Best Practices for WordPress On Azure App Service Whitepaper

Best Practices for WordPress on Azure App Service

Abstract: This document provides system administrators with guidance on how to get started with WordPress on Azure App Service, best practices of deployment topology while handling the user page response time within user acceptable limits. It also explains how to scale-up/scale-out /scaledown for operational efficiencies.

Introduction:

WordPress is an open-source blogging tool and content management system (CMS) based on PHP and MySQL that is used to power anything from personal blogs to high-traffic websites. It is the most popular Content Management System, powering 43.2% of all websites on the internet (Source: W3techs). It is a free and Open-Source software written in PHP and paired with a MySQL database. Originally created as a blogging platform, WordPress has evolved to become a platform that supports portfolio websites, forums, online stores, and marketplaces, learning management platforms, social networks, podcast websites, job boards, Wikis, and e-auction platforms, among others. WordPress has a strong OSS community, and it is evolving better day by day.

Taking advantage of Azure App Service capabilities, we have launched the offering as a managed, performant, secure and scalable hosting solution for WordPress.

Considerations:

The discussion starts with a single web server deployment. There may be occasions when you outgrow it, for example:

• The App Service that your WordPress website is deployed on is a single point of failure. A problem with this instance causes a loss of service for your website.

• Scaling resources to improve performance can be achieved either by "vertical scaling;" that is, by increasing the App Service SKU of your WordPress website or by "horizontal scaling" by increasing the number of App Service instances

Hosting plans supported by WordPress:

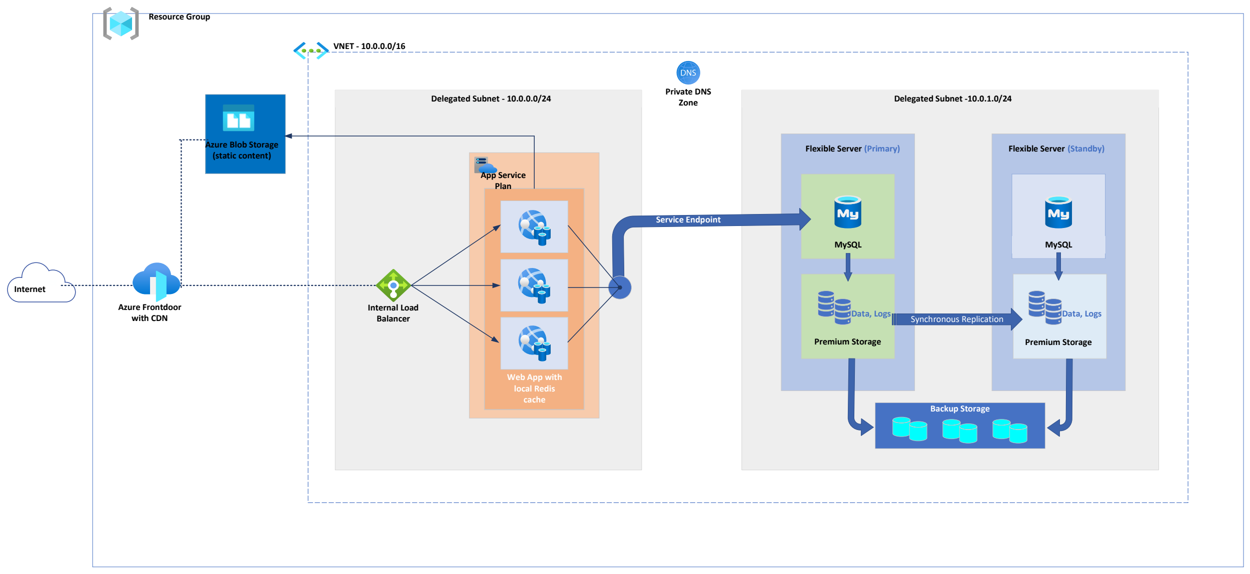
WordPress is supported on 3 different hosting plans:

Basic: This is a hosting plan that can be used primarily for initial exploration of the offering. WordPress runs on B1 SKU(1 Core. 1.75 GB RAM, 10 GB Storage) of App Service integrated with MySQL Flexible Server hosted on Burstable B1(B1s (1vCore, 1GB RAM, 20GB Storage, Auto IOPS)

