Abstract

Cloud computing is the set of technologies and infrastructure capabilities being offered in a utility based consumption model. Windows Azure is Microsoft’s Cloud Computing offering to help its customers realize the benefits of cloud computing.

This paper is an introduction on Windows Azure and provides insights into different aspects of Azure based development especially for those who are interested in adopting Windows Azure within their Enterprise IT landscape. The key aspects we shall be discussing in this paper include design principles required for building Azure based applications, development lifecycle on Azure and key challenges enterprises would have to tackle in the process of on-boarding Windows Azure in the enterprise.

For more information, contact askus@infosys.com

July 2010
Introduction

Cloud computing is the set of technologies and infrastructure capabilities being offered in a utility based consumption model. Following are the key value propositions of Cloud:

1. Lower upfront capital investments
2. Improved business agility and ability to innovate with reduced entry barriers
3. Pay-per-Use model which significantly aligns IT investment to operational expenditure
4. Elastic scale which helps meet on-demand needs more dynamically

Cloud computing today can be delivered in multiple forms and which can be broadly categorized as:

Infrastructure as a Service (IaaS)

IaaS providers abstract IT infrastructure resources such as CPU, storage and memory as services. A Cloud provider manages the physical infrastructure; provisions virtualized instances of operating systems to the cloud consumer. The consumer here is given complete ownership of the virtual image which he/she can choose to configure as they think appropriate.

Drivers to use IaaS delivery models:

A cloud consumer has better control and flexibility at a system level, however, on the flip side higher administration and licensing overheads will have to be borne by the customer. Also, upgrades & patches deployment is the responsibility of the cloud consumer.

Amazon, Enamoly, Rackspace are a few of the leading IaaS providers.

Microsoft has announced that in the near future, Windows Azure platform would be providing users with greater control over their cloud Virtual Machines (VM) and also the ability to deploy any standard VM image on Azure. Click here for details. When this happens Azure extends from being a PaaS platform to an IaaS based offering as well.

Platform as a Service (PaaS)

PaaS is the next higher level of abstraction in which not only do the technical infrastructure resources, as discussed above, but essential application infrastructure services such as computation, messaging, structured data persistence, connectivity, access control, etc, form a part of the service offering. These are also referred to as the building blocks of cloud applications.

Drivers to use PaaS delivery models:

Here, cloud consumers have lesser control and flexibility in terms of defining the underlying IT configuration. However, there is lesser administration and licensing overheads when compared to the earlier model discussed. Also, service consumers would not be responsible to upgrade systems or apply patches on the platform. This would be completely managed and automated behind the scenes by the PaaS providers.

Microsoft’s Azure, Google AppEngine, Force.com are the leading PaaS providers in the industry today. In addition to being...
PaaS enabled, the Virtual Machine Role will enable Microsoft to position the Windows Azure platform as an IaaS offering which gives customers the capability to create, deploy and manage VM images on premises and host them on the Microsoft managed datacenters.

**Software as a Service (SaaS)**

SaaS is the highest level of abstraction on the cloud. Here, key application functionalities such CRM, SCM are abstracted and delivered over the cloud.

**Drivers to use SaaS delivery models:**

Cloud consumers do not have any control in defining the underlying infrastructure configuration and so there is no management and operational overheads in maintaining the application.

Salesforce.com and Amitive are a few examples of SaaS based applications.

**What is Azure?**

Azure is Microsoft's Cloud Computing offering for developers to build and deploy applications on a pay-per-use basis. Azure is a comprehensively managed hosted platform for running applications and services.

> “The Windows Azure Platform is an internet-scale cloud services platform hosted in Microsoft data centers, which provides an operating system and a set of developer services that can be used individually or together. Azure's flexible and interoperable platform can be used to build new applications to run from the cloud or enhance existing applications with cloud-based capabilities. Its open architecture gives developers the choice to build web applications, applications running on connected devices, PCs, servers, or hybrid solutions offering the best of online and on-premises.”

Microsoft (http://www.microsoft.com/presspass/events/pdc/docs/StrataFS.doc)

Windows Azure provides a scalable infrastructure for customer to run their web services and applications.

**Azure provides following Runtime services**

- Sandbox VM environment for processing and hosting web applications
- Storage system for files and structured content
- Connectivity services for integration with on premise applications
- SQL server relational capabilities
- Content delivery network

**Design Time Services**

- .NET SDK for building applications for Azure
- Visual Studio Tools for packaging and deploying applications to Azure
- Eclipse Plug-in for building PHP and Ruby applications for Azure
- Simulation environment for testing Azure applications and storage locally

**Operations Services**

- Web based console for provisioning and managing Azure accounts
- Web based console for management and monitoring applications

**Azure supports:**

<table>
<thead>
<tr>
<th>Programming language</th>
<th>.NET, PHP, Java, Ruby</th>
</tr>
</thead>
<tbody>
<tr>
<td>Standards and protocols</td>
<td>SOAP, REST, and XML</td>
</tr>
<tr>
<td>Development tools</td>
<td>Visual Studio, Eclipse</td>
</tr>
</tbody>
</table>
This enables the platform to be interoperable and be able to support several complex heterogeneous scenarios pertinent to enterprises which may span both, on-premise as well as cloud.

The various capabilities of the platform, which we shall be discussing later in this paper, are made available to Azure customers as consumable services. What this offers is the ability for customers to be able to utilize these services in a utility based manner by simply plugging in the services within their applications and have the usage metered on a pay-as-you-go consumption model. However, behind this simplicity of usage lies the true value of the platform. Azure at the core is run by colossal but efficient data centers which provide users with global scale capabilities.

**Azure Core**

The Windows Azure Platform is just the tip of the iceberg, as is simplistically represented in the figure below. Azure, at the core, is essentially driven by a set of software and hardware components geographically distributed across the globe, which is completely abstracted by a services layer on the top.

![Azure Core Diagram](image)

**Azure Data Centers**

Microsoft has and will continue to invest in building huge data centers across the globe. The investments Microsoft makes is essential to achieve the economies of scale that would help Microsoft run Azure more cost effectively. A lot of research and investments have also gone in making the data centers run with high efficiency to provide the access to environment friendly sustainable technologies to its customers.

**Azure Hardware Nodes**

The Azure data center is a host for several technical infrastructure components maintained in large numbers. These components include fiber optic cables, high-speed switches, routers, hardware balancers, UPSs, servers, commodity hardware systems, etc., and that too in large numbers.

**Azure Fabric**

The Fabric is the actual lifeline of the Azure Platform. All above mentioned infrastructure components are interconnected to form a mesh or a fabric. A fabric is a web of inter-connected nodes where the inter-connections are facilitated by high-speed switches through fiber optic channels and the nodes in the fabric are the commodity hardware, servers, hardware balancers and other technical infrastructure. These infrastructure pieces are glued together by the Azure Fabric controller which governs the proper functioning of Azure. The Azure Fabric controller maintains, manages and provisions machines to host the applications created to be deployed on Azure. It offers services outside the mesh which are responsible for managing all the nodes and other edge devices such as load balancers, servers, switches, routers etc. on the fabric.

An operational view of the Fabric controller is shown below; Azure cloud will consist of several virtualized nodes and be regularly managed and monitored by the fabric controller. The node in the cloud is an instance of the Azure role(s) i.e. either Web Role/Worker Roles.

A role is a template defining the application type which would be run on the cloud. The template translates into some predefined specifications around fault and upgrades, software requirements, and machine level resources. The fabric
controller utilizes the configurations defined in the role to govern the provisioning and management of nodes and additional technical infrastructure resources on the cloud.

Additionally the fabric controller is designed to ensure that applications are always available. The fabric controller replicates application instances on multiple nodes and monitors for any faults or any service upgrades to ensure the application is running round the clock. These are also known as ‘Fault Domains’ and ‘Update Domains’ respectively. And, availability in terms of monitoring and managing the various application nodes is handled as a part of these domains. The Fabric controller is also responsible to manage load balancers, switches and other edge resources which have to be provisioned to run global scale applications on the cloud. All this is implicit and Enterprises would not have to bother.

**Point of View**

Windows platform provides its users:
1. Ability to provision infrastructure on demand
2. Platform which can scale with a response time in minutes
3. Infrastructure which is managed using automated tools
4. Environment which is current and is optimized to run web applications efficiently

Here are few scenarios where we see Azure to be leveraged:
1. Experiencing seasonal or fluctuating workloads
2. Workloads which experience massive scale
3. Enterprises looking at consolidating their existing infrastructure and moving towards Green IT
4. Enterprises that want to have a quick and seamless way to integrate with their partners and suppliers
5. Provide collaboration solution for desk-less or the mobile workforce
6. Software Development teams seeking quick and cost effective ways to set up their development infrastructure
7. Offsite facility to backup and archive data
8. Backend processes which have high demand for processing computational intensive tasks such as billing, brokerage calculations, processing media files, etc
9. Applications needed for relatively shorter period (tactical applications/non strategic/opportunistic)
10. Applications which must either fail or scale fast
11. Azure based drives help realize scenarios where data needs to be shared across multiple applications
12. In the future, hosting standard apps would also be supported.

Some typical application stereotypes which can be deployed on Azure are:
- Social Apps
- Cross Enterprise Integration apps
- Web apps
- Backup & Archival
- Standard Apps will also be supported soon
Similarly, ISV's or SME's alike can leverage the platform and utilize best of breed technology to run their business which are otherwise not accessible to them, either owing to the high entry barriers such as high cost of purchasing licenses and setting up the infrastructure and lack of experience or skilled resource availability.

Let's look at the Windows Azure Platform along with the services available today and which will help realize the above scenarios.

**The Windows Azure Platform**

The complexity which underlies the Azure Service layer is abstracted through the services layer which is a part of the Azure Service Platform.

The key design principles behind the Windows Azure Platform are:

1. Access to on-demand computing power
2. Infinite storage capacity
   a. Anytime and anywhere availability
   b. Utility based model of usage

The Windows Azure Platform technology stack is as shown below:

![Windows Azure Platform Stack](image1)

The two layers, mentioned below, are the main constituents of the Windows Azure Platform

1. Windows Azure Platform
2. Building Block Services

Let us look into the details of the Windows Azure Platform:


**Windows Azure Platform**

**Windows Azure**

Windows Azure is one of the core constituents of the Windows Azure platform, also known as the Windows OS on the cloud. It provides key infrastructure capabilities like compute and storage as services. A separate set of management and monitoring services also form a part of this layer and are primarily responsible for handling provisioning, management and monitoring of applications on Azure.

Windows Azure allows customers to have their application code executed and managed within the Microsoft data centers. You can visualize it as your application code being hosted in optimized version of IIS7 and run from a virtualized instance of a non-standard Azure specialized version of the Windows Server OS.

The Windows Azure OS leverages core components of Server 2008, but it is highly optimized and slipstreamed with Windows Azure OS specific bits. Also HWC, utilizing only the core IIS functionality, is a new feature in IIS 7 that enables developers to create applications.

With Azure’s automated service management process, as soon as an application code is deployed, the Azure fabric automatically copies a virtual image to a machine in the Azure fabric mesh and spins off a new virtual instance of the OS. Once the OS is loaded on the fabric, the application code is then copied and based on the role definition, a specialized application process is spun up; say, if the role definition is that of a web role, an HWC instance is spun up as a new process to host the user's web role.

**Compute**

This is the computational core of the Windows Azure layer. It provides a scalable on-demand hosting environment to run Azure applications. Azure provides a virtualized compute environment leveraging core components of the Windows Server running Windows Server 2008, Hyper-V and IIS7 which have been highly optimized specifically for Azure to support the highly demanding cloud workloads.

In the future, Windows Azure will provide users with capabilities to run their virtual images in the form of VHDs on Azure. This would provide users with more options to realize more scenarios on cloud, especially in cases where users prefer having more control of the target environment than which is supported today. This will potentially attempt to reduce, if not completely eliminate, some of the deployment asymmetries which exist with the current Azure model with running an application both on-premises as well as on the Azure with the same code base.

As also touched upon in the previous section, Windows Azure currently provides support for two roles, i.e. Web Role and Worker Role, with support for role customization being provided in the future:

1. **Web Role**

A web role is a process which is specifically configured for directing Azure to hosting web based applications such as Asp.Net, WCF services (SOAP/REST), ASMX services, FastCGI (eg.PHP), Java, Python application.

The Web role can be used to host the following application types on Azure:

- Social applications
- Custom Web apps
- Platform based services

2. **Worker Role**

A Worker role is similar to the windows service and can support generalized application developments which may not necessarily be web oriented in nature. The model is useful in running tasks which are offline in nature and do not need to behave in real time. Additionally, it also provides the framework to deploy non-Microsoft technologies on Azure. This allows technologies such as Java to leverage capabilities of Azure.

The worker role plays a key function in realizing scenarios which are required to meet high scale computation intensive tasks. With the Worker role, developers program to work with queues, building applications to asynchronously process messages that arrive on queue. This design helps to able to load balance workloads across multiple instances of worker
roles and parallelize any task execution. This pattern would be useful where the user needs to implement batch or store-
n-forward application styles. The load gets balanced and also the risk of any node failure impacting the overall system also
diminishes. Each such node in this case is a configured worker role application instance, which can be dynamically added or
removed through the Azure developer portal without the user having to physically configure or deploy the nodes on separate
machines.

Some real-life scenarios which the worker role model will help address on the cloud is:
1. Running Maintenance & Housekeeping operations such as data cleanup, backup & archival, etc
2. Executing resource intensive operations such as generation of mathematical, geographical models, image processing
   algorithms, etc
3. Processing large volumes of data in scenarios such as billing computations, report generations, etc
4. Act as a container to expose a self-hosted Internet-exposed service such as a custom application server; the likes of
   Tomcat
5. Make open source packages such as MySQL, MediaWiki, Subversion, Drupal, WordPress, Memcache, etc., to be
   deployed on Azure

Also bundled with Windows Azure is load balancing support which is implicit within the platform provided for both Web
and Worker Roles.

**Storage**

Windows Azure provides multiple storage services that are highly durable, scalable as well as constantly available. Azure
storage provides users with following capabilities to persist both structured as well as unstructured data:

- Anywhere and anytime access
- Store data for any length of time
- Scale to store any amount of data
- Pay for only what is used/stored

Azure offers three types of storage services, which cater to unstructured, structured as well as transient data requirements,
such as:
1. BLOB
2. Table
3. Queues

To get access to storage services, a user has to provision a separate storage subscription as would have been done previously
while provisioning a compute hosted service, following which, the below storage options would be made available:
a. Containers to persist unstructured Blob data  
b. Tables to persist structured data objects represented as Entities  
c. Queues to persist transient messages  

1. BLOB

BLOB storage provides the capability to persist small to very large unstructured objects such as images, media, documents and XML on the cloud. Access to BLOB data is provided through REST based HTTP interfaces which enables the storage services to be accessible from any platform including non-Microsoft ones. Access to the BLOB services can be done programmatically also, storage SDK’s have been provided which enables developers to easily access and rapidly use the BLOB services from their application code.

There are two types of blogs: block blobs and page blob. A “bock blob” can store up to 200 GB of data and is optimized for streaming workloads. While a “page blob” supports up to 1 TB of data and is meant for random access.

BLOB storage capability can be compared with that of NTFS (local file system) on a Windows OS. Each object is stored in a Container and can also have properties associated with it. The properties can be used to store any meta-data information relevant to the object persisted.

Running on top of page blobs is a feature known as Windows Azure XDrive. The XDrive can be mounted as if it were a NTFS formatted hard drive, allowing it to work with normal file I/O APIs. This feature is useful in scenarios where traditional applications that require access to local file system for logging, file handling, etc., are to be migrated to Windows Azure.

2. Table

Table storage provides non-relational, schema-less but structured storage facility. It is built to provide massively scalable, constantly available and durable structured storage.

In the Table storage, a unit of data is persisted as an Entity. Each entity is defined as a set of attributes that constitute the properties of the entity. There is no limit on how many entities can be stored in one table but you can typically store billions of entities in one table. Also, the storage has no hard limits and should easily scale into the terabytes range.

Additionally, entities can also be grouped into partitions. A partition key along with row key are mandatory attributes of any entity object stored in Tables. Partition key is a user defined value used to logically group related set of entities and a Row key is a user defined unique value within the partition that the entity belongs to.

Partitions are required to support scale-out as well as highly durable storage capabilities to the cloud enabled applications on Azure.

- Partitions enable logically related group of entities with the common partition keys to be always scaled out as set of entities grouped together across different storage nodes. This provide users with the capability to run queries more efficiently over large entity sets
- Partitions support high durability by replicating data across multiple storage nodes. Data is replicated across multiple servers automatically based on the partition key provided. Tables monitored by the fabric and data may be moved to different storage endpoints as and when the usage gets high. Based on the partition, table storage is also highly durable as data is always replicated at least 3 times.

For programmatic access, Azure provides easy to use .NET APIs and REST interfaces, similar to the ones as we have seen with BLOB services.

From a provisioning standpoint, each Storage Account can have one or more tables and each table can contain one or more entities of similar as well as different schema types.

3. Queues

Queues provide reliable storage and delivery of messages in an application. It is used to support Store & Forward disconnected type architectural patterns which are essentially required to build highly scalable applications on Azure. Queues are used to transfer work between different roles (web role and worker role); hence it helps in asynchronous work dispatch. Its programming semantics ensure that a message can be processed at least once. Queues are accessible using REST interfaces. The message size for Azure Queues is limited to less than 8KB.
Management

The management services provide automated management of the infrastructure and the application developed on Azure. This service manages instance provisioning, application deployment, and changing service configuration.

The Azure Management aspects can be broadly categorized to comprise of the following four areas:

Provision - Deploy - Manage - Monitor

a) Provisioning

In order for Enterprises to start using the Azure platform, an Azure account will have to be first provisioned. An account is provisioned only after the registration process is successful. In the registration process, an Enterprise will have to register for the services from the Microsoft Azure Portal using a Live ID. Following which an invitation code through email would be sent. This invitation code is then used to activate the Azure account, which successfully completes the registration process. Note that the invitation codes are a means to provision hosted services and storage accounts to a particular Live ID account. A user will now be able to create new hosted services and storage accounts on Azure.

Azure supports the concept of Geo-Location/Availability Zones. This capability enables users to define the locations in which they would want the users to deploy their application or run the data storage from. The benefit it provides is:

1. To make sure that storage accounts and hosted services can be very close to each other, in the same data center, thus ensuring calls between the running application and data have high bandwidth and low latency to extract the best performance benefits.

2. For customers to be able to control the location where their data can reside, so as to meet any regional regulatory requirements especially around data privacy laws such as the US Patriot act, EU Privacy laws, etc., which may exist.

b) Deploy

The deployment process is another area managed by the Windows Azure portal. Here, it follows a two step process: Upload to Staging and Promotions to Production.

Developers upload their compiled cloud packages and configuration files through the Portal. These uploaded packages are first deployed in the staging area where a publically accessible test URL is generated, akin to dynamically generated GUID, for users to test the application. As a best practice, users should test their applications before promoting to Production.

Once the application has successfully run in staging, the application is to be promoted into production. After being promoted, the application is made accessible using a user friendly URL which is publically accessibly on the cloud.

c) Manage

The hosted application and storage accounts can be managed from the Azure Services Portal. Based on workload demand surges, the service portal allows users to configure the number of separate instances of the application that can be started. A user can easily scale-up or scale-down by specifying the number of nodes required in the configuration settings. The operation of a hosted application is governed by the configuration file which is the model that Azure acts on. This manage-by-model approach allows users to govern the behavior of application and the execution of its instances at runtime in a highly configurable manner.

Additionally, Windows Azure also offers Management Services and APIs which provide programmatic access to much of the functionality which is available through the Windows Azure Portal.

d) Monitor

In the new cloud scheme of things, IT Administrator tasks have been considerably alleviated. In the traditional model, an administrator's major chunk of work involved managing the infrastructure, included Provisioning, DR (Disaster Recovery), upgrades, etc., and which with Azure can be completely offloaded to be handled by Microsoft. However, now the administrator's task would have to be tuned to focus towards monitoring and tracking the usage of services deployed and running on Azure.
To address the same, the platform provides application usage analytics to administrators in the following ways:

- An Analytics dashboard on Azure developer portal to graphically display usage across different Azure services.
- Diagnostic and Monitoring service APIs to ascertain production metrics and programmatically integrate with user application to realize new scenarios such as raise notifications, enable dynamic scaling etc.
- REST based APIs to inter-operate with third-party monitoring products such as HP, BMC, etc.

**Windows Azure Platform AppFabric**

These extend key capabilities of the .NET framework on the cloud. Flexible business connectivity, service registry, messaging and federated access control of applications being some of the key capabilities supported.

a) Service Bus

Enterprise systems are usually confined and locked within the boundaries governed by the corporate firewalls. Architects and developers face a daunting task when it is required for information from these systems to be provided to partners such as suppliers, vendors, regulators, merged/acquired entities, etc., outside the corporate boundaries without compromising on the security policies of the enterprise. Some usual practices adopted in such scenarios are:

- The data is made available offline as extracts from an isolated ftp location
- For real-time data, a private point-to-point pipe is set up connecting to the internal systems with several checks and balances in place

These integration points entail high maintenance and development costs whenever new partners have to be onboarded to access the enterprise systems.

The AppFabric provides connectivity services akin to a service bus on the cloud that enables unlocking information locked within the enterprise systems and make it available to its partners. The service bus provides the technology infrastructure which allows applications to cross over corporate boundaries in a seamless and secure manner.

This is achieved by the connectivity services in the AppFabric Service Bus stack. The AppFabric Service Bus supports various connectivity models that enable communication across multiple applications which may be geo-distributed. The AppFabric Service Bus connectivity services support various communication modes such as one-way, request-reply, multi-broadcast, publish-subscribe, routing along with communication protocols such as TCP, HTTP/REST and SOAP protocols.

Enterprises will be able to quickly realize different integration scenarios spanning different platforms and geographies by the messaging capabilities supported by the AppFabric Service Bus on the cloud.
b) Access Control Services

Security concerns within an Enterprise primarily revolve around answering the following two questions:

1. Who is the caller?
2. What is the caller supposed to do?

The first, “Who” part, is about Authentication which is usually handled by the Enterprise Identity systems such as Active Directory, OpenId, OpenAuth, etc. The second, “What” part, is about Authorization which is what the AppFabric Access Control Services (ACS) is primarily responsible for.

Historically, several enterprise applications have built their own, often silos of access control mechanisms, embedded tightly within the application code. As a result of which the rules which govern access to these applications have become isolated and often difficult to maintain. Any change in policy will involve substantial impact in duplicating changes across several effected applications. One of the biggest challenges enterprises today face is the sheer complexity of handling cross-organizational federated Identity and its management, especially considering all the different vendors that play in this space.

What has been observed in the past is that, due to the departmentalization of business functions, enterprises have to provide support for authorized access in their applications, which have led to inclusion of lot of spaghetti code which is primarily written to deal with identities and the rules to govern access of the application functionality. As the number of systems involved in such scenarios increases, the complexity starts to become pretty overwhelming and unmanageable. This is very difficult to maintain. Additionally, if you expand the horizon of the systems which looks at a federated scenario, the complexity increases many folds. A federated scenario such as those involving partners, government agencies, etc., that require access in to Enterprise systems can be quite a challenge to implement, especially with each party having their own identity platform providers and security token issuers which may have to be integrated.

The solution here is to somehow factor out the access control logic from the application code base. This will get the enterprises in a much better position to handle Identities and the rules governing the access control across applications, across enterprises.

ACS fits in here to provide a remotely located centralized location to separately manage the access rules for applications. The ACS helps to decouple access control concerns from the application and offers a convenient place to configure and apply the access control rules. Not only does the ACS helps provide access control within applications but also manage federated access control. The platform will help manage identity across multiple domains and multiple identity providers in a centralized and thus a more manageable manner.

The AppFabric Access Control service is an elegant yet powerful & convenient way of externalizing your authorization logic. Additionally ACS helps provide an additional security layer over your services published on the AppFabric.

Source: Windows Azure Platform Training Kit
SQL Azure

These services provide SQL Server like rich functionality on the cloud, basically a SQL Server running on the cloud. With SQL Azure now available on the Azure platform, enterprises can build Data centric applications that demand rich relational data handling capabilities. Enterprises can offload data workload from their existing on-premise SQL instances and host it on the cloud with very minimal effort involved in doing the migration and, being able to leverage cloud benefits such as 24x7 availability, the low cost usage based model of consumption and faster time to market. The current SQL Azure release will support the following:

RDMS capabilities on the cloud:
- a. Tables, indexes and views
- b. Stored Procedures
- c. Triggers
- d. Constraints
- e. Table variables
- f. Session temp tables (#t)

SQL Azure has been designed with the intent of combining the best features of SQL Server running at scale with low friction and being highly compatible with existing on-premise SQL Server offerings. With the current SQL Server capabilities that have been enabled, it will attempt to address most of scenarios relevant for web and departmental applications. A few advanced SQL Server capabilities which may be enabled in the future could be:
- a. Distributed Transactions
- b. Distributed Query
- c. CLR compatibility
- d. Service Broker
- e. Spatial
- f. Physical server or catalog DDL and views

Building Block Services

The Windows Azure Platform also provides a set of building block services for developing cloud enabled applications. These are higher-level developer services which are programmable components, often exposed through standard SOAP or open REST-based endpoints that can be consumed from within any service friendly application. Being standards based, these services can be composed not only within applications running on Azure but also on platforms which are non-Azure or as a matter of non-Microsoft thus further increasing the adoption of Azure services. In fact, users can selectively choose to use the building block services independently from the rest of the Windows Azure Platform minimizing complete dependency on Azure for building applications. The Azure building block services are:

Codename “Dallas”

With the growth of the Internet and the dawn of Information explosion, has given rise to the unavailability of “Authentic and Authorized data”. Today, with enormous levels of information available on the Internet, access to information can be easily made available to any user by running casual searches through a browser, but then can we trust the data that we receive in the search results? Data available on the Internet could be stale, misleading or even inaccurate.

Also, consuming data in your application to meet business requirements such as address validation, geo-location coordinates, user demographics, etc., can be a big challenge considering that information is not easily consumable through online searches. Although today you may be able to receive consumable information through the several Information Content Providers which are out there. Content providers expose their data in the form of feeds or messages so that the consumer can easily make use of the data. However to search for the right content provider and trust the credentials of the same can be an area of concern. Also, in consuming data from the different Content Providers, the interoperability mechanisms have to be evaluated especially since there are no standards governing this space. This result in additional effort spent in retrofitting these integration points within the application design and development and further challenges with maintaining these disparate interfaces are also high.
Businesses require a trusted entity where information is accessible from a single location, easily consumable and at the same time, authentic and accurate. Hence, the need for an “Information Market Place” arises; a market place is a place where information can be treated as a commodity.

Azure provides a marketplace which is codenamed as “Dallas”. Dallas brings information together into a single location. Be this information in any form like images, documents, real time data, etc. Unified provisioning and billing framework are very important characteristics of this information market place. Dallas provides the required setup for a market place and brings together the content providers and the consumers. As enterprise would desire to trade information like a commodity, Dallas strives to provide this desire. Discovery is another key distinctive of any market place. Dallas provides the discovery services to be able to find content providers for a specific business domain. Contents can be viewed on the portal itself to get a quick snapshot of the information. Dallas also supports an inbuilt billing framework to provide consumption details to consumer as well as revenue details to the content providers.

Dallas has very simple provisioning process where a consumer needs to discover and select the provider, take a unified view of the data on Dallas itself and start using and benefitting from it. From the business user's perspective, the dataset available from the content providers are rendered within the Dallas portal along with a very powerful analytics tool called “Power Pivot”. As an independent analyst, you buy a subscription, use power pivot and take home what you need. Dallas provides capabilities to bring data from disparate sources at one place and then capabilities to slice and dice the data, analyze it and eventually empowering you to make decisions. Analytics capabilities are extended to consume these contents from within Excel using Power Pivot, Microsoft Office and SQL Server for rich reporting and analytics. Your transactional data combined with Reference data brought from Dallas gives richer data points for analysis.

For developers too, Dallas provides service based API’s for integration with the application code which can help developers build rich mash-up business applications composed of contents such as maps, news, weather conditions, geo-spatial data, etc., merged with enterprise data.

**Pinpoint**

Rapid growth in Cloud computing and virtualization has commoditized the software and hardware infrastructure.

Commodities need a market place where they can be bought and sold. Microsoft has made an effort to provide such a market place by launching “Pinpoint” where service providers and subscribers or consumers can quickly sell and find what they need. The benefits perceived from Pinpoint are manifold, to summarize - it perfectly connects the software life cycle with the demand supply link. Pinpoint provides marketing services, products and offerings, try before you buy features, competitor analysis, review comments from the existing consumers and finding a suitable (technically and geographically) service.
provider is made easy. Apart from providing a market place for software end products, professional services and solutions, Microsoft Pinpoint provides platforms for specialized application building blocks or components which can be used to develop complex set of applications.

How strange is it that we check out all vendors and products while buying a gadget like a cell phone or a compatible Bluetooth device for the cell phone, but can’t do it while we are looking for solution providers for our critical business processes. Microsoft Pinpoint would help finding the best fit IT solution required for meeting business needs.

**SQL Azure DataSync**

SQL Azure DataSync provides symmetry between SQL Azure and SQL Server through bi-directional data synchronization. It provides capabilities to synchronize on-premise SQL Server data with SQL Azure.

Enterprise will have several scenarios where the need for bi-directional data synchronization between on-premise and cloud will be required. Some of these scenarios are:

1. Migrating traditional on-premise applications to Azure
2. Extending on-premise process to cloud and making the data available on the cloud such as those from remote offices or mobile devices via cloud
3. Scheduled backup & archival of on-premise data

Architects and Developers can make use of these rich set of Azure services to develop cloud enriched applications and services.

**Azure Development Lifecycle**

Azure development lifecycle will primarily follow a two stage process. The two stages being:

**Application Development**

In the application development stage, Azure application code is constructed locally on a developer's workstation. The Application Development comprise of the following two phases done on-premise:

**Construct & Test**

In the build & testing phase, a Windows Azure application is developed in the Visual Studio IDE (Visual Studio 2008 and above). For developers building non-Microsoft applications to run on Windows Azure or consuming Azure services can do so by using their existing development platforms made possible by the availability of community built libraries available such as Eclipse Plug-ins, SDKs for Java, PHP or Ruby. However Visual Studio provides developers with the best development platform to build Windows Azure applications or consume Azure services.
Visual Studio provides Web Role and Worker Role project templates out-of-the-box, which are made available to ease the configuration required to build a specific project type. In an Azure Team development scenario, where multiple developers would be working simultaneously on the same Azure solution, configuration management would be an essential piece of the development lifecycle. From the Microsoft engineering stack, Visual studio Team foundation Server (TFS) is the source code configuration manager. It can be used to maintain, version and manage source code of the application which will be deployed on Azure. The user at any point of time is in complete control of the source code.

Once the application is developed, unit testing of the code is done by the developer locally using the Visual Studio suite of testing tools such as automated unit tests, web test, etc. The Azure developer tools contain emulators for both compute and storage known as Dev Fabric and Dev Storage respectively. These tools are provided for developers to unit test their application code before having it deployed on to Azure. In Team development scenarios, multiple developers can unit test their functionality from their respective workstations using the development emulators. And considering the application can be tested locally, the access to the Azure hosted account need not be provisioned for every developer.

To ensure a smooth transition of the application code to the cloud, developers should test their application using a three-phased approach which is depicted in the figure below:

In the absence of run-time debugging support on the cloud, this approach should be used so as to ensure that application functionality can be sufficiently tested locally, with the help of the development fabric and development storage emulators, before moving to the cloud.

Deploy

In this phase, successfully checked-in and unit tested code is base-lined and versioned. Following which the code is compiled and a package is published to be uploaded on the Azure platform. The deployment steps are outlined as below:

Build

In a Visual Studio Team Foundation Server (TFS) Team development scenario, the build process can be automated by means of using Build Verification Tests (BVTs). BVTs are a broad suite of tests that are run to verify the overall quality of a particular build; they can be enabled for projects developed for Azure as well. The same TFS build infrastructure along with MSBuild can help to enable BVTs for Azure projects as well. Successful completion of tests executed in the TFS BVT will result in the build process being initiated and which is programmed using MSBuild tasks.

Publish

The publish process involves creation of a deployment package and a configuration file as the output which has to be subsequently uploaded on the Windows Azure Platform. A package is a unit of deployment on the Azure platform. It will bundle all the binaries, content and service metadata information of the projects that are a part of an Azure development solution into a single compressed and encrypted file. Azure’s publish process packages through MSBuild scripts. Additionally, CSPack, a command line executable, is a part of the Azure developer tools which allow developers to publish their Azure services from outside the Microsoft Visual Studio environment. This is useful in scenarios where developers would want to deploy apps developed on other platforms such as Eclipse.

Upload

16 | Infosys – White Paper
To be able to deploy a package on Azure, the user would have to first provision a hosted service component on the Azure Portal. The hosted service component will host, execute and manage the deployed application in the Azure data center. Users can provision a hosted service component by purchasing a subscription. After the hosted component has been provisioned, the user is then free to manually upload a deployment package to either a staging or a production region of the hosted service.

To automate the upload process, as the Azure publish process packages through an MSBuild script, here developers can add custom tasks to the script which can help automate uploading packages directly to Azure Blob storage. And further Service Management APIs can be used to deploy deployment packages from the Blob storage to the compute environment provisioned to the user.

**Deployment & Release**

This stage of the Azure development lifecycle is carried out on the Azure development portal. A single Azure compute service provides access to two managed environments i.e. Staging and Production. A user can choose to upload an application either in the staging or directly to the production environment, along with the possibility for users to interchangeably swap the deployed code between the two environments.

From a development process standpoint, it is recommended to have the user's application services package deployed first to the staging region so that the application can be tested in a separate QA like environment and then subsequently promoted to production for release. This approach provides users with the capability to first test an application in a production like environment and release the application for public use only after it has been sufficiently tested thus improving the quality of the code released. The Deployment & Release stage involves the following two key phases:

**System Testing**

In this phase, different application tests are carried out. The tests can start with a basic “Smoke Test” to ensure that all basic functionality of the deployed services is running fine on the cloud. This can be followed by “Integration Testing” to ensure that all external touch points to the services are functioning as expected. Subsequently followed by a “User Acceptance Test", the services would be tested by a sample set of the application users. These are a representative set of tests, not necessarily limited to, that can be carried out as a part of the development lifecycle on the Azure platform, which an Enterprise can run as a part of their standard project delivery practices.

All these tests are carried out on the Azure platform, without any investments required to procuring or setting up separate test environments on-premise. These tests can be carried out in the default staging environment, as discussed previously or as the case may be with large projects, separate environments to carry out each test may be required and for which additional hosted service components would have to be provisioned separately.

**Production Release**

On completion of all the test cycles, the tested Azure services code can be released into production by promoting the services from the Windows Azure portal. The promoted services will execute from the production regions of the Azure data center fabric. As a part of the promotion process, the hosted services will have a public facing URL like “http://[project].cloudapp.net”, where [project] is the name of the project defined at the time of creating a new hosted instance. In the production stage, the services are configured, managed and monitored from the Windows Azure portal. Some activities which an administrator can control and manage are:

a. Configuring the number of running instances of the hosted service
b. Start/Stop service instances
c. Upgrade or delete deployed service packages
d. Monitor the usage of the hosted services
Enterprise Readiness Requirement

Azure is a new style of computing, and as we have seen so far, is radically different from the traditional computing models which have so strongly embedded within the enterprise IT fabric. This represents change and listed below are some of the challenges which IT would need to pay heed to while on-boarding apps on to Azure in the enterprise:

1. Cloud workload evaluation

Not all workloads are suitable for the cloud. Certain factors which influence this decision include regional government regulations, data governance, performance or mission criticality, TCO, SLAs, etc. These criteria’s should be used to carefully evaluate scenarios which can be identified as potential candidates for cloud.

Portfolio analysis and assessment of the enterprise application landscape will help with identifying applications which can be offloaded to Azure.

2. Unbeknownst usage leading to governance failure

With cloud, users will have on-demand access to computing resources at the swipe of a card. The traditional IT procurement cycle is drastically shortened if not completely disappeared; this has the potential to give rise to the proliferation of opportunistic apps which may not be under the direct control of IT. IT should be aware of the business use of cloud services, which will help IT, ensure that the customer and business requirements are being met. It would otherwise result in “Shadow IT” and increase overall business as well as IT risks.

Centralized provisioning and quota management will help the enterprise better manage its cloud usage. Also governance practices would need to be put in place to monitor and govern cloud computing adoption in the Enterprise. The intent would be to set expectations with the business and IT, as well as manage any business risks that may be posed by it.

3. Enterprise security implications

Security in an enterprise has been traditionally centralized and internal. The user identity is housed in a single enterprise-wide identity store and applications would trust it to allow users access. Whereas with the Azure model, the trust boundaries and control procedure would be different, the boundaries are more distributed and external spanning the context of the cloud service provider.

With the cloud, IT will have to rely on a trust based federated security model for their hosted applications. In this model, enterprise will have to trust transactions arriving from the service provider and vice-versa so as to allow the enterprise users secured and seamless access to its resources both on-premise as well as on cloud, which is also often known as the “Single Sign-on” experience. For IT, enterprise security compliances and audit policies will have to be upgraded to factor these newer models.

Additionally, access control and audit trail mechanisms to the cloud platform have to be put in place so as to prevent unauthorized access as well as to bring about control in safeguarding usage of the enterprise account and applications hosted on the cloud.

4. Enterprise compliance policies

Compliances in the enterprise are usually governed by defined policies. These policies will have to be updated to factor vendor trusted procedures, which in this case would be the agreements and procedures laid down by Microsoft.

5. Managing and Monitoring

Unlike traditional on-premise systems, Azure does not provide full control of the computing environment and resources to administrators. As we have seen earlier in this paper, the Azure data center is fully managed and controlled by Microsoft and the users are only provided with private virtualized instances, packaged as services, being the unit of computation and storage for hosting their apps. This will require a shift in the IT support operations, requiring changes in the techniques and processes by which IT administrator’s service requests for its internal customers currently. This is because Azure today does not permit administrators to deploy any custom or third-party tools, utilities, agents, etc., that might be in use extensively with the existing operational process to support and administer. Administrators use these tools to investigate into production related issues such as poor performance, crashes, etc.
On the Azure platform, the responsibility of the enterprise IT administrator will be to manage and monitor the virtual instances and the applications deployed. Administrators would be able to manage and monitor these resources from the Analytics dashboard and service management configurations available on the Azure developer portal. Additionally special purpose APIs (RESTful) are provided by the Windows Azure platform for the same. These APIs when integrated with existing enterprise management and monitoring processes/tools will help to align existing operations and provide a unified platform for not only monitoring but also managing both on-premise as well as cloud. Hence a deep re-assessment of existing enterprise operations in terms of tools and process will help to improve the effectiveness of the operations management on the Azure platform.

Design Considerations for Building Applications on Azure

The Azure platform has been designed to serve and meet global scale demands. However, this does not mean that an application deployed on Azure can automatically inherit scale-free behavior and instantly be expected to handle internet scale. In order for applications deployed on Azure to service internet scale demands, architects will have to design their apps based on certain key design principles, the basis of which forms the core to defining the Architecture of application deployed on Azure. These design principles are as mentioned below:

**Loose Coupling**

From an Architectural standpoint, design your application as independent components which are not tightly coupled. The loosely your architecture components are coupled; the better is the ability of your application to scale. Applications designed without clear separation of concerns will result in monolithic code which will not be able to scale and optimally use the scalability capabilities which are built into Azure. Designing everything as a Black Box with functionality exposed through publicly accessible interfaces will help you build loosely coupled components.

**Service Orientation**

Service oriented design approaches will help build loosely coupled systems that can scale on the Azure platform. This is possible as good service design principles recommend that service exposed functionality is available as atomic units accessible through explicitly defined service interfaces. These principles result in the formation of loosely coupled atomic components, also known as Black Box; the execution of code being self contained - that is it contains no call-level interdependence. Being atomic would make the components execute without any interdependence and provide the ability to be deployed and operated independently. The components can be processed in parallel across distributed worker roles which on the Azure platform would allow effective utilization of the scale capabilities build into Azure.
Message Queuing based Communication

Data transmission between the web and worker roles should be message based and the communication channel that have to be used are Queues acting as buffers. Communication is in the form of messages and queues are one of the key design criteria's that should be adopted while building loosely coupled systems.

Consider implementing the “store and forward” design patterns on the cloud to scale out your application to process large quantities of messages. All incoming requests on the cloud are buffered in the queues which are further picked up by activated worker roles for processing. As demand rises and the workload increases, the message arrival rate also increases. In order to service the increasing demand, the application owner can activate new instances of the worker roles almost instantaneously.

Data Partitioning

In a typical web application on the cloud, scaling out could be as easy as starting up additional web roles instances to handle the user requests and load balancing between them. However, you may find that parts of your overall architecture will start to become points of contention because everything will get busy at the same time. An example in case here would be a single database that handles the requests from all these web roles. As and when the application usage starts increasing and the data starts growing, your queries start taking a lot of time to return the results. A good strategy to overcome such situations would be to adopt an appropriate partition strategy for your application. Put simply, this involves breaking up a single piece of the information architecture of your application into smaller, logical, more manageable chunks. Partitioning that single element into smaller chunks allows the azure platform to scale them out and this is exactly the technique several large sites use to ensure that their architectures scale.

The intent is to concentrate related data sets as close together as possible, by partitioning the entities and co-locating related data partitions which in table storage is achieved by using similar partitions keys across related entities.

Stateless

Build your service components to be stateless. A stateful component limits the capability of your application to scale. This is one essential criterion which should be considered while building scalable applications on Azure. More so, since the Azure platform implicitly distributes workload across various instances of the roles by a pre-configured load balancer and that too without any affinity based support, there would be no guarantee of subsequent requests hitting the same running instance which had service the request previously. Hence, stateless components can simply be scaled out and the work load balanced between them, ideally with all instances of the component running in an active manner. Also, adopting partition strategies will come in handy if there is state that needs to be maintained, you need to find a workload partitioning strategy that will allow you to have multiple instances of those stateful components, where each instance is responsible for a distinct subset of the work and/or data.
A few typical Azure application architecture patterns are shown below:

```
<table>
<thead>
<tr>
<th>Patterns</th>
<th>Scalable Web apps</th>
<th>Large scale computation apps</th>
<th>Platform Inter-op</th>
<th>Software + Service</th>
</tr>
</thead>
<tbody>
<tr>
<td>Context</td>
<td>Build rich UI centric OLTP web applications to serve on-demand scale load 24/7.</td>
<td>Build compute intensive applications executing tasks in an offline mode.</td>
<td>Leverage instant scale capabilities in your existing open source non-microsoft apps.</td>
<td>Build composite applications combining traditional on-premise software with remote services to provide a consistent and seamlessly integrated user experience across devices and form factors.</td>
</tr>
</tbody>
</table>
| Azure Technology leveraged | • Azure Web Role  
• Azure Table storage  
• SQL Azure | • Azure Worker Role  
• Azure Queues  
• Azure Blob Storage | Azure Worker Role  
Azure AppFabric | |
| Examples | • Web 2.0 apps  
• Departmental apps | • Batch Jobs  
• Data Analytics | • Media Wiki, WordPress, Subversion  
• Memcache, MySQL | • Cross Enterprise Integration  
• Federation Access |
```

**Key benefits of the Windows Azure Platform**

**An Enterprise perspective**

With the emergence of cloud computing and platforms such as Azure we may experience the birth of a new generation of Enterprise applications which may converge the two otherwise opposing forces of IT & Business in the Enterprise.
Let's explore what benefits both IT & Business may realize by adapting to Azure and in turn converge to unite.

<table>
<thead>
<tr>
<th>IT</th>
<th>Business</th>
</tr>
</thead>
<tbody>
<tr>
<td>Building an internet scale application can be accomplished with speed and efficiency</td>
<td>Flexibility is one of the fundamental benefits of cloud computing. Better ability to handle Peaks and Troughs in demand without any upfront investments required to handle peaks</td>
</tr>
<tr>
<td>Developers focus more on building functionality and business logic other than worrying about nonfunctional requirements of scalability, availability, etc.</td>
<td>Enables cost reduction and better cost management by moving from a Cap-ex driven model to a more Op-ex based transactional model</td>
</tr>
<tr>
<td>Azure Platform manages the heavy lifting required for scalability and fault tolerance</td>
<td>Deliver systems which are more agile and responsive to meet business needs</td>
</tr>
<tr>
<td>Managing hardware environments and software deployment is abstracted</td>
<td>Help companies to innovate faster &amp; cheaply. Businesses no longer risk expensive investments, they can now fail fast and cheaper too</td>
</tr>
</tbody>
</table>

A Developers Perspective

1. Web/Social

The Azure™ Services Platform provides web developers easy to use development tools and cloud infrastructure to build rich internet applications targeted for the browser and digital devices. Create socially aware solutions and connect with a network of over 460 million live users.

2. Dev Tools/Experience

Any web developer can use the platform. Developers familiar with .NET and Visual Studio can use their existing skills to extend or create new cloud-based applications that dynamically scale. Applications written in PHP or Ruby can also run on and take advantage of the Azure infrastructure and services.

3. Interoperable

Azure services use REST and SOAP web communications standards to interoperate with other platforms and services; run applications on any browser, create and expose your own services, or utilize the services regardless of platform or programming language.

4. Power of Choice

- The Windows Azure Platform allows developers to take advantage of one, all, or a combination of services.
- Author new applications or augment on-premises software with cloud services to create a new breed of rich internet-based solutions.

5. Economical

The Windows Azure Platform reduces onsite infrastructure needs and allows developers to continue using skills they already know from familiar development tools, all leading to lower cost and faster time to market.

Conclusion

In this paper we have discussed various on-demand services offered on the Windows Azure platform with architectural patterns representing the way these services can be leveraged by enterprises to build cloud based applications. Azure development lifecycle has been described to provide Enterprise project teams with insights into how existing methodologies would have to be altered for those projects targeted to be developed on Azure. Also challenges faced today on the platform were highlighted with strategies for enterprise to ready themselves before on-boarding on to Azure.

External References

1. MSDN
2. Windows Azure website
3. Windows Azure platform training kits
4. Introducing the Azure Services Platform - David Chappell

About the Author

Sidharth Ghag (Senior Technology Architect - Infosys Ltd)

Sidharth Ghag works as a Senior Technology Architect with the Microsoft Technology Center (MTC) in Infosys. With over ten years of industry experience, he currently leads solutions in Microsoft Technologies in the area of Cloud Computing. In the past he has also worked in the areas of SOA and service-enabling mainframe systems. He has been instrumental in helping Infosys clients with the service orientation of their legacy mainframe systems. Currently he helps customers adopt Cloud computing within their Enterprise. He has authored papers on Cloud computing and service-enabling mainframe systems. Sidharth blogs at http://www.infosysblogs.com/cloudcomputing

Email Sid - Sidharth_Ghag@infosys.com

Reviewer

Vikram Rajkondawar (Architect Advisor - Microsoft India)

Vikram Rajkondawar is an Architect Advisor with Microsoft’s Developer and Platform Evangelism team. Vikram has more than 13+ years of professional experience and has worked with multiple customers during his tenure. Currently, Vikram is the DPE Architect Advisor for Infosys Alliance. He leads technical engagement with Infosys and drive readiness, enablement and adoption of current and upcoming Microsoft Technologies within Infosys Developer and Architect community.

E-mail Vikram - Vikram.Rajkondawar@microsoft.com

About Infosys

Many of the world’s most successful organizations rely on Infosys to deliver measurable business value. Infosys provides business consulting, technology, engineering and outsourcing services to help clients in over 30 countries build tomorrow’s enterprise.

For more information about Infosys (NASDAQ:INFY), visit www.infosys.com.