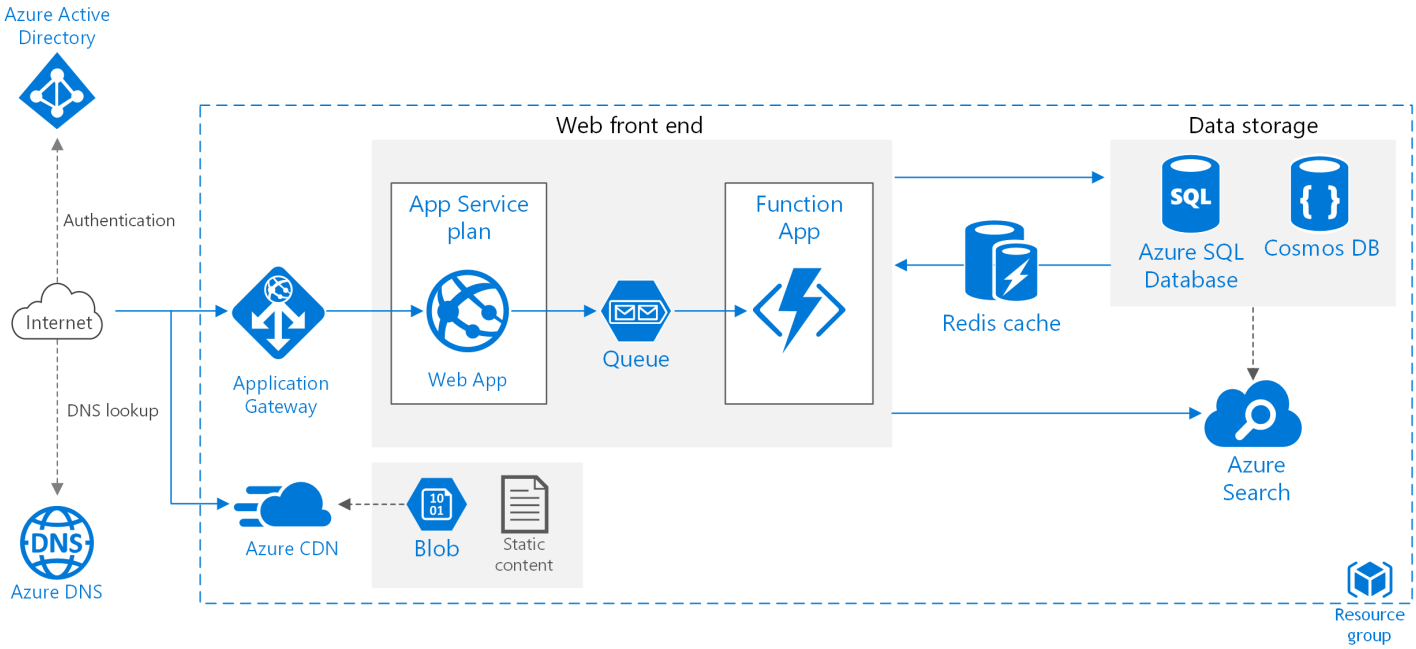
Improve scalability in an Azure web application

10/25/2018 • 6 minutes to read • Contributors all

In this article

This reference architecture shows proven practices for improving scalability and performance in an Azure App Service web application.

A reference implementation for this architecture is available on GitHub.



Download a Visio file of this architecture.

Architecture

This architecture builds on the one shown in Basic web application. It includes the following components:

- **Resource group**. A resource group is a logical container for Azure resources.
- **Web app**. A typical modern application might include both a website and one or more RESTful web APIs. A web API might be consumed by browser clients through AJAX, by native client applications, or by server-side applications. For considerations on designing web APIs, see API design guidance.
- **Function App**. Use Function Apps to run background tasks. Functions are invoked by a trigger, such as a timer event or a message being placed on queue. For long-running stateful tasks, use Durable Functions.
- **Queue**. In the architecture shown here, the application queues background tasks by putting a message onto an Azure Queue storage queue. The message triggers a function app. Alternatively, you can use Service Bus queues. For a comparison, see Azure Queues and Service Bus queues - compared and contrasted.
- **Cache**. Store semi-static data in Azure Redis Cache.