Standard: This plan is suitable for test/ staging environment to define all the customizations before moving to production. SKU details P1v2 SKU(1 core, 3.5 GB RAM, 250 GB Storage) and DB is on B2s (2vCore, 4GB RAM, 128GB Storage, Auto IOPS)

Premium: This configuration is recommended for production environments that can manage your workloads seamless. App Service SKU Details-P1v3(2 cores, 8 GB RAM, 250 GB storage) and Database is with General Purpose (2 vCores, 8 GB RAM, 128 GB storage, Auto IOPS)

System Architecture:



Co-locate WordPress site and database

It seems obvious. When you create a WordPress site using our Marketplace templates, we automatically select the database in the same region as your Web App. If you choose to use a different database server, please double check the connection string in wp-config.php to ensure your database is in the same region as your Web App.

Network latency can increase the page load time for your website (esp. admin pages) if the site needs to go around the world to make a call to the database. Keeping the site and database components in the same region will reduce the network latency and improve the page load time for your website.

Local Storage Cache: A relatively common problem with WordPress is that the Admin dashboard has slow performance even when the frontend works fast for the users. Most of the performance enhancements (Azure CDN, Azure Blob Storage) are limited to speeding up your frontend and do not improve the performance of your Admin dashboard. So we have enabled App Service local storage cache which will server all WordPress plugin and theme files from the local drive speeding up admin page rendering faster. For more details please refer to : How to improve performance of WP Admin - WordPress on Azure App Service - Microsoft Community Hub

Improving performance and cost efficiency: The WordPress container has a local Redis Server installed. Redis server is configured to use at most 20% of the total available memory. It is recommended to utilize this caching to reduce the load on the App Service and database. By default, WordPress in Marketplace is configured to use Redis server cache using the **W3 Total Cache** plugin enabled with Page Cache, Object Cache & Database Cache. For more details refer to: wordpress-linux-appservice/WordPress/wordpress_local_redis_cache.md at main · Azure/wordpress-linux-appservice · GitHub

Database caching: It can significantly reduce latency and increase throughput for read heavy application workloads like WordPress. Application performance is improved by storing frequently accessed pieces of data in memory for low-latency access (for example, the results of input/output (I/O)-intensive database queries). When a large percentage of the queries is served from the cache, the number of queries that need to hit the database is reduced, resulting in a lower cost associated with running the database. Although WordPress has limited capabilities out-of-the-box, the out of the box W3 Total Cache plugin is a good example.

Accelerating content delivery: The Azure WordPress site is powered with fully managed CDN(Microsoft CDN) to handle the static file content and preloaded Redis caching to reduce load on your site's resources with our caching solution, which is capable of moving thousands of hits per day through our system. You can scale out the platform to manage the traffic spikes. There was thorough validation done on WordPress on Azure leveraging the caching & CDN abilities helped gain optimal response times. For more details refer to: wordpress-linux-appservice/WordPress/wordpress_azure_cdn.md at main · Azure/wordpress-linux-appservice · GitHub

URL Routing with caching: Azure Front Door is a modern content delivery network (CDN), with dynamic site acceleration and load balancing capabilities. When caching is configured on your route, the edge site that receives each request checks its cache for a valid response. Caching helps to reduce the amount of traffic sent to your origin server. If no cached response is available, the request is forwarded to the origin. For more details : wordpress-linux-appservice/WordPress/wordpress_afd_configuration.md at main · Azure/wordpress-linux-appservice · GitHub

Scaling the web tier: There are two workflows for scaling, scale up and scale out. For more details please refer to : Scale up features and capacities - Azure App Service | Microsoft Learn

- Scale up: Get more CPU, memory, disk space, and extra features like dedicated virtual machines (VMs), custom domains and certificates, staging slots, autoscaling, and more. You scale up by changing the pricing tier of the App Service plan that your app belongs to.
- Scale out: Increase the number of VM instances that run your app. You can scale out to as many as 30 instances, depending on your pricing tier. It can also be configured to auto-scale the App Services based on the workloads and more details about autoscaling can be seen here: How to enable automatic scaling - Azure App Service | Microsoft Learn