**Azure App Service Overview**

Azure App Service is a rich and powerful platform designed to make hosting web applications as simple as possible, and you can use it to host regular websites or web APIs, as well as to implement backends for your mobile apps.

One of the most attractive things about Azure App Service is that it comes with many features that are ideally suited for hosting web applications. For example, you can configure custom domains and SSL certificates for your sites.

You can scale up to a more powerful web server or scale out to multiple instances of your web server. You can configure staging slots to try out a new version of your application, and when you're ready, swap the staging version into production.

It's really easy to manage environment variables and connection strings for your web applications. There are rich monitoring options in the portal, allowing you to easily see if lots of users are getting 400 or 500 errors. And there are lots of options to integrate with a CI/CD pipeline, so it's easy to automate deployment of new versions of your web app.

And there are many security features such as configuring IP address whitelisting or requiring Azure AD authentication on all incoming requests. And there's lots more as well like easily enabling a CDN for your web application. So App Service is a Platform as a Service offering design so you can simply provide your web app and then configure as many of those features as you need for your scenario.

With App Service, you start off by creating an App Service plan. And the App Service plan is what you pay for, and you can host multiple web apps on the same App Service plan. And there are several pricing tiers such as Basic, Standard, and Premium, each of which brings additional functionality and performance.

Now, App Service doesn't require you to containerize your web apps. Out of the box, it already supports many of the most popular web development frameworks. So, as you'd expect, being from Microsoft, there's great support for ASP.NET applications, but you can also run Node.js, Java, PHP, and Python web apps.

However, a while ago they added the ability to provide your web application as a container with Linux as the underlying hosting operating system, and more recently, they've also opened up the possibility of running Windows containers on App Service, although that's still in preview. And Microsoft refers to the feature enabling you to host containers on App Service as Web App for Containers. Web App for Containers supports most, but not all of the features offered by regular web apps on App Service. So, for example, you can still configure custom domains and SSL certificates, you can scale up to multiple servers, and you can use the staging slots feature to test the next version of your site in staging before pushing it live.

**Why Web App for Containers?**

So if Azure App Service is already able to run web apps built with a whole host of common web frameworks, why would you want to use Web App for Containers? First of all, you might just really like the consistent deployment model that containerizing applications offers. Maybe you're already using Docker for all the other components in your application, and so you might want to keep things the same way for your web apps. And by containerizing them, you've got the flexibility to easily host them elsewhere if you want to.

Secondly, although App Services supports many common web frameworks, it doesn't support everything, so providing your application as a container means that you can use any web development framework that can be hosted in a container.

Third, even if you are using something like ASP.NET, because App Service is a Platform as a Service offering, you don't have direct control over exactly what version of the .NET Framework is installed on the host service. For example, Azure App Service has .NET 4.7.1 installed by default, but next month it's going to be upgraded to .NET 4.7.2, but I've got no control over that. In most cases, that's a good thing. The framework is getting patched automatically, and I don't need to do anything special to enable it. But sometimes we do want control over the exact versions of the frameworks that our applications are running on, and containerizing them gives us that control.

Another interesting capability that Web App for Containers offers is for your web app to consist of multiple containers, and you can use the Docker Compose or Kubernetes configuration file formats to define the containers that make up your application.

Another benefit of containerized web apps is that Azure App Service puts applications into a sandboxed environment to protect other web apps running on the same host, meaning that there are some restrictions on what you can do in a regular web app. For example, certain Windows APIs aren't allowed to be called. However, if you provide your web app as a container, then Docker is providing strong isolation, and so your web app is allowed to do anything that you can do inside a container.

Finally, although the regular App Service offers loads of great CI/CD options, using Web App for Containers opens up an interesting possibility that you can trigger an automated deployment whenever a new version of an image is pushed to your container registry. You can set it up to pull that new image and run it in a staging slot, allowing you to swap slots once you're happy that the updated version is working correctly. So there are plenty of great reasons for choosing Web App for Containers in preference to regular App Service web apps. And of course, the main reason you'd be hosting your containers on App Service in the first place is because you're creating a web application or a web API and you want to take advantage of the many great web hosting-related features that we looked at earlier.