + Scale vertically (instance size) or horizontally (instance count) based on demand.
  + Built-in load balancing for high availability.

**3**. **DevOps Integration:**

* + Continuous Deployment (GitHub, Azure DevOps, Bitbucket).
  + Staging slots for A/B testing and blue-green deployments..

**App Service Environment:**

An App Service Environment is an Azure App Service feature that provides a fully isolated and dedicated environment for running App Service apps securely at high scale. Unlike the App Service public multitenant offering where supporting infrastructure is shared, with App Service Environment, compute is dedicated to a single customer.

**An App Service Environment can host your:**

* Windows web apps
* Linux web apps
* Docker containers (Windows and Linux)
* Functions
* Logic apps (Standard) - in [supported regions](https://azure.microsoft.com/explore/global-infrastructure/products-by-region/?products=logic-apps&regions=all)

**App Service Environments are appropriate for application workloads that require:**

* High scale.
* Isolation and secure network access.
* High memory utilization.
* High requests per second (RPS). You can create multiple App Service Environments in a single Azure region or across multiple Azure regions. This flexibility makes an App Service Environment ideal for horizontally scaling stateless applications with a high RPS requirement.

## Automatic authentication provided by App Service:

App Service provides built-in authentication and authorization support, so you can sign in users with no code in your web app. Using the optional App Service authentication/authorization module simplifies authentication and authorization for your app.

**App service authentication provides:**

* Easily turn on and configure through the Azure portal and app settings.
* No SDKs, specific languages, or changes to application code are required.​
* Several identity providers are supported:
  + Microsoft Entra
  + Microsoft Account
  + Facebook
  + Google

**Authentication solutions:**

* **Azure App Service built-in authentication** - Allows you to sign users in and access data by writing minimal or no code in your web app, RESTful API, or mobile back end. It’s built directly into the platform and doesn’t require any particular language, library, security expertise, or even any code to use.
* **Microsoft Authentication Library (MSAL)** - Enables developers to acquire security tokens from the Microsoft identity platform to authenticate users and access secured web APIs. Available for multiple supported platforms and frameworks, these are general purpose libraries that can be used in various hosted environments. Developers can also integrate with multiple sign-in providers, like Microsoft Entra, Facebook, Google.
* **Microsoft.Identity.Web** - A higher-level library wrapping MSAL.NET, it provides a set of ASP.NET Core abstractions that simplify adding authentication support to web apps and web APIs integrating with the Microsoft identity platform. It provides a single-surface API convenience layer that ties together ASP.NET Core, its authentication middleware, and MSAL.NET.

**Types of Azure App Services:**

1. **Web Apps**
2. **API Apps**
3. **Logic Apps**
4. **Mobile Apps**
5. **Function Apps**
6. **Web Apps**

* Web Apps enable us to host our web applications without risk about the infrastructure plumbing that is required.
* In a hosting mechanism, we need to make sure the Server is up, the OS is updated and IIS is running
* Hosting in the Azure web app removes all this burden and the Service fabric layer below it makes sure that the app is up and running.
* Deploying applications in web app service in Azure helps developers focus on delivering business values rather than consuming time on severe updates or OS patches.

1. **API Apps:**

API apps are offering of App Service that helps to host Web APIs. This enables us to expose existing or new APIs. This is also a part of the platform as a service and we don’t need to worry about infrastructure plumbing to bring our APIs up and running. It also supports identity providers to secure the APIs. The API Apps support, Java, Python, and Node.js to build and deploy Web APIs.

1. **Logic Apps**

Logic apps enable us to create functional workflows by orchestrating software as a service component. These are basically used to connect different components of a solution to manage and trigger events and perform the desired action on some other service. For example, we can build a logic app that triggers an event of a new file uploaded on blob storage and performs an action of sending a notification to a user.

1. Logic apps facilitate workflows by using triggers, connectors, and actions.
2. Logic apps can be triggered manually, or at any scheduled time. Moreover, logic apps can also be triggered on the basis of some event on any connected component.
3. **Mobile Apps**

Mobile Apps enable us to build a backend for Mobile applications. It can provide capabilities to mobile client applications. This can be considered to be the same as a web service to support mobile client scenarios. The client can be Windows Universal apps, IOS apps, windows apps etc. They use Mobile app SDK to connect with the backend. There are certain unique capabilities with mobile apps:

1. They are cross-platform, which means Apps built for any platform Android, Windows, or iOS can consume them.
2. Mobile Apps also support secured client connections, which allow client applications to connect with the default identity providers, such as Active Directory and Microsoft accounts.
3. **Functions Apps:**

[Azure Functions](https://www.scholarhat.com/tutorial/azure/exploring-azure-functions) are event-driven components that eliminate the need for a server to host a piece of logical code and process. They can intercept events occurring in any Azure service, third-party service, or on-prem system. They are an evolution of Azure web jobs, which are a feature of Azure App Services.

For example, Azure Functions can be triggered on Event Hubs, Service Bus topics or queues, or via a timer.

1. An Azure function can run any executable. Azure Functions are also referred to as Serverless. It's not that Azure functions do not run on servers they do. They run on Azure service fabric. But we do not need to manage the server.

**App Service Plan:**

All the apps that run under app service are governed and observed by a contract with the cloud service provider known as the App Service plan. This acts as a container for the applications and defines the boundary limitations of resources available to consume and scale. An app service plan comes with measured compute resources that keep our app running. These compute resources include fixed computing power which can be consumed by different applications deployed in the same app service plan.

**The app service plan is categorized by its pricing tiers as below:**

1. **Free**

This App Service plan uses a single VM for multiple app service plans and can host multiple applications with some limited computing power. Also, we cannot scale our apps in this app service plan and applications deployed in this plan cannot be provided with custom domain names.

1. **Shared**

The shared app service tier runs in a similar environment as that of a Free tier. This tier allocates CPU quotas to each app that runs on the shared resources, and the resources cannot scale out. We can add a custom domain to the apps in this tier.

**App Service Scaling:**

Web apps offer 2 types of scaling based on our needs – Vertical Scaling (Scale up, scale down) and Horizontal Scaling(Scale Out and Scale In). Scaling is important for couple of reasons:

1. As users accessing our app grow, we want them to have a seamless experience with the app.
2. We only want to pay for the amount of computing power we use.

**Vertical Scaling:**

1. **Scale up**

Increasing the computing power of infrastructure to support heavy workloads by increasing the CPU power and storage efficiency.

1. **Scale Down**

Decreasing the computing power of infrastructure in case the website hit goes down by decreasing the CPU power and storage efficiency.

**Horizontal Scaling:**

1. **Scale Out**

This is also called horizontal scaling. To distribute the traffic load, the number of app instances is increased.

1. **Scale In**

The number of instances of the app is decreased to reduce cost in the off season when traffic on the web app goes low.