- **CDN**. Use Azure Content Delivery Network (CDN) to cache publicly available content for lower latency and faster delivery of content.
- **Data storage**. Use Azure SQL Database for relational data. For non-relational data, consider Cosmos DB.
- **Azure Search**. Use Azure Search to add search functionality such as search suggestions, fuzzy search, and language-specific search. Azure Search is typically used in conjunction with another data store, especially if the primary data store requires strict consistency. In this approach, store authoritative data in the other data store and the search index in Azure Search. Azure Search can also be used to consolidate a single search index from multiple data stores.
- **Azure DNS**. Azure DNS is a hosting service for DNS domains, providing name resolution using Microsoft Azure infrastructure. By hosting your domains in Azure, you can manage your DNS records using the same credentials, APIs, tools, and billing as your other Azure services.
- **Application gateway**. Application Gateway is a layer 7 load balancer. In this architecture, it routes HTTP requests to the web front end. Application Gateway also provides a web application firewall (WAF) that protects the application from common exploits and vulnerabilities.

Recommendations

Your requirements might differ from the architecture described here. Use the recommendations in this section as a starting point.

App Service apps

We recommend creating the web application and the web API as separate App Service apps. This design lets you run them in separate App Service plans so they can be scaled independently. If you don't need that level of scalability initially, you can deploy the apps into the same plan and move them into separate plans later if necessary.

> **Note**
>
> For the Basic, Standard, and Premium plans, you are billed for the VM instances in the plan, not per app. See App Service Pricing

Cache

You can improve performance and scalability by using Azure Redis Cache to cache some data. Consider using Redis Cache for:

- Semi-static transaction data.
- Session state.
- HTML output. This can be useful in applications that render complex HTML output.

For more detailed guidance on designing a caching strategy, see Caching guidance.

CDN

Use Azure CDN to cache static content. The main benefit of a CDN is to reduce latency for users, because content is cached at an edge server that is geographically close to the user. CDN can also reduce load on the application, because that traffic is not being handled by the application.

If your app consists mostly of static pages, consider using CDN to cache the entire app. Otherwise, put static content such as images, CSS, and HTML files, into Azure Storage and use CDN to cache those files.

> **Note**

Azure CDN cannot serve content that requires authentication.

For more detailed guidance, see Content Delivery Network (CDN) guidance.

Storage

Modern applications often process large amounts of data. In order to scale for the cloud, it's important to choose the right storage type. Here are some baseline recommendations.

What you want to store	Example	Recommended storage
Files	Images, documents, PDFs	Azure Blob Storage
Key/Value pairs	User profile data looked up by user ID	Azure Table storage
Short messages intended to trigger further processing	Order requests	Azure Queue storage, Service Bus queue, or Service Bus topic
Non-relational data with a flexible schema requiring basic querying	Product catalog	Document database, such as Azure Cosmos DB, MongoDB, or Apache CouchDB
Relational data requiring richer query support, strict schema, and/or strong consistency	Product inventory	Azure SQL Database

See Choose the right data store.

Scalability considerations

A major benefit of Azure App Service is the ability to scale your application based on load. Here are some considerations to keep in mind when planning to scale your application.

App Service app

If your solution includes several App Service apps, consider deploying them to separate App Service plans. This approach enables you to scale them independently because they run on separate instances.

Similarly, consider putting a function app into its own plan so that background tasks don't run on the same instances that handle HTTP requests. If background tasks run intermittently, consider using a consumption plan, which is billed based on the number of executions, rather than hourly.

SQL Database

Increase scalability of a SQL database by *sharding* the database. Sharding refers to partitioning the database horizontally. Sharding allows you to scale out the database horizontally using Elastic Database tools. Potential benefits of sharding include:

- Better transaction throughput.
- Queries can run faster over a subset of the data.

Azure Search

Azure Search removes the overhead of performing complex data searches from the primary data store, and it can scale to handle load. See Scale resource levels for query and indexing workloads in Azure Search.

Security considerations

This section lists security considerations that are specific to the Azure services described in this article. It's not a complete list of security best practices. For some additional security considerations, see Secure an app in Azure App Service.

Cross-Origin Resource Sharing (CORS)

If you create a website and web API as separate apps, the website cannot make client-side AJAX calls to the API unless you enable CORS.

> **Note**
>
> Browser security prevents a web page from making AJAX requests to another domain. This restriction is called the same-origin policy, and prevents a malicious site from reading sensitive data from another site. CORS is a W3C standard that allows a server to relax the same-origin policy and allow some cross-origin requests while rejecting others.

App Services has built-in support for CORS, without needing to write any application code. See Consume an API app from JavaScript using CORS. Add the website to the list of allowed origins for the API.

SQL Database encryption

Use Transparent Data Encryption if you need to encrypt data at rest in the database. This feature performs real-time encryption and decryption of an entire database (including backups and transaction log files) and requires no changes to the application. Encryption does add some latency, so it's a good practice to separate the data that must be secure into its own database and enable encryption only for that database.