8KMiles Software Services, Inc

Migrating Onpremise Web Applications to

Microsoft Azure PaaS

Cookbook – Ver 1.0

8K Miles

TABLE OF CONTENTS

Table of Contents

TABLE OF CONTENTS

Introduction

Microsoft first released Classic ASP, a server side scripting language, in 1996 with IIS 3.0 as an add-on. Classic ASP opened many opportunities for developers to build dynamic websites for businesses. Later Microsoft released ASP.NET 1.0, an advanced platform built on top of.NET Common language runtime with full support for

- SOAP messages
- XML
- Object oriented programming
- Web forms

From 2002 to 2009, Microsoft released several enhancements and new versions of ASP.NET Runtime. The evolution continued and in 2010 Microsoft embraced MVC architecture and released ASP.NET MVC, a cross platform development methodology to build rich internet applications using Model-View-Controller Architecture.

ASP.Net MVC promotes

- Distributed application development methodology for building scalable applications
- Empowers coders to take full control of HTML rendered, TDD (Test Drive Development)
- Provides support for REST, without state management components like View State or Session State mechanisms.

Although there are lot of advantages with the new ASP.NET MVC platform, there are plenty of Internal portals, public facing applications, commercial SaaS applications, built using various older versions of ASP.NET such as ASP.NET 1.1, 2.0,3..5, 4.0 web forms which rely on View state and Session state component. This isn't to say applications built on older platforms are not scalable or less performant, but some characteristics of ASP. NET Webforms Platform is not cloud friendly and are anti-stateless which applies additional load on the webserver.

This whitepaper is focused on the important considerations that developers must keep in mind while migrating On-premises hosted ASP.NET applications to Microsoft Azure public cloud.

Background

Many clients approach 8KMiles with a common set of business problems such as:

- "We have public facing web applications, built on ASP.Net 1.1 hosted on our premises. We are having scalability and performance issues. Can you help us?" – A leading pharmaceuticals company.
- "We have an internal Learning Management System. It was built on ASP.Net 2.0, and we'd like to host it on Microsoft's Windows Azure Websites. We understand you have expertise in this space."

- "We have a hybrid public/private cloud application. We built it using Classic ASP, and we've uncovered some potential security exposures in it. We're hoping to contract your services to upgrade it to the latest ASP.NET platform."
- "We're struggling with legacy applications a contractor built for us on Classic ASP. We have real issues with interoperability. Our current situation won't allow for integrations across the platforms we run in our environment. We're hoping you can help us transition to a more cohesive set of solutions!"

These are a few scenarios, but there are many other cases that we come across on a daily basis. Usually while consulting on migrations or upgrade engagements, our discovery methodology is as follows:

Phase I - We sit with business stakeholders understand the strategic nature of their applications

Phase II - We dive deep into application study and dependency Identification

At 8KMiles, we have leveraged our expertise from many successful projects, to consolidate 26 key considerations you should keep in mind while migrating critical applications to Microsoft Azure.

Environment Considerations (On-premises/SaaS/PaaS/IaaS)

After deciding to migrate self-hosted On-premises applications to Microsoft Azure, the next critical decision you must make is the type of cloud service to host your applications. Such as Platform as a Service or Infrastructure as a Service (PaaS or IaaS).

PaaS provides the complete hardware and software portfolio, including the runtime environment. It empowers the developer to focus strictly on their applications, without the need to maintain the underlying infrastructure.

IaaS, on the other hand provides just the virtualized hardware alone, which transfers responsibility of installing, patching the operating systems, along with management of runtime components to the IaaS service provider. Infrastructure maintenance can be a painful process, unless you have an IT team who has the time and expertise to handle this directly or involve vendors/contractors to take care of this activity.

From our experiences consulting with many enterprise clients through application implementations, migrations and upgrades, we found PaaS the most suitable for most requirements.

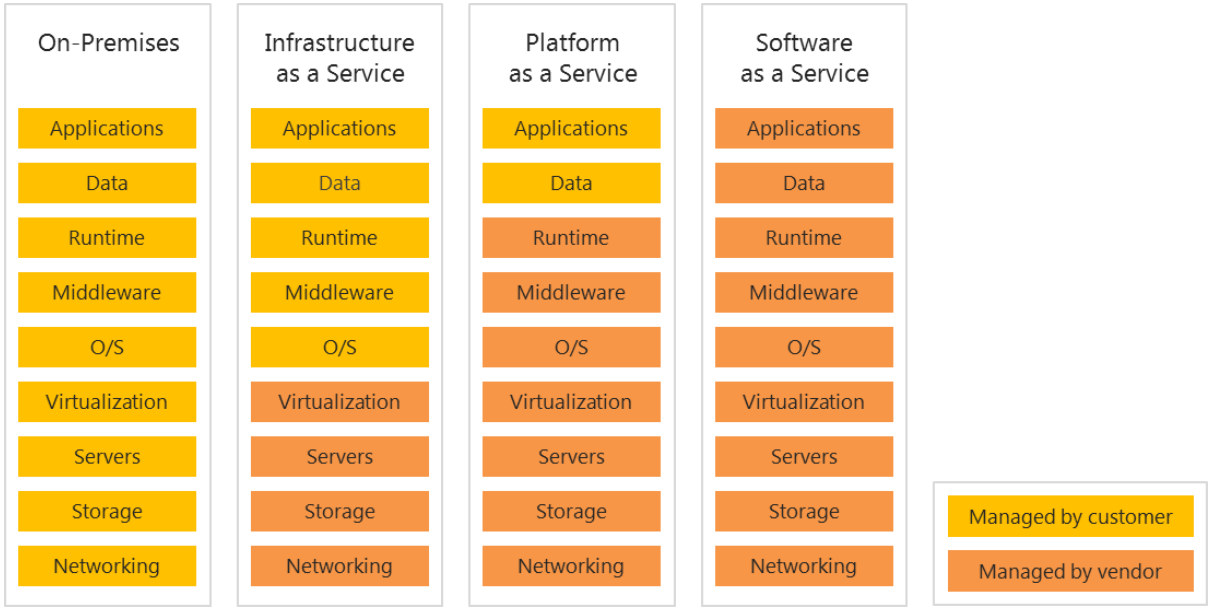
PaaS benefits include:

- Outsourced OS update management
- Integrated deployment management
- Scalable storage capacity and functionality
- Elastic availability and performance
- Cost containment and/or reduction

Although PaaS can really fit for most applications, but some customers/enterprises choose Azure Virtual Machines, (VMs) which is simply their individual preference.

It's important to understand the basic differences between the service models. Microsoft has documented the common differences between these offerings that help you choose the best one for your business. Please refer the comparison chart below to decide on the most suitable offering for your applications.

Our recommendation is to consider the feature comparison between the three Azure offerings WebApps, Cloud Services (Web & Worker Roles) and VM. Choose the one which best suit your requirements either on a per-project basis, or over all for the long haul.



(Image Credit: Microsoft)

Database Considerations

The database application, and this likely won't surprise you, a fundamental component of any data-driven business solution. Usually, SQL Server has been the default choice for Classic ASP or ASP.NET applications. Though some may have deployed their solution on an Oracle or MySQL database. Whatever may be the database backend, Microsoft Azure has native database solutions for clients who migrate to Azure Cloud.

As far as Microsoft SQL Server is concerned, Developers have 2 major options to choose from

1. Azure SQL (SQL server as a service)
2. Self-Hosted SQL Server on VM (IaaS)

Azure SQL

Azure SQL is a fully functional SQL Server Database as a Service. Delivered via the cloud, and fully managed/serviced by Microsoft. Azure SQL offers many benefits, and a natural fit for applications which are migrated from SQL Server On-Premise.

The primary benefits of Azure SQL include:

- Elasticity on demand
- No installation required
- Automated patching
- Zero maintenance
- High availability
- Pay as you go Pricing model
- Cost effective than regular SQL Server
- Unlimited storage(with elastic sharding)

Along with these benefits of Azure SQL, there are significant differences which exists due to the inherent distributed nature of this service. These differences are discussed in detail in the following sections. Azure SQL is offered in three service tiers:

- Basic (GA)
- Standard (GA)
- Premium Tier (GA)

Regardless of the tier, all Azure SQL editions provide 99.99 % Uptime and Security SLAs. The primary differences are Database Size, Replicability, Database Throughput Units (DTUs).

The Standard tier is recommended for most applications which have moderate to high input/output, because it provides a maximum 250GB of storage and up to 100 DTUs and standard geo replication capability. However, if your database is currently large, and requires more DTUs its recommended to choose premium Tier or the new SQL Database Service Version (V12) in (Preview) which offers five times better performance than the Premium Edition.

A few benefits of Azure SQL Database Service

- T-SQL support with common language runtime
- XML Indexing support
- Support for In-memory column store for better performance

Note: Refer this article to understand the complete benefits of Azure SQL.

Self-Hosted SQL Server on VM

Self-Hosted SQL Server on VM, is another option where the customer can host SQL Server on their own with the overhead of all the benefits mentioned in the Azure SQL section above. With respect to the licensing, customers can opt for a bundled license model or BYOL (Bring your own license) Self-

Hosted SQL on VM is ideal for situations such as a "Lift and Shift" model of deployment, and a quick migration path to Azure.

Here are some key considerations to bear in mind during the SQL Server migration to Azure

Distributed Transactions

Distributed Transactions are not yet fully supported in Azure SQL. If your application depends heavily on transactions, distributed transactions aren't really feasible. The only exception where it might work is if you are ready to re-write the data access layer of your application, and you build a custom transaction module to handle the data integrity. If you don't want to risk future complications with the application, and you want a simple Lift, Shift and Run model deployment, then SQL Server on VM is your best option.

SQL Server Federation/Sharding

To overcome the limitations of single-server hardware failure, and to achieve a higher level of query performance, many applications use SQL Server Federation, also known as Horizontal Sharding. Sharding is primarily used in scenarios where there is significant data growth in a particular database, or multi-tenant SaaS scenarios, where each customer's data must be stored in a separate database. This approach ensures better data isolation and data security.

Many legacy applications have implemented custom sharding mechanisms because of their unique business scenarios. If the applications that you are migrating have sharding already implemented, you can achieve the same by using Azure SQL Elastic Scale. It's technically the same as SQL Server Federation or Sharding, but it's built exclusively for Azure SQL and doesn't support SQL Server on VM.

There are three different Sharding techniques available. They are

1. Federation on SQL Server
2. Azure SQL Federation introduced in Web & Business Tier (Deprecated from September 2015)
3. Azure SQL Elastic Preview

If your choice is Azure SQL, your existing Sharding infrastructure have to be entirely re-written using the Azure Elastic Scale API, because it provides a new implementation, and an SDK which extends the elasticity and scalability benefits of Azure SQL.

Similarly, if you use the sharding infrastructure of the Azure Premium Tier (formerly Web and Business), Microsoft recommends you to adopt Azure SQL Elastic API.

If re-writing or modifying your existing code base is not an option, you need to run SQL Server on VM.

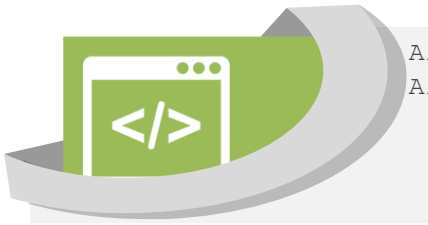
SQL Datetime Considerations

SQL Date time manipulation is one of the most intriguing topics in terms of dealing with applications with time-sensitive data. Applications hosted on single server, or on multiple servers located within the same datacenter will not impact date time processing, since they are located in a single time zone. This is not the same in the cloud. Since Microsoft offers datacenters in many different regions, and many different time zones, it's important to be cautious with time sensitive information.

Conventionally, developers have used the Date time objects of .NET 2.0 and SQL Server to store Datetime data with additional overhead of serializing, parsing and date time conversion.

The Datetime object provides the Date and Time of a particular server's calendar, and where the server resides but doesn't give any other additional information, such as the regional time zone. Some applications store Datetime in UTC format and handle the Datetime convention at the application layer which is a good practice to follow.

The alternative is ***DateTimeOffset***, which represents a point in time, typically expressed as a date and time of day, relative to Coordinated Universal Time (UTC) which uniquely and identifies a single point in time. It is advisable to update all the Datetime Objects, both in the application and database objects to Datetime Offset for better handling of the Datetime value in Azure SQL.



```
Altertable[8kmiles].[Employees]
AltercolumnJoinedDateDatetimeOffset
```

Indexing

SQL Database tables usually have primary key AKA Clustered indexes on each of the tables when they design the database, but sometime they leave if they don't have proper awareness or don't understand the database design principles. If Azure SQL is chosen as the database, it doesn't support tables which doesn't contain Primary Key constraints. It is important to make sure all the tables are updated with clustered index before migrating to Azure SQL.

Beyond clustered indexing, it's also a good practice to create non-clustered indexes for the columns that are often queried upon. Indexes help you to improve the performance for the OLTP workloads.

While migrating the database, index fragmentation needs to be considered. For example, developers should rebuild and reorganize the index in each and every table. The best practice for indexing is always measuring index fragmentation in percentage.

For example:

- If the index is fragmented by thirty percent or less, you have to reorganize the index
- If fragmentation exceeds thirty percent you should rebuild the index

Syntax for reorganizing and rebuilding the index

```
--Index Rebuild
USE <<dbname>>
GO
ALTER INDEX ALL ON [table name] REBUILD
GO

--Index Reorganize
USE <<dbname>>
GO
ALTER INDEX ALL ON [Object Name] REORGANIZE
GO
```

Azure SQL Security

Azure SQL is inherently secure, in all respects. By default, the only open port is TCP 1433. The internal firewall must be instructed to accept the connection from your local desktop, or from a designated server. All SQL connections are accessed through an encrypted connection. If your application tries to connect without https encryption, intruders can execute a "Man in the Middle" attack. To test for an encrypted connection, set "*Encrypt=True and TrustServerCertificate=False*" in the connection string.

Azure Security Best Practices

- Use the latest updated Azure SDKs and respective libraries to prevent security vulnerabilities.
- Protect your application from cross-site scripting attacks
- Usage of parameterized queries is advised to avoid SQL injections.

User Defined Datatypes

If the applications you are migrating contains user-defined Common Language Runtime (CLR) datatypes or objects, update your application to adapt to Azure SQL Supported Datatypes. Please refer the Azure supported SQL datatypes. Usually developers develop user defined data types for cases like

- Phone numbers in a specific format

- Alpha numeric employee ID's
- IP addresses

These datatypes are defined for better application consistency, but Azure SQL doesn't currently support these user defined data types. The alternative solution, if you don't want to change the application code, is to choose a SQL Server installed on a VM.

Alternate Database Options

Generally existing applications primarily used SQL Server irrespective of the type of data, be it Relational Data, Non-Relational Data, Map Data, Object Data, Graph etc, due to the unavailability of database technologies which are available now. Hence all the different variety of data was dumped only to SQL server or any other Relational Database for that matter.

But this is not the case today, with the services like

- Document DB (Document based non-relational storage, Ideal for building next general scalable web applications),
- Azure Table, a columnar storage to store Key pair Values
- Azure Queue Store, a large message storage service

While migrating your database and related information to the cloud, it is also best to:

- Analyze the kind of data which resides in the SQL Store
- Determine the best storage solution for better performance and scalability

NoSQL databases, like Azure DocumentDB and Table Storage are best suited to traffic intensive, user content-oriented web applications, which require massive scale and performance. If, on the other hand, your applications heavily depend on ACID principles** and you are comfortable with your current performance, it's a best practice to continue with SQL stores like Azure SQL or SQL Server.

**-Atomicity, Concurrency, Integrity, and Durability

Database Backup and Restore

With an On-Premises deployment, you may have configured different kinds of complex database backup and recovery mechanisms. They may be manual, or automated. Moving to Azure SQL will help you to achieve higher resiliency, advanced database replication and redundancy.

By default, Azure SQL creates one primary database, and two secondary replicas of your database in a separate physical node away from the primary server. Should the primary database fail, Azure SQL will seamlessly and automatically promote the secondary database to primary, and prevent loss of your information assets.

Along with default data backup and recovery options, Azure provides additional database recovery solutions for business executives who are risk averse, and concerned with data protection. They are

1. Standard Geo Replication

2. Active Geo Replication

These solutions you assist you to design and build web solutions which are

- Highly available
- Resilient against failure or security breach
- Data retention driven

Beyond the Azure Basic Tier, these options are available with both Standard and Premium tiers. To understand the difference between these 2 options it is important to understand the below terminologies

- **Estimated Recovery Time (ERT)**–The time it takes to restore your database to active state, when a disaster occurs at your primary datacenter
- **Recovery Point Objective (RPO)**–The time interval to when the most significant volume of data changes/adds/deletions which your application could lose after recovery.

Note: Standard GEO Replication and Active Geo Replication offers ERT* less than thirty seconds & RPO† less than five seconds.

Standard GEO Replication allows the creation of a non-readable secondary replica to another region which can be auto/manual auto failured in case the Primary goes down. On the other hand, Active Geo replications allows us to create up to four readable replicas which can solve two major purposes:

1. Database load balancing
2. Database failover

To summarize, Standard Geo Replication is targeted towards application Medium to Large applications with moderate update rates and cost centric customers. Active Geo replication is for high intensity applications, with heavy write data load and a higher license cost. Based on your application, you can evaluate these suggestions, and choose the right disaster recovery solutions for Azure SQL for your environment.

SQL Server to Azure Migration Wizard

Finally, the migration (Schema + Data) is the final and the most important part of Azure Migration process. A database migration best executed following this three step process

1. Create Schema objects in Azure SQL
2. Load or move the data from your local database to Azure SQL
3. Make sure the schema and data in Azure SQL 100% syncs with local database

There are a handful of tools to help developers with data migration process, but it is up to the developer to make sure the schema and the data adhere to Azure SQL Guidelines and Limitations. Make sure there are no custom user defined datatypes, and clustered indexes are created before moving the schema objects to Azure SQL.

With respect to the data migration and synchronization tools, there are multiple ways to do the data migration:

1. Use SQL Server Management Studio to create scripts for the schema and insert scripts for the data, and execute the same against Azure SQL
2. SQL Database Migration Wizard (SQLAzureMW) is an open source, community-developed tool, designed to help you migrate your SQL Server 2005/2008/2012/2014 databases to Azure SQL database. As a precaution, this tool doesn't validate UDFs, or the existence of a clustered index, so it's advisable to ensure those attributes are defined and validated before using the tool. For more information, Read TheSQLAzureMW Community's "CookBook" for migrating Databases to Azure SQL Database update V12
3. SQL Server 2008 Integration Services (SSIS)
4. The bulk copy utility (BCP.exe)

Note: Once the SQL DB migration is completed, make sure the schema, objects and data are identical, and same as the On-Premises database setup.

To summarize, we recommend you review the limitations, and see if they affect your current SQL server implementations. Often these limitations are relaxed and removed with the introduction of new Azure SQL versions.

Attribute	Azure SQL	SQL on Azure VM
Size	Max 500 GB	50+ Tb
Ports	only 1433	User based customization is also available
Backup database	Either we can script the database or table using standard customized SQL scripts.	By default backup and restore via SSMS
Restore database	Create a database and then execute the series of scripts	By default backup and restore via SSMS
Distributed transaction	Not supported	On demand we can use MSDTC
Scheduling and automation	Not available	By default SQL agent service will be executing series of TSql command
Login Security	2 basic Server level security not allowed for windows authentication	By default Server level security/Database level security
Database HA	Highly Available	Log shipping /Mirroring
Table level HA	Highly Available	Replication[Transactional/merge/peer to peer replication]
Instance level HA	Highly Available	Clustering [active/active, active/passive]
Disaster Recovery Solution	Active Geo Replication up to 4 online secondary's only for premium tiers	Multiple availability Zone or native database technology
Max No Database support	150 including Master	32767 including system database

SQL Server Collation Support	Only database level col SQL_LATIN1_GENERAL_CP1_CI_AS	It is available for Instance/database/table/Column

Application Considerations

.NET versions

Investigate applications built on different older versions of .NET, for example .NET 1.1, 2.0 0r 3.5.It's important to identify the current version of the .NET runtime of your application, because while you are deploying your application on the cloud, you must choose the targeted .NET framework.

Currently, the default .NET versions supported by Azure WebApp are .NETv3.5 and v4.5.

During the migration, it's recommended you recompile your application to target either v3.5 or v4.5 for better performance and security. However during the recompilation process, you may encounter certain libraries which are obsolete, and need to be replaced with the new libraries of the targeted framework, for example, v4.5.

We recommend you refactor the code, and replace obsolete binaries with the recommended alternates. In certain cases, your application might use third party tools and components which might expect a specific older version on which they are built. In such cases, you can go back to the respective vendor for the latest compatible component versions for the target application framework. Failing this, you may continue to support those old frameworks by including the configuration below into your *web.config* file.

```
<configuration>
<startup>
<supportedRuntime version="v1.1.4322"/>
<supportedRuntime version="v1.0.3705"/>
</startup>
</configuration>
```

Migrating your application to the latest version of .NET can be best accomplished through Visual Studio IDE.

1. To convert a.NETv2.0 application to .NETv4.5, open your solution using Visual Studio IDE.
2. The Visual Studio Conversion Wizard will pop up and will request you to back up the application.
3. Click "Yes" to backup, or "No" to proceed with the conversion.

4. On the next screen, select the targeted framework for your application. It is recommended you install the targeted version of .Net before the starting conversion, if not the conversion wizard will assist you to download and install the respective .Net framework. After the successful conversion, you may proceed with deploying the application to Azure right from Visual Studio IDE.

ASP.NET Caching

Caching helps to store frequently accessed data stored and retrieved from the cache memory of the server. ExistingASP.NET applications rely heavily on *InProc* Cache memory for storing and retrieving

- Output Buffers
- View State
- Session State
- Named Caches

InProc is the fastest caching technique, as it stores the caching data within the application server. The major downfall with InProc cache is, it will lose the objects when the server or the IIS gets restarted. Similarly, if your application is hosted in more than 1 server *inproc* is not an ideal solution as it depends on the Cache memory of the host server.

Azure offers three different caching solutions

1. Azure Redis Cache (Preferred)
2. Azure AppFabric Managed Cache
3. Azure AppFabric InRole Cache

Of these three solutions Azure Redis Cache is the newest offering, based on popular open source Redis Cache. Azure Redis Cache has many advantages over the other two options. It supports running Atomic operations on these types, such as:

- Appending to a string
- Incrementing the value of a hash
- Pushing values to a list
- Computing set intersection
- Identifying data unions and differences
- Defining the member with highest ranking in a sorted set

Other features include:

- Support for transactions
- Publish/Subscribe (Pub/Sub)
- Keys with a limited lifespan
- Configuration settings to make Redis behave more like a traditional cache

Azure App Fabric Managed Cache offers providers for storing

- View state
- Session State
- Output Buffer

Azure App Fabric InRole Cache is a self-hosted alternative, where the developers must take care of patching and updating their data application themselves.

Store Session State in Redis Cache

Session State dependent Web applications can easily move from InProc Cache to Azure Redis Cache in by following two steps.

1. Create a Azure Cache Node
2. Update your ***Web.config*** to refer Azure Redis Cache instead of InProc Cache

Comment out the Following Configurations in the *web.config* File

```
<!--<sessionState                    mode="InProc"
customProvider="DefaultSessionProvider">
<providers>
<add                name="DefaultSessionProvider"
type="System.Web.Providers.DefaultSessionStateP
rovider, System.Web.Providers, Version=1.0.0.0,
Culture=neutral,
PublicKeyToken=31bf3856ad364e35"
connectionStringName="DefaultConnection" />
</providers>
</sessionState> -->
```

Include the following configuration in the web.config File

```
<sessionState mode="Custom"
customProvider="8KMilesSessionStateStore">
<providers>
<!--Syntax
<add name="MySessionStateStore"  host = "127.0.0.1"
[String] port = "" [number]  accessKey = "" [String]
ssl = "false" [true|false] throwOnError = "true"
[true|false] retryTimeoutInMilliseconds = "0"
[number]
databaseId = "0" [number] applicationName = ""
[String]  connectionTimeoutInMilliseconds = "5000"
[number]
        operationTimeoutInMilliseconds = "5000"
[number]  />
    -->
<add name="8KMilesSessionStateStore "
type="Microsoft.Web.Redis.RedisSessionStateProvider"
host="8KMilesSessionStateStore.redis.cache.windows.ne
t"
    accessKey="..." ssl="true" />
</providers>
</sessionState>
```

To programmatically read and write objects into Microsoft Azure Redis Cache, download *theStackExchange.Redisprovider* from NuGet via Visual Studio IDE. Please refer the below example for reference

1-855-8KMILES (855-856-4537
info@8kmiles.com

```
ConnectionMultiplexer connection =
ConnectionMultiplexer.Connect("8KMilesSessionSt
ateStore.redis.cache.windows.net,ssl=true,passw
ord=...");
IDatabase cache = connection.GetDatabase();

// Perform cache operations using the cache
object...
// Simple put of integral data types into the
cache
cache.StringSet("key1", "value");
cache.StringSet("key2", 25);

// Simple get of data types from the cache
string key1 = cache.StringGet("key1");
int key2 = (int)cache.StringGet("key2");
```

Event Logs

Handling and management of errors, events and logs is a cumbersome but a mandatory process to ensure application health. It is always essential to analyze the errors, and resolve them on a timely basis.

Most existing ASP.NET applications do use Logging libraries and SDKs or build a custom logging mechanism to capture the logs and monitoring infrastructure. These logs are usually retained from a couple of weeks, to a month to reduce the storage cost of these logs. Usually developers store the log files in a text file and store them in a centralized location. To avoid the overhead of parsing and conversion to feed the data to log monitoring applications, another option is to store the log data in the SQL Server Database.

Microsoft Azure provides native support for Log Tracing and Diagnostics. Azure provides three different ways to collect and store the log information. They are:

File System Storage

This storage format helps you to store the logs in the text format on the host server, and then you can configure to move the data to table storage or Azure SQL on a timely intervals for advanced monitoring and analysis.

Table Storage

This option will directly store the data in Azure Table storage without having to storage the data in the host server.

Blob Storage

Blob storage is also very similar to File System storage but this will save the log files in the CSV format.

Another option is "Azure Operational Insights" exclusive for applications hosted on VM/Physical Machine hosted on premise and private or hybrid clouds. It's the cloud based Log/Security/Capacity Planning service fully managed by Microsoft. With Operational Insights, you can completely automate, store, analyze the logs.

While migrating applications to Azure, it's not advisable to move old log files into Azure to avoid raising confusions between the Azure Server Logs, and Application or On-premises logs.

Static File Considerations (Image/Scripts/CSS/Video/Audio)

Assets such as images and CSS are the bulky and bandwidth intensive parts of your web applications. Hosting these files within your application environment has number of problems including

1. Increasing the overhead of the application server to serve static contents as well
2. Increasing the bandwidth cost of the application hosting server
3. Increasing the size of the application packages during frequent deployment

Offloading these static assets from your application host to a dedicated storage account is a recommended approach. While migrating your application to Microsoft Azure, you can move these static files to Azure Storage. Directly serve the static content from the dedicated storage account, instead of your application server. This load separation will overcome the disadvantages mentioned above, along with benefits of:

- Boosting throughput performance
- Reducing your existing hosting costs
- Providing the ability to update static files, by replacing them in a Content Delivery Network (CDN) without having to deploy entire website.

When you adopt Blob storage, you have to update all the physical path to logical paths. Refer to the configuration snippet below to fix this challenge in the web.config file.

```
<rewrite>
<rules>
<rule name="images-to-azure">
<match url="images/(.*)" />
<action                              type="Redirect"
url="https://xxxxx.vo.msecnd.net/images/{R:1}"
/>
</rule>
<rule name="css-to-azure">
<match url="css/(.*)" />
<action                              type="Redirect"
url="https://xxxxx.vo.msecnd.net/css/{R:1}" />
</rule>
</rules>
</rewrite>
```

Check out the Community-developed Azure Storage Explorer (Version 6). It can help you connect your Azure hosted application to your storage account, and manage your static files.

Content Delivery Networks (CDNs)

Content management systems, document management systems and repositories of other major user generated content deal with large amounts of data. Apart from this, integral website files such as JavaScript, CSS, Images and other media files which are downloaded on every page request takes a great deal of bandwidth of your origin/host server.

Usually on-premises applications neglect page load performance and bandwidth issues. You can offload these static resources from your application server to the purpose-built Azure CDN. By doing this, the bandwidth usage of the origin server the page loading performance can actually double.

Another reason to adopt the Azure CDN is to keep these static files close to your customers. It will also reduce multi network hops, and you can gain significant page performance. As of this writing, Microsoft Azure has around 31 POP locations spread across United States, Europe, Asia, Australia and South America.

Adopting a CDN is a 2 step process:

1. Create a storage account if it is not already created
2. Create a CDN repository and map the storage account in the origin domain and finally create an endpoint to access the static assets.

Not all applications require a CDN. A CDN is best suited for heavy traffic websites with lot of static assets and targeted users which are spread across many different regions. Analyze your IIS Logs to

understand how much bandwidth your static assets consume compared to dynamic content, if your static content bandwidth is less, probably CDN will not make sense.

Active Directory

Many internal and enterprise Line of Business applications leverage Windows Active Directory for authenticating and authorizing their users. These applications have many complex configurations related to the integration of on-premises Windows Active Directory setup. ASP.NET 2.0 provides support for Windows AD integration through ***Active Directory Membership Provider***. Developers integrate Windows Active Directory with their applications through this provider mechanism.

Migrating to the cloud doesn't affect, or replace Windows Active directory integration with your application, except for simplifying the cluttered configuration process. You can adopt the same capability by migrating your applications to Azure Cloud, by using Azure Active Directory with minimal configuration and code changes. Integrating Azure Active Directory with your cloud application is a three step process.

1. Setup Active Directory on Azure
2. Setup authentication through the authorization configuration wizard, using the Configure tab of the website. Associate the newly created Active Directory you set up in Step 1.
3. Select or create the Azure Active Directory app for the Website

Once you complete the steps mentioned above, your application will only allow users from the associated Active Directory. However there are known limitations such as

- The target framework must be running on .NET 4.5
- All users in the configured directory will have access to the application.
- The entire site will require a login and a password
- Headless authentication/authorization for API scenarios or service to service scenarios are not currently supported.
- There is no distributed log-out, so logging the user out will only do so for this application, and not all sessions globally.

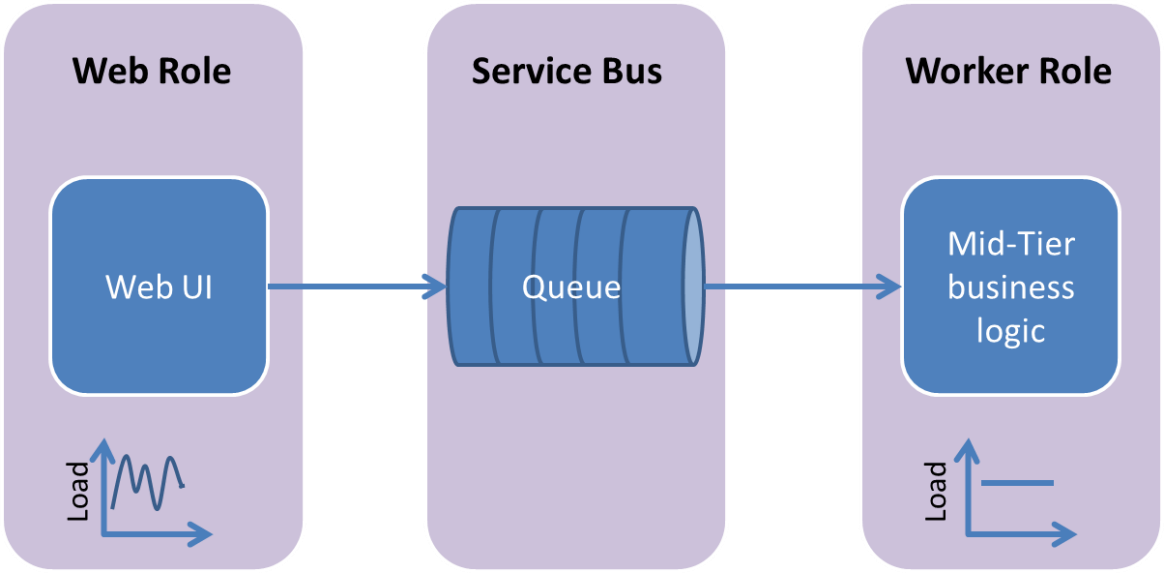
If you are satisfied with your application given current restrictions, it's wise to choose the Default Authentication/Authorization capability of Azure WebApps.

Queue/Service Bus

Queues are not new. They have existed ever since Multi-Tiered and Multi-Layered applications have been developed. Queues have helped our applications to distribute messages between components, servers and layers. Popular use cases include uploading images to a web server and processing it in a distinct component. Prior to Queues or Service Busses, developers used to build custom Queue mechanisms using SQL Server and managed manipulation like insertion, retrieval and deletion.

The process of migrating applications which use custom Queue messaging can be replaced with Azure Queue storage solution. It offers SDKs for a wide variety of programming languages including

- Java
- PHP
- Python
- .NET



(Image Credit: Microsoft)

Azure offers scalable and reliable messaging queues, however adopting Azure Queues requires developers to modify the existing application, according to the Azure Queue SDK.

Carefully analyze

- The current Queue infrastructure
- Any complexities involved in adopting Azure Queues
- Benefits of Azure Queue

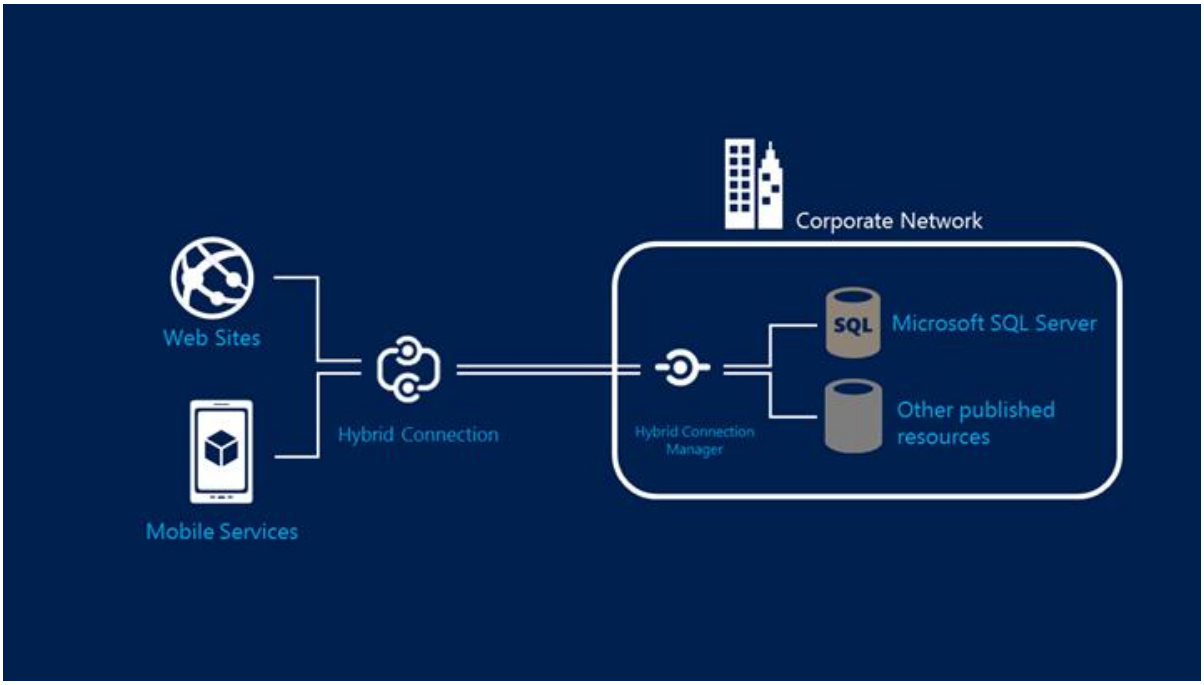
Consider both the benefits and challenges involved to choose the most suitable solution for your environment.

Cloud – On premise Connectivity

While migrating applications to the cloud, the preliminary step is to identify any dependent systems such as

- Databases
- Third Party Tools
- Data feeds etc

Should you have concerns regarding data security, compliance, or legal restrictions, you might be forced to keep the database on your premises, and migrate the application alone to Azure. Microsoft Azure provides Azure Hybrid Connectivity Tools, which helps developers to just migrate applications and still consume the data from an On-premises database or any custom sources.



(Image Credit: Microsoft)

Hybrid Connection Manager is a part of BizTalk services, provided by Microsoft to bridge the connectivity between on-premises solutions and Azure Cloud Services. It can be installed on a dedicated or shared server, inside your corporate firewall to allow Azure to connect to your designated databases including SQL Server, MYSQL & Oracle.

Hybrid Connection Manager uses Shared Access Signature (SAS) to secure the connectivity between your Azure account and your on-premises database. It creates separate security keys for the application and the database; Developers can individually revoke and roll these keys over for security reasons.

Ports to be opened on your on-premises network

Port	Description
80	HTTP port; Used for certificate validation.

443	HTTPS port
5671	Used to connect to Azure. If TCP port 5671 is unavailable, TCP port 443 is used.
9352	Used to push and pull data. If TCP port 9352 is unavailable, TCP port 443 is used.

Search Considerations

Many existing mission-critical LOB applications use SQL Server full-text search indexing engines for better search performance. Typically, these would incur high costs, since developing these Indexing systems onsite, and writing custom coding in the application layer. Developers who choose not to modify the search section of the application, but want to bring the same search indexing capability to Azure can implement their application using SQL Server 2008 or 2012 editions.

Another alternative is to use Azure Search, a PaaS offering which offloads the search indexing functionality into scalable and high availability Azure Search Services.

Batch Jobs (Background Tasks)

Usually, applications process lot of data in bulk, known as Batch processing. Batch execution includes variety of jobs including:

- Bulk database updates
- Automated transaction processing
- ETL processing
- Digital images processing (resize, convert, watermark, or otherwise edit image files)
- Files conversion from one format to another

The bulk execution of jobs executed sequentially, time based execution for e.g. morning 8.00 am to 9.00. Depends upon the application, server infrastructure, based on the size of the data, based on the logic may take as less minutes to hours based on the intense of the data.

If the volume of the data is large, and system capacity is low, it may take more time for batch processing jobs to complete.

(Image Credit: Microsoft)

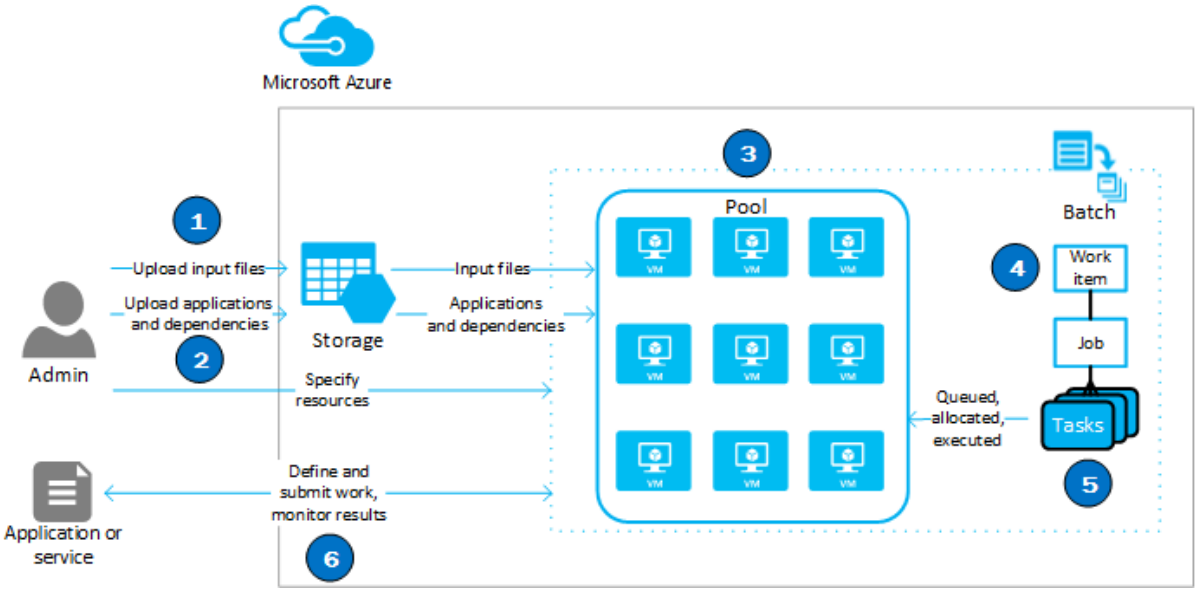
If other applications are dependent on the output of this system, the subsequent process might also get delayed. On-premises applications usually provision less system resources for batch systems, and architects often fail to allow for scalability in such scenarios.

With Azure cloud, developers now have a wealth of resources at their disposal. Instead of running a batch job on a single server for ten hours, they can boot ten different servers, and complete the job within an hour. The dependent applications can perform faster with the same cost.

Migrating these batch processes to Microsoft Azure has various benefits including:

- On demand resources
- Hyper scale parallel processing
- Azure Storage Integration

Azure Batch is a new service built by Microsoft. It follows a distributed job execution architecture. Existing custom-built Batch processing structures cannot be easily migrated to the new Azure Batch processing system.



(Image Credit: Microsoft)

Existing Batch processing systems have to be rebuilt using *Microsoft.Azure.Batch* SDKs, which can obtained from NuGet packages. If the application has a complex batch processing system and you don't want to re-build or customize it, we recommend execute batch processing in a VM environment.

Schedulers

Schedulers are important parts of the batch jobs if they are time-bound. Traditionally, developers use Windows Server's task scheduler to execute tasks in a fixed-time interval. This is good for

- Executing a job within the server or in a remote server
- Providing a basic monitoring window to display the current tasks
- Enabling developers/administrators to view task properties and history etc.

This system is not very scalable or reliable, simply because if the server where the scheduler configured goes down, all the jobs will ultimately fail. Unless your customer complains, or unless you have a monitoring system for the hosted server it is likely nobody will notice this.

You can replace your existing scheduling mechanism with Azure Scheduler, which provides scalable and reliable Scheduling functionality in Azure.

Benefits of Azure Scheduler include:

- Call HTTP/HTTPS services inside your Azure subscription and resources on your premises
- Run jobs on any schedule—now, later or recurring
- Use Azure Storage queues for long-running or offline jobs

Scalability and High Availability Considerations

A key aspect of the application migration is to enable the existing application to leverage the elastic scalability (up and down) features of Azure cloud. It's important to analyze and refactor the application, to adapt to the scalability attributes of Azure Cloud. This doesn't often require any rewriting or modifying of your existing application. It usually can be achieved using configurations with minimal effort.

Scalability strategies may differ from one application to another, and from time zone to time zone. For example, a multinational company might have branches all over the globe. Their applications might have heavy traffic around the clock. A smaller local organization, only having employees only in a specific country or region might scale down the CRM application after hours due to less traffic.

Identifying the scalability patterns of your applications and configuring based on meeting scalability requirements will help you to optimize your return on investment, and realize the benefits of low total cost of ownership of adopting Azure Cloud.

Business Continuity (Backup& Resiliency Data Recovery)

On-premises applications suffered backup deficiency due to:

- The lack of efficient storage availability
- Manual labor/overhead involved
- Connectivity
- Distributed geographic locations of the technology involved
- High Cost

Out of many applications we analyzed, we found very few applications had Backup and Data recovery planned. The remainder had only manual file and/or database back up practices in place.

Whether the application you are migrating has, or doesn't currently have a Backup and Data recovery strategy, it's recommended to setup Application and Database backups on Azure Cloud. Azure Backup & Site Recovery is a suite of backup solutions for your applications, databases and VMs.

TCO & ROI

Microsoft Azure provides a new breed of services to help enterprise and SMBs to move their technology spend from the Capex to Opex. Many of your legacy applications might have suffered from

variety of hardware and software limitations, due to Capex availability shortages, and the manual effort required to acquire, install and configure new equipment.

Many enterprises have either over-provisioned or under-powered their technology infrastructure, which can high IT costs and/or resource-intensive ongoing maintenance. With Azure Cloud, robust virtualized hardware and software resources are available for to companies on demand. Azure provides a simplified pay-as you-go model with clearly defined service level agreements, as opposed to complex software licensing terms and conditions.

From a TCO & ROI point of view, Migrating to cloud may look like a costly initiative in a short term, but effectively designing and developing your application to leverage key cloud advantages and value-added features such as Elastic Scalability, Bundled license options and Service Level Agreements, enabling you to pay only for what you use will reduce the overall TCO of your IT investments.

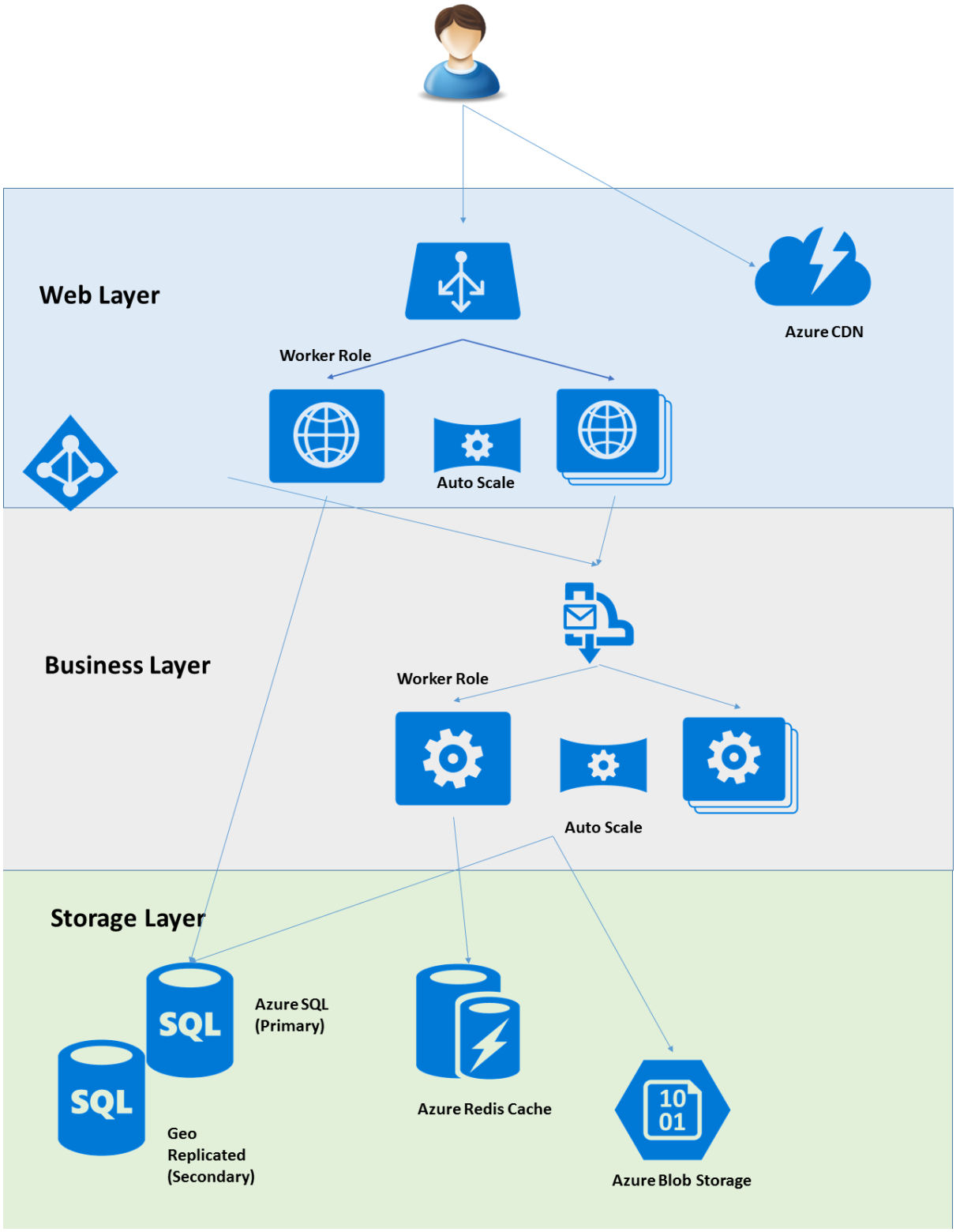
Below is a simple cost estimation for a medium sized 3-tier web application with the components like Compute, Database, and Storage. But this is just an indicative cost estimation, but the actual cost may vary depend on your unique application requirements.

Resource	Description	Monthly Cost
Web Role	Standard D2 (2 cores, 7 GB RAM, 100 GB SSD) Count 2	**$473.19**(0.636/hr)
Worker Role	Standard A2 (2 cores, 3.5 GB RAM)	**$238.08**($0.32/hr)
SQL Database	Standard (Performance Level S2, 5 DBs)	**$375.00**
Redis Cache	Basic (6 GB)	**$133.92**($0.18/hr)
Storage (Geo Redundant)		
Block Blobs	100 GB (Geo Redundant)	**$4.80**
Tables & Queues	100 GB (Geo Redundant)	**$9.50**
Storage Transaction	6 Millions	**$0.22**
Storage Backup	200 GB	**$9.60**

Bandwidth	United States + Europe egress (10GB)	**$0.44**
		$1244. 75

** Cost estimation above is based on the Azure Calculator.

8K Miles Software Services Inc 2007-2015 1-855-8KMILES (855-856-4537
info@8kmiles.com



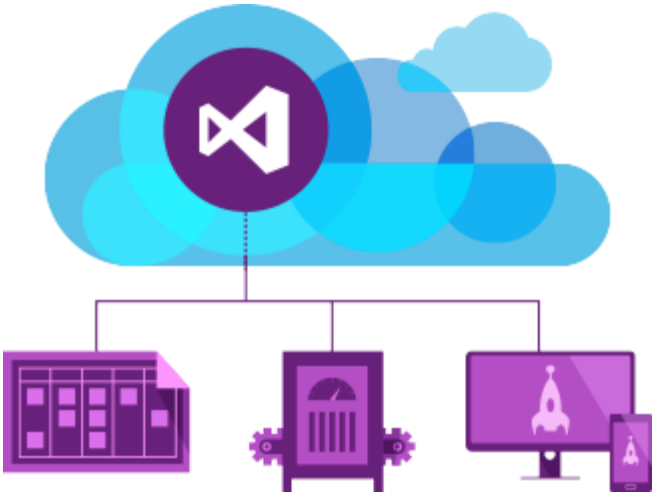
Three tier Web application architecture on Microsoft Azure

Deployment/Continuous integration & Maintenance

The biggest benefit of adopting cloud is the simplification of deployment and continuous integration of solutions to production, thereby releasing new features, enhancement and bug fixes in nearly minutes and hours compared to days and weeks. Conventional deployment mechanisms allowed developers with very few options for deployment, such as installing to production servers via FTP. Configuring the development, staging and Testing/QA environments was difficult, and often took weeks or months to configure and provision. Beyond these challenges, synchronizing product to development or development to staging was often a struggle.

By adopting Azure cloud, developers are empowered with many options. Azure provides:

- Simplified mechanisms to develop and deploy their solutions with a single mouse click.
- Development environments, preloaded with all the necessary software.



(Image Credit: Microsoft)

- Tools to replicate the production data to the staging environment and duplicating your development environment to UAT testing is now simplified.

Visual Studio IDE is a one stop solution for developers to develop, debug (locally as well as remotely) stress test and deploy the application to Azure Cloud. Visual Studio (VS natively integrates with Team Foundation Server (TFS). VS + TFS online lets you to collaborate, version control, continuously build, integrate and test your solution on the cloud.

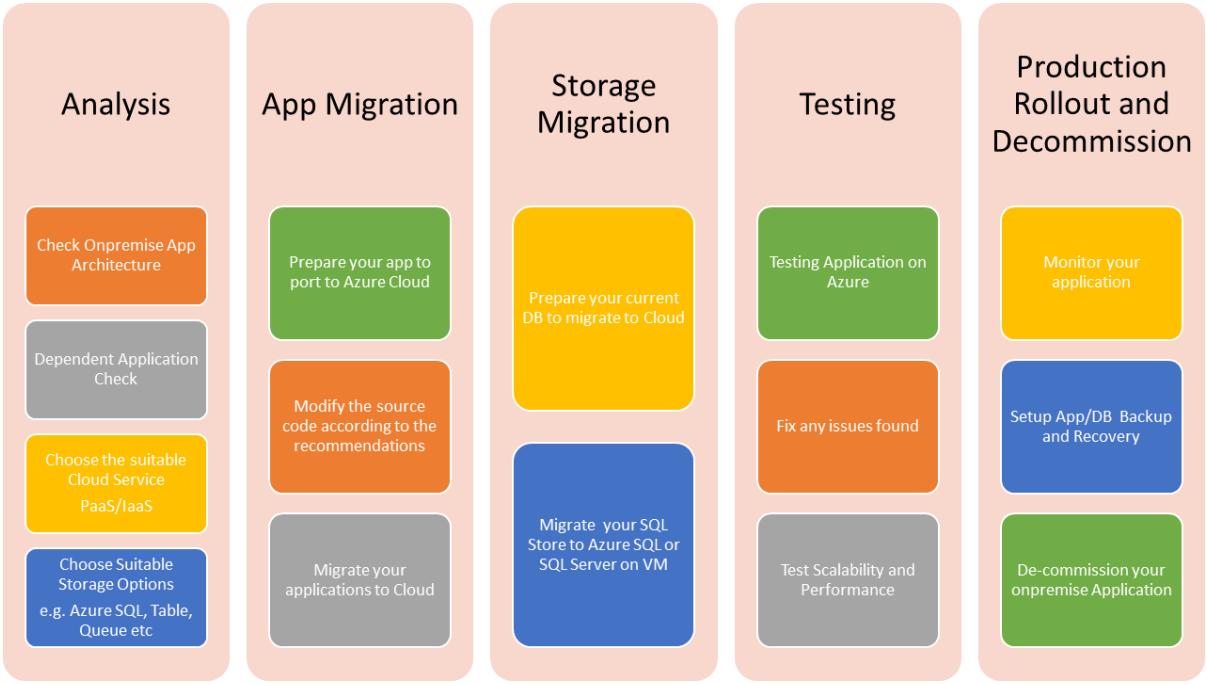
Developers can also choose Open Source alternatives, like GitHub to build a collaborative development and deployment environment and use *NuGet* to consume and distribute packages within your team and others.

Overall Migration Approach

Migrating your On-premises applications to Azure cloud may be challenging, or in many cases it takes a much simpler path. It depends on the complexity of your application, its size and how it is designed and/or developed. Regardless of the application, you have to work through the various application migration stages including:

1. Application discovery
2. Dependency identification
3. Data/Application migration
4. Testing the production application.

We recommend spending at least 30-40% of time allocated to analysis phase and do a detailed analysis and developing a migration plan.



Application Migration Approach

Conclusion

In this whitepaper, we have captured the various real-time scenarios and use cases usually found in migrations from on-premises to Azure cloud. We described various strategies and recommendations for suitable migration approaches. However, not all applications are equal! We recommend you work with an experienced Azure expert consultant, to define the best approach based on your unique business environment. Take a methodical, step by step approach and migrate your applications to cloud based on criteria including technical, compliance, and business factors.

Need assistance in your application analysis, deliver your “Lift and Shift”?

Contact 8KMiles!

We would be glad to help you out with your unique migration requirements.

8KMiles has the:

- Expertise from many successful Azure migration projects
- A proven methodology for successful delivery
- Intellectual capital which provides our team with the tools to execute on standard and complex engagements

References

http://blogs.msdn.com/b/davidrickard/archive/2012/04/07/system-datetime-good-practices-and-common-pitfalls.aspx

http://stackoverflow.com/questions/4331189/datetime-vs-datetimeoffset

http://azure.microsoft.com/en-in/pricing/details/sql-database/

https://msdn.microsoft.com/en-us/library/azure/ee336245.aspx

https://msdn.microsoft.com/en-us/library/ms187752.aspx

http://searchsqlserver.techtarget.com/tip/Three-tips-for-a-better-SQL-Azure-migration

https://msdn.microsoft.com/en-us/library/azure/jj156153.aspx

https://msdn.microsoft.com/en-us/library/azure/dn690524.aspx

http://azure.microsoft.com/en-in/documentation/articles/cache-dotnet-how-to-use-service/

http://www.dotnetlogging.com/

http://azure.microsoft.com/blog/2014/09/18/enabling-cdn-for-azure-websites/

https://msdn.microsoft.com/en-us/library/azure/gg680302.aspx

http://azure.microsoft.com/en-in/documentation/articles/cdn-how-to-use/

http://azure.microsoft.com/blog/2014/11/13/azure-websites-authentication-authorization/

https://msdn.microsoft.com/en-us/library/ff650308.aspx

https://msdn.microsoft.com/en-us/library/dn589776.aspx

http://microsoftazurewebsitescheatsheet.info/

http://azure.microsoft.com/en-us/documentation/articles/integration-hybrid-connection-overview/

http://azure.microsoft.com/en-us/documentation/articles/web-sites-hybrid-connection-connect-on-premises-sql-server/

http://azure.microsoft.com/en-us/documentation/articles/batch-dotnet-get-started/

About The Author

Ilyas is a Microsoft Certified Azure Solution Architect and Cloud Evangelist at 8KMiles. He specializes in Application Architecture, IaaS/PaaS Consultation, Application migrations, exclusively on Microsoft Azure. He has over 9 years of experience in designing, developing, consulting and delivering world-class software products and solutions.

Ilyas has provided solutions for several customers to leverage the robust applications and services that run on the Microsoft Azure Cloud platform. He is very passionate about technology. You can find his blogs at http://www.8kmiles.com/blog & his personal blog http://www.bornoncloud.com

About 8KMiles

8Kmiles Software Services Inc is a solutions company that is focused on helping organizations of all sizes to integrate Cloud, Identity and Big Data into their IT and business strategies. 8KMiles' team of experts, located in North America and India offer a host of services and solutions such as Cloud / Federated Identity Consulting, Cloud Engineering, Migration, Big Data services, Managed Services on Amazon Web Services. 8KMiles offers specialized expertise in matured verticals such as Pharma, Retail, Media, Travel, and Healthcare. Visit us at http://www.8kmiles.com/

1-855-8KMILES (855-856-4537
info@8kmiles.